

KNOWLEDGE REPRESENTATION

From Logic Programming Semantics to the Consistency of Syntactical Treatments of Knowledge and Belief

Thomas Bolander

Informatics and Mathematical Modelling
Technical University of Denmark
tb@imm.dtu.dk

Abstract

This paper concerns formal theories for reasoning about the knowledge and belief of agents. It has seemed attractive to researchers in artificial intelligence to formalise these propositional attitudes as predicates of first-order predicate logic. This allows the agents to express stronger introspective beliefs and engage in stronger meta-reasoning than in the classical modal operator approach. Results by Montague [1963] and Thomason [1980] show, however, that the predicate approach is prone to inconsistency. More recent results by des Rivières & Levesque [1988] and Morreau & Kraus [1998] show that we can maintain the predicate approach if we make suitable restrictions to our set of epistemic axioms. Their results are proved by careful translations from corresponding modal formalisms. In the present paper we show that their results fit nicely into the framework of logic programming semantics, in that we show their results to be corollaries of well-known results in this field. This does not only allow us to demonstrate a close connection between consistency problems in the syntactic treatment of propositional attitudes and problems in semantics for logic programs, but it also allows us to strengthen the results of des Rivières & Levesque [1988] and Morreau & Kraus [1998].

1 Introduction

The approach most often used in constructing formal theories for reasoning about multiagent systems is to formalise the agents' beliefs and knowledge through modal operators. An alternative approach is to formalise these propositional attitudes as predicates of a first-order predicate logic. This has several advantages, which have been widely discussed in the literature [Davies, 1990; Attardi and Simi, 1995; Carlucci Aiello *et al.*, 1995; McCarthy, 1997; Grant *et al.*, 2000]. Most importantly, it allows us to quantify over the propositional objects of knowledge and belief as for instance in “agent 1 believes that *everything* known by agent 2 is also known by agent 1”, formalised by

$$B_1(\ulcorner \forall x (K_2(x) \rightarrow K_1(x)) \urcorner).$$

This formula has no counterpart in the classical modal operator approach, since if K_1 and K_2 were modal operators, we would not be able to apply them directly to the variable x (modal operators only apply to well-formed formulas). Thus the predicate approach gives us more expressive power and the ability of agents to refer to the totality of their own and others beliefs, which is important in meta-reasoning.

Unfortunately, the predicate approach easily becomes inconsistent, since the added expressive power allows the agents to express self-referential beliefs that in some cases turn out to be paradoxical. This was proved by Montague [1963] and Thomason [1980]. They prove that certain axiom schemes describing natural properties of knowledge and belief are inconsistent with formal arithmetic. Their results are reviewed in Section 3. Des Rivières & Levesque [1988] and Morreau & Kraus [1998] have shown there to be a way out of these inconsistency results: to suitably restrict the set of sentences that we instantiate our axiom schemes of knowledge and belief with. These results are reviewed in Section 3 as well.

In this paper we will show that the results of des Rivières, Levesque, Morreau and Kraus can be reduced to well-known results in logic programming semantics. This is carried out in Section 4. In Section 5 we give a strengthening of their results, using again the connection to logic programming semantics.

2 Terminology and Notation

We will be using theories of first-order predicate logic to formalise propositional attitudes of agents. To prove the consistency of these first-order theories, we use results from logic programming semantics. Below we introduce the kinds of logic programs and first-order languages we will be considering.

2.1 Logic Programs

All logic programs considered in this paper will be propositional. Thus, an **atom** is simply a propositional letter, and a **literal** is either a propositional letter p or its negation $\neg p$. We take the symbols *true* and *false* to be among our propositional letters with the obvious intended interpretation. A **clause** is a formula of propositional logic on the form

$$H \leftarrow L_1 \wedge L_2 \wedge \cdots \wedge L_n$$

where H is an atom, $n \geq 1$ and all L_i are literals. A **propositional program** (or simply a **program**) is a (possibly infinite) set of clauses. **Herbrand models** of programs are defined in the usual way. We require that all models assign the truth-value true to the propositional letter *true* and false to *false*.

Given a program P , $\text{comp}(P)$ denotes its Clark completion. Since we work only with propositional programs, the Clark completion is particularly simple. The **Clark completion** of P is the following set of equivalences in infinitary propositional logic: for each atom A in P ,

- if A does not appear as head of any clause in P , then $A \leftrightarrow \text{false} \in \text{comp}(P)$.
- otherwise we have $A \leftrightarrow \bigvee_{i \in I} B_i \in \text{comp}(P)$, where $\{A \leftarrow B_i \mid i \in I\}$ is the set of clauses in P with head A .

Let P be a propositional program. The **dependency graph** of P is the directed graph with signed edges defined as follows. The nodes of the graph are the atoms (propositional letters) occurring in P excluding the special atoms *true* and *false*. There is a positive edge from A to B , denoted by $\langle A, B, + \rangle$, if and only if there is a clause $A \leftarrow L_1 \wedge \dots \wedge L_n$ in P such that $L_i = B$ for some $1 \leq i \leq n$. If $L_i = \neg B$ then there is a negative edge from A to B , denoted $\langle A, B, - \rangle$. We say that A **depends** on B , denoted by $A < B$, if there is a proper path from A to B in the graph. We say that A **depends negatively** on B , denoted by $A <_1 B$, if there is a path from A to B containing at least one negative edge. A program P is called **locally stratified** if the relation $<_1$ in the dependency graph of P is well-founded.

2.2 First-Order Languages

We use L to range over languages of first-order predicate logic. We take the connectives of first-order logic to be $\neg, \wedge$ and $\exists$. When using $\vee, \rightarrow, \leftrightarrow$ and $\forall$ in formulas, these formulas are simply abbreviations of formulas containing only $\neg, \wedge$ and $\exists$. We require all languages L to contain the one-place predicate symbols T and $\mathcal{P}$. $\mathcal{P}$ will be used as a predicate that picks out a set of (codes of) formulas in L . T will, depending on the context, be used to express one of our syntactic attitudes *belief* or *knowledge*. By $L - \{T\}$ we denote the language L with the predicate symbol T removed. We will assume that all considered languages contain a parametrised coding. By a **parametrised coding** in L we understand an injective map $\ulcorner \cdot \urcorner$ from the formulas of L into the terms of L satisfying:

- (i) For any formula φ in L , the term $\ulcorner \varphi \urcorner$ has the same free variables as φ (but $\ulcorner \varphi \urcorner$ is not itself a variable).
- (ii) For any formula $\varphi(x)$ in L and any term τ which is free for x in $\varphi(x)$, $\ulcorner \varphi(\tau) \urcorner$ is the term obtained by substituting τ for all free occurrences of x in $\ulcorner \varphi(x) \urcorner$.
- (iii) The coding is *well-founded*, that is, there is no infinite sequence of formulas $\varphi_0, \varphi_1, \varphi_2 \dots$ such that $\ulcorner \varphi_{i+1} \urcorner$ is a term in φ_i for all $i \in \mathbb{N}$.

We refer to [Feferman, 1984] for the construction of a parametrised coding. Feferman's coding does not satisfy (ii), but a simple variant of it will. $\ulcorner \varphi \urcorner$ is called the **code** of

φ . The intended interpretation of a formula $T(\ulcorner \varphi \urcorner)$ is that “ φ is known” or “ φ is believed”. We assume all first-order languages L to contain the language of Peano arithmetic. Throughout the paper, by *formal arithmetic* we mean Robinson's arithmetic, though any other standard formalisation of arithmetic could have been used in its place. We identify first-order languages L with their sets of sentences. By a *sentence* in L we understand a closed formula, that is, a formula without any occurrences of free variables. The set of ground terms of L is denoted $\text{Terms}(L)$.

To avoid confusion between formulas of propositional programs and formulas of first-order languages we will use Latin letters for the former and Greek letters for the latter.

2.3 Regular Formulas and RPQ Formulas

We now define the sets of first-order sentences which we intend to prove that our axiom schemes of knowledge and belief can consistently be instantiated with.

Definition 1. Let L be a first-order language. The set of **regular formulas** of L is the least set satisfying:

- (i) Any atomic formula of $L - \{T\}$ is a regular formula.
- (ii) If φ and ψ are regular formulas and x is a variable, then $\varphi \wedge \psi$, $\neg \varphi$ and $\exists x \varphi$ are regular formulas.
- (iii) If φ is a regular formula then $T(\ulcorner \varphi \urcorner)$ is a regular formula.

Our definition differs slightly from the one given by Moreau and Kraus [1998]. Instead of using a parametrised coding, they have an $(n + 1)$ -place predicate symbol T^n for each n , such that instead of writing $T(\ulcorner \varphi(x_1, \dots, x_n) \urcorner)$, where $x_1, \dots, x_n$ are the free variables of φ , they would be writing $T^n(\ulcorner \varphi \urcorner, x_1, \dots, x_n)$ where $\ulcorner \cdot \urcorner$ is then a standard (non-parametrised) Gödel coding. To simplify matters, we have chosen to take care of the free variables by using a parametrised coding rather than by introducing infinitely many predicate symbols of different arities.

As an example of a regular formula we have, for a suitable choice of L ,

$$\exists x T(\ulcorner \text{telephone}(\text{Mike}) = x \urcorner),$$

expressing that the agent knows Mike's telephone number. If we have more than one agent, we can of course introduce a predicate symbol T_i for each agent i . In that case the following sentence also becomes regular:

$$T_1(\ulcorner \forall x (T_2(\ulcorner \text{departure-time}(\text{train}, x) \urcorner) \rightarrow \text{departure-time}(\text{train}, x)) \urcorner),$$

expressing that agent 1 believes agent 2 to have correct beliefs about the departure time of the train. As an example of a non-regular formula we have

$$T_{\text{user}}(\ulcorner \forall x (\text{utter}(\text{system}, x) \rightarrow T_{\text{system}}(x)) \urcorner),$$

expressing that the user believes that the system only utters what it believes to be the case. It is non-regular because T_{system} is applied directly to a variable and not to the code of a formula. To allow expressing beliefs such as this one, Moreau and Kraus [1998] extended the set of regular formulas to a more inclusive class called the RPQ formulas.

Definition 2. Let L be a first-order language. The set of **RPQ formulas** of L is the least set satisfying:

- (i) Any atomic formula of $L - \{T\}$ is an RPQ formula.
- (ii) If φ and ψ are RPQ formulas and x is a variable, then $\varphi \wedge \psi$, $\neg\varphi$ and $\exists x\varphi$ are RPQ formulas.
- (iii) If φ is an RPQ formula then $T(\ulcorner\varphi\urcorner)$ is an RPQ formula.
- (iv) If φ is **any formula** in L , then $\exists x(\mathcal{P}(x) \wedge \varphi)$ is an RPQ formula.

This definition also differs from the one given by Morreau and Kraus [1998]. It defines a slightly more inclusive set of formulas, and at the same time it is simpler, since it avoids Morreau and Kraus' use of two distinct collections of variables.

By **regular sentence** we understand a closed regular formula, and by **RPQ sentence** a closed RPQ formula.

3 Review of Previous Results

Consider the following axiom schemes in a first-order language L :

- A1. $T(\ulcorner\varphi\urcorner) \rightarrow \varphi$
- A2. $T(\ulcorner T(\ulcorner\varphi\urcorner) \rightarrow \varphi\urcorner)$
- A3. $T(\ulcorner\varphi \rightarrow \psi\urcorner) \rightarrow (T(\ulcorner\varphi\urcorner) \rightarrow T(\ulcorner\psi\urcorner))$
- A4. $T(\ulcorner\varphi\urcorner)$, if φ is a theorem in formal arithmetic.
- A5. $T(\ulcorner\varphi\urcorner) \rightarrow T(\ulcorner T(\ulcorner\varphi\urcorner)\urcorner)$
- A6. $\neg T(\ulcorner\varphi \wedge \neg\varphi\urcorner)$

As already mentioned, $T(\ulcorner\varphi\urcorner)$ is intended to denote either “ φ is known” or “ φ is believed”. Thus, for instance, the first axiom scheme expresses that everything known (believed) is true. It seems reasonable to characterise knowledge by the axiom schemes A1–A4 and belief by A2–A6. But the following theorem shows that this is not always possible.

Theorem 3 (Montague [1963], Thomason [1980]). Let L be a first-order language. Formal arithmetic extended with any of the following sets of axioms is **inconsistent**.

- (a) The axiom schemes of knowledge, A1–A4, instantiated over the sentences of L .
- (b) The axiom schemes of belief, A2–A6, instantiated over the sentences of L .

The inconsistency of (a) is Montague's result, and the inconsistency of (b) is Thomason's result. A way out of these inconsistency results is to restrict the set of sentences that we instantiate A1–A6 with. This strategy gives us the following positive results.

Theorem 4 (des Rivières and Levesque [1988]). Let L be a first-order language. Formal arithmetic extended with any of the following sets of axioms is **consistent**.

- (a) The axiom schemes of knowledge, A1–A4, instantiated over the regular sentences of L .
- (b) The axiom schemes of belief, A2–A6, instantiated over the regular sentences of L .

Theorem 5 (Morreau and Kraus [1998]). Theorem 4 still holds when we replace “regular sentences” with “RPQ sentences”.

Theorem 4 is proved in [des Rivières and Levesque, 1988] by a careful translation from a corresponding first-order modal logic. Theorem 5 is proved in [Morreau and Kraus, 1998] by a similar translation from a corresponding second-order modal logic. In the following section we give proofs of their results taking a completely different route. We show that the problems can be reduced to problems of consistency of particular logic programs.

Instead of working directly with the axiom schemes A1–A6, we will most of the time be working with the *truth scheme* which is the following axiom scheme:

$$T(\ulcorner\varphi\urcorner) \leftrightarrow \varphi.$$

This is often sufficient since, as the following lemma shows, instances of axiom schemes A1–A6 are logical consequences of corresponding instances of the truth scheme. To prove that the axiom schemes A1–A6 instantiated over a set of sentences S are consistent, it thus suffices to prove the consistency of the truth schema instantiated over that same set.

Lemma 6. Let L be a first-order language, and let S be a set of sentences in L satisfying:

if φ and ψ are in S then $T(\ulcorner\varphi\urcorner)$, $\neg\varphi$ and $\varphi \wedge \psi$ are in S .

Let $\mathcal{M}$ be a model of L in which $T(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ holds for all φ in S . Then all of A1–A6 hold in $\mathcal{M}$ for all φ, ψ in S .

Proof. That A1 holds in $\mathcal{M}$ when φ is in S is a trivial consequence of the fact that $T(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ holds in $\mathcal{M}$. To see that A2 holds in $\mathcal{M}$, we first note that if φ is in S then $\neg(T(\ulcorner\varphi\urcorner) \wedge \neg\varphi)$ is in S as well, by assumption on S . This sentence is an abbreviation of $T(\ulcorner\varphi\urcorner) \rightarrow \varphi$, so we get that the following instance of the truth schema holds in $\mathcal{M}$:

$$T(\ulcorner T(\ulcorner\varphi\urcorner) \rightarrow \varphi\urcorner) \leftrightarrow (T(\ulcorner\varphi\urcorner) \rightarrow \varphi).$$

Using this together with the fact that A1 holds in $\mathcal{M}$, we get that $T(\ulcorner T(\ulcorner\varphi\urcorner) \rightarrow \varphi\urcorner)$ holds in $\mathcal{M}$. That is, A2 holds in $\mathcal{M}$. A3–A6 are proved to hold in $\mathcal{M}$ in a similar manner. $\square$

4 From LP Semantics to Consistent Treatments of Knowledge and Belief

The results of this paper are based on the following lemma.

Lemma 7 (Przymusiński [1987], Sato [1990]). If a program P is locally stratified then $\text{comp}(P)$ has a Herbrand model.

Our formulation is taken from [Sato, 1990]. It should be noted that Sato is not considering infinite programs in his paper, but his proof carries over without modification to this more general framework. This is because Sato is considering the set of ground instances of non-propositional programs rather than these programs themselves. The set of ground instances of a finite non-propositional program is in general an infinite propositional program, that is, the kind of logic program we are considering in this paper.

Definition 8. Let L be a first-order language, and let S be a set of sentences in L . We define an infinite program $P_{L,S}$ as follows. For every sentence φ in L , the program $P_{L,S}$ contains a propositional atom denoted p_φ . The clauses of $P_{L,S}$ are given by:

$$\begin{aligned} p_{\varphi \wedge \psi} &\leftarrow p_\varphi \wedge p_\psi, & \text{for all } \varphi, \psi \in L. \\ p_{\neg \varphi} &\leftarrow \neg p_\varphi, & \text{for all } \varphi \in L. \\ p_{\exists x \varphi(x)} &\leftarrow p_{\varphi(\tau)}, & \text{for all } \exists x \varphi(x) \in L \text{ and } \tau \in \text{Terms}(L). \\ p_{T(\ulcorner \varphi \urcorner)} &\leftarrow p_\varphi, & \text{for all } \varphi \in S. \end{aligned}$$

The relation between models of the program $P_{L,S}$ and models of the first-order language L is given by the following lemma.

Lemma 9. Let L and S be as above. If $\text{comp}(P_{L,S})$ has a Herbrand model $\mathcal{M}$ then L has a Herbrand model $\mathcal{N}$ satisfying:

- (i) For every sentence φ in L ,

$$\mathcal{M} \models p_\varphi \Leftrightarrow \mathcal{N} \models \varphi. \quad (1)$$
- (ii) $\mathcal{N} \models T(\ulcorner \varphi \urcorner) \leftrightarrow \varphi$, for all $\varphi \in S$.

Proof. Assume $\mathcal{M}$ is a model of $\text{comp}(P_{L,S})$. $\text{comp}(P_{L,S})$ is the following set of equivalences:

$$\begin{aligned} p_{\varphi \wedge \psi} &\leftrightarrow p_\varphi \wedge p_\psi, & \text{for all } \varphi, \psi \in L. & (2) \\ p_{\neg \varphi} &\leftrightarrow \neg p_\varphi, & \text{for all } \varphi \in L. & (3) \\ p_{\exists x \varphi(x)} &\leftrightarrow \bigvee_{\tau \in \text{Terms}(L)} p_{\varphi(\tau)}, & \text{for all } \exists x \varphi(x) \in L. & (4) \\ p_{T(\ulcorner \varphi \urcorner)} &\leftrightarrow p_\varphi, & \text{for all } \varphi \in S. & (5) \end{aligned}$$

Let $\mathcal{N}$ be the following Herbrand model of L :

$$\mathcal{N} = \{\varphi \in L \mid \varphi \text{ is an atom and } \mathcal{M} \models p_\varphi\}.$$

(i) is proved by induction on the syntactic complexity of φ . If φ is an atom then (1) holds by definition of $\mathcal{N}$. To prove (1) for sentences of the form $\varphi \wedge \psi$, $\neg \varphi$ and $\exists x \varphi(x)$ we simply use (2), (3) and (4), respectively. For the case of $\neg \varphi$ the proof is:

$$\begin{aligned} \mathcal{M} \models p_{\neg \varphi} &\Leftrightarrow \mathcal{M} \models \neg p_\varphi \Leftrightarrow \mathcal{M} \not\models p_\varphi \stackrel{\text{ih}}{\Leftrightarrow} \\ &\mathcal{N} \not\models \varphi \Leftrightarrow \mathcal{N} \models \neg \varphi, \end{aligned}$$

where the first equivalence is by (3) and the third is by induction hypothesis. The two remaining cases are proved similarly. Thus (i) holds. Furthermore, using (i) and (5), we get for all $\varphi \in S$:

$$\mathcal{N} \models T(\ulcorner \varphi \urcorner) \Leftrightarrow \mathcal{M} \models p_{T(\ulcorner \varphi \urcorner)} \Leftrightarrow \mathcal{M} \models p_\varphi \Leftrightarrow \mathcal{N} \models \varphi,$$

and thus $\mathcal{N} \models T(\ulcorner \varphi \urcorner) \leftrightarrow \varphi$, proving (ii). $\square$

Lemma 10. Let L be a first-order language and let R be the set of regular sentences in L . The propositional program $P_{L,R}$ is locally stratified.

Proof. To simplify matters we will throughout this proof be identifying every propositional letter p_φ with the corresponding first-order sentence φ in L . It should always be clear from the context whether φ is used to denote the first-order sentence or the corresponding propositional letter. Thus, by the identification, the nodes of the dependency graph of $P_{L,R}$ are all sentences in L . The edges are:

- $\langle \varphi \wedge \psi, \varphi, + \rangle$ and $\langle \varphi \wedge \psi, \psi, + \rangle$, for all $\varphi, \psi \in L$.
- $\langle \neg \varphi, \varphi, - \rangle$, for all $\varphi \in L$.
- $\langle \exists x \alpha(x), \alpha(\tau), + \rangle$, for $\exists x \alpha(x) \in L$ and $\tau \in \text{Terms}(L)$.
- $\langle T(\ulcorner \varphi \urcorner), \varphi, + \rangle$, for all $\varphi \in R$.

Edges of the first type are called $\wedge$ -edges, edges of the second type are called $\neg$ -edges, edges of the third type are called $\exists$ -edges and edges of the last type T -edges.

We have to prove that $P_{L,R}$ is locally stratified. Actually, we will be proving something slightly stronger. We will prove that the relation $<$ in the dependency graph of $P_{L,R}$ is well-founded. That is, we will prove that there does not exist any path of infinite length in the graph. Assume the opposite, that is, assume the existence of an infinite path σ .

Claim. σ contains infinitely many T -edges.

Proof of claim. Assume the opposite. Then there will be an infinite subpath σ' of σ containing no T -edges. Thus all edges on σ' must be $\wedge$ -, $\neg$ - or $\exists$ -edges. But note that for any such edge, the start node will have higher syntactic complexity than the end node. Thus, along σ' the syntactic complexity will be strictly decreasing, which contradicts σ' being infinite. This proves the claim. $\diamond$

With every formula φ in L we now associate a natural number $d(\varphi)$, called the T -degree of φ . The T -degree is defined recursively by

- $d(\varphi) = 1 + d(\psi)$, if $\varphi = T(\ulcorner \psi \urcorner)$ for some ψ .
- $d(\varphi) = 0$, if φ is any other atomic formula.
- $d(\varphi) = \max\{d(\psi) \mid \psi \text{ is a subformula of } \varphi\}$, otherwise.

The well-foundedness of the parametrised coding ensures that d is well-defined. By the above claim, σ contains an infinite number of T -edges. Let φ be the end node of such an edge. Then φ is regular. Let σ' be the infinite subpath of σ having φ as its start node. Then every node on σ' must be a regular formula (c.f. the definition of a regular formula). This implies that every edge on σ' is

- (i) either a $\wedge$ -, $\neg$ - or T -edge,
- (ii) or of type $\langle \exists x \alpha(x), \alpha(\tau), + \rangle$, where $\alpha(x)$ does not contain $T(x)$ as a subformula.

Item (ii) follows from that fact that when x is a variable then $T(x)$ is not a regular formula, and therefore no formula having $T(x)$ as a subformula can be regular either. Now note that on any edge of type (i) or (ii), the T -degree of the end node will be less than or equal to the T -degree of the start node. Thus the T -degree will be monotonically decreasing along σ' and must therefore be constant from some point. But then from this point it can not contain any T -edges, since the T -degree of the end node of such an edge is always one less than the T -degree of the start node. This contradicts the claim above. $\square$

Lemma 11. Let L be a first-order language and let S be a set of sentences in L . If $P_{L,S}$ is locally stratified then any Herbrand model of $L - \{T\}$ can be expanded into a Herbrand model of L in which $T(\ulcorner \varphi \urcorner) \leftrightarrow \varphi$ holds for all φ in S .

Proof. Let $\mathcal{M}$ denote a Herbrand model of $L - \{T\}$. Let P be the program $P_{L,S}$ extended with the following clauses:

- $p_\varphi \leftarrow \text{true}$, if φ is an atom in $L - \{T\}$ and $\mathcal{M} \models \varphi$.
- $p_\varphi \leftarrow \text{false}$, if φ is an atom in $L - \{T\}$ and $\mathcal{M} \models \neg\varphi$.

$P_{L,S}$ is assumed to be locally stratified, and since P has the same dependency graph as $P_{L,S}$, then P must be locally stratified as well. Therefore $\text{comp}(P)$ has a Herbrand model $\mathcal{M}'$, by Lemma 7. Finally, Lemma 9 gives us the existence of a Herbrand model $\mathcal{N}$ of L in which the equivalences $T(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ hold for all φ in S . To see that $\mathcal{N}$ expands $\mathcal{M}$ we just have to note that if φ is an atom in $L - \{T\}$ then

$$\mathcal{M} \models \varphi \Rightarrow p_\varphi \leftarrow \text{true} \in P \Rightarrow \mathcal{M}' \models p_\varphi \Rightarrow \mathcal{N} \models \varphi$$

and

$$\mathcal{M} \models \neg\varphi \Rightarrow p_\varphi \leftarrow \text{false} \in P \Rightarrow \mathcal{M}' \models \neg p_\varphi \Rightarrow \mathcal{N} \models \neg\varphi,$$

where the last implications are by (i) in Lemma 9. $\square$

Theorem 12. *Let L be a first-order language and let U be a theory in $L - \{T\}$ containing formal arithmetic. If U has a Herbrand model then U extended with any of the following sets of axioms has a Herbrand model.*

- (i) *The axiom scheme $T(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ instantiated over the regular sentences of L .*
- (ii) *The axiom schemes of knowledge, A1–A4, instantiated over the regular sentences of L .*
- (iii) *The axiom schemes of belief, A2–A6, instantiated over the regular sentences of L .*

Proof. Assume U has a Herbrand model $\mathcal{M}$. Let R denote the set of regular sentences in L . By Lemma 10, $P_{L,R}$ is locally stratified. So by Lemma 11, $\mathcal{M}$ can be expanded into a Herbrand model $\mathcal{N}$ in which $T(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ holds for all regular φ . This proves (i). (ii) and (iii) then immediately follow, using Lemma 6. $\square$

Theorem 4 is an immediate consequence of Theorem 12, when taking U to be formal arithmetic. The machinery we have introduced can also be applied to prove Theorem 5. It is an immediate consequence of the following corollary to Theorem 12.

Corollary 13. *Theorem 12 still holds when we replace “regular sentences” with “RPQ sentences”. Furthermore, the extension of $\mathcal{P}$ in the Herbrand model constructed will be the set of codes of regular sentences.*

Proof. Let S denote the set of RPQ sentences of L . Modify the program $P_{L,S}$ by removing every clause of the form

$$p\exists x(\mathcal{P}(x) \wedge \varphi(x)) \leftarrow p\mathcal{P}(\tau) \wedge \varphi(\tau),$$

where τ is not the code of any regular sentence. Call the new program $Q_{L,S}$. It is easy to see that $Q_{L,S}$ is locally stratified, using the argument given in the proof of Lemma 10. Lemma 11 still holds when we use $Q_{L,S}$ instead of $P_{L,S}$, so any Herbrand model of U can be expanded into a model of L in which $T(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ holds for all RPQ sentences. This proves (i) in Theorem 12 with “regular sentences” replaced by “RPQ sentences”. (ii) and (iii) then follows from Lemma 6. $\square$

5 Strengthening the Results

We now strengthen the results obtained above. We want to define a set of formulas more inclusive than the RPQ formulas that the truth scheme $T(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ can safely be instantiated with. For this we need a couple of new definitions.

Definition 14. *Let L be a first-order language and let φ be a formula in L . The set of formulas **occurring** in φ is defined as the least set containing φ and satisfying:*

- *If β is a subformula of a formula α occurring in φ , then β is occurring in φ .*
- *If $T(\ulcorner\alpha\urcorner)$ is occurring in φ , then α is occurring in φ .*

Assume ψ is a formula occurring in φ . The occurrence is said to be **negative** if ψ occurs in a formula α where $\neg\alpha$ occurs in φ . Otherwise the occurrence is called **positive**. An occurrence of ψ in φ is said to be **protected** if ψ occurs in a formula α where $\exists x(\mathcal{P}(x) \wedge \alpha)$ occurs in φ .

Thus, for instance, φ occurs in formulas such as $T(\ulcorner\varphi\urcorner) \wedge \neg\psi$ and $T(\ulcorner\neg T(\ulcorner\varphi\urcorner) \wedge \psi\urcorner)$ but not in $A(\ulcorner\varphi\urcorner)$ when $A \neq T$. φ has positive occurrence in $T(\ulcorner\varphi\urcorner) \wedge \neg\psi$ but negative occurrence in $T(\ulcorner\neg T(\ulcorner\varphi\urcorner) \wedge \psi\urcorner)$.

Definition 15. *Let L be a first-order language. A formula φ in L is said to be **weakly RPQ** if, for any variable x , the formula $T(x)$ only occurs positively or protected in φ .*

Note that in an RPQ formula, every occurrence of $T(x)$ for some variable x is protected, so every RPQ formula is also weakly RPQ. Thus the set of RPQ formulas is a subset of the set of weakly RPQ formulas. It is furthermore a proper subset, since among the weakly RPQ formulas we have e.g. $\exists x(\text{about.love}(x) \wedge T(x))$, which is not RPQ. The previously obtained results can be extended to the weakly RPQ formulas.

Theorem 16. *Let L be a first-order language and let U be a theory in $L - \{T\}$. If U has a Herbrand model, then U extended with the axiom scheme*

$$T(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$$

instantiated over the set of weakly RPQ sentences has a Herbrand model.

Proof. Let S denote the set of weakly RPQ formulas. Using Lemma 11, it suffices to prove that $P_{L,S}$ is locally stratified. As in the proof of Corollary 13, we can consider the modified program $Q_{L,S}$ instead. To obtain a contradiction, assume $Q_{L,S}$ is not locally stratified. Then $<_1$ in the dependency graph of $Q_{L,S}$ is not well-founded, that is, there must exist an infinite path σ containing infinitely many negative edges. As in the proof of Lemma 10, we get that σ must contain infinitely many T -edges. Let φ be the end node of such an edge. Then φ is weakly RPQ. Let σ' be the infinite sub-path of σ having this node as its start node. Then every node on σ' must be weakly RPQ. As noted in the proof of Lemma 10, if every edge on σ' is

- (i) either a $\wedge$ -, $\neg$ - or T -edge,
- (ii) or of type $(\exists x\alpha(x), \alpha(\tau), +)$ where $\alpha(x)$ does not contain $T(x)$ as a subformula,

then σ' can not be infinite. Thus, every node on σ' must have an occurrence of $T(x)$ for some variable x . Since all nodes are weakly RPQ, in each of these $T(x)$ is either positive or protected. But if $T(x)$ occurs protected in φ , there can be no infinite path starting at φ . Thus, in every formula φ on σ' , $T(x)$ must occur positively (for some variable x). But this implies that all edges in the path are positive, which contradicts our assumption. $\square$

The above theorem also relates to a result by Perlis [1985]. Perlis showed that a modified truth scheme $T(\ulcorner\varphi\urcorner) \leftrightarrow \varphi^*$ is consistent with arithmetic. It is easily seen that the set of instances of the (unmodified) truth scheme shown to be consistent by Perlis' result is contained in the set of instances shown to be consistent by Theorem 16.

In view of the results by Montague and Thomason (Theorem 3), there is a limit to how many instances of our axiom schemes we can add while still retaining consistency. The set of weakly RPQ sentences is quite close to this limit, as is made clear by the following example.

Example. Let L be a first-order language containing three one-place predicate symbols A , P and T . One of the simplest examples of a formula which is not weakly RPQ in L is the formula $\exists x (A(x) \wedge \neg T(x))$. Let us call this formula ψ . In ψ , $T(x)$ occurs negatively and unprotected (since $A \neq P$). Let U be the theory consisting of the following axioms

$$A(\ulcorner\psi\urcorner) \quad (6)$$

$$\neg A(\tau), \text{ when } \tau \text{ is a term } \neq \ulcorner\psi\urcorner. \quad (7)$$

U obviously has a Herbrand model $\mathcal{M}$, but we will show that U extended with the single axiom

$$T(\ulcorner\psi\urcorner) \leftrightarrow \psi \quad (8)$$

does not have a Herbrand model. This shows that Theorem 16 no longer holds if we to the weakly RPQ sentences add a sentence such as ψ . Assume, to obtain a contradiction, that there exists a Herbrand model $\mathcal{N}$ in which all of (6), (7) and (8) holds. Then we obtain the following contradiction:

$$\begin{aligned} \mathcal{N} \models \psi &\Leftrightarrow \mathcal{N} \models \exists x (A(x) \wedge \neg T(x)) \Leftrightarrow \\ \mathcal{N} \models A(\tau) \wedge \neg T(\tau) &\text{ for some } \tau \stackrel{(7)}{\Leftrightarrow} \\ \mathcal{N} \models A(\ulcorner\psi\urcorner) \wedge \neg T(\ulcorner\psi\urcorner) &\stackrel{(6)}{\Leftrightarrow} \mathcal{N} \models \neg T(\ulcorner\psi\urcorner) \Leftrightarrow \\ \mathcal{N} \not\models T(\ulcorner\psi\urcorner) &\stackrel{(8)}{\Leftrightarrow} \mathcal{N} \not\models \psi. \end{aligned}$$

6 Conclusion

We have been showing that results on the consistency of the predicate approach to knowledge and belief can be proved through the use of well-known results from logic programming semantics. This connects the two research fields in a new and interesting way, and have furthermore allowed us to strengthen the previously known results on the consistency of the predicate approach. It is expected that the connection between the two fields can be pursued further to get even better consistency results. This might be done by using some of the results from the literature on logic programming semantics that strengthen Lemma 7.

Acknowledgments

I would like to thank the following persons for guiding me in the direction that led to this paper: Jørgen Fischer Nilsson, Mai Gehrke, João Alexandre Leite and Ken Satoh. Special thanks to Roy Dyckhoff for his interest in my work and his many valuable comments.

References

- [Attardi and Simi, 1995] Guiseppe Attardi and Maria Simi. A formalization of viewpoints. *Fundamenta Informaticae*, 23(2-4):149–173, 1995.
- [Carlucci Aiello *et al.*, 1995] Luigia Carlucci Aiello, Marta Cialdea, Daniele Nardi, and Marco Schaerf. Modal and meta languages: consistency and expressiveness. In *Meta-logics and logic programming*, pages 243–265. MIT Press, 1995.
- [Davies, 1990] Nick Davies. A first order logic of truth, knowledge and belief. *Lecture Notes in Artificial Intelligence*, 478:170–179, 1990.
- [des Rivières and Levesque, 1988] Jim des Rivières and Hector J. Levesque. The consistency of syntactical treatments of knowledge. *Computational Intelligence*, 4:31–41, 1988.
- [Feferman, 1984] Solomon Feferman. Toward useful type-free theories I. *The Journal of Symbolic Logic*, 49(1):75–111, 1984.
- [Grant *et al.*, 2000] John Grant, Sarit Kraus, and Donald Perlis. A logic for characterizing multiple bounded agents. *Autonomous Agents and Multi-Agent Systems*, 3(4):351–387, 2000.
- [McCarthy, 1997] John McCarthy. Modality si! Modal logic, no! *Studia Logica*, 59(1):29–32, 1997.
- [Montague, 1963] Richard Montague. Syntactical treatments of modality, with corollaries on reflection principles and finite axiomatizability. *Acta Philosophica Fennica*, 16:153–166, 1963.
- [Morreanu and Kraus, 1998] Michael Morreanu and Sarit Kraus. Syntactical treatments of propositional attitudes. *Artificial Intelligence*, 106(1):161–177, 1998.
- [Perlis, 1985] Donald Perlis. Languages with self-reference I. *Artificial Intelligence*, 25:301–322, 1985.
- [Przymusiński, 1987] T. Przymusiński. On the declarative semantics of deductive databases and logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, 1987.
- [Sato, 1990] Taisuke Sato. Completed logic programs and their consistency. *Journal of Logic Programming*, 9(1):33–44, 1990.
- [Thomason, 1980] Richmond H. Thomason. A note on syntactical treatments of modality. *Synthese*, 44(3):391–395, 1980.

Inverse Circumscription

Hubie Chen

Department of Computer Science
Cornell University
Ithaca, NY 14853, USA
hubes@cs.cornell.edu

Abstract

Inverse (or *identification*) problems involve deciding whether or not an explicitly given set of data points have an implicit description, for instance, in the form of a constraint network. Such problems provide insight into the relationships among various representations of knowledge, which may have differing computational properties. This paper formalizes and studies the *inverse circumscription* problem, which (roughly speaking) is to decide, given a set of models, if there exists a formula whose circumscription describes the input set.

1 Introduction

The relationship between *implicit* and *explicit* descriptions of information is a central theme in knowledge representation and reasoning. For example, the core reasoning task of *propositional satisfiability* is to decide, given a propositional formula, whether or not the explicit set of models described implicitly by the formula is non-empty.

Recently, Kavvadias and Sideri [1998] studied a complementary task, which they called *inverse satisfiability*: given a set of models, is there a propositional formula with *exactly* the given set as its satisfying assignments? Without any restrictions on the class of formulas considered, the question is trivial, since for a given set of models M , it is always possible to create a formula in disjunctive normal form (DNF) capturing exactly M . Kavvadias and Sideri consider this question with respect to the class of formulas expressible by a pre-specified *constraint language* – a set of relations which can be used to express constraints. Their intriguing main result is that the complexity of the inverse satisfiability problem reflects exactly that of the classical satisfiability problem: for a fixed constraint language, inverse satisfiability is intractable if and only if satisfiability is intractable! For instance, INVERSE 3-SAT is intractable, as 3-SAT is intractable; but, INVERSE 2-SAT is tractable, as 2-SAT is tractable.

The inverse satisfiability problem is akin to the *identification* problems studied by Dechter and Pearl [1992], which involve deciding whether or not explicitly given relational data in the form of tuples has an implicit description as a constraint network with particular features. Studying such

inverse or *identification* problems is valuable for a variety of reasons, as articulated in [Dechter and Pearl, 1992; Kavvadias and Sideri, 1998]. First, certain restricted forms of propositional formulas, such as Horn formulas, facilitate efficient reasoning. Hence, the ability to decide when an explicitly given set of models can be represented in such a restricted form, is desirable. Second, model-based representations of information have been proposed as a viable alternative to formula-based representations [Kautz *et al.*, 1995; Khardon and Roth, 1996]; inverse problems address the relative expressibility of these two types of representations. Third, discovering structure in empirical data is a crucial component of scientific activity, and so understanding when structure discovery is computationally feasible sheds light on the nature of such activity.

In this paper, we further investigate the relationship between implicit and explicit descriptions of data by considering the complexity of *inverse circumscription*, an analog of the inverse satisfiability problem for circumscription. Circumscription is a well-studied non-monotonic reasoning formalism intended to embody common sense, and provides an alternative semantics for propositional formulas [McCarthy, 1980]. Intuitively, the models of the circumscription of a formula are those models of the original formula which make as few “assumptions” as necessary. The inverse circumscription problem addresses the expressiveness of circumscription: this problem involves deciding, roughly speaking, if a set of models can be described by a circumscribed formula (over some constraint language). Studying the complexity of this problem is natural not only for the aforementioned reasons, but also because circumscribed formulas are more *space-efficient* than uncircumscribed formulas: knowledge can be represented more succinctly with the former than with the latter, in a sense made precise in [Cadoli *et al.*, 1997].

There are at least two generalizations of the inverse satisfiability problem to the context of circumscription. One generalization is the problem of deciding, given a set of models, whether or not there exists a circumscribed formula (over a specified constraint language) with *exactly* the input set as its models. A second generalization is the problem of deciding, given a set of models, whether or not there exists a formula such that all models of the input set are models of the circumscribed formula – in other words, the problem is to decide the existence of an *approximating* formula whose circumscribed

models covers the input set of models. We note that the tightness of the approximating formula is not what is at stake. In our formulation of this problem, it is possible to efficiently compute a candidate formula ψ such that if there is any formula ϕ covering the input set of models, then ψ also covers the input set and ψ has fewer models than ϕ (that is, the model set of ψ is a subset of the model set of ϕ).

As we will show, the complexity of the first generalization, for almost all constraint languages, can be characterized fairly easily using previous results. Consequently, the focus of this paper is on the complexity of the second generalization, which we call the inverse circumscription problem. Our main result is a *dichotomy theorem* in the complexity of inverse circumscription, which states that for *all* constraint languages, inverse circumscription is either in P, or is co-NP-complete.

Complexity dichotomy theorems are important because they demonstrate a full understanding of the complexity of a problem, with respect to a particular form of problem restriction (in our case, a restriction on the constraint language). The first dichotomy theorem was obtained by Schaefer, who showed that the propositional satisfiability problem over a fixed constraint language is either in P, or is NP-complete. The non-trivial tractable cases of satisfiability given by this result are 2-SAT, HORN SAT, and XOR-SAT (where each constraint is a linear equation in the field with two elements). Since Schaefer's result, many other dichotomy theorems have been established [Creignou *et al.*, 2001].

Kirousis and Kolaitis [2001a] very recently established a dichotomy theorem in the complexity of model checking for circumscription. The model checking problem is to decide, given an assignment and a formula, whether or not the assignment is a model of the circumscribed formula. Their theorem states that, for any constraint language, the model checking problem is either in P or is co-NP-complete. Interestingly, we show that (for any constraint language) the complexity of inverse circumscription is *exactly* the complexity of this model checking problem, giving a correspondence analogous to that between the complexity of inverse satisfiability and the complexity of satisfiability. In fact, to establish hardness of inverse circumscription, we will reduce from the model checking problem.

2 Preliminaries

In this section, we present the definitions and notation that will be used throughout the paper. We first introduce the notation of a constraint; constraints will be the building blocks of the propositional formulas we consider.

Definition 2.1 A subset of $\{0, 1\}^k$ (for some $k \geq 1$) is called a logical relation, and is said to have arity k .

A constraint over variable set V is a logical relation R paired with a list of variables $x_1, \dots, x_k \in V$ where k is the arity of R , and is written $R(x_1, \dots, x_k)$.

A constraint with constants over variable set V is a logical relation R and a list of variables or constants $y_1, \dots, y_k \in V \cup \{0, 1\}$ where k is the arity of R , and is written $R(y_1, \dots, y_k)$.

We next introduce the notion of a S -formula, which is a propositional formula built using the relations of a constraint language S as “templates.”

Definition 2.2 A constraint language S is a finite set of logical relations (which may contain relations of different arities).¹ The maximum arity of S is the maximum over the arities of the relations in S .

An S -formula over variable set V is a set of constraints (over V) where the logical relation of each constraint is in S .

An S -formula with constants over variable set V is a set of constraints with constants (over V) where the logical relation of each constraint is in S .

We move on to describe the set of models associated with a formula. Intuitively, a model of a formula is an assignment to the variable set which obeys all constraints.

Definition 2.3 Let ϕ be an S -formula (possibly with constants) over variable set V .

An assignment to ϕ is a function $f : V \rightarrow \{0, 1\}$. When W is a subset of V , $f|_W$ denotes the restriction of f to W ; and, $\bar{f} : V \cup \{0, 1\} \rightarrow \{0, 1\}$ denotes the extension of f where $\bar{f}(0) = 0$, $\bar{f}(1) = 1$, and $\bar{f}(v) = f(v)$ for all $v \in V$.

The assignment f satisfies a constraint $R(x_1, \dots, x_k)$ of ϕ if the tuple $(\bar{f}(x_1), \dots, \bar{f}(x_k))$ is in R .

The assignment f satisfies ϕ if it satisfies all constraints in ϕ ; in this case, it is said to be a satisfying assignment or model of ϕ .

Define $\text{Models}(\phi)$ to be the set containing all models of ϕ .

For a fixed S , the problem of deciding whether or not an input S -formula has a satisfying assignment is in NP, since in non-deterministic polynomial time, a satisfying assignment can be guessed and verified. The following example demonstrates that there is an S for which this satisfiability problem is equivalent to that for 3-SAT, and hence NP-complete.

Example 2.4 Let $S^3 = \{R_0, R_1, R_2, R_3\}$, where $R_0 = \{0, 1\}^3 \setminus \{(0, 0, 0)\}$, $R_1 = \{0, 1\}^3 \setminus \{(1, 0, 0)\}$, $R_2 = \{0, 1\}^3 \setminus \{(1, 1, 0)\}$, and $R_3 = \{0, 1\}^3 \setminus \{(1, 1, 1)\}$. Every 3-SAT formula can be efficiently converted into a S^3 -formula with exactly the same set of satisfying assignments, and vice-versa. For example, the 3-SAT formula

$$(x \vee y \vee \neg z) \wedge (\neg w \vee y \vee \neg z) \wedge (z \vee \neg y \vee \neg w)$$

is equivalent to the S^3 -formula

$$\{R_1(z, x, y), R_2(w, z, y), R_2(y, w, z)\}.$$

We generalize the definition of S^3 in Example 2.4 as follows.

Definition 2.5 Let S^k (for $k \geq 1$) be the set of relations $\{R_i^k : i \in \{0, \dots, k\}\}$ where $R_i^k = \{0, 1\}^k \setminus \{t_i^k\}$, and $t_i^k \in \{0, 1\}^k$ is the tuple with 1 in the first i coordinates, and 0 in the remaining $k - i$ coordinates.

¹Throughout this paper, we assume that all logical relations and constraint languages are non-empty.

Having defined the notion of a model of a formula, we now define what it means for a model to be minimal. The circumscription of a formula is considered to have, as its models, the minimal models of the original formula [McCarthy, 1980]. We consider minimality with respect to a subset P of the variable set: a model is minimal if there is no model “below” it in a comparison based on the variables in P .

Definition 2.6 Let ϕ be an S -formula (possibly with constants) over variable set V , and let P be a subset of V . Let $\leq$ denote the standard total ordering on $\{0, 1\}$ where $0 \leq 1$.

Suppose that $\alpha : V \rightarrow \{0, 1\}$ and $\beta : V \rightarrow \{0, 1\}$ are both assignments over the same variable set V . We write $\alpha =_P \beta$ if for all $v \in P$, $\alpha(v) = \beta(v)$. We write $\alpha \leq_P \beta$ if for all $v \in P$, $\alpha(v) \leq \beta(v)$. We write $\alpha <_P \beta$ if $\alpha \leq_P \beta$ and it is not the case that $\alpha =_P \beta$. We write $\alpha \perp_P \beta$ if neither $\alpha \leq_P \beta$ nor $\beta \leq_P \alpha$ holds. When the set P is equal to V (the entire domain of α and β), we sometimes omit it when writing the above relations.

We say that β is a minimal satisfying assignment (or minimal model) of ϕ with respect to P if β is a model of ϕ ; and, for all models α of ϕ , $\alpha \leq_P \beta$ implies $\alpha =_P \beta$.

Define $\text{MinModels}(\phi, P)$ to be the set containing all minimal models of ϕ with respect to P .

We now introduce some terminology that can be used to describe constraint languages; this terminology will prove to be quite wieldy in describing many of the complexity dichotomy theorems presented in the next section. As usual, we define a CNF-SAT formula over variable set V to be a conjunction of clauses, where each clause is the disjunction of literals from V . (A *literal* from variable set V is either a variable $v \in V$ itself, or the negation of a variable $\neg v$. Literals of the former type are called *positive*, whereas those of the latter type are called *negative*.) An assignment $f : V \rightarrow \{0, 1\}$ satisfies a CNF-SAT formula ϕ if every clause of ϕ contains a literal evaluating to true under f .

Definition 2.7 A *bijunctive formula* is a CNF-SAT formula where each clause contains exactly two literals.

A (dual) *Horn formula* is a CNF-SAT formula where each clause contains at most one positive (negative) literal.

An *affine formula* is a conjunction of equations of the form $x_1 \oplus \dots \oplus x_m = c$, where the x_i are variables and $c \in \{0, 1\}$ is a constant. (The symbol $\oplus$ denotes the logical “exclusive or.”)

Definition 2.8 Let R be a logical relation of arity n , and S be a constraint language.

The relation R is 0-valid if it contains the all-zeroes tuple $(0, \dots, 0)$ of arity n , and is 1-valid if it contains the all-ones tuple $(1, \dots, 1)$ of arity n .

The relation R is *bijunctive* (Horn, dual Horn, affine) if there exists a bijunctive (respectively Horn, dual Horn, affine) formula over $\{v_1, \dots, v_n\}$ with the same set of satisfying assignments as the $\{R\}$ -formula $\{R(v_1, \dots, v_n)\}$.

The constraint language S is 0-valid (1-valid, bijunctive, Horn, dual Horn, affine) if every relation contained in S is 0-valid (respectively 1-valid, bijunctive, Horn, dual Horn, affine).

The constraint language S is *Schaefer* if at least one of the following four conditions hold: S is bijunctive, S is Horn, S

is dual Horn, S is affine. The constraint language S is non-Schaefer if it is not Schaefer.

3 Related Work

This section reviews relevant work done previously; throughout, S is used to denote a constraint language. We first mention the seminal work of Schaefer, who proved a complexity dichotomy theorem on the satisfiability problem for S -formulas.

Definition 3.1 The $\text{SAT}(S)$ decision problem.

Input: An S -formula ϕ .

Question: Is ϕ satisfiable?

The $\text{SAT}_C(S)$ decision problem is identical, except the input S -formula can have constants.

In other words, the $\text{SAT}(S)$ problem is to decide, for a given S -formula ϕ , if the set $\text{Models}(\phi)$ is non-empty. Remarkably, Schaefer proved that for any constraint language S , $\text{SAT}(S)$ is either in P or is NP -complete; in addition, he gave a precise description of which constraint languages yield a tractable satisfiability problem, and which do not.

Theorem 3.2 [Schaefer, 1978] Let S be a constraint language.

If S is 0-valid, 1-valid, or Schaefer, then $\text{SAT}(S)$ is in P ; otherwise, $\text{SAT}(S)$ is NP -complete.

If S is Schaefer, then $\text{SAT}_C(S)$ is in P ; otherwise, $\text{SAT}_C(S)$ is NP -complete.

Kavvadias and Sideri [1998] studied the “inverse satisfiability problem” (denoted here by $\text{INVERSE SAT}_C(S)$). In the “standard” satisfiability problem the goal is to decide, given a formula, whether or not there exists a satisfying assignment. In the “inverse” problem, the given input is a set of assignments, and the goal is to determine whether or not there exists a formula with exactly the given assignments as its set of satisfying assignments.

Definition 3.3 The $\text{INVERSE SAT}_C(S)$ decision problem.

Input: Set A of assignments over the same variable set V .

Question: Is there an S -formula ϕ with constants such that $A = \text{Models}(\phi)$?

A dichotomy theorem was established by Kavvadias and Sideri, which shows that the inverse satisfiability problem is always in P , or is co - NP -complete. Intriguingly, the complexity of the inverse problem reflects exactly the complexity of the satisfiability problem: $\text{SAT}_C(S)$ is intractable if and only if $\text{INVERSE SAT}_C(S)$ is intractable!

Theorem 3.4 [Kavvadias and Sideri, 1998] Let S be a constraint language.

If S is Schaefer, then $\text{INVERSE SAT}_C(S)$ is in P ; otherwise, $\text{INVERSE SAT}_C(S)$ is co - NP -complete.

We now formalize the model checking problem for circumscription, which was called “minimal satisfiability” in [Klirousis and Kolaitis, 2001a].

Definition 3.5 The $\text{MIN SAT}(S)$ decision problem.

Input: An S -formula ϕ over variable set V , an assignment $\alpha : V \rightarrow \{0, 1\}$ satisfying ϕ , and a subset P of V .

Question: Is the assignment α a minimal model of ϕ with respect to P ?

The $\text{MIN SAT}_C(S)$ decision problem is identical, except the input S -formula can have constants.

The $\text{MIN SAT}(S)$ problem is in **co-NP**, as deciding whether or not α is a minimal model of a formula amounts to verifying – for all assignments β – that if β is a model, then β is not strictly below α in the ordering $\leq_P$. Cadoli [1992] showed that this problem is **co-NP**-complete in general, and also identified some tractable cases. More recently, the following full dichotomy theorem was proved concerning the complexity of $\text{MIN SAT}(S)$.

Theorem 3.6 [Kirousis and Kolaitis, 2001a, Theorem 4.2]²

Let S be a constraint language.

*If S is Schaefer, then $\text{MIN SAT}_C(S)$ is in P ; otherwise, $\text{MIN SAT}_C(S)$ is **co-NP**-complete.*

*If S is 0-valid or Schaefer, then $\text{MIN SAT}(S)$ is in P ; otherwise, $\text{MIN SAT}(S)$ is **co-NP**-complete.*

4 The Inverse Circumscription Problem

The inverse satisfiability problem (defined formally in the previous section) is to decide, given a set of assignments, whether or not there is a formula with precisely the given input set as its models. The *inverse circumscription problem*, denoted by $\text{INVERSE MIN SAT}(S)$, is similar in that the input is also a set of assignments and the task is to decide whether or not there is a formula describing the input set. However, in the inverse circumscription problem, the question is whether there exists a formula ϕ such that all of the given assignments are *minimal* models of ϕ .

Definition 4.1 The $\text{INVERSE MIN SAT}(S)$ decision problem.

Input: Set A of assignments over the same variable set V , and a subset P of V .

Question: Is there an S -formula ϕ such that $A \subseteq \text{MinModels}(\phi, P)$?

The $\text{INVERSE MIN SAT}_C(S)$ decision problem is identical, except the question is to decide if there is an S -formula with constants satisfying the stated condition.

As mentioned in the introduction, there is a natural variant of the $\text{INVERSE MIN SAT}_C(S)$ problem which has an identical description, except the $\subseteq$ symbol is replaced with an $=$ symbol in the “question.” Let us denote this variant by $\text{INVERSE EXACT MIN SAT}_C(S)$. We have the following results concerning this question.

Theorem 4.2 Let S be a constraint language.

If S is bijunctive or Horn, then

$\text{INVERSE EXACT MIN SAT}_C(S)$ is in P .

²We note that the notation of [Kirousis and Kolaitis, 2001a] is different from ours. In particular, their decision problem $\text{MIN SAT}(S)$ involves checking, given an assignment and formula, if the assignment is minimal with respect to *all* variables. It is easily verified that our definition of $\text{MIN SAT}_C(S)$ is equivalent to their $(P; Q; Z) - \text{MIN SAT}(S)$, and that our definition of $\text{MIN SAT}(S)$ is equivalent to their $(P; \emptyset; Z) - \text{MIN SAT}(S)$. The theorem is stated here with respect to our notation.

*If S is non-Schaefer, then $\text{INVERSE EXACT MIN SAT}_C(S)$ is **co-NP**-complete.*

For non-Schaefer constraint languages S , hardness of $\text{INVERSE EXACT MIN SAT}_C(S)$ can be shown by first establishing the hardness of $\text{INVERSE EXACT MIN SAT}_C(S^3)$ by reduction from $\text{INVERSE SAT}_C(S^3)$, and then reducing from $\text{INVERSE EXACT MIN SAT}_C(S^3)$ to $\text{INVERSE EXACT MIN SAT}_C(S)$, using a technique in [Kavvadias and Sideri, 1998]. For Schaefer constraint languages S , the existence of an output polynomial time algorithm for computing minimal models of a S -formula implies the tractability of $\text{INVERSE EXACT MIN SAT}_C(S)$; see [Kavvadias et al., 2000] for such algorithms in the case of constraint languages S that are bijunctive or Horn.

For the remainder of this paper, we focus on the $\text{INVERSE MIN SAT}(S)$ decision problem. Given an input set A of this problem, it is possible to efficiently compute a “candidate formula” having the property that if there is *any* formula containing A in its set of minimal models, then the candidate formula is such a formula. That is, the candidate formula serves to witness that “yes” is the answer to the decision question of Definition 4.1 – so long as some formula does.

Definition 4.3 Suppose that S is a constraint language and that A is a set of assignments over the same variable set V . Define the candidate S -formula for A to be the set containing all constraints (over V and with relation in S) that are satisfied by every assignment in A . Similarly, define the candidate S -formula for A with constants to be the set containing all constraints with constants (over V and with relation in S) that are satisfied by every assignment in A .

Lemma 4.4 Suppose that S is a constraint language and that A is a set of assignments over a variable set V . Let P be a subset of V , and let ψ_C be the candidate S -formula (with constants) for A . There exists an S -formula (with constants) ϕ such that $A \subseteq \text{MinModels}(\phi, P)$ if and only if $A \subseteq \text{MinModels}(\psi_C, P)$.

For a fixed constraint language S , the candidate S -formula for a set of assignments A can be computed in polynomial time (measured with respect to the size of the representation of A). By the key property of the candidate formula (Lemma 4.4), it follows that $\text{INVERSE MIN SAT}(S)$ is in **co-NP**: an assignment β which is not included in A , satisfies the S -candidate formula of A , and is strictly below an assignment in A (with respect to $\leq_P$), serves as a succinct and efficiently checkable proof that A is a “no” instance of $\text{INVERSE MIN SAT}(S)$. (By similar reasoning, $\text{INVERSE SAT}_C(S)$ and $\text{INVERSE EXACT MIN SAT}_C(S)$ can be shown to be in **co-NP**, as discussed in [Kavvadias and Sideri, 1998].)

Whenever the model checking problem $\text{MIN SAT}(S)$ is in P , the inverse problem $\text{INVERSE MIN SAT}(S)$ will be in P . This is because deciding whether or not an input set A to the $\text{INVERSE MIN SAT}(S)$ problem is a “yes” instance amounts to verifying that every assignment in A is a minimal model of the candidate formula for A ; clearly, this can be done in polynomial time when $\text{MIN SAT}(S)$ is in P . To prove a full dichotomy theorem on the complexity of

INVERSE MIN SAT(S), it remains to describe the complexity of INVERSE MIN SAT(S) for the constraint languages S such that MIN SAT(S) is not in P .

5 Dichotomy Theorem

In this section, we describe completely the complexity profile of the “inverse circumscription problem.” In particular, we show that for those constraint languages S such that MIN SAT(S) is **co-NP-hard**, INVERSE MIN SAT(S) is also **co-NP-hard** (and similarly for MIN SAT_C(S) and INVERSE MIN SAT_C(S)).

Our first step is to prove hardness of the INVERSE MIN SAT(S) problem for 9-SAT formulas, where the circumscription is performed with respect to all of the variables. This initial hardness result is then used to establish the hardness of INVERSE MIN SAT(S) for other constraint languages S .

Theorem 5.1 *The problem INVERSE MIN SAT(S^9) is co-NP-hard, even with the restriction that the subset P must be equal to the entire variable set.*

The proof of this theorem is sketched in Appendix A; the hardness result claimed by the theorem is achieved by reduction from the MIN SAT(S) problem. The hardness of INVERSE MIN SAT(S^9) can then be leveraged to establish the hardness of INVERSE MIN SAT_C(S), for the remaining constraint languages S .

Theorem 5.2 *Let S be a constraint language. If S is non-Schaefer, then INVERSE MIN SAT_C(S) is co-NP-hard.*

Then, constants can be “removed” in such a way that allows the hardness of INVERSE MIN SAT(S) to be established, based on the hardness of INVERSE MIN SAT_C(S).

Theorem 5.3 *Let S be a constraint language. If S is neither 0-valid nor Schaefer, then INVERSE MIN SAT(S) is co-NP-hard.*

Collecting together the theorems of this section as well as the discussion at the end of Section 4, we have the following dichotomy theorem.

Theorem 5.4 *Let S be a constraint language.*

If S is Schaefer, then INVERSE MIN SAT_C(S) is in P ; otherwise, INVERSE MIN SAT_C(S) is co-NP-complete.

If S is 0-valid or Schaefer, then INVERSE MIN SAT(S) is in P ; otherwise, INVERSE MIN SAT(S) is co-NP-complete.

6 Conclusions and Future Work

In this paper, we formalized and studied *inverse circumscription*. We established a full dichotomy theorem in the complexity of this problem (Theorem 5.4). A fascinating phenomenon is that the complexity of inverse circumscription reflects *exactly* the complexity of model checking for circumscription (Theorem 3.6). This correspondence in complexity parallels the intimate relationship between the complexity of inverse satisfiability (Theorem 3.4) and that of satisfiability (Theorem 3.2). The resemblance between the results on circumscription and those on propositional logic seems quite

strong, as the hardness of inverse circumscription is established by reduction from model checking for circumscription – just as the hardness of inverse satisfiability is established by reduction from satisfiability [Kavvadias and Sideri, 1998].

It would be of great interest to investigate further the relationship between “inverse” problems (mapping an explicit description to an implicit description) and more classical “forward” problems (mapping an implicit description to an explicit description). A concrete goal for future work is to study inverse problems for other non-monotonic reasoning formalisms.

Acknowledgements. The author wishes to thank Bart Selman for useful discussions and suggestions, and Joe Halpern for his advice on the preparation of the final version of this paper.

A Proof Sketch of Theorem 5.1

Definition A.1 *Let A be a set of assignments over the variable set V . For $k \geq 1$, we say that an assignment $b : V \rightarrow \{0, 1\}$ is k -compatible with A if for every size k subset W of V , there exists an assignment $a \in A$ such that $a|_W = b|_W$.*

Lemma A.2 *Let S be a constraint language with maximum arity k , and let A be a set of assignments over the variable set V . Suppose that ψ_C is the candidate S -formula for A (with $k \geq 1$). If b is r -compatible with A for all $r = 1, \dots, k$, then b satisfies ψ_C . Moreover, the converse holds if $S = S^k$.*

Definition A.3 *Let S be a constraint language. For $r \geq 1$, an r -pattern of an S -formula ϕ over V is a pair (W, T) such that $|W| = r$, $W \subseteq V$, $T : W \rightarrow \{0, 1\}$ is an assignment to the variables of W , and for all clauses $R(x_1, \dots, x_k) \in \phi$, if T is defined on the variables of $R(x_1, \dots, x_k)$ (that is, $\{x_1, \dots, x_k\} \subseteq W$), then T satisfies $R(x_1, \dots, x_k)$.*

In other words, an r -pattern for a formula ϕ is a subset W of the variable set of ϕ of size r along with an assignment to W which does not falsify any clause of ϕ .

Before giving the proof, we introduce the following notation. A vector $\vec{x}$ of length k is an ordered list of variables $(x_1, \dots, x_k)$. If θ is a bit vector $(b_1, \dots, b_k) \in \{0, 1\}^k$ and $\vec{x}$ is a vector of length k , we let $\theta(\vec{x})$ denote the function mapping x_i to b_i for all $i = 1, \dots, k$. If $\mu : V_1 \rightarrow \{0, 1\}$ and $\nu : V_2 \rightarrow \{0, 1\}$ are assignments with disjoint domains (that is, $V_1 \cap V_2 = \emptyset$), by $\mu \cup \nu$ we denote the function with domain $V_1 \cup V_2$ equal to μ on V_1 , and equal to ν on V_2 . When $\vec{z}$ is a vector of variables and f is an assignment defined on the variables of $\vec{z}$, we let $f|_{\vec{z}}$ denote the restriction of f to the variables of $\vec{z}$. For a positive integer k , we let $[k]$ denote the set $\{1, \dots, k\}$.

Proof. By [Kirosis and Kolaitis, 2001a, Theorem 3.8], the version of MIN SAT(S^3) where the variable set P is promised to be the entire variable set is **co-NP-complete**. To prove hardness of INVERSE MIN SAT(S^9), we give a reduction from this version of MIN SAT(S^3). In particular, given a S^3 -formula ϕ and an assignment a satisfying ϕ , we create a set of assignments A over a variable set X such that there is an S^9 formula ψ with $A \subseteq \text{MinModels}(\psi, X)$ if and only if a is

a minimal model of ϕ . (By a minimal model of ϕ , we mean a minimal model with respect to the entire variable set of ϕ .)

We let $V = \{v_1, \dots, v_n\}$ denote the variable set of ϕ , and we assume without loss of generality that $n > 9$. For all $i \in [n]$, let $\vec{x}_i^1, \vec{x}_i^2$, and $\vec{x}_i^3$ be vectors of length 3. For all $i \in [n]$ and $j \in [m]$, let $\vec{y}_i^j$ be a vector of length 2. Let X_i denote the set containing all $9 + 2m$ variables in the vectors $\vec{x}_i^1, \vec{x}_i^2, \vec{x}_i^3$ and $\vec{y}_i^1, \dots, \vec{y}_i^m$. Let $X = \cup_{i \in [n]} X_i$.

Let $\alpha = (0, 0, 1)$, $\beta = (1, 0, 1)$, and $\gamma = (0, 1, 0)$. Let $\sigma = (0, 1)$ and $\tau = (1, 0)$. The essential properties of these vectors that we will use are that for any vector $\vec{z}$ of length 3, $\alpha(\vec{z}) < \beta(\vec{z})$, $\alpha(\vec{z}) \perp \gamma(\vec{z})$, and $\beta(\vec{z}) \perp \gamma(\vec{z})$; and, for any vector $\vec{z}$ of length 2, $\sigma(\vec{z}) \perp \tau(\vec{z})$.

For $i \in [n]$, define $\alpha_i : X_i \rightarrow \{0, 1\}$ to be $\{\alpha(\vec{x}_i^1), \alpha(\vec{x}_i^2), \alpha(\vec{x}_i^3)\} \cup \cup_{k \in [m]} \tau(\vec{y}_i^k)$. Similarly, for $i \in [n]$, define $\beta_i : X_i \rightarrow \{0, 1\}$ to be $\{\beta(\vec{x}_i^1), \beta(\vec{x}_i^2), \beta(\vec{x}_i^3)\} \cup \cup_{k \in [m]} \tau(\vec{y}_i^k)$. For $i \in [n]$ and $j \in [m]$, define $\gamma_i^j : X_i \rightarrow \{0, 1\}$ to be $\{\gamma(\vec{x}_i^1), \gamma(\vec{x}_i^2), \gamma(\vec{x}_i^3)\} \cup \{\sigma(\vec{y}_i^j)\} \cup \cup_{k \in [m], k \neq j} \tau(\vec{y}_i^k)$.

Let $(W_1, T_1), \dots, (W_m, T_m)$ be all of the 9-patterns of ϕ . Our set of assignments A will be of size $m + 1$: it will have one assignment $a^i : X \rightarrow \{0, 1\}$ for each of the 9-patterns (W_i, T_i) , as well as one assignment $\tilde{a} : X \rightarrow \{0, 1\}$ encoding the assignment $a : V \rightarrow \{0, 1\}$. The partial assignments $\{\gamma_i^j\}$, $\{\alpha_i\}$, and $\{\beta_i\}$ will be used as building blocks to define the assignments in A .

Define

$$a_i^j = \begin{cases} \gamma_i^j & \text{if } v_i \notin W_j \\ \alpha_i & \text{if } v_i \in W_j \text{ and } T_j(v_i) = 0 \\ \beta_i & \text{if } v_i \in W_j \text{ and } T_j(v_i) = 1 \end{cases}$$

Define $a^j = \cup_{i \in [n]} a_i^j$.

Also, for any assignment $f : V \rightarrow \{0, 1\}$, define $\tilde{f} : X \rightarrow \{0, 1\}$ to be the assignment $\cup_{i \in [n], f(v_i)=0} \alpha_i \cup \cup_{i \in [n], f(v_i)=1} \beta_i$.

Define A , the output of the reduction, to be the set $\{a^1, \dots, a^m\} \cup \{\tilde{a}\}$.

Let us say that a pair of assignments $(d : X \rightarrow \{0, 1\}, c : X \rightarrow \{0, 1\})$ is a *violating pair* if the following four conditions are met: 1) $d \notin A$, 2) $c \in A$, 3) $d < c$, and 4) d is 9-compatible with A . Intuitively, a violating pair (d, c) is evidence that there is no S^9 -formula ψ such that $A \subseteq \text{MinModels}(\psi, X)$.

More formally, observe that a violating pair exists if and only if there exists a $d \notin A$ and $c \in A$ such that $d < c$ and d satisfies the candidate S^9 -formula ψ_C for A , by Lemma A.2. This occurs if and only if it is not the case that $A \subseteq \text{MinModels}(\psi_C, X)$, since (as is easily verified) the assignments in A are pairwise incomparable. By Lemma 4.4, it is not the case that $A \subseteq \text{MinModels}(\psi_C, X)$ if and only if for all S^9 -formulas ψ , it is not the case that $A \subseteq \text{MinModels}(\psi, X)$. Hence, a violating pair exists if and only if there is no S^9 -formula ψ such that $A \subseteq \text{MinModels}(\psi, X)$.

The remainder of the proof, omitted due to space constraints, establishes that a violating pair exists if and only if a

is not a minimal model of ϕ . This implies the correctness of our reduction, as then we have that there exists no S^9 -formula ψ such that $A \subseteq \text{MinModels}(\psi, X)$ if and only if a is not a minimal model of ϕ . $\square$

References

- [Cadoli, 1992] Marco Cadoli. The Complexity of Model Checking for Circumscriptive Formulae. *Information Processing Letters*, 44(3):113-118, 1992.
- [Cadoli et al., 1997] Marco Cadoli, Francesco M. Donini, Marco Schaerf, and Riccardo Silvestri. On Compact Representations of Propositional Circumscription. *Theoretical Computer Science*, 182:183-202, 1997.
- [Creignou et al., 2001] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. SIAM Monographs on Discrete Mathematics and Applications, SIAM, 2001.
- [Dechter and Pearl, 1992] Rina Dechter and Judea Pearl. Structure identification in relational data. *Artificial Intelligence*, 58:237-270, 1992.
- [Kautz et al., 1995] Henry Kautz, Michael Kearns, and Bart Selman. Horn Approximations of Empirical Data. *Artificial Intelligence*, 74(1):129-145, 1995.
- [Kavvadias and Sideri, 1998] Dimitris Kavvadias and Martha Sideri. The Inverse Satisfiability Problem. *SIAM Journal on Computing*, 28(1):152-163, 1998.
- [Kavvadias et al., 2000] Dimitris J. Kavvadias, Martha Sideri and Elias C. Stavropoulos. Generating all maximal models of a Boolean expression. *Information Processing Letters*, 74:157-162, 2000.
- [Khaldon and Roth, 1996] Roni Khaldon and Dan Roth. Reasoning with Models. *Artificial Intelligence*, 87:187-243, 1996.
- [Klousis and Kolaitis, 2001a] Lefteris M. Klousis and Phokion G. Kolaitis. The Complexity of Minimal Satisfiability Problems. In *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2010 of *Lecture Notes in Computer Science*, pages 407-418. Springer, 2001. Full version at: Electronic Colloquium on Computational Complexity – (www.eccc.uni-trier.de/eccc), Report No. 82, 2000.
- [Klousis and Kolaitis, 2001b] Lefteris M. Klousis and Phokion G. Kolaitis. A Dichotomy in the Complexity of Propositional Circumscription. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science – LICS 2001*, pages 71-80, 2001.
- [McCarthy, 1980] John McCarthy. Circumscription—A Form of Non-Monotonic Reasoning. *Artificial Intelligence*, 13:27-39, 1980.
- [Schaefer, 1978] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symposium on Theory of Computing*, pages 216-226, 1978.

A Theory of Average-Case Compilability in Knowledge Representation

Hubie Chen

Department of Computer Science

Cornell University

Ithaca, NY 14853, USA

hubes@cs.cornell.edu

Abstract

Compilability is a fundamental property of knowledge representation formalisms which captures how succinctly information can be expressed. Although many results concerning compilability have been obtained, they are all “worst-case” results. We develop a theory of average-case compilability which allows for the formal comparison and classification of knowledge representation formalisms “on average.”

1 Introduction

By now, a multitude of knowledge representation formalisms have been proposed and studied in the literature, for example, propositional logic, default logic, and circumscription – to name a few. The comparison of these formalisms has been a major research theme over the past decade. In particular, many results have been obtained on the computational complexity of inference and model checking – core reasoning tasks associated with each formalism. The fundamental property of *compilability*, which measures how efficiently or succinctly a formalism represents knowledge, has also been studied.¹ There is now a rich body of results concerning compilability; however, all of these results address the “worst-case”, and are thus susceptible to the complaint that they do not address “average” or “typical” compilability. The main contribution of this paper is a theoretical framework providing a language and tools for comparing and classifying the compilability of formalisms *on average*. Our framework is built on notions and insights from the theory of compilability classes due to Cadoli et al. [2000a; 2000b] and the theory of average-case time complexity due to Levin [1986].

1.1 Background

Compilability. Informally, a formalism A is compilable to a formalism B if for every knowledge base x in formalism A , there is a knowledge base in formalism B representing the same information as x with size polynomial in the length

of x .² Intuitively, this means that formalism B is at least as space efficient as formalism A : whatever can be expressed in formalism A can be expressed, with about the same level of succinctness, in formalism B .

A number of papers compared the compilability of different formalisms [Kautz *et al.*, 1995; Gogic *et al.*, 1995; Khardon and Roth, 1996; Cadoli *et al.*, 1996; 1997; 1999; 2000a; 2000b]. In many cases, these papers rigorously prove that one formalism B is strictly more succinct than another formalism A – that is, A is compilable to B , but B is not compilable to A . This means that there is *no* way to translate knowledge bases in B to knowledge bases in A , unless the translation is allowed to increase the size of knowledge bases by a super-polynomial amount. In other words, any translation of knowledge bases in B to knowledge bases in A is *inherently exponential* in size.³ Observe that this statement has the same flavor as the famous “P does not equal NP” conjecture from classical complexity theory. This conjecture holds that no algorithm can solve an NP-complete problem, unless the algorithm is allowed to take super-polynomial time on some inputs – or, put differently, any algorithm solving a NP-complete problem is *inherently exponential* in time.

Notice that the above definition of “formalism A is compilable to formalism B ” does not take into account the difficulty or complexity of computing the translation between knowledge bases in A and knowledge bases in B . This non-uniformity is a key feature of the definition: the *existence* of a succinct translation is sufficient; an efficiently computable translation is not necessary. It turns out that many proofs of non-compilability – that is, proofs of statements of the form “formalism B is not compilable to formalism A ” – rely on results from *non-uniform* complexity theory. Such proofs often are not unconditional, but are contingent upon the widely believed complexity-theoretic assumption that the polynomial hierarchy does not collapse.⁴

²In the technical portion of this paper, this definition will be captured formally by the $\leq_{\text{nu-comp}}$ reducibility.

³Here, by “exponential” we mean exceeding every polynomial infinitely often.

⁴The polynomial hierarchy is a collection of complexity classes which includes P, NP, co-NP, and other classes which are in essence generalizations of these three classes. For the intents and purposes of this paper, this assumption can be thought of as being similar to the (perhaps better known) assumption that P does not equal NP. (In

¹In previous work, compilability has also been called *space efficiency* and *succinctness*.

Compilability classes. The initial papers that demonstrated non-compilability results [Kautz *et al.*, 1995; Gogic *et al.*, 1995; Khardon and Roth, 1996; Cadoli *et al.*, 1996; 1997; 1999] were based on *ad hoc* proofs, each of which isolated two particular formalisms and then demonstrated non-compilability of one to the other. In [Cadoli *et al.*, 2000a; 2000b], new complexity classes measuring compilability, called *compilability classes*, were introduced; for every classical complexity class C , it is possible to define a compilability class analog of C . The theory of compilability classes made it possible to systematize proofs of non-compilability much in the way classical complexity theory makes it possible to systematize proofs of intractability. A proof that a language is NP-complete is demonstration that there is no polynomial-time algorithm for the language, and that the language has the same time complexity (up to a polynomial) as all other NP-complete languages. Likewise, a proof that a formalism is complete for the compilability class analog of NP is demonstration that it is not compilable to any formalism in the compilability class analog of P, and that the language is compilable to and from all other formalisms complete for the compilability class analog of NP.⁵ (More generally, when C is a class from the polynomial hierarchy, a proof that a formalism is complete for the compilability class analog of C is demonstration that it is not compilable to any formalism in the compilability class analog of C' , if C' is below C in the polynomial hierarchy.)

In addition to providing a methodology for comparing formalisms with respect to compilability, the compilability classes capture formally the notion of off-line *preprocessing*. Preprocessing a knowledge base off-line can be of great utility if the resulting, processed knowledge base is in a form that allows for quick, on-line response to queries (and if many queries are expected). Membership of a formalism A in the compilability class analog of P will mean that any knowledge base x of A can be preprocessed into a form that does not unreasonably increase the size of x , but permits queries to x to be processed efficiently (that is, in polynomial time).

The compilability classes are part of a formal framework for discussing compilability, which includes robust notions of reduction and completeness. Not only is this framework extremely appealing from a theoretical point of view, but the compilability classes are rife with *natural complete problems* – the *sine qua non* of complexity classes purporting to be useful in performing problem classification. Indeed, there do not appear to be any knowledge representation formalisms which defy classification as complete for a compilability class. On the downside, many of the existing classification results are quite negative, showing that a formalism is complete for (the compilability class analog of) NP, coNP, or a higher level of the polynomial hierarchy.

fact, if the polynomial hierarchy does not collapse, then P does not equal NP.)

⁵Our discussion presumes that the polynomial hierarchy does not collapse.

1.2 Motivations and Approach

Worst-case versus average-case. Although crisp theoretical results can be obtained concerning compilability and non-compilability, these are *worst-case* notions: non-compilability of formalism B to formalism A implies that there is an infinite family of knowledge bases in B that cannot be succinctly translated into knowledge bases in A . Non-compilability does *not* say anything about the density or frequency of instances from the family of untranslatable knowledge bases. This observation calls into question the real-world utility of studying compilability (as defined above); if formalism B is compilable to formalism A for all but a pathological family of knowledge bases arising infrequently in practice, then formalism B is, for pragmatic purposes, compilable to formalism A . Therefore, a theory which permits formal results of compilability *on average* is necessary. This paper lays the foundations for a theory of average-case compilability.

Notice that our objection to the worst-case nature of non-compilability is not truly novel. It has long been observed that NP-completeness of a language L does not imply hardness of L on typical or real-world instances. Our objection is really this old observation, masquerading in the new context of compilability.

One theory that was developed in response to this old observation is the theory of average-case time complexity (ACTC), initiated by Levin [1986]. Our theory of average-case compilability will be built on a key notion of ACTC – that of “polynomial on average.” Moreover, there are useful analogies between our theory and the theory of ACTC. Consequently, we provide a brief overview of ACTC.

Average-case time complexity. By definition, a language L (consisting of strings) is in P if there is an algorithm deciding membership for L in polynomial time. The idea behind ACTC is to relax the requirement in this definition that a suitable algorithm is one that *always* runs in polynomial time; this is done by placing a probability distribution on all strings and allowing an algorithm to take super-polynomial times on unlikely strings.

A language paired with a probability distribution over all strings is called a *distributional language*. While languages are the objects classified by classical complexity theory, distributional languages are the objects classified by ACTC. A distributional language (L, μ) is in the average-case version of P, *average-P*, if there is an algorithm deciding membership for L in time *polynomial on μ -average* – a concept to be discussed formally later in this paper.

There is a vast literature on ACTC; for more information, we recommend the overviews/surveys [Johnson, 1984; Gurevich, 1989; 1991a; 1991b; Impagliazzo, 1995; Wang, 1997; Goldreich, 1997] as starting points.

Average-case compilability. In laying down a theory for average-case compilability, we want to “soften” the definition of “formalism A is compilable to formalism B ” by relaxing the requirement that there needs to be a translation of knowledge bases of strictly polynomial size. This is done roughly in

analogy to the described development of ACTC. We first define a distributional formalism to be a formalism paired with a probability distribution over all knowledge bases. Then, a distributional formalism (A, μ) is (informally) *compilable on average* to a formalism B if there is a translation of knowledge bases of size polynomial on μ -average.⁶

Using the notion of “compilable on average,” we define average-case analogs of compilability classes. These average-case analogs contain distributional formalisms, in contrast to the compilability classes themselves, which contain pure formalisms. Intuitively, membership of a distributional formalism (A, μ) in the average-case compilability class analog of P will mean the following: any knowledge base x of A can be preprocessed in a way that tends not to increase the size of x by more than a polynomial (with respect to μ), and that allows for rapid processing of queries to x .

2 Preliminaries

In this section, we present notation and assumptions that will be used throughout the paper.

We assume Σ to be a fixed finite alphabet which is used to form strings. We will at times assume that pairs of strings (that is, elements of $\Sigma^* \times \Sigma^*$) are represented as strings (that is, elements of Σ^*); when this assumption is made, we assume that the representation is via a pairing function $\langle \cdot, \cdot \rangle$ such that the length of $\langle x, y \rangle$ is linear in $|x| + |y|$. For a string $x \in \Sigma^*$, we let $|x|_u$ denote $1^{|x|}$, that is, the length of x written in unary notation.

We assume that the reader has familiarity with basic notions of computational complexity theory – in particular, the classes of the polynomial hierarchy (P , NP , $\text{co-}NP$, Σ_2^P , Π_2^P , etc.) and the polynomial many-one reduction $\leq_m^P$ [Balcázar *et al.*, 1995]. A *language* is a subset of Σ^* , that is, a set of strings. A *complexity class* is a set of languages. When C is a complexity class and $\leq$ is a reduction, we say that C is *compatible* with $\leq$ if for all languages A and B , $A \leq B$ and $B \in C$ imply that $A \in C$. We will assume throughout that every complexity class C is compatible with the polynomial many-one reduction $\leq_m^P$. We say that a language B is *complete* for a complexity class C under $\leq$ reductions if $B \in C$ and for all $A \in C$, $A \leq B$.

A function $f : \Sigma^* \rightarrow \Sigma^*$ is *polynomial-size* if there exists a polynomial p such that for all $x \in \Sigma^*$, $|f(x)| \leq p(|x|)$. A function $f : \Sigma^* \rightarrow \Sigma^*$ is *polynomial-time computable* if there exist a polynomial p and a Turing machine M such that for all $x \in \Sigma^*$, the Turing machine M , on input x , produces $f(x)$ in time less than $p(|x|)$.⁷

Definition 2.1 A knowledge representation formalism (*KRF*) is a subset of $\Sigma^* \times \Sigma^*$. When F is a KRF and $x \in \Sigma^*$, we let F_x denote the set $\{y \in \Sigma^* : (x, y) \in F\}$.

Intuitively, each $x \in \Sigma^*$ can be thought of as a knowledge base (KB), representing the information F_x . The following

⁶In the technical portion of this paper, this definition will be captured formally by the $\leq_{\text{dist-nu-comp}}$ reducibility.

⁷Notice that every polynomial-time computable function is polynomial-size, but not every polynomial-size function is polynomial-time computable.

are examples of KRFs which capture model checking and inference for 3-SAT formulas.

Propositional-Logic-MC $\stackrel{\text{def}}{=}$

$\{(x, y) : x \text{ is a 3-SAT formula and } y \text{ is a model of } x\}$

Clause-Inference $\stackrel{\text{def}}{=}$

$\{(x, y) : x \text{ is a 3-SAT formula, } y \text{ is a 3-clause, and } x \models y\}$

Notice the generality of the definition of a KRF; the knowledge represented (the various F_x) may be models, formulas, or some altogether different combinatorial structures.

3 Compilability classes and reductions

This section reviews the theory of compilability classes. There are two different types of compilability classes, uniform and non-uniform, but our focus is on the latter, for reasons discussed below. *We emphasize that none of the definitions, theorems, or insights in this section are our own, but rather, are due to [Cadoli *et al.*, 2000a; 2000b], on which our presentation is based.*

The formal definition of non-uniform compilability may by itself look non-intuitive, so we first attempt to describe some of the ideas behind this definition, before giving the actual definition. Intuitively, we want to say that a KRF F is compilable if membership queries $y \in F_x$ can be decided efficiently after the KB x is preprocessed into a new KB $f(x)$. We want to constrain the size of the new KB $f(x)$; otherwise, it may be prohibitively large to store. We arrive at a candidate definition of compilability: say that a KRF F is in **comp- C** if for some polynomial-size function f and a second KRF $F' \in C$, the property

$$\text{for all pairs } (x, y) \in \Sigma^* \times \Sigma^*, (x, y) \in F \text{ if and only if } (f(x), y) \in F'$$

holds. Notice that the class C constrains the difficulty of deciding a post-processing query $y \in F'_{f(x)}$; of course, such queries can be decided efficiently when $C = P$. However, to permit fine classification of KRFs, we leave C as a parameter to the definition of **comp- C** .

The above definition misses one important detail – non-uniformity. If F is a KRF such that F_x is always a finite set, F can never be complete for **comp- C** when C is a class of the polynomial hierarchy above P . At the same time, there are KRFs with this “finiteness” property that are provably *not* in **comp- P** . As an example, **Clause-Inference** is in **comp-co- NP** , but is neither **comp-co- NP** -complete nor in **comp- P** .⁸ Hence, the classes **comp- C** fail to capture the compilability of some very natural KRFs. (For more information on these matters, as well as proofs of the claims, we refer the reader to [Cadoli *et al.*, 2000a].) To resolve this issue, we allow the *length* of a query to be known to the compilation mapping f .

Definition 3.1 (nu-comp- C) Let C be a complexity class. A KRF F belongs to **nu-comp- C** if there exists a binary polynomial-size function $f : \Sigma^* \times 1^* \rightarrow \Sigma^*$ and a KRF $F' \in C$ such that the following property holds:

⁸This discussion presumes that the polynomial hierarchy does not collapse.

for all pairs $(x, y) \in \Sigma^* \times \Sigma^*$, $(x, y) \in F$ if and only if $(f(x, |y|_u), y) \in F'$.

The fact that the compilation mapping may use the length of y may seem a bit strange; after all, we wish to capture, after preprocessing on a knowledge base x , the difficulty of answering queries of the form $y \in F_x$. However, it is reasonable to assume that we will never be interested in such a query when the length of y greatly exceeds that of x – since such a query requires a large amount of time to even write down. Under this assumption, Definition 3.1 is equivalent to the “uniform” definition given above. (Formally, if there exists a polynomial p such that $(x, y) \in F$ implies that $|y| \leq p(|x|)$, then F is in **nu-comp-C** if and only if F is in **comp-C**.)

The following notion of reduction, associated with the **nu-comp-C** classes, allows one to compare the compilability of different KRFs.

Definition 3.2 (*nu-comp reducibility*) A KRF F is **nu-comp reducible** to a KRF F' (denoted by $F \leq_{\text{nu-comp}} F'$) if there exist binary polynomial-size functions $f_1, f_2 : \Sigma^* \times 1^* \rightarrow \Sigma^*$ and a binary polynomial-time computable function $g : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ such that the following property holds:

for all pairs $(x, y) \in \Sigma^* \times \Sigma^*$, $(x, y) \in F$ if and only if $(f_1(x, |y|_u), g(f_2(x, |y|_u), y)) \in F'$.

Theorem 3.3 *The nu-comp reduction is transitive and is compatible with the class nu-comp-C (for every complexity class C).*

We note that **Clause-Inference** is **nu-comp-co-NP**-complete, under **nu-comp** reductions.

Finally, we observe that the **nu-comp-C** analog of the polynomial hierarchy does not collapse.

Theorem 3.4 *If C_1 and C_2 are classes of the polynomial hierarchy such that C_2 is higher than C_1 , then **nu-comp- C_1** is properly contained in **nu-comp- C_2** (under the assumption that the polynomial hierarchy does not collapse).*

4 Average-case compilability

In this section, we present our new theory of average-case compilability.

Definition 4.1 A probability distribution μ is a *real-valued function from $\Sigma^* \times 1^*$ to $[0, 1]$ such that $\sum_{(x, l) \in \Sigma^* \times 1^*} \mu(x, l) = 1$.*

The **nu-comp-C** classes are used to classify and compare KRFs; our new classes will be used to classify and compare what we call **distributional KRFs**.

Definition 4.2 A **distributional KRF (DKRF)** is a pair (F, μ) consisting of a KRF F and a probability distribution μ .

We now define the notion of “polynomial on average.” This definition is exactly that used in the theory of average-case time complexity (up to appropriate changes of the domain and range of the functions whose size we wish to measure).

Definition 4.3 Let μ be a probability distribution. A function $f : \Sigma^* \times 1^* \rightarrow \Sigma^*$ is of size polynomial on μ -average if there exists an $\epsilon > 0$ such that $\sum_{(x, l) \in \Sigma^* \times 1^*} |f(x, l)|^\epsilon |\langle x, l \rangle|^{-1} \mu(x, l) < \infty$.

We are now ready to define our new “average-compilability” classes; formally, each class is a set of DKRFs. The definition can be viewed as a relaxation of Definition 3.1. The difference is that the translation mapping $f : \Sigma^* \times 1^* \rightarrow \Sigma^*$ need no longer be of strictly polynomial size, but is now only required to be of size polynomial on average.

Definition 4.4 (*avg-nu-comp-C*) Let C be a complexity class. A DKRF (F, μ) belongs to **avg-nu-comp-C** if there exists a binary function $f : \Sigma^* \times 1^* \rightarrow \Sigma^*$ of size polynomial on μ -average and a KRF F' in C such that the property of Definition 3.1 holds.

There is an alternative way to define **avg-nu-comp-C**, in terms of the following type of reduction, which relates DKRFs to KRFs.

Definition 4.5 (*dist-nu-comp reducibility*) A DKRF (F, μ) is **dist-nu-comp reducible** to a KRF F' (denoted by $F \leq_{\text{dist-nu-comp}} F'$) if there exist binary polynomial-size functions $f_1, f_2 : \Sigma^* \times 1^* \rightarrow \Sigma^*$ and a binary polynomial-time computable function $g : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ such that $\langle f_1, f_2 \rangle$ is of size polynomial on μ -average⁹ and the property of Definition 3.2 holds.

Theorem 4.6 *Suppose that (F, μ) is a DKRF and that C is a complexity class. The DKRF (F, μ) is in **avg-nu-comp-C** if and only if $(F, \mu) \leq_{\text{dist-nu-comp}} F'$ for some F' in **nu-comp-C**.*

This theorem has an important corollary: if **nu-comp-C** has a complete KRF F' , then the DKRFs contained in **avg-nu-comp-C** are exactly those which reduce to F' . In order to establish the corollary, the following lemma is needed.

Lemma 4.7 *Suppose that (F, μ) is a DKRF and that F' and F'' are KRFs. If $(F, \mu) \leq_{\text{dist-nu-comp}} F'$ and $F' \leq_{\text{nu-comp}} F''$, then $(F, \mu) \leq_{\text{dist-nu-comp}} F''$.*

Roughly, Lemma 4.7 can be viewed as a proof of transitivity: if (F, μ) reduces to F' and F' reduces to F'' , then (F, μ) reduces to F'' (under the right notions of reduction).

Corollary 4.8 *Suppose that (F, μ) is a DKRF, C is a complexity class and the KRF F' is **nu-comp-C**-complete under **nu-comp** reductions. Then, the DKRF (F, μ) is in **avg-nu-comp-C** if and only if $(F, \mu) \leq_{\text{dist-nu-comp}} F'$.*

It is worth noting here that the class **nu-comp-C** has a complete KRF under **nu-comp** reductions whenever the underlying complexity class C has a complete language under $\leq_m^p$ reductions [Cadoli et al., 2000a].

We now give a notion of reduction for the comparison of DKRFs. When μ and ν are probability distributions, we say that ν *dominates* μ if there exists a polynomial p such that for all $(x, l) \in \Sigma^* \times 1^*$, $\mu(x, l) \leq p(|\langle x, l \rangle|) \nu(x, l)$.

Definition 4.9 (*avg-nu-comp reducibility*) A DKRF (F, μ) is **avg-nu-comp reducible** to a DKRF (F', μ') if there exist a **nu-comp** reduction (f_1, f_2, g) from F to F' and a probability distribution μ_1 dominating μ such that $\mu'(y, m) \geq \sum \mu_1(x, l)$, where the sum is over all (x, l) such that $[f_1(x, l) = y$ and there exists $z \in \Sigma^*$ with $|z| = l$ and $|g(f_2(x, l), z)| = m]$.

⁹By $\langle f_1, f_2 \rangle$, we denote the function which maps a pair $(x, y) \in \Sigma^* \times 1^*$ to $\langle f_1(x, y), f_2(x, y) \rangle$.

Theorem 4.10 *The avg-nu-comp reduction is transitive and is compatible with the class avg-nu-comp-C (for every complexity class C).*

5 Discussion

We now discuss some of the considerations behind the definitions in the previous section. When $\mu : \Sigma^* \times 1^* \rightarrow [0, 1]$ is a probability distribution and S is a subset of $\Sigma^* \times 1^*$, let $(\mu|S)$ denote the conditional distribution of μ on S , that is, the function defined on $(x, l) \in S$, with value equal to $\mu(x, l) / [\sum_{s \in S} \mu(s)]$.

• *Why is Levin’s notion of polynomial-on-average used instead of the naive formulation of expected polynomial size?*

According to the naive formulation, a function $f : \Sigma^* \times 1^* \rightarrow \Sigma^*$ has expected polynomial size (with respect to μ) if there exists $k \geq 1$ such that for all n ,

$$\sum_{|x, l|=n} |f(x, l)| (\mu(x, l) |\{(x', l') : |x', l'| = n\}|) = O(n^k).$$

It is well known that this notion of expected polynomial is *not* closed under polynomials (see for example [Goldreich, 1997]). For instance, there exist functions $f, f' : \Sigma^* \times 1^* \rightarrow \Sigma^*$ and a distribution μ such that $|f'(x, l)| = |f(x, l)|^2$ and f has expected polynomial size, but f' fails to have expected polynomial size.

In our average-case compilability theory, this “lack of closure under polynomials” problem manifests itself in the following way. Suppose that, using the alternative definition of avg-nu-comp-C given in Theorem 4.6, the KRF F' is witness to the membership of DKRF (F, μ) in avg-nu-comp-C, that is, $(F, \mu) \leq_{\text{dist-nu-comp}} F'$. Surely, if F'' is at least as space efficient as F' (that is, $F' \leq_{\text{nu-comp}} F''$), then one expects that F'' would also witness the membership of (F, μ) in avg-nu-comp-C; this is the content of Lemma 4.7, on which Corollary 4.8 relies. However, if expected polynomial size is used instead of polynomial on average in Definition 4.3, Lemma 4.7 breaks down – then, there would exist a DKRF (F, μ) and KRFs F' and F'' such that $(F, \mu) \leq_{\text{dist-nu-comp}} F'$, $F' \leq_{\text{nu-comp}} F''$, and (F, μ) does *not* $\leq_{\text{dist-nu-comp}} F''$.

• *Why does a DKRF have one probability distribution – over all elements of $\Sigma^* \times 1^*$ – as opposed to an ensemble of distributions, each of which is over some finite subset of $\Sigma^* \times 1^*$?*

Defining a DKRF to be a KRF paired with an ensemble of distributions (each of which is, say, defined on a different string length) may seem more natural than the given definition, where there is only one distribution, on all strings.

However, the literature on ACTC contains many alternative formulations of “polynomial on average” (equivalent to that given in Definition 4.3) and sufficient conditions for a function to be polynomial on average. (For example, there is a different formulation of Definition 4.3 in terms of ensembles of distributions, each having finite support, in [Impagliazzo, 1995].) These give rise to alternative formulations of avg-nu-comp-C, and so the particular characterizations we give for avg-nu-comp-C (in Definition 4.4 and Theorem 4.6) were, in some sense, chosen over provably equal characterizations on purely aesthetic grounds.

• *Why is a probability distribution over $\Sigma^* \times 1^*$ as opposed to over Σ^* ? After all, the knowledge bases are represented by elements of Σ^* , and an element of $\Sigma^* \times 1^*$ is just a “slice” of a knowledge base.*

Let F be a KRF conforming to the “bounded-query-length” assumption discussed in Section 3, that is, suppose that there exists a polynomial p such that $(x, y) \in F$ implies $|y| \leq p(|x|)$. Then, a distribution over Σ^* naturally induces a distribution over $\Sigma^* \times 1^*$ for the padded version of F defined as $F' = \{(x, z) : z = y\Box^k, |z| = p(|x|), (x, y) \in F\}$ (where $\Box$ is an extra padding symbol), which has the property that all strings in F'_x have the same length, for any x . Note that all of the example KRFs in this paper already have this property.

6 Example: model checking for circumscription

In this section, we illustrate the use of our new theory by showing that model checking for circumscription on 3-SAT formulas (formalized below and denoted by Circumscription-MC(d)) is in avg-nu-comp-P, under a natural probability distribution where formulas are generated by including each clause independently with identical probability. Since Circumscription-MC(d) is nu-comp-co-NP-hard,¹⁰ this result gives a natural DKRF which is contained in avg-nu-comp-P, but which has a KRF which is provably *not* in nu-comp-P (Theorem 3.4).

Suppose that ϕ is a 3-SAT formula over $V_n \stackrel{\text{def}}{=} \{v_1, \dots, v_n\}$, and S is a subset of V_n . We say that a model $a : V_n \rightarrow \{0, 1\}$ of ϕ is a *S-minimal model* (of ϕ) if for all other models $b : V_n \rightarrow \{0, 1\}$ of ϕ , $[b(s) \leq a(s)]$ for all $s \in S$ implies that $[b(s) = a(s)]$ for all $s \in S$.¹¹ In addition, a model $a : S \rightarrow \{0, 1\}$ of ϕ is said to be a *minimal model* if it is the restriction of a *S-minimal model* $a : V_n \rightarrow \{0, 1\}$ (of ϕ).

We define Circumscription-MC(d) to be the set

$$\{(\phi, a) : \phi \text{ is a 3-SAT formula on variable set } V_n \text{ and } a : \{v_1, \dots, v_{\lfloor dn \rfloor}\} \rightarrow \{0, 1\} \text{ is a minimal model of } \phi\}.$$

Let W_n denote the set $\{(\phi, a) : \phi \text{ is a 3-SAT formula on } V_n \text{ and } a : \{v_1, \dots, v_{\lfloor dn \rfloor}\} \rightarrow \{0, 1\} \text{ is an assignment}\}$, which is the set of all syntactically well-formed pairs that may or may not be in Circumscription-MC(d).

Theorem 6.1 *Let $\nu_{n,c}$ denote the distribution on 3-SAT formulas over V_n where a formula is generated by including each of the $\binom{n}{3}$ 3-clauses independently with probability c/n^2 (for all real numbers $c > 0$ and natural numbers $n \geq 1$). Let $\mu_{c,d}$ (with $c > 0$ and $d \in (0, 1)$) be any distribution such that*

- $(\mu_{c,d}|W_n) = \nu_{n,c}$ for all n , and
- $\mu_{c,d}(\phi, y) = 0$ if ϕ is a 3-SAT formula on V_n and $|y| \neq \lfloor dn \rfloor$.

¹⁰This can be shown using a result in [Gogic et al., 1995].

¹¹Here, $\leq$ denotes the usual total ordering on $\{0, 1\}$ where $0 \leq 1$.

If the inequality $2^{1-d}e^{-c/6}(2 - e^{-c/2})^d < 1$ holds, then (*Circumscription-MC*(d), $\mu_{c,d}$) is in *avg-nu-comp-P*.

In particular, (*Circumscription-MC*(0.9), $\mu_{3.73,0.9}$) is in *avg-nu-comp-P*.

This theorem is established using techniques from probabilistic combinatorics. It is worth noting that at $c = 3.73$, the expected number of satisfying assignments that a random 3-SAT formula has is exponential.

7 Conclusions and future work

In [Papadimitriou, 1996], Papadimitriou writes:

The ultimate and most conclusive criterion for comparing knowledge representation formalisms is to compare their expressive power not on arbitrary sets of models, but on the “interesting” sets of models, the ones that come up in the “real world.” We still hope that a meaningful and convincing formulation of this important problem may be possible.

In this paper, we presented a robust and flexible theory of average-case compilability in which the intuitive notion of “interesting sets of models” can be formalized as a DKRF (Definition 4.2), DKRFs can be classified (Definition 4.4), and DKRFs can be compared (Definition 4.9). We illustrated the use of this theory by showing that, under a natural probability distribution, model checking for circumscription is compilable to the analog of the complexity class **P**, in our theory.

There are many open questions for future investigation; to conclude the paper, we list a few.

- Can additional DKRFs be classified as being inside *avg-nu-comp-P*? We conjecture in particular that *Circumscription-MC*(d) – along with the “standard” 3-SAT probabilistic model with $p = c/n^2$ for a sufficiently low c – is in *avg-nu-comp-P*.
- Are there DKRFs complete for *avg-nu-comp-C* (where C is from the polynomial hierarchy) under the *avg-nu-comp* reduction, or some other notion of reduction compatible with the *avg-nu-comp-C* classes?
- Can any questions concerning the new *avg-nu-comp-C* classes defined here be related to better known complexity-theoretic hypotheses?

References

- [Balcázar *et al.*, 1995] J. L. Balcázar and J. Díaz and J. Gabarró. *Structural Complexity I*. Springer-Verlag, Berlin, 1995.
- [Cadoli *et al.*, 2000a] Marco Cadoli, Francesco M. Donini, Paolo Liberatore, and Marco Schaerf. Preprocessing of intractable problems. *Information and Computation*, 176(2):89–120, 2000.
- [Cadoli *et al.*, 1999] Marco Cadoli, Francesco M. Donini, Paolo Liberatore, and Marco Schaerf. The size of a revised knowledge base. *Artificial Intelligence*, 115(1), 25–64.
- [Cadoli *et al.*, 2000b] Marco Cadoli, Francesco M. Donini, Paolo Liberatore, and Marco Schaerf. Space Efficiency of Propositional Knowledge Representation Formalisms. *Journal of Artificial Intelligence Research*, 13:1–31, 2000.
- [Cadoli *et al.*, 1997] Marco Cadoli, Francesco M. Donini, Marco Schaerf, and Riccardo Silvestri. On Compact Representations of Propositional Circumscription. In *Theoretical Computer Science*, 182:183–202.
- [Cadoli *et al.*, 1996] Marco Cadoli, Francesco M. Donini, and Marco Schaerf. Is Intractability of Non-Monotonic Reasoning a Real Drawback? *Artificial Intelligence*, 88:215–251.
- [Gogic *et al.*, 1995] Goran Gogic, Henry Kautz, Christos Papadimitriou, and Bart Selman. The Comparative Linguistics of Knowledge Representation. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 862–869, Montreal, Canada, 1995.
- [Goldreich, 1997] Oded Goldreich. Notes on Levin’s Theory of Average-Case Complexity. *Electronic Colloquium on Computational Complexity (ECCC)* 4(58): 1997.
- [Gurevich, 1991a] Yuri Gurevich. Average case completeness. *Journal of Computer and System Sciences*, 42:346–398, 1991.
- [Gurevich, 1991b] Yuri Gurevich. Average case complexity. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, volume 510 of Lecture Notes in Computer Science, Springer, pages 615–628, 1991.
- [Gurevich, 1989] Yuri Gurevich. The challenger-solver game: variations on the theme of $P = ?$ NP. *EATCS Bulletin*, pages 112–121, 1989.
- [Impagliazzo, 1995] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th Conference on Structure in Complexity Theory*, IEEE Computer Society Press, pages 134–147, 1995.
- [Johnson, 1984] David Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, 5:284–299, 1984.
- [Kautz *et al.*, 1995] Henry Kautz, Michael Kearns, and Bart Selman. Horn approximations of empirical data. *Artificial Intelligence*, 75:129–145.
- [Kharden and Roth, 1996] Roni Kharden and Dan Roth. Reasoning with Models. *Artificial Intelligence* 87:187–213, November 1996.
- [Levin, 1986] Leonid Levin. Average case complete problems. *SIAM Journal on Computing*, 15:285–286, 1986.
- [Papadimitriou, 1996] Christos Papadimitriou. The Complexity of Knowledge Representation. In *Proceedings of the Eleventh Annual IEEE Conference on Computational Complexity*, pages 244–248, Philadelphia, Pennsylvania, May 1996.
- [Wang, 1997] Jie Wang. Average-case computational complexity theory. (L. Hemaspaandra and A. Selman, eds.), *Complexity Theory Retrospective II*, Springer-Verlag, pages 295–328, 1997.

LADDER: A Language to Describe Drawing, Display, and Editing in Sketch Recognition

Tracy Hammond and Randall Davis
Massachusetts Institute of Technology
200 Technology Square, NE43
Cambridge, MA 02139
{hammond, davis}@ai.mit.edu

Abstract

We have created LADDER, the first language to describe how sketched diagrams in a domain are drawn, displayed, and edited. The difficulty in creating such a language is choosing a set of predefined entities that is broad enough to support a wide range of domains, while remaining narrow enough to be comprehensible. The language consists of predefined shapes, constraints, editing behaviors, and display methods, as well as a syntax for specifying a domain description sketch grammar and extending the language, ensuring that shapes and shape groups from many domains can be described. The language allows shapes to be built hierarchically (e.g., an arrow is built out of three lines), and includes the concept of “abstract shapes”, analogous to abstract classes in an object oriented language. Shape groups describe how multiple domain shapes interact and can provide the sketch recognition system with information to be used in top-down recognition. Shape groups can also be used to describe “chain-reaction” editing commands that effect multiple shapes at once. To test that recognition is feasible using this language, we have built a simple domain-independent sketch recognition system that parses the domain descriptions and generates the code necessary to recognize the shapes.

1 Introduction

To date, sketch recognition systems have been domain-specific, with the recognition details of the domain hard-coded into the system. Developing such a sketch interface is a substantial effort. We propose instead that recognition be performed by a single domain-independent recognition system that uses a domain specific sketch grammar (an approach used with some success in speech recognition [Zue *et al.*, 1990; Hunt and McGlashan, 2002]). Programmers could then create new sketch interfaces simply by writing a sketch grammar describing the domain-specific information.

We have created LADDER, a sketch description language that can be used to describe how shapes and shape groups are drawn, edited, and displayed. These descriptions primarily concern shape, but may include other information helpful to

the recognition process, such as stroke order or stroke direction. The specification of editing behavior allows the system to determine when a pen gesture is intended to indicate editing rather than a stroke. Display information indicates what to display after strokes are recognized.

The language consists of predefined shapes, constraints, editing behaviors, and display methods, as well as a syntax for specifying a domain description and extending the language. The difficulty in creating such a language is ensuring that domain descriptions are easy to specify, and that the descriptions provide enough detail for accurate sketch recognition. To simplify the task of creating a domain description, shapes can be built hierarchically, reusing low-level shapes. Shapes can extend abstract shapes, which describe shared shape properties, preventing the application designer from having to redefine these properties several times. The language has proven powerful enough to describe shapes from several domains. The language enables more accurate sketch recognition by supporting both top-down and bottom-up recognition. Descriptions of how shapes may combine can aid in top-down recognition and can be used to describe “chain-reaction” editing commands.

Our contribution is in creating LADDER, the first sketching language to incorporate editing, display, and shape group information. To test our language, we have built a simple domain-independent sketch recognition system that parses the domain description and successfully recognizes shapes based on these descriptions.

Section 2 describes the components of the language, including the predefined shapes, constraints, editing behaviors, and display methods available. Section 3 describes the syntax and content of a sketch grammar designed in the language. Section 4 describes a system we have implemented to test the language and ensure that shapes in a domain can be recognized based on their descriptions. Section 5 describes related work done in the development in sketch languages.

2 Language Contents

The language consists of predefined shapes, constraints, editing behaviors, and display methods. Figure 2 shows an example description for **OpenArrow** drawn in (Figure 1). The description of a shape contains a list of *components* (the elements from which the shape is built), geometric *constraints* on those components, a set of *aliases* (names that can be used

to simplify other elements in the description), *editing* behaviors (how the object should react to editing gestures), and *display* methods indicating what to display when the object is recognized.

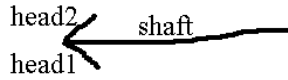


Figure 1: An open arrow.

```
(define shape OpenArrow
  (description "An arrow with an open head")
  (components
    (Line shaft)
    (Line head1)
    (Line head2))
  (constraints
    (coincident shaft.p1 head1.p1)
    (coincident shaft.p1 head2.p1)
    (coincident head1.p1 head2.p1)
    (equal-length head1 head2)
    (acute-meet head1 shaft)
    (acute-meet shaft head2))
  (aliases
    (Point head shaft.p1)
    (Point tail shaft.p2))
  (editing
    ( (trigger (click_hold_drag shaft))
      (action
        (translate this)
        (set-cursor DRAG)
        (show-handle MOVE tail head)))
      ( (trigger (click_hold_drag head))
        (action
          (rubber-band this head tail)
          (show-handle MOVE head)
          (set-cursor DRAG)))
      ( (trigger (click_hold_drag tail))
        (action
          (rubber-band this tail head)
          (show-handle MOVE tail)
          (set-cursor DRAG))))
    (display (original-strokes)))
```

Figure 2: The description for an arrow with an open head

The power of the language is derived in part from carefully chosen predefined building blocks.

2.1 Predefined Shapes

The language includes a number of predefined primitive and non-primitive shapes, usable as building blocks in describing other shapes. The primitive shapes are **Shape**, **Point**, **Path**, **Line**, **BezierCurve**, and **Spiral**. **Circle**, **Arc**, and **Ellipse** are examples of non-primitive shapes included in the language library; all three are more specific versions of the primitive shape **Spiral**. The **OpenArrow** in Figure 2 is a non-primitive shape built out of three primitive shapes.

The language uses an inheritance hierarchy; **Shape** is an abstract shape which all other shapes extend. **Shape** provides a number of components and properties for all shapes, including *boundingbox*, *centerpoint*, *width*, and *height*. Each predefined shape may have additional components and properties; a **Line**, for example, also has *p1*, *p2* (the endpoints), *midpoint*, *length*, *angle*, and *slope*. Components and properties for a shape can be used hierarchically in shape descriptions. When defining a new shape the components and properties are those defined by **Shape**, and those defined by the *components* and *aliases* section.

2.2 Predefined Constraints

New shapes are defined in terms of previously defined shapes and constraints between them. For instance, the **OpenArrow** in Figure 2 contains the constraint (*acute-meet head1 shaft*), which indicates that *head1* and *shaft* meet at a point and form an acute angle in a counter-clockwise direction from *head1* to *shaft*. (Angles are measured in a counter-clockwise direction.)

A number of predefined constraints are included in the language, including: *perpendicular*, *parallel*, *collinear*, *same-side*, *opposite-side*, *coincident*, *connected*, *meet*, *intersect*, *tangent*, *contains*, *concentric*, *larger*, *near*, *draw-order*, *equal-length*, *=*, *<*, *<=*, *angle*, *angle-dir*, *acute*, *obtuse*, *acute-meet*, and *obtuse-meet*. If a sketch grammar consists of only the constraints above, the shape is rotationally invariant.

There are also predefined constraints that are valid only in a particular orientation, including *horizontal*, *vertical*, *pos-slope*, *neg-slope*, *left-of*, *right-of*, *above*, *below*, *same-H-pos*, *same-V-pos*, *above-left*, *above-right*, *below-left*, *below-right*, *centered-below*, *centered-above*, *centered-left*, *centered-right*, and *angleL*, where (*angleL line1 degrees*) specifies that the angle between a horizontal line pointing right and *line1* is *degrees*.

There is an additional constraint: *is-rotatable*, which implies the shape can be found in any orientation. If *is-rotatable* is specified along with an orientation-dependent constraint, there must be an *angleL*, *horizontal*, or *vertical* constraint specified, which serves to define the orientation and set a relative coordinate system. For example, the two *angle-meet* constraints could have been replaced with:

```
(is-rotatable) (horizontal shaft)(neg-slope head1) (pos-slope
head2) (left-of shaft.p1 shaft.p2) (left-of head1.p2 shaft.p2)
(left-of head2.p2 shaft.p2),
```

in which case the *shaft* is the reference line.

2.3 Predefined Editing Behaviors, Actions, and Triggers

Describing editing gestures permits the recognition system to discriminate between sketching (pen gestures intended to leave a trail of ink) and editing gestures (pen gestures intended to change existing ink), and permits us to describe the desired behavior in response to a gesture.

In order to encourage interface consistency, the language includes a number of predefined editing behaviors described using the actions and triggers above. One such example is *DragInside*, defines that if you *click-hold-drag* the pen starting inside of the bounding box of a shape, the entire shape automatically moves with it.

When defining a new editing behavior particular to a domain, there are two things to specify: the trigger – what signals an editing command – and the action – what should happen when the trigger occurs. The language has a number of predefined triggers and actions to aid in describing editing behaviors.

For instance, in Figure 2, the **OpenArrow** contains three editing behaviors. The first editing behavior says that if you click and hold the pen over the **shaft** of the **OpenArrow**, when you drag the pen, the entire **OpenArrow** will translate along with the movement of the arrow. The second editing behavior states that if you click and hold the pen over the **head**

of the arrow, the **head** of the arrow will follow the motion of the pen, but the **tail** of the arrow will remain fixed and the entire **OpenArrow** will stretch like a rubber band (translating, scaling, and rotating) to satisfy these two constraints and keep the **OpenArrow** as one whole shape. All of the editing behaviors also change the pen's cursor as displayed to the sketcher, and display moving handles to the sketcher to let the sketcher know that she performing an editing command.

The possible editing actions include *wait*, *select*, *deselect*, *color*, *delete*, *translate*, *rotate*, *scale*, *resize*, *rubber-band*, *show-handle*, and *set-cursor*. To give an example:

```
(rubber-band shape-or-selection fixed-point move-point
[new-point])
```

translates, scales, and rotates the *shape-or-selection* so that the *fixed-point* remains in the same spot, but that the *move-point* translates to the *new-point*. If *new-point* is not specified, *move-point* translates according to the movement of the pen.

The possible triggers include: *click*, *double-click*, *click-hold*, *click-hold-drag*, *draw*, *draw-over*, *scribble-over*, and *encircle*. Possible triggers also include any action listed above, to allow for "chain-reaction" editing.

Shape groups allow designers to define "chain-reaction" editing behaviors. For instance, the designer may want to specify that when we move a rectangle, if there is an arrow head inside of this rectangle, the arrow should move with the rectangle.

2.4 Predefined Display Methods

An important part of a sketching interface is controlling what the user sees after shapes are recognized. The designer can specify that the original strokes should remain, or instead that a cleaned version of the strokes should be displayed. In the cleaned version, the original strokes are fit to straight lines, clean curves, clean arcs, or a combination.

Another option is to display the ideal version of the strokes. In this case, lines that are supposed to connect at their end points actually connect and lines that are supposed to be parallel are actually shown as parallel. In the ideal version of the strokes, all of the noise from sketching is removed.

It may be that we don't want to show any version of the strokes at all, but some other picture. In this case, we can either place an image at a specified location, size, and rotation, or we can create a picture built out of predefined shapes, such as circles, lines, and rectangles.

The pre-defined display methods include: *original-strokes*, *cleaned-strokes*, *ideal-strokes*, *circle*, *line*, *point*, *rectangle*, *text*, *color*, and *image*. Each method includes color as an optional argument.

3 Specifying a Domain Description

A domain description contains a list of the domain shapes and shape groups, as well as definitions for each of them. Descriptions can be hierarchical and can refer to other shapes in the language. This section provides examples from the domain description sketch grammar of UML (Unified Modelling Language) class diagrams [Booch *et al.*, 1998].

3.1 Indicating Domain Definitions

The compiler uses a list of domain shapes and shape groups to confirm that each shape is properly defined and to speed recognition by creating recognizers only for sub-shapes needed by the domain.

3.2 Defining Shapes

A domain shape is a shape that is meaningful in the domain. Geometric shapes usually occur in several domains and are the building blocks of the domain shapes. For instance, in the domain of UML class diagrams, the domain shapes (followed by their geometric shape component) are: general classes (represented by rectangles), interface classes (circles), interface associations (lines), dependency associations (open-headed arrows), aggregation associations (diamond-headed arrows), inheritance associations (triangle-headed arrows), information associations (dotted lines or dotted open arrows).

A shape definition includes primarily geometric information, but can include other drawing information that may be helpful to the recognition process, such as stroke order or stroke direction. A shape definition is composed of seven sections. All sections are optional except the *components* section.

1. The *description* contains a textual description of the shape, e.g., "an arrow with a triangle-shaped head."
2. The *is-a* section specifies any class of abstract shapes (Section 3.3) that the shape may be a part of. This is similar to the extends property in Java. All shapes extend the abstract shape **Shape**.
3. The *components* section lists the components of the shape. For example, the **TriangleArrow** in Figure 3 is built out of the **OpenArrow** from Figure 2 and a **Line**. Components can be accessed hierarchically.
4. The *constraints* section specifies relationships between the components. For example, in the **TriangleArrow** in Figure 3, (*coincident head3.p1 head1.p2*) specifies that an endpoint of *head3* and an endpoint of *head1* are located at the same point.
The *constraints* section can specify both hard constraints, such as the one listed above, and soft constraints, which are specified by the keyword *soft*. Hard constraints are always satisfied in the shape, but soft constraints may not be. Soft constraints can aid recognition by specifying relationships that usually occur. For instance, in Figure 3 the shaft of the arrow is commonly drawn before the head of the arrow, but the arrow should still be recognized even if this constraint is not satisfied.
5. The *aliases* section allows us to compute certain properties and name them for use later. For instance, in Figure 3, *head1* is defined and used in a constraint for simplicity. Components specified in the *aliases* section can be accessed hierarchically. For instance, **TriangleArrow** uses *head* and *tail* from the **OpenArrow** in Figure 2.
6. An *editing* section specifies how the shape can be edited. Common editing commands involve movement

and deletion of the shape. Each editing behavior must specify a trigger and an action listed in Section 2.3.

Shapes can be defined to be moved as a whole rather than having to move individual lines. For instance, in Figure 3, one editing behavior for the **TriangleArrow** indicates that if the user presses and holds the pen on the **shaft** for a brief period, the pen will drag the entire **TriangleArrow** when moved.

7. A *display* section specifies what should be displayed on the screen when the shape is recognized. This section is generally included only for domain shapes, not for geometric shapes. In the **UMLInheritanceAssociation** in Figure 4, the arrow will be displayed using straight lines for the arrow head and the original stroke for the shaft.

```
(define shape TriangleArrow
  (description "An arrow with a triangle-shaped head")
  (components
    (OpenArrow oa)
    (Line head3))
  (aliases
    (Line shaft oa.shaft)
    (Line head1 oa.head1)
    (Line head2 oa.head2)
    (Point head oa.head)
    (Point tail oa.tail))
  (constraints
    (coincident head3.p1 head1.p2)
    (coincident head3.p2 head2.p2)
    (soft draw-order shaft head1)
    (soft draw-order shaft head2))
  (editing
    ( (trigger (click_hold_drag shaft))
      (action
        (translate this)
        (set-cursor DRAG)
        (show-handle MOVE tail head))))))
```

Figure 3: The description for an arrow with a triangle-shaped head.

```
(define shape UMLInheritanceAssociation
  (is-a UMLGeneralAssociation)
  (components
    (TriangleArrow arrow))
  (aliases
    (Point head arrow.head)
    (Point tail arrow.tail)
    (Line shaft arrow.shaft))
  (display
    (original_strokes arrow.shaft)
    (cleaned_strokes arrow.head1 arrow.head2 arrow.head3)))
```

Figure 4: The domain shape UML Inheritance Association is defined by the geometrical shape **TriangleArrow** from Figure 3.

3.3 Defining Abstract Shapes

In the **UMLInheritanceAssociation** defined in Figure 4, the *is-a* section specifies that the **UMLInheritanceAssociation** is an extension of the abstract shape **UMLGeneralAssociation**. Abstract shapes have no concrete shape associated with them; they represent a class of shapes that have similar attributes or editing behaviors. These attributes can be defined once in the abstract shape description rather than for each domain shape. For instance, in Figure 2 and Figure 3, notice that the **OpenArrow** and the **TriangleArrow** have identical editing behaviors defined. Rather than repeatedly listing these

editing behaviors in each shape, we could create an abstract shape which specifies these editing behaviors.

An abstract shape is defined similarly to a regular shape, except it has a *required* section instead of a *components* section. Each shape that extends the abstract shape must define each variable listed in the *required* section, in its *components* or *aliases* section.

Figure 5 presents a diagram of the inheritance hierarchy for the abstract and non-abstract shapes in the UML class diagrams domain. In UML, **UMLDependencyAssociation**, the **UMLInheritanceAssociation**, the **UMLAggregationAssociation**, the **UMLInformationAssociation**, and the **InterfaceAssociation** are all links represented by arrows or lines and all have the same editing behavior. Thus, we can create the abstract shape **UMLAssociation**, which lists the editing behavior of these shapes. Figure 6 shows the abstract shape description of the **UMLAssociation**. Notice that the required variables are used when defining editing behaviors.

```
(define abstract-shape UMLAssociation
  (is-a Shape)
  (required
    (Point head)
    (Point tail)
    (Line shaft))
  (editing
    ( (trigger (click_hold_drag shaft))
      (action
        (translate this)
        (set-cursor DRAG)
        (show-handle MOVE tail head))))))
```

Figure 6: The description for the abstract class UMLAssociation.

3.4 Defining Shape Groups

A shape group is a collection of domain shapes that are commonly found together in the domain. Defining shape groups provides two significant benefits. Shape groups can be used by the recognition system to provide top-down recognition, and “chain-reaction” editing behaviors can be applied to shape groups, allowing the movement of one shape to cause the movement of another.

In the domain of UML class diagrams, there are six legal shape groups that describe the visual relationship between UML associations and UML classes. For example, one shape group consists of an association combined with a general class, such that the tail of the association is inside or near the general class shown in Figure 7 and described in Figure 8. A shape group definition is similar to that of a shape definition.



Figure 7: An association attached to a class at its tail.

If a single shape in a sketch can be part of many instances of a shape group, then we place the key word *multiple* before the component shape of the shape group. In UML Class Diagrams, for example, a single **UMLAssociation** can be part of only one instance of a shape group, while a single **UMLClass** can be part of many instances of **UMLGenClassGenAssociationTail**.

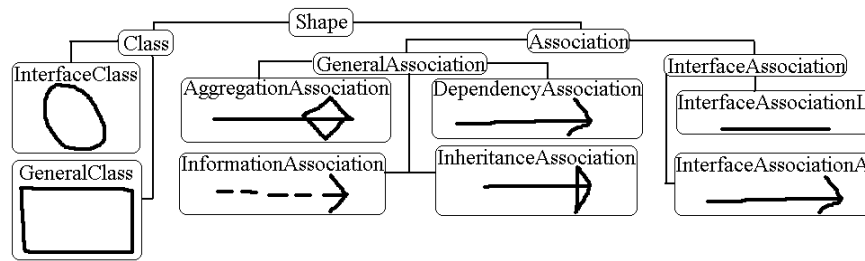


Figure 5: The inheritance diagram of UML Class Diagram shapes.

```
(define shape-group
  UMLGenClassGenAssociationTail
  (description "A general class attached to
    the tail of a general association")
  (is-a UMLAssociationAttachedTail)
  (components
    (multiple (GeneralClass ct))
    (GeneralAssociation r)))
```

Figure 8: Description of the shape group from Figure 7.

3.5 Defining Abstract Shape Groups

Abstract shape groups definitions allow the reuse of shared properties across multiple shape groups. The definition of a **UMLAssociationAttachedTail** in Figure 9 indicates that the tail, but not the head, of the association is inside the class, preventing us from having to redefine the constraints many times, and allows us to define one general editing behavior for many shapes. An editing behavior for the **UMLAssociationAttachedTail** indicates that whenever you move a **UMLClass** that is attached to the tail of a **UMLAssociation**, the head of the **UMLAssociation** remains fixed in its original location, but the tail of the **UMLAssociation** remains attached to the **UMLClass** as it moves; the **UMLAssociation** acts like a rubber band (translating, scaling, and rotating) to satisfy these constraints.

```
(define abstract-shape-group
  UMLAssociationAttachedTail
  (required
    (Association r)
    (Class ct))
  (constraints
    (contains ct r.tail)
    (!contains ct r.head))
  (editing
    (trigger (translate ct))
    (action (rubber-band r r.head r.tail))))
```

Figure 9: Definition for an abstract shape group.

3.6 Defining Constraints

We believe the language contains sufficient constraints to define a broad range of domains. When an additional constraint is needed, it can be defined using a macro facility. The sections of a sketch-constraint definition include *description*, *components*, and *constraints*.

4 Testing

4.1 Examples of Shapes Described in the Language

We have evaluated the language by showing that it can describe a wide variety of symbols from a number of different domains. We have used it to describe over a hundred shapes

```
(define constraint centered-in
  (description "Tests if shape1 is centered inside shape2")
  (components
    (Shape s1)
    (Shape s2))
  (constraints
    (contains s2 s1)
    (coincident s1.center s2.center)))
```

Figure 10: Definition for the constraint centered-in.

from the domains of UML class diagrams, mechanical engineering diagrams, course of action diagrams, and letters of the alphabet. Illustrative examples are given below.

Polygon

A **PolyLine** (shown in Figure 11a), may contain a variable number of line segments. A variable number of components is specified by the key word *vector* and must specify the minimum and maximum number of components. If the maximum number can be infinite, the variable *n* is listed. For instance the **PolyLine** must contain at least two lines, and each line must be connected with the previous. The definition of a **Polygon** easily follows from the definition of the **PolyLine**

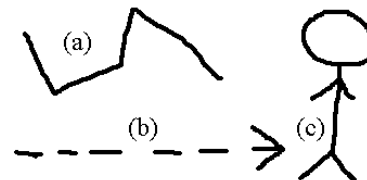


Figure 11: (a)Poly Line (b) Dashed Open Arrow (c) Stick Figure

```
(define shape PolyLine
  (components (vector Line vl[2,n]))
  (constraints (coincident vl[i].p2 vl[i+1].p1))
  (aliases (Point head vl[0].p1)(Point tail vl[n].p2)))

(define shape Polygon
  (components (PolyLine poly))
  (constraints (coincident poly.head poly.tail)))
```

Figure 12: Shape Description of a Polygon.

Dashed Open Arrow

A **DashedOpenArrow** (Figure 11b) is made from a **OpenArrow**, and a **Dashedline**, which in turn contains at least two line segments. When given a third argument specifying a length, the constraint *near* states that two points are near to each other relative to a given length.

Stick Figure

The definition of a stick figure (Figure 11c) shows how we can create new components to help describe shapes. It creates

```
(define shape DashedLine
  (components (vector Line vl[2,n]))
  (constraints (collinear vl[i].p1 vl[i].p2 vl[i+1].p1)
    (! (intersect vl[i] vl[i+1])
      (near vl[i].p2 vl[i+1].p1 vl[i].length))
    (aliases (Point head vl[0].p1)(Point tail vl[n].p2))))

(define shape DashedOpenArrow
  (components (OpenArrow oa)(DashedLine dl))
  (constraints (near oa.tail dl.head oa.shaft))
  (aliases
    (Point head oa.head)(Point tail dl.tail)))
```

Figure 13: Description of a dashed line and a dashed open arrow.

a new line between the feet for use in defining constraints, ensuring that both feet lie below the body.

```
(define shape StickFigure
  (description "a stickfigure with two arms and two legs
    all sloping down at 45 degrees")
  (components (Circle head)(Line body)
    (Line larm)(Line rarm)(Line lleg)(Line rleg))
  (alias (Line feet_space (new Line (lleg.p2 rleg.p2))))
  (constraints (meet head body.p1)(!(intersect body head)
    (is-rotatable) (vertical body) (meet body larm.p1)
    (meet body rarm.p1) (coincident larm.p1 rarm.p1)
    (acute larm body)(acute body rarm)(left-of larm rarm)
    (coincident body.p2 lleg.p1)(coincident body.p2 rleg.p1)
    (obtuse body lleg)(obtuse body rleg)
    (perpendicular larm rarm)(perpendicular lleg rleg)
    (near body.p1 rarm.p1)(parallel rarm rleg)
    (parallel larm lleg)(!(intersect feet_space body))
    (equal-length lleg rleg)
    (equal-length larm rarm)))
```

Figure 14: Description of a stick figure.

4.2 System Implementation

We built a simple domain-independent recognition system to test whether sketches can be recognized from our domain descriptions. The system parses a domain description into Java code and Jess (a rule-based system that interfaces with Java) [Friedman-Hill, 1995] rules, and uses them to recognize sketches. For example, using the domain description for UML, the system successfully recognized hand-drawn sketches of all of the shapes in Figure 5 regardless of overlap.

Domain Description Parsing

The domain description is parsed to create recognition code, creating at least one Jess rule (containing the shape recognition information), and one Java file (describing the shape), for each shape description. The system then uses the Jess rules to recognize sketches.

Jess Rule Example

The rule automatically generated for the **TriangleArrow** from Figure 3 is shown in Figure 15. If a shape description contains a vector, such as that of the **DashedArrow** in Figure 13, two Jess rules are created, one containing the base case, and the second containing a recursive rule.

Stroke Preprocessing

The recognition system has several stages of recognition. First, each time a stroke is drawn, the stroke is pre-processed [Sezgin *et al.*, 2001] into either Point, Line, Curve, Arc, or a combination thereof, allowing users to draw objects in a single stroke or with multiple strokes. These primitive shapes are then asserted to a Jess database.

```
(defrule TriangleArrowCheck
  ;;Get the parts of the triangle arrow
  ?f0 <- (Subshapes OpenArrow ?oa $?oa_list)
  ?f1 <- (Subshapes Line ?head3 $?head3_list)
  (OpenArrow ?oa ?oa_shaft ?oa_head1 ?oa_head2 ?oa_head ?oa_tail)
  (Line ?head3 ?head3_p1 ?head3_p2)

  ;;Make sure that the openarrow and line don't share any subparts.
  (test (uniquefields $?oa_list $?head3_list))

  ;; Get the subpart of the open arrow since they are referenced
  (Line ?oa_shaft ?shaft_p1 ?shaft_p2)
  (Line ?oa_head1 ?head1_p1 ?head1_p2)
  (Line ?oa_head2 ?head2_p1 ?head2_p2)

  ;; test for that the constraints hold
  (test (coincident ?head3_p1 ?head1_p2))
  (test (coincident ?head3_p2 ?head2_p2))
  =>

  ;; Triangle Arrow has been successfully found
  ;; Set the aliases
  (bind ?shaft ?oa_shaft)(bind ?head1 ?oa_head1)
  (bind ?head2 ?oa_head2)(bind ?head ?oa_head)
  (bind ?tail ?oa_tail)

  ;; Tell the recognition system that there is a Triangle Arrow
  (bind ?nextnum (addshape TriangleArrow ?oa $?head3_list ?shaft
    ?head1 ?head2 ?head ?tail))

  ;; Tell the Jess system that there is a Triangle arrow
  (assert (TriangleArrow ?nextnum ?oa ?head3 ?shaft ?head1 ?head2
    ?head ?tail))
  (assert (Subshapes TriangleArrow ?nextnum (union$ $?oa_list
    $?head3_list)))

  ;; Triangle Arrow is a domain shape. Assert it.
  ;; Conflicts will be resolved elsewhere.
  (assert (DomainShape TriangleArrow ?nextnum (time)))
  (assert (CompleteSubshapes OpenArrow ?oa $?oa_list)))
```

Figure 15: Automatically generated Jess Rule for the Triangle Arrow.

Recognition of Shapes

Recognition is handled by the Jess rule based system. We have automatically generated templates for the Jess system to fill in. Once Jess finds the appropriate components, the rule is fired and the constraints are tested. The constraints are Java functions with which Jess interacts. All possible shapes are found, even if the shapes share lines or other components. If the shape can be a domain shape (i.e., a final shape in the domain), the shape is asserted as a domain shape.

Domain Shapes

When domain shapes are created, a rule fires in Jess confirming that no two found domain shapes share the same sub-components. If two domain shapes do share subcomponents, one domain shape is retracted. The domain shape chosen to remain is the one containing more primitive components (following Ockham's Razor); if the two shapes contain the same number of components, the shape created first is chosen, since previously chosen recognitions have higher precedence. If a domain shape is found, then the recognition system displays the shape as the designer specified and editing commands can then be performed on the shape.

5 Related Work

Shape description languages, such as shape grammars, have been around for a long time [Stiny and Gips, 1972]. Shape grammars are studied widely within the field of architecture, and many systems are continuing to be built using shape grammars [Gips, 1999]. However, shape grammars were developed for shape generation rather than recognition, and don't provide for non-graphical information, such as stroke

order, that may be helpful in recognition. They also lack ways for specifying shape editing.

Within the field of sketch recognition, there have been other attempts to create languages for sketch recognition. Bimber et. al [2000] describe a simple sketch language using a BNF-grammar. The language describes three-dimensional shapes hierarchically. This language allows a programmer to specify only shape information and lacks the ability to specify other helpful domain information such as stroke order or direction and editing behavior, display, or shape interaction information.

Mahoney [2002] uses a language to model and recognize stick figures. The language currently is not hierarchical, making large objects cumbersome to describe. Caetano et. al. [2002] use fuzzy relational grammars to describe shape. Both Mahoney and Caetano lack the ability to describe editing, display, or shape grouping information.

The Electronic Cocktail Napkin project [Gross and Do, 1996] allows users to define domain shapes by drawing them. A shape is described by the shapes it is built out of and the constraints between them. The Cocktail Napkin's language is able to describe only shape.

Jacob [Jacob et al., 1999] has created a software model and language for describing and programming fine-grained aspects of interaction in a non-WIMP user interface, such as a virtual environment. The language is very low-level making it difficult to define new interactions, and, in the domain of sketching, does not provide a significant improvement to coding the domain-dependent recognition system from scratch.

The language described in this paper is being used in several other systems, including sketch learning from example [Veselova, 2002], smart compiling using HMM's [Sezgin, 2002], and an intelligent Bayesian network context oriented sketch recognition system [Alvarado et al., 2002].

6 Conclusion

6.1 Future Work

We plan to examine more domains to ensure that the language contains the appropriate primitives. We would also like to test our syntax on a wide user base. Large domains benefit from visual diagrams, such as the one in Figure 5. We plan to automatically generate some of these visual diagrams to help with the grammar-writing process.

6.2 Contributions

We have created LADDER, the first language to describe how sketched diagrams in a domain are drawn, displayed, and edited. The language consists of pre-defined shapes, constraints, editing-behaviors, and display methods, as well as a syntax for specifying a sketch grammar and extending the language, ensuring that shapes and shape groups from many domains can be described. The syntax simplifies the definition of new shapes by allowing shapes to be built hierarchically and by providing abstract shapes to contain common shape properties. Shape groups describe how domain shapes interact, and can provide information to be used in top-down as well as bottom-up recognition. Shape groups can also be used to describe "chain-reaction" editing commands. We have built a simple domain-independent sketch recognition

system for testing that recognition is feasible based on the descriptions provided.

Acknowledgments

This work is supported in part by the MIT Oxygen Collaboration. The authors would like to thank Jacob Eisenstein for the name LADDER and Jacob Eisenstein, Michael Oltmans, Mark Foltz, and Vineet Sinha for their help in reviewing this paper.

References

- [Alvarado et al., 2002] C Alvarado, M Oltmans, and R Davis. A framework for multi-domain sketch recognition. *AAAI Spring Symposium on Sketch Understanding*, pages 1–8, March 25–27 2002.
- [Bimber et al., 2000] O Bimber, LM Encarnao, and A Stork. A multi-layered architecture for sketch-based interaction within virtual environments. *Computer and Graphics*, 2000.
- [Booch et al., 1998] G Booch, J Rumbaugh, and I Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, Reading, MA, 1998.
- [Caetano et al., 2002] A Caetano, N Goulart, M Fonseca, and J Jorge. Javasketchit: Issues in sketching the look of user interfaces. *AAAI Spring Symposium on Sketch Understanding*, 2002.
- [Friedman-Hill, 1995] E Friedman-Hill. Jess, the rule engine for the java platform. <http://herzberg.ca.sandia.gov/jess/>, 1995.
- [Gips, 1999] J Gips. Computer implementation of shape grammars. *NSF/MIT Workshop on Shape Computation*, 1999.
- [Gross and Do, 1996] MD Gross and EYL Do. Demonstrating the electronic cocktail napkin: a paper-like interface for early design. *ACM Conference on Human Factors in Computing*, pages 5–6, 1996.
- [Hunt and McGlashan, 2002] A Hunt and S McGlashan. Speech recognition grammar specification version 1.0, w3c candidate recommendation. <http://www.w3.org/TR/speech-grammar>, 26 June 2002.
- [Jacob et al., 1999] RJK Jacob, L Deligiannidis, and S Morrison. A software model and specification language for non-WIMP user interfaces. *ACM Transactions on Computer-Human Interaction*, 6(1):1–46, 1999.
- [Mahoney and Fromherz, 2002] JV Mahoney and MPJ Fromherz. Three main concerns in sketch recognition and an approach to addressing them. *AAAI Spring Symposium on Sketch Understanding*, pages 105–112, March 25–27 2002.
- [Sezgin et al., 2001] TM Sezgin, T Stahovich, and R Davis. Sketch based interfaces: Early processing for sketch understanding. *Perceptive User Interfaces Workshop*, 2001.
- [Sezgin, 2002] M Sezgin. Generating domain specific sketch recognizers from object descriptions. *MIT Student Oxygen Workshop*, 2002.
- [Stiny and Gips, 1972] G Stiny and J Gips. Shape grammars and the generative specification of painting and sculpture. *Information Processing*, pages 1460–1465, 1972.
- [Veselova, 2002] O Veselova. Perceptually based learning of shape descriptions from one example. *MIT Student Oxygen Workshop*, 2002.
- [Zue et al., 1990] V Zue, J Glass, M Phillips, and S Seneff. The mit speech recognition system: phonological modeling and lexical access. *1990 International Conference on Acoustics, Speech and Signal Processing*, pages 49–52, March 25–27 1990.

Evaluating Significance of Inconsistencies

Anthony Hunter

Department of Computer Science
University College London
Gower Street
London WC1E 6BT, UK
a.hunter@cs.ucl.ac.uk

Abstract

Inconsistencies frequently occur in knowledge about the real-world. Some of these inconsistencies may be more significant than others, and some knowledgebases (sets of formulae) may contain more inconsistencies than others. This creates problems of deciding whether to act on these inconsistencies, and if so how. To address this, we provide a general characterization of inconsistency, based on quasi-classical logic (a form of paraconsistent logic with a more expressive semantics than Belnap's four-valued logic, and unlike other paraconsistent logics, allows the connectives to appear to behave as classical connectives). We analyse inconsistent knowledge by considering the conflicts arising in the minimal quasi-classical models for that knowledge. This is used for a measure of coherence for each knowledgebase, and for a measure of significance of inconsistencies in each knowledgebase. In this paper, we formalize this framework, and consider applications in managing heterogeneous sources of knowledge.

1 Introduction

An approach to measuring inconsistency is to use propositional quasi-classical (QC) logic [Hunter, 2002]. In this, each inconsistent set of formulae is reflected in the QC models for the set, and then the inconsistency is measured in the models. Obviously, this is not possible in classical logic, or indeed many non-classical logics, because there is no model of an inconsistent set of formulae. QC logic, a form of paraconsistent logic, can model inconsistent sets of formulae. There are other paraconsistent logics that we could consider, for example Belnap's four-valued logic [Belnap, 1977], or Levesque's 3-interpretations [Levesque, 1984], or Grant's generalizations of classical satisfaction [Grant, 1978], but these, as we will illustrate, involve the consideration of too many models. This increases the number of models that need to be analysed and it underspecifies the nature of the conflicts.

However, the original proposal for measuring inconsistency based on QC logic does not provide an evaluation of the significance of inconsistencies. As an illustration of the need to evaluate significance, consider two news reports on a

World Cup match, where the first report says that Brazil beat Germany 2-0, and the second report says that Germany beat Brazil 2-0. This is clearly a significant inconsistency. Now consider two news reports on the same football match, where the first report says that the referee was Pierluigi Collina and the second report says that the referee was Ubaldo Aquino. This inconsistency would normally be regarded as relatively insignificant.

In this paper, we extend the idea of measuring inconsistency via QC models by giving a new framework for measuring the significance of inconsistencies in a knowledgebase. To do this, we review aspects of QC logic and a measurement of inconsistency based on it. We then present our framework for evaluating the significance of inconsistencies. We show this leads to a generalization of a four-valued semantics where instead of a single truth value to denote inconsistency, we have an infinite sequence of inconsistent truth values B_i where $i \in [0, 1]$. We then discuss how we can use QC logic together with significance functions for reasoning and analysing with heterogeneously sourced information such as news from different newsfeeds.

2 Review of QC Logic

We review the propositional version of quasi-classical logic (QC Logic) [Besnard and Hunter, 1995; Hunter, 2000].

Definition 1 *The language of QC logic is that of classical propositional logic. We let $\mathcal{L}$ denote a set of formulae formed in the usual way from a set of atom symbols $\mathcal{A}$, and the connectives $\{\neg, \vee, \wedge\}$. If $\Gamma \in \wp(\mathcal{L})$, then $\text{Atoms}(\Gamma)$ returns the set of atom symbols used in Γ .*

Definition 2 *Let α be an atom, and let $\sim$ be a complementation operation such that $\sim \alpha$ is $\neg \alpha$ and $\sim (\neg \alpha)$ is α . The $\sim$ operator is not part of the object language, but it makes some definitions clearer.*

Definition 3 *Let $\alpha_1 \vee \dots \vee \alpha_n$ be a clause that includes a disjunct α_i and $n > 1$. The **focus** of $\alpha_1 \vee \dots \vee \alpha_n$ by α_i , denoted $\otimes(\alpha_1 \vee \dots \vee \alpha_n, \alpha_i)$, is defined as the clause obtained by removing α_i from $\alpha_1 \vee \dots \vee \alpha_n$.*

Example 1 *Let $\alpha \vee \beta \vee \gamma$ be a clause where α, β , and γ are literals. Hence, $\otimes(\alpha \vee \beta \vee \gamma, \beta) = \alpha \vee \gamma$.*

Focus is used to capture a form of resolution in the semantics of QC logic. A model in QC logic is a form of Herbrand model.

Definition 4 Let $\mathcal{A}$ be a set of atoms. Let $\mathcal{O}$ be the set of objects defined as follows, where $+\alpha$ is a positive object, and $-\alpha$ is a negative object.

$$\mathcal{O} = \{+\alpha \mid \alpha \in \mathcal{A}\} \cup \{-\alpha \mid \alpha \in \mathcal{A}\}$$

We call any $X \in \wp(\mathcal{O})$ a **QC model**. So X can contain both $+\alpha$ and $-\alpha$ for some atom α .

For each atom $\alpha \in \mathcal{L}$, and each $X \in \wp(\mathcal{O})$, $+\alpha \in X$ means that in X there is a **reason for** the belief α and that in X there is a **reason against** the belief $\neg\alpha$. Similarly, $-\alpha \in X$ means that in X there is a **reason against** the belief α and that in X there is a **reason for** the belief $\neg\alpha$. This effectively gives us a four-valued semantics. Though for non-atomic formulae the semantics, defined next, is significantly different to [Belnap, 1977].

Definition 5 Let $\models_s$ be a satisfiability relation called **strong satisfaction**. For a model X , we define $\models_s$ as follows, where $\alpha_1, \dots, \alpha_n$ are literals in $\mathcal{L}$, $n > 1$, and α is a literal in $\mathcal{L}$.

$X \models_s \alpha$ iff there is a reason for the belief α in X

$$\begin{aligned} X \models_s \alpha_1 \vee \dots \vee \alpha_n \\ \text{iff } [X \models_s \alpha_1 \text{ or } \dots \text{ or } X \models_s \alpha_n] \\ \text{and } \forall i \text{ s.t. } 1 \leq i \leq n \\ [X \models_s \sim \alpha_i \text{ implies } X \models_s \otimes(\alpha_1 \vee \dots \vee \alpha_n, \alpha_i)] \end{aligned}$$

For $\alpha, \beta, \gamma \in \mathcal{L}$, we extend the definition as follows,

$$\begin{aligned} X \models_s \alpha \wedge \beta &\text{ iff } X \models_s \alpha \text{ and } X \models_s \beta \\ X \models_s \neg\alpha \vee \gamma &\text{ iff } X \models_s \alpha \vee \gamma \\ X \models_s \neg(\alpha \wedge \beta) \vee \gamma &\text{ iff } X \models_s \neg\alpha \vee \neg\beta \vee \gamma \\ X \models_s \neg(\alpha \vee \beta) \vee \gamma &\text{ iff } X \models_s (\neg\alpha \wedge \neg\beta) \vee \gamma \\ X \models_s \alpha \vee (\beta \wedge \gamma) &\text{ iff } X \models_s (\alpha \vee \beta) \wedge (\alpha \vee \gamma) \\ X \models_s \alpha \wedge (\beta \vee \gamma) &\text{ iff } X \models_s (\alpha \wedge \beta) \vee (\alpha \wedge \gamma) \end{aligned}$$

Definition 6 For $X \in \wp(\mathcal{O})$ and $\Delta \in \wp(\mathcal{L})$, let $X \models_s \Delta$ denote that $X \models_s \alpha$ holds for every α in Δ . Let $\text{QC}(\Delta) = \{X \in \wp(\mathcal{O}) \mid X \models_s \Delta\}$ be the set of QC models for Δ .

A key feature of the QC semantics is that there is a model for any formula, and for any set of formulae.

Example 2 Let $\Delta = \{\neg\alpha \vee \neg\beta \vee \gamma, \neg\alpha \vee \gamma, \neg\gamma\}$, where $\alpha, \beta, \gamma \in \mathcal{A}$, and let $X = \{-\alpha, -\beta, -\gamma\}$. So $X \models_s \neg\alpha$, $X \models_s \neg\beta$ and $X \models_s \neg\gamma$. Also, $X \models_s \sim \gamma$. Hence, $X \models_s \neg\alpha \vee \gamma$, and $X \models_s \neg\alpha \vee \neg\beta$, and so, $X \models_s \neg\alpha \vee \neg\beta \vee \gamma$. Hence every formula in Δ is strongly satisfiable in X .

Strong satisfaction is used to define a notion of entailment for QC logic. There is also a natural deduction proof theory for propositional QC logic [Hunter, 2000] and a semantic tableau version for first-order QC logic [Hunter, 2001]. Entailment for QC logic for propositional CNF formulae is coNP-complete, and via a linear time transformation these formulae can be handled using classical logic theorem provers [Marquis and Porquet, 2001].

The definitions for QC models and for strong satisfaction provide us with the basic concepts for measuring inconsistency. QC logic exhibits the nice feature that no attention needs to be paid to a special form that the formulae in a set of premises should have. This is in contrast with other paraconsistent logics where two formulae identical by definition

of a connective in classical logic may not yield the same set of conclusions. For example, in QC logic, β is entailed by both $\{(\neg\alpha \rightarrow \beta), \neg\alpha\}$ and $\{\alpha \vee \beta, \neg\alpha\}$ and γ is entailed by $\{\gamma \wedge \neg\gamma\}$ and $\{\gamma, \neg\gamma\}$. QC logic is much better behaved in this respect than other paraconsistent logics such as C_ω [da Costa, 1974], and consistency-based logics such as [Benferhat et al., 1993]. Furthermore, the semantics of QC logic directly models inconsistent sets of formulae.

Definition 7 Let $\Delta \in \wp(\mathcal{L})$. Let $\text{MQC}(\Delta) \subseteq \text{QC}(\Delta)$ be the set of minimal QC models for Δ , defined as follows:

$$\text{MQC}(\Delta) = \{X \in \text{QC}(\Delta) \mid \text{if } Y \subset X, \text{ then } Y \notin \text{QC}(\Delta)\}$$

Example 3 Consider the following sets of formulae.

$$\begin{aligned} \text{MQC}(\{\alpha \wedge \neg\alpha, \alpha \vee \beta, \neg\alpha \vee \gamma\}) \\ &= \{\{+\alpha, -\alpha, +\beta, +\gamma\}\} \\ \text{MQC}(\{\neg\alpha \wedge \alpha, \beta \vee \gamma\}) \\ &= \{\{+\alpha, -\alpha, +\beta\}, \{+\alpha, -\alpha, +\gamma\}\} \\ \text{MQC}(\{\alpha \vee \beta, \neg\alpha \vee \gamma\}) \\ &= \{\{+\beta, +\gamma\}, \{+\alpha, +\gamma\}, \{-\alpha, +\beta\}\} \end{aligned}$$

Whilst four-valued logic [Belnap, 1977] also directly models inconsistent sets of formulae, there are too many Belnap models in many situations. Consider for example $\{\alpha \vee \beta, \neg\alpha\}$. There is one minimal QC model $\{-\alpha, +\beta\}$, but there are a number of Belnap models that satisfy this set. QC logic has a reduced number of models because of the constraint in the definition of strong satisfaction for disjunction that ensures that if the complement of a disjunct holds in the model, then the resolvent should also hold in the model. This strong constraint means that various other proposals for many-valued logic will tend to have more models for any given knowledgebase than QC logic. In particular, the shortcomings of Belnap's four-valued logic also apply to three-valued logics such as 3-interpretations by [Levesque, 1984], and a similar proposal by [Grant, 1978].

3 Measuring coherence of QC models

We now consider a measure of inconsistency called coherence [Hunter, 2002]. The opinionbase of a QC model X is the set of atomic beliefs (atoms) for which there are reasons for or against in X , and the conflictbase of X is the set of atomic beliefs with reasons for and against in X .

Definition 8 Let $X \in \wp(\mathcal{O})$.

$$\begin{aligned} \text{Conflictbase}(X) &= \{\alpha \mid +\alpha \in X \text{ and } -\alpha \in X\} \\ \text{Opinionbase}(X) &= \{\alpha \mid +\alpha \in X \text{ or } -\alpha \in X\} \end{aligned}$$

In finding the minimal QC models for a set of formulae, minimization of each model forces minimization of the conflictbase of each model. As a result of this minimization, if $\Delta \in \wp(\mathcal{L})$, and $X, Y \in \text{MQC}(\Delta)$, then (1) $\text{Conflictbase}(X) = \text{Conflictbase}(Y)$ and (2) either $\text{Opinionbase}(X) = \text{Opinionbase}(Y)$ or $\text{Opinionbase}(X)$ is not a subset of $\text{Opinionbase}(Y)$.

Increasing the size of the conflictbase, with respect to the size of the opinionbase, decreases the degree of coherence, as defined below.

Definition 9 The Coherence function from $\wp(\mathcal{O})$ into $[0, 1]$, is given below when X is non-empty, and $\text{Coherence}(\emptyset) = 1$.

$$\text{Coherence}(X) = 1 - \frac{|\text{Conflictbase}(X)|}{|\text{Opinionbase}(X)|}$$

If $\text{Coherence}(X) = 1$, then X is a totally coherent, and if $\text{Coherence}(X) = 0$, then X is totally incoherent, otherwise, X is partially coherent/incoherent.

Example 4 Let $X \in \text{MQC}(\{\neg\alpha \wedge \alpha, \beta \wedge \neg\beta, \gamma \wedge \neg\gamma\})$, $Y \in \text{MQC}(\{\alpha, \neg\alpha \vee \neg\beta, \beta, \gamma\})$, and $Z \in \text{MQC}(\{\neg\alpha, \beta, \neg\gamma \wedge \gamma\})$. So $\text{Coherence}(X) = 0$, $\text{Coherence}(Y) = 1/3$, and $\text{Coherence}(Z) = 2/3$.

Different minimal QC models for the same knowledgebase are not necessarily equally coherent.

Example 5 Let $\Delta = \{\alpha, \neg\alpha, \beta \vee \gamma, \beta \vee \delta\}$, and let $X = \{+\alpha, -\alpha, +\beta\}$ and $Y = \{+\alpha, -\alpha, +\gamma, +\delta\}$. So $\text{MQC}(\Delta) = \{X, Y\}$, and $\text{Coherence}(X) = 1/2$ and $\text{Coherence}(Y) = 2/3$.

We extend coherence to knowledgebases as follows.

Definition 10 Let $\Delta \in \wp(\mathcal{L})$. Assign $\text{Coherence}(\Delta)$ the maximum value in $\{\text{Coherence}(X) \mid X \in \text{MQC}(\Delta)\}$.

Example 6 Let $\Delta = \{\alpha \wedge \neg\alpha, \beta \wedge \neg\beta, \alpha \vee \beta \vee (\gamma \wedge \delta)\}$ and $\Delta' = \{\alpha \wedge \neg\alpha, \alpha \vee \beta\}$. Here $\text{Coherence}(\Delta) = \text{Coherence}(\Delta') = 1/2$.

Example 7 Let $\Delta = \{\phi \wedge \neg\phi, \alpha \vee (\beta \wedge \gamma \wedge \delta)\}$ and $\Delta' = \{\phi \wedge \neg\phi, (\alpha \wedge \beta) \vee (\gamma \wedge \delta)\}$. Also let $X_1 = \{+\phi, -\phi, +\alpha\}$, $X_2 = \{+\phi, -\phi, +\beta, +\gamma, +\delta\}$, $Y_1 = \{+\phi, -\phi, +\alpha, +\beta\}$, and $Y_2 = \{+\phi, -\phi, +\gamma, +\delta\}$. So, $\text{MQC}(\Delta) = \{X_1, X_2\}$ and $\text{MQC}(\Delta') = \{Y_1, Y_2\}$. Also, $\text{Coherence}(X_1) = 1/2$, $\text{Coherence}(X_2) = 3/4$, $\text{Coherence}(Y_1) = 2/3$, and $\text{Coherence}(Y_2) = 2/3$. So $\text{Coherence}(\Delta) > \text{Coherence}(\Delta')$.

4 Significance functions

We now present a new framework for measuring the significance of inconsistencies arising in QC models, and thereby in sets of formulae. The approach is based on specifying the relative significance of incoherent models using the notion of a mass assignment which is defined below.

Definition 11 Let $\Omega \in \wp(\mathcal{O})$ such that $\text{Coherence}(\Omega) = 0$. A mass assignment m for Ω is a function from $\wp(\Omega)$ into $[0, 1]$ such that:

- (1) If $X \subseteq \Omega$ and $\text{Coherence}(X) = 1$, then $m(X) = 0$
- (2) $\sum_{X \subseteq \Omega} m(X) = 1$

We have the constraint $\text{Coherence}(\Omega) = 0$ to ensure that for all $\alpha \in \mathcal{A}$, we have $+\alpha \in \Omega$ iff $-\alpha \in \Omega$. Condition 1 ensures mass is only assigned to models that contain conflicts and condition 2 ensures the total mass distributed sums to 1. Given some $\Omega \in \wp(\mathcal{O})$, a mass assignment can be localized on small subsets of Ω , spread over many subsets of Ω , or limited to large subsets of Ω . A mass assignment can be regarded as a form of metaknowledge, and so it needs to be specified for a domain, where the domain is characterized by Ω , and so the possible models of the domain are subsets of Ω .

Example 8 Let $\Omega = \{+\alpha, -\alpha, +\beta, -\beta\}$. A mass assignment m is given by $m(\{+\alpha, -\alpha\}) = 0.2$ and $m(\{+\beta, -\beta\}) = 0.8$. Another mass assignment m' is given by $m'(\{+\alpha, -\alpha\}) = 0.2$, $m'(\{+\alpha, -\alpha, -\beta\}) = 0.6$, and $m'(\{+\alpha, -\alpha, +\beta, -\beta\}) = 0.2$.

A significance function gives an evaluation of the significance of the conflicts in a QC model. This evaluation is in the range $[0, 1]$ with 0 as least significant and 1 as most significant.

Definition 12 Let m be a mass assignment for Ω . A significance function for Ω , denoted S_Ω , is a function from $\wp(\Omega)$ into $[0, 1]$. A mass-based significance function for m , denoted S_Ω^m , is a significance function defined as follows for each $X \in \wp(\Omega)$.

$$S_\Omega^m(X) = \sum_{Y \subseteq X} m(Y)$$

The definitions for mass assignment and mass-based significance correspond to mass assignment and belief functions (respectively) in Demspter-Shafer theory [Shafer, 1976]. However, here they are used to formalise significance rather than uncertainty. To ease reading in the following, we drop the superscript and subscript for significance functions.

Proposition 1 Let m be a mass assignment for Ω . If S is a significance function for m , then the following property of simple cumulativity holds for all $X, Y \in \wp(\Omega)$: $X \subseteq Y$ implies $S(X) \leq S(Y)$.

Given that simple cumulativity holds, we see that specifying significance in terms of mass assignment is more efficient than directly specifying the significance.

Proposition 2 Let m be a mass assignment for Ω . Let S be a mass-based significance function for m . For all $X, Y \in \wp(\Omega)$,

- (1) $S(X \cup Y) \geq (S(X) + S(Y) - S(X \cap Y))$
- (2) $S(X) + S(X^c) \leq 1$

So mass-based significance is not additive. Also the remaining significance need not be for the complement of X (ie, X^c). Some may be assigned to models not disjoint from X . We now consider some constraints on mass assignments that give useful properties for mass-based significance.

Definition 13 Let m be a mass assignment for Ω . m is focal iff for all $X \in \wp(\Omega)$ $m(X) \geq 0$ when $\text{Coherence}(X) = 0$ and $m(X) = 0$ when $\text{Coherence}(X) > 0$. m is solo iff for all $\{+\alpha, -\alpha\} \in \wp(\Omega)$ $m(\{+\alpha, -\alpha\}) \geq 0$ and for all other $X \in \wp(\Omega)$, $m(X) = 0$.

A focal mass assignment puts the mass onto the totally incoherent models, and a solo mass assignment puts the mass on the smallest totally incoherent models.

Proposition 3 If m is a solo mass assignment for Ω , then m is focal mass assignment for Ω .

Significance is additive for totally incoherent models when the mass assignment is solo.

Proposition 4 Let m be a solo mass assignment for Ω . Let S be a mass-based significance function for m and let $X \in \wp(\Omega)$. If $\text{Coherence}(X) = 0$, then $S(X) + S(X^c) = 1$.

A useful feature of a focal mass-based significance function is that as the number of conflicts rises in a model, then the significance of the model rises. This is formalized by the following notion of conflict cumulativity. It does not hold in general (see Example 9).

Proposition 5 *Let m be a focal mass assignment for Ω . If S is a significance function for m , then the following property of conflict cumulativity holds for all $X, Y \in \wp(\Omega)$: $\text{Conflictbase}(X) \subseteq \text{Conflictbase}(Y)$ implies $S(X) \leq S(Y)$.*

Example 9 *Let $\Omega = \{+\alpha, -\alpha, +\beta, -\beta, +\gamma, -\gamma\}$. Let $X = \{+\alpha, -\alpha, +\gamma\}$ and $Y = \{+\alpha, -\alpha, +\beta, -\beta\}$. Let m be an mass assignment for Ω such that $m(X) = 0.6$ and $m(Y) = 0.4$. So we have $\text{Conflictbase}(X) \subseteq \text{Conflictbase}(Y)$, but $S(X) \not\leq S(Y)$.*

In general, $S(X) \leq S(Y)$ holding does not necessarily imply $\text{Conflictbase}(X) \subseteq \text{Conflictbase}(Y)$, nor does it necessarily imply $\text{Conflictbase}(Y) \subseteq \text{Conflictbase}(X)$. Also $\text{Opinionbase}(X) \subseteq \text{Opinionbase}(Y)$ does not necessarily imply $S(X) \leq S(Y)$, nor does it necessarily imply $S(Y) \leq S(X)$. This is illustrated in the next example.

Example 10 *Let $\Omega = \{+\alpha, -\alpha, +\beta, -\beta\}$ and let m be an mass assignment for Ω where $m(X) = 0.2$ and $m(Y) = 0.8$. We consider four cases: (1) If $X = \{+\alpha, -\alpha\}$ and $Y = \{+\beta, -\beta\}$, then $S(X) \leq S(Y)$ but $\text{Conflictbase}(X) \not\subseteq \text{Conflictbase}(Y)$; (2) If $X = \{+\alpha, -\alpha\}$ and $Y = \{+\alpha, -\alpha, +\beta, -\beta\}$, then $S(X) \leq S(Y)$ but $\text{Conflictbase}(Y) \not\subseteq \text{Conflictbase}(X)$; (3) If $X = \{+\alpha, -\alpha, +\beta, -\beta\}$ and $Y = \{+\alpha, -\alpha, +\beta\}$, then $\text{Opinionbase}(X) \subseteq \text{Opinionbase}(Y)$ but $S(X) \not\leq S(Y)$; (4) If $X = \{+\alpha, -\alpha\}$ and $Y = \{+\alpha, -\alpha, +\beta\}$, then $\text{Opinionbase}(X) \subseteq \text{Opinionbase}(Y)$ but $S(Y) \not\leq S(X)$.*

We now extend the significance functions to knowledgebases. Since $\text{MQC}(\Delta)$ is not necessarily a singleton, the significance for a set of formulae Δ is the lowest significance obtained for an $X \in \text{MQC}(\Delta)$. This means we treat the information in Δ as a “disjunction” of QC models, and we regard each of those models as equally acceptable, or equivalently we regard each of those models as equally representative of the information in Δ .

Definition 14 *Let $\Delta \in \wp(\mathcal{L})$. We extend the definition for a significance function S to knowledgebases as follows:*

$$S(\Delta) = \min(\{S(X) \mid X \in \text{MQC}(\Delta)\})$$

Some knowledgebases have zero significance. Clearly, if $\Delta \not\models \perp$, then $S(\Delta) = 0$.

Example 11 *Let $\Omega = \{+\alpha, -\alpha, +\beta, -\beta, +\gamma, -\gamma\}$. Let $m(\{+\alpha, -\alpha\}) = 0.6$, $m(\{+\alpha, -\alpha, +\beta\}) = 0.3$, and $m(\{+\beta, -\beta, +\gamma\}) = 0.1$. So $S(\{\alpha \wedge \neg\alpha, \beta \vee \gamma\}) = 0.6$*

In order to determine the set Ω for which a mass function is defined, we can use the delineation function as follows.

Definition 15 *For $\Delta \in \wp(\mathcal{L})$, $\text{Delineation}(\Delta) = \{+\alpha, -\alpha \mid \alpha \in \text{Atoms}(\Delta)\}$.*

Example 12 *Let $\Delta_1 = \{\neg\alpha, \alpha \vee \beta, \neg\beta\}$, $\Delta_2 = \{\alpha \vee \beta, \neg\alpha \wedge \alpha\}$, and $\Delta_3 = \{\beta, \neg\alpha \vee \neg\beta\}$. Let $\Omega =$*

$\text{Delineation}(\Delta_1 \cup \Delta_2 \cup \Delta_3) = \{+\alpha, -\alpha, +\beta, -\beta\}$. Also let $m(\{+\alpha, -\alpha, +\beta, -\beta\}) = 0.2$ and $m(\{+\alpha, -\alpha\}) = 0.8$. So $S(\Delta_1) = 1$, $S(\Delta_2) = 0.8$, and $S(\Delta_3) = 0$.

Since the conflictbase is common for all minimal QC models for a knowledgebase, we obtain the following.

Proposition 6 *Let $\Delta \in \wp(\mathcal{L})$ and let m be a focal mass assignment for $\text{Delineation}(\Delta)$. If S is a significance function for m , then for all $X, Y \in \text{MQC}(\Delta)$, $S(X) = S(Y)$.*

The next two results capture notions of monotonicity for mass-based significance.

Proposition 7 *Let $\Delta \in \wp(\mathcal{L})$ and $\alpha \in \mathcal{L}$. Let m be a mass assignment for $\text{Delineation}(\Delta \cup \{\alpha\})$. If S is a significance function for m , then $S(\Delta) \leq S(\Delta \cup \{\alpha\})$.*

Proposition 8 *Let $\phi, \psi \in \mathcal{L}$. Let m be a mass assignment for $\text{Delineation}(\{\phi, \psi\})$. Let S be a significance function for m . If for all $X \in \wp(\mathcal{O})$, $X \models_s \phi$ implies $X \models_s \psi$, then $S(\{\psi\}) \leq S(\{\phi\})$.*

In this section, we have augmented the measurement of inconsistency in QC models with an evaluation of significance. In the next section, we provide a semantics.

5 Semantics for mass-based significance

A mass assignment can be regarded as transforming the four-valued semantics of QC logic into a many-valued logic.

Definition 16 *The knowledge lattice is a pair $(K, \preceq)$ where $K = \{N, T, F\} \cup B^*$, and $B^* = \{B_n \mid n \in [0, 1]\}$, and $+$ is a (commutative and associative) addition operation over truth values, and the ordering relation $\preceq$ is defined by conditions (1) to (4) below, where $x \simeq y$ denotes $x \preceq y$ and $y \preceq x$ and $x \prec y$ denotes $x \preceq y$ and $y \not\preceq x$.*

- (1) $N \prec T$ and $N \prec F$ and $T \not\preceq F$
and $F \not\preceq T$ and $T \prec B_0$ and $F \prec B_0$
- (2) for all $B_i, B_j \in B^*$, $i \leq j$ iff $B_i \preceq B_j$
- (3) for all $B_i, B_j, B_k \in B^*$, $k = i + j$ iff $B_k \simeq B_i + B_j$
- (4) for all $B_i \in B^*$, $B_i \simeq (B_i + T)$
and $B_i \simeq (B_i + F)$ and $B_i \simeq (B_i + N)$

The knowledge lattice is illustrated in Figure 1. It is a distributive lattice. The key difference between Belnap’s lattice and ours is the value *Both* has been split into a chain of truth values $B_0, \dots, B_1$. If we equate the truth values $\{N, T, F, B_0\}$ with the Belnap truth values $\{\text{Neither}, \text{True}, \text{False}, \text{Both}\}$, respectively, then Belnap’s four-valued lattice is a sublattice of the knowledge lattice.

Definition 17 *A many-valued model is a tuple $(K, \preceq, t)$ where $(K, \preceq)$ is the knowledge lattice and t is a truth assignment function from the set of atoms $\mathcal{A}$ to K .*

In the following, we restrict consideration to solo mass assignments, though this restriction can be relaxed with slightly more complex definitions.

Definition 18 *Let m be a solo mass assignment for Ω , and let $(K, \preceq, t)$ be many-valued model. We describe (X, m) and*

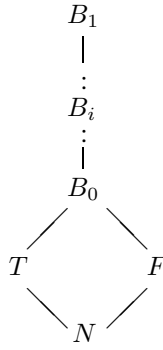


Figure 1: The Knowledge Lattice

$(K, \preceq, t)$ as being isomorphic models when $X \subseteq \Omega$ and the following conditions hold:

$$\begin{aligned} +\alpha \in X \text{ and } -\alpha \notin X &\text{ iff } t(\alpha) = T \\ +\alpha \notin X \text{ and } -\alpha \in X &\text{ iff } t(\alpha) = F \\ +\alpha \notin X \text{ and } -\alpha \notin X &\text{ iff } t(\alpha) = N \\ +\alpha, -\alpha \in X \text{ and } m(\{+\alpha, -\alpha\}) = i &\text{ iff } t(\alpha) = B_i \end{aligned}$$

We can define a valuation function that extends a truth assignment to any formula by an inductive definition on the structure of α . However, for brevity, we will adopt a shortcut based on the $\models_s$ relation.

Definition 19 Let (X, m) and $(K, \preceq, t)$ be isomorphic models. A **valuation function**, denoted V_t , from $\mathcal{L}$ to K is defined so that $V_t(\alpha) = t(\alpha)$ when α is an atom, and as follows when α is non-atomic.

$$\begin{aligned} V_t(\alpha) \simeq B_i &\text{ iff } X \models_s \alpha \text{ and } X \not\models_s \neg\alpha \\ &\text{ and } \text{Atoms}(\alpha) = \{\gamma_1, \dots, \gamma_n\} \\ &\text{ and } B_i \simeq t(\gamma_1) + \dots + t(\gamma_n) \\ V_t(\alpha) \simeq T &\text{ iff } X \models_s \alpha \text{ and } X \not\models_s \neg\alpha \\ V_t(\alpha) \simeq F &\text{ iff } X \not\models_s \alpha \text{ and } X \models_s \neg\alpha \\ V_t(\alpha) \simeq N &\text{ iff } X \not\models_s \alpha \text{ and } X \not\models_s \neg\alpha \end{aligned}$$

Since we could have defined the valuation function equivalently in an inductive definition on the structure of the formulae, we can obtain propositions such as the following that relate the valuation function applied to a formula and applied to its subformulae.

Proposition 9 If $(K, \preceq, t)$ is a many-valued model and $\alpha_1 \vee \alpha_2$ is a clause, then: (1) $V_t(\alpha_1 \vee \alpha_2) \simeq B_i$ iff $\text{Atoms}(\{\alpha_1\}) \neq \text{Atoms}(\{\alpha_2\})$ and $t(\alpha_1) = B_{i_1}$ and $t(\alpha_2) = B_{i_2}$ and $B_i \simeq (B_{i_1} + B_{i_2})$; (2) $V_t(\alpha_1 \vee \alpha_2) \simeq T$ iff $t(\alpha_1) \simeq T$ or $t(\alpha_2) \simeq T$; (3) $V_t(\alpha_1 \vee \alpha_2) \simeq F$ iff $[t(\alpha_1) \succeq F \text{ and } t(\alpha_2) \simeq F]$ or $[t(\alpha_1) \simeq F \text{ and } t(\alpha_2) \succeq F]$; and (4) $V_t(\alpha_1 \vee \alpha_2) \simeq N$ iff $[t(\alpha_1) \simeq N \text{ and } t(\alpha_2) \not\succeq T]$ or $[t(\alpha_1) \not\succeq T \text{ and } t(\alpha_2) \simeq N]$.

Example 13 Let $X = \{+\alpha, -\alpha, +\beta, -\beta, +\gamma\}$ with mass $m(\{+\alpha, -\alpha\}) = 0.2$ and $m(\{+\beta, -\beta\}) = 0.8$. Let t be a truth assignment where (X, m) and $(K, \preceq, t)$ are isomorphic models, then $t(\alpha) = B_{0.2}$, $t(\beta) = B_{0.8}$, and $t(\gamma) = T$.

$$\begin{aligned} V_t(\alpha \vee \beta) &\simeq B_{1.0} & V_t(\beta) &\simeq B_{0.8} \\ V_t(\neg\beta) &\simeq B_{0.8} & V_t(\gamma \vee \alpha) &\simeq T \\ V_t(\alpha \wedge \neg\alpha) &\simeq B_{0.2} & V_t(\alpha \wedge \beta) &\simeq B_{1.0} \\ V_t(\beta \wedge \neg\beta) &\simeq B_{0.8} & V_t(\beta \vee \neg\gamma) &\simeq F \end{aligned}$$

Example 14 Let $X = \{+\alpha, -\alpha, +\beta\}$ with mass assigned by $m(\{+\alpha, -\alpha\}) = 0.2$ and $m(\{+\beta, -\beta\}) = 0.8$. Let t be a truth assignment where (X, m) and $(K, \preceq, t)$ are isomorphic models, then $t(\alpha) = B_{0.2}$, $t(\beta) = T$, $t(\gamma) = N$.

$$\begin{aligned} V_t(\alpha \vee \beta) &\simeq T & V_t(\beta) &\simeq T \\ V_t(\neg\beta) &\simeq F & V_t(\gamma) &\simeq N \\ V_t(\alpha \wedge \neg\alpha) &\simeq B_{0.2} & V_t(\alpha \wedge \beta) &\simeq T \\ V_t(\alpha) &\simeq B_{0.2} & V_t(\alpha \vee \gamma) &\simeq N \end{aligned}$$

This semantics relates to QC logic strong satisfaction relation with the mass-based significance function as follows.

Proposition 10 Let m be a solo mass assignment for Ω . Also let $X \subseteq \Omega$ and t be a truth assignment where (X, m) and $(K, \preceq, t)$ are isomorphic models. If $\phi \in \mathcal{L}$, then $V_t(\phi) \succeq T$ iff $X \models_s \phi$.

Proposition 11 Let $\phi \in \mathcal{L}$. Let $X \in \text{MQC}(\{\phi\})$, where $\text{Coherence}(X) = 0$, and let m be a solo mass assignment for Ω . Let $(K, \preceq, t)$ be a many-valued model where $X \subseteq \Omega$ and (X, m) and $(K, \preceq, t)$ are isomorphic models. Let S be a significance function for m .

$$V_t(\phi) \simeq B_i \text{ iff } S(\{\phi\}) = i$$

The semantics therefore extends QC logic so that in case of inconsistency, the B_i truth value reflects the significance of the inconsistency, where B_0 is the least significant and B_1 is the most significant. It is straightforward to extend this semantics to focal mass assignments.

6 Significance of inconsistencies in news

In this section, we consider news reports. Having some understanding of the “degree of inconsistency” of a news report can help in deciding how to act on it. Moreover, inconsistencies between information in a news report and domain knowledge can tell us important things about the news report. For this we use a significance function to give a value for each possible inconsistency that can arise in a news report in a given domain. We may also use significance in the following ways: (1) to reject reports that are too inconsistent; (2) to highlight unexpected news; (3) to focus on repairing significant inconsistencies; and (4) to monitor sources of information to identify sources that are unreliable.

How we use the significance function depends on the application. In our simple example below, we assume a news report is a set of ground predicates. Each term is used to represent a piece of information in a news report and may be equivalent to a word, or number, or a simple phrase. The predicate symbol is used to tag or categorise each piece of information.

Example 15 Each of $\Gamma_1 - \Gamma_5$ is a news report:

$$\begin{aligned} \Gamma_1 &= \{\text{temp}(30C), \text{pptn}(\text{snow}), \text{pollen}(\text{high})\} \\ \Gamma_2 &= \{\text{temp}(30C), \text{pptn}(\text{snow}), \text{pollen}(\text{low})\} \\ \Gamma_3 &= \{\text{temp}(10C), \text{pptn}(\text{snow}), \text{pollen}(\text{high})\} \\ \Gamma_4 &= \{\text{temp}(10C), \text{pptn}(\text{snow}), \text{pollen}(\text{low})\} \\ \Gamma_5 &= \{\text{temp}(0C), \text{pptn}(\text{snow}), \text{pollen}(\text{high})\} \end{aligned}$$

A potential inconsistency that can arise in a news report is any set of literals that may be rebutted by the domain knowledge.

Definition 20 Let Φ be a news report and let Δ be domain knowledge. For $\{\phi_1, \dots, \phi_n\} \subseteq \Phi$, $\{\phi_1, \dots, \phi_n\}$ is **rebutted** by Δ iff $\Delta \vdash \neg(\phi_1 \wedge \dots \wedge \phi_n)$.

Example 16 Domain knowledge for weather reports may include:

- (clause 1) $\neg \text{temp}(30\text{C}) \vee \neg \text{pptn}(\text{snow})$
- (clause 2) $\neg \text{temp}(10\text{C}) \vee \neg \text{pptn}(\text{snow})$
- (clause 3) $\neg \text{pollen}(\text{high}) \vee \neg \text{pptn}(\text{snow})$

So with the report Γ_1 in Example 15, we get that $\{\text{temp}(30\text{C}), \text{pptn}(\text{snow})\}$ is rebutted by clause 1, and that $\{\text{pollen}(\text{high}), \text{pptn}(\text{snow})\}$ is rebutted by clause 3.

Example 17 Continuing Example 15, we adopt the mass assignment as follows:

$$\begin{aligned} m(\{+\text{temp}(30\text{C}), -\text{temp}(30\text{C}), \\ +\text{pptn}(\text{snow}), -\text{pptn}(\text{snow})\}) &= 0.6 \\ m(\{+\text{temp}(10\text{C}), -\text{temp}(10\text{C}), \\ +\text{pptn}(\text{snow}), -\text{pptn}(\text{snow})\}) &= 0.3 \\ m(\{+\text{pollen}(\text{high}), -\text{pollen}(\text{high}), \\ +\text{pptn}(\text{snow}), -\text{pptn}(\text{snow})\}) &= 0.1 \end{aligned}$$

So the reports Γ_1 to Γ_5 in Example 15, together with the domain knowledge in Example 16, denoted here by Δ , gives the following significance evaluations. $S(\Delta \cup \Gamma_1) = 0.7$, $S(\Delta \cup \Gamma_2) = 0.6$, $S(\Delta \cup \Gamma_3) = 0.4$, $S(\Delta \cup \Gamma_4) = 0.3$, $S(\Delta \cup \Gamma_5) = 0.1$. If we set the threshold of acceptability for a news report Γ_i at a significance evaluation of 0.3, then only Γ_4 and Γ_5 would be acceptable, the others would be rejected.

This small example illustrates how we may find some inconsistencies acceptable and others unacceptable, thereby select some news reports in preference to others. For larger examples, it may be necessary to specify a mass assignment for many possible incoherent models. One solution for this is to generate a mass assignment from a partially specified mass assignment. Mass is assigned to particular models and then mass for further models is obtained by interpolation. In particular, domains with some numerical data, such as sport reports or weather reports, can be addressed with this approach. For example, if we are only interested in inconsistencies about temperature, and we have a range of possibilities from $\text{rain}(1\text{cm}), \dots$, to $\text{rain}(50\text{cm})$, then the mass could be assigned to two models such as $\{+\text{rain}(1\text{cm}), -\text{rain}(1\text{cm})\}$ and $\{+\text{rain}(50\text{cm}), -\text{rain}(50\text{cm})\}$, and then the remaining mass assigned by interpolation.

7 Discussion

Whilst a number of approaches to handling inconsistent information touch on the issue of measurement of inconsistency, the topic is underdeveloped. Information theory can be used to measure the information content of sets of inconsistent formulae [Lozinskii, 1994]. This increases with additions of consistent information and decreases with additions of inconsistent information. However, it does not provide a direct measure of inconsistency since for example, the value for $\{\alpha\}$ is the same as for $\{\alpha, \neg\alpha, \beta\}$. Another approach to handling inconsistent information is that of possibility theory [Dubois et al., 1994]. Let (ϕ, α) be a weighted formula where ϕ is a

classical formula and $\alpha \in [0, 1]$. A possibilistic knowledge-base B is a set of weighted formulae. An α -cut of a possibilistic knowledgebase, denoted $B_{\geq \alpha}$, is $\{(\psi, \beta) \in B \mid \beta \geq \alpha\}$. The inconsistency degree of B , denoted $\text{Inc}(B)$, is the maximum value of α such that the α -cut is inconsistent. However, this approach does not discriminate between different inconsistencies. For this, there is a need for an underlying paraconsistent logic such as QC logic.

References

- [Belnap, 1977] N Belnap. A useful four-valued logic. In G Epstein, editor, *Modern Uses of Multiple-valued Logic*, pages 8–37. Reidel, 1977.
- [Benferhat et al., 1993] S Benferhat, D Dubois, and H Prade. Argumentative inference in uncertain and inconsistent knowledge bases. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 1449–1445. Morgan Kaufmann, 1993.
- [Besnard and Hunter, 1995] Ph Besnard and A Hunter. Quasi-classical logic: Non-trivializable classical reasoning from inconsistent information. In *Symbolic and Quantitative Approaches to Uncertainty*, volume 946 of *LNCS*, pages 44–51, 1995.
- [da Costa, 1974] N C da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15:497–510, 1974.
- [Dubois et al., 1994] D Dubois, J Lang, and H Prade. Possibilistic logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 439–513. Oxford University Press, 1994.
- [Grant, 1978] J Grant. Classifications for inconsistent theories. *Notre Dame Journal of Formal Logic*, 19:435–444, 1978.
- [Hunter, 2000] A Hunter. Reasoning with conflicting information using quasi-classical logic. *Journal of Logic and Computation*, 10:677–703, 2000.
- [Hunter, 2001] A Hunter. A semantic tableau version of first-order quasi-classical logic. In *Symbolic and Quantitative Approaches to Uncertainty*, volume 2143 of *LNCS*, pages 544–556, 2001.
- [Hunter, 2002] A Hunter. Measuring inconsistency in knowledge via quasi-classical models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'02)*, pages 68–73. MIT Press, 2002.
- [Levesque, 1984] H Levesque. A logic of implicit and explicit belief. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'84)*, pages 198–202, 1984.
- [Lozinskii, 1994] E Lozinskii. Information and evidence in logic systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 6:163–193, 1994.
- [Marquis and Porquet, 2001] P Marquis and N Porquet. Computational aspects of quasi-classical entailment. *Journal of Applied Non-classical Logics*, 11:295–312, 2001.
- [Shafer, 1976] G Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

LEARNING

LEARNING

CLUSTERING AND BAYES NET LEARNING

Data Clustering: Principal Components, Hopfield and Self-Aggregation Networks*

Chris H.Q. Ding

NERSC Division, Lawrence Berkeley National Laboratory
University of California, Berkeley, CA 94720. chqding@lbl.gov

Abstract

We present a coherent framework for data clustering. Starting with a Hopfield network, we show the solutions for several well-motivated clustering objective functions are principal components. For MinMaxCut objectives motivated for ensuring cluster balance, the solutions are the nonlinearly scaled principal components. Using scaled PCA, we generalize to multi-way clustering, constructing a self-aggregation network, where connection weights between different clusters are automatically suppressed while connection weights within same clusters are automatically enhanced.

1 Introduction

Principal component analysis (PCA) is widely adopted as an effective unsupervised dimension reduction method. PCA is extended in many different directions [Hastie and Stuetzle, 1989; Kramer, 1991; Lee and Seung, 1999; Scholkopf *et al.*, 1998; Collins *et al.*, 2001].

The main justification is that PCA uses singular value decomposition (SVD) which is the best low rank approximation in L_2 norm to original data due to Eckart-Young theorem. However, this results alone is inadequate to explain the effectiveness of PCA. Here, we provide a new derivation of PCA based on optimizing suitable clustering objective functions and show that principal components are actually cluster indicator vectors in clustering.

Hopfield network [Hopfield, 1982] provide a convenient framework of our study. In particular, the self-aggregation network proposed in this work uses Hebb rule to encode pattern vectors. One feature of Hopfield associative-memory networks is that it can be adopted to solve hard combinatorial problems [Haykin, 1998 2nd ed].

Another thread of this work is the spectral graph partitioning [Fiedler, 1973; Pothen *et al.*, 1990; Hagen and Kahng, 1992; Shi and Malik, 2000; Ding *et al.*, 2001a; 2001b; Ng *et al.*, 2001; Meila and Shi, 2001], which uses Laplacian matrix of a graph. This arises naturally for balancing the clusters (see §2.4). Our approach differs from others mainly in

well-motivated clustering objective functions. A further point is the recognition that spectral graph clustering is embedded in the scaled PCA thus leading to self-aggregation networks.

This paper combines above three threads and develop a coherent framework for clustering. We begin with two-way clustering in §2 and generalize to multi-way clustering in §3.

2 Two-way clustering

We start by formulate two-way clustering as a Hopfield network and derive PCA as cluster indicator vectors.

2.1 Hopfield Network and PCA

Given n data points and properly defined similarity or association $w_{ij} = w_{ji} \geq 0$ between points i, j , we form a network (a weighted graph) G with w_{ij} as the connection between nodes i, j . We wish to partition it into two clusters c_1, c_2 . The result of clustering can be represented by an indicator vector $\mathbf{q}$, where

$$q(i) = \begin{cases} 1 & \text{if } i \in c_1 \\ -1 & \text{if } i \in c_2 \end{cases} \quad (1)$$

Consider the clustering objective,

$$J_1 = \mathbf{q}^T W \mathbf{q} = s(c_1, c_1) + s(c_2, c_2) - 2s(c_1, c_2) \quad (2)$$

where $s(c_1, c_2) = \sum_{i \in c_1, j \in c_2} w_{ij}$ is the overlap between c_1, c_2 ; $s(c_1, c_1) = \sum_{i, j \in c_1} w_{ij}$ is the within-cluster similarity of c_1 , and analogously for $s(c_2, c_2)$.

Now we propose a *min-max clustering principle*: data points are grouped into clusters such that the overlap $s(c_1, c_2)$ between different clusters are minimized while within-cluster similarities $s(c_1, c_1)$, $s(c_2, c_2)$ are maximized

$$\min s(c_1, c_2), \quad \max s(c_1, c_1), \quad \max s(c_2, c_2) \quad (3)$$

These conditions can be simultaneously satisfied by maximizing the energy (objective function) J_1 . Using Hopfield model [Hopfield, 1982], the solution is obtained by the update rule

$$q^{(t+1)}(i) = \text{sgn} \left[\sum_j w_{ij} q^{(t)}(j) \right].$$

or in vector form $\mathbf{q}^{(t+1)} = \text{sgn}[W \mathbf{q}^{(t)}]$. If one relaxes $q(i)$ from discrete indicators to continuous values in $(-1, 1)$, the solution $\mathbf{q}$ is given by

$$W \mathbf{q} = \lambda \mathbf{q}. \quad (4)$$

*Work supported by Department of Energy (Office of Science, through a LBNL LDRD) under contract DE-AC03-76SF00098.

Since the matrix entries in W are non-negative, the first principal eigenvector $\mathbf{q}_1$ has all positive (or all negative) entries. Thus the desired solution is $\mathbf{q}_2$.

2.2 Hopfield network for bipartite graph

Here we extend the Hopfield networks for clustering bipartite graph. An example of bipartite graph is a $m \times n$ term-document association matrix $B = (b_{ij})$, where each row represents a word, each column represents a document, and b_{ij} the counts of co-occurrence of row r_i and column c_j . We show that the solution for clustering indicators is precisely the Latent Semantic Indexing [Deerwester *et al.*, 1990].

We wish to partition the r -type nodes of R into two parts R_1, R_2 and simultaneously partition the c -type nodes of C into two parts C_1, C_2 , based on the clustering principle of minimizing between-cluster association and maximizing within-cluster association. We use indicator vector $\mathbf{f}$ to determine how to split R into R_1, R_2 : $f(i) = 1$, or -1 depending on $r_i \in R_1$ or $r_i \in R_2$. We use $\mathbf{g}$ to determine how to split C into C_1, C_2 : $g(i) = 1$, or -1 depending on $c_i \in C_1$ or $c_i \in C_2$.

(For presentation purpose, we index the nodes such that nodes within same cluster are indexed contiguously. The clustering algorithms presented are independent to this assumption. Bold face lower case letters are vectors. Matrices are denoted by upper case letters.) Thus we may write $\mathbf{f} = \begin{pmatrix} \mathbf{f}^{(+)} \\ \mathbf{f}^{(-)} \end{pmatrix}$, $\mathbf{g} = \begin{pmatrix} \mathbf{g}^{(+)} \\ \mathbf{g}^{(-)} \end{pmatrix}$ With this indexing, the association matrix is

$$B = \begin{pmatrix} B_{R_1, C_1} & B_{R_1, C_2} \\ B_{R_2, C_1} & B_{R_2, C_2} \end{pmatrix} \quad (5)$$

It is convenient to convert the bipartite graph into an undirected graph. We follow standard procedure and combine the two types nodes to one by setting

$$\mathbf{q} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}, \quad W = \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix}, \quad (6)$$

This induces an undirected graph G , whose adjacency matrix is the symmetric weight matrix W .

Consider the following objective function,

$$J_2(C_1, C_2; R_1, R_2) = \frac{1}{2} \mathbf{q}^T W \mathbf{q} \quad (7)$$

$$= s(R_1, C_1) + s(R_2, C_2) - s(R_1, C_2) - s(R_2, C_1)$$

where $s(R_1, C_2) \equiv \sum_{r_i \in R_1, c_j \in C_2} b_{ij}$, the sum of association between R_1 and C_2 , and $s(R_1, C_2)$ is the sum of association between R_1 and C_2 . $s(R_1, C_2)$ and $s(R_2, C_1)$ should be minimized. $s(R_1, C_1)$ is the sum of association within cluster 1 (see Fig.1), and $s(R_2, C_2)$ is the sum of association within cluster 2. $s(R_1, C_1)$ and $s(R_2, C_2)$ should be maximized. These conditions are simultaneously satisfied by maximizing J_2 .

We can write down the Hopfield network update rule for $\mathbf{q}$. If one relaxes $q(i)$ from discrete indicators to continuous values, the solution $\mathbf{q}$ satisfies Eq.(4). Now utilizing the explicit structures of W and $\mathbf{q}$, we have

$$\begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}. \quad (8)$$

The solutions to this equation are the singular value decomposition (SVD) of B : $\{\mathbf{f}_i\}$ are left singular vectors and $\{\mathbf{g}_i\}$ are right singular vectors of the SVD of B ,

$$B = \sum_{k=1}^m \mathbf{f}_k \lambda_k \mathbf{g}_k^T = F_m \Lambda_m G_m^T. \quad (9)$$

Again, since the matrix entries in W are non-negative, the first principal components $\mathbf{u}_1, \mathbf{v}_1$ have entries of same sign; thus the desired solutions are $\mathbf{u}_2, \mathbf{v}_2$. Note that this is also the SVD employed in LSI. We summarize these results in

Theorem 1. Principal components are solutions for clustering indicators for clustering undirect graphs and bipartite graphs under appropriate Hopfield network models.

2.3 Principal component clustering

In the objective J_1 , we may explicitly enforce a balance of clusters C_1, C_2 . From Eq.1, we need to minimize $|\sum_i q(i)|$. Thus we consider the clustering objective,

$$\tilde{J}_1 = \mathbf{q}^T W \mathbf{q} - \mu \left(\sum_i q(i) \right)^2 = \mathbf{q}^T (W - \mu \mathbf{e} \mathbf{e}^T) \mathbf{q} \quad (10)$$

where $\mu > 0$ is a parameter and $\mathbf{e} = (1, \dots, 1)^T$. We adjust μ to control the level of balance between C_1, C_2 . The principal eigenvector $\mathbf{q}_1$ of $\tilde{W} = W - \mu \mathbf{e} \mathbf{e}^T$ is the desired indicator vector.

What is the reasonable choice of μ ? A natural choice is to set $\mu = w_{..}/n^2$, $w_{..} = \sum_{ij} w_{ij}$, the average of w_{ij} . With this choice, the modified weight matrix satisfies the sum-to-zero condition:

$$\sum_{ij} \tilde{w}_{ij} = \sum_{ij} (w_{ij} - w_{..}/n^2) = 0. \quad (11)$$

This condition can be further refined by centering each column and each rows, such that

$$\sum_i \bar{w}_{ij} = 0, \forall j; \quad \sum_j \bar{w}_{ij} = 0, \forall i. \quad (12)$$

where

$$\bar{w}_{ij} = w_{ij} - w_{i.}/n - w_{.j}/n + w_{..}/n^2 \quad (13)$$

(here column and row sums, $w_{i.} = \sum_j w_{ij}$, $w_{.j} = \sum_i w_{ij}$ and $w_{..}$ are standard notations in statistics.) Now the desired cluster indicator vector is the principal eigenvector of

$$\bar{W} \mathbf{q} = \lambda \mathbf{q}$$

The fully centered $\bar{W}$ has a useful property that all eigenvectors of $\bar{W}$ with nonzero eigenvalues have the sum-to-zero property: $\sum_i \bar{q}(i) = 0$. This is because (1) $\bar{\mathbf{q}}_0 = \mathbf{e}$ is an eigenvector of $\bar{W}$ with $\lambda_0 = 0$; (2) all other eigenvectors are orthogonal to $\bar{\mathbf{q}}_0$, i.e., $\bar{\mathbf{q}}^T \mathbf{e} = 0$.

The sum-to-zero condition $\sum_i \bar{q}(i) = 0$ does not necessarily imply that the sizes of the two cluster, n_1, n_2 , should be equal. In fact, the cluster indicator vector should be refined to

$$\bar{q}(i) = \begin{cases} \sqrt{n_2/n_1} & \text{if } i \in C_1 \\ -\sqrt{n_1/n_2} & \text{if } i \in C_2 \end{cases} \quad (14)$$

(in this paper, all vectors are implicitly normalized to 1 using L_2 norm). Correspondingly, the clustering objective becomes

$$\bar{J}_1 = \bar{\mathbf{q}}^T \bar{W} \bar{\mathbf{q}} = n_1 n_2 \left[\frac{s(c_1, c_1)}{n_1^2} + \frac{s(c_2, c_2)}{n_2^2} - 2 \frac{s(c_1, c_2)}{n_1 n_2} \right] \quad (15)$$

Clearly, the first two terms represent the average within-cluster similarities which are maximized, and the 3rd term represent average between-cluster similarities which are minimized. The factor $n_1 n_2$ encourages cluster balance, since $n_1 + n_2 = n$, and $\max(n_1 n_2)$ is reached when $n_1 = n_2 = n/2$. We summarize these results in

Theorem 2. Solution for clustering objective $\bar{J}_1$ is given by $\bar{\mathbf{q}}_1$; and solution for clustering objective $\bar{J}_1$ is $\bar{\mathbf{q}}_1$.

We call these clustering schemes with objective functions $J_1, \bar{J}_1, \tilde{J}_1$ as principal component clustering methods because of the use of principal components.

All above discussions apply to bipartite graphs. For example, $\bar{J}_1$ becomes

$$\bar{J}_2 = \sqrt{n_{R_1} n_{R_2} n_{C_1} n_{C_2}} \cdot \left[\frac{s(R_1, C_1)}{n_{R_1} n_{C_1}} + \frac{s(R_2, C_2)}{n_{R_2} n_{C_2}} - \frac{s(R_1, C_2)}{n_{R_1} n_{C_2}} - \frac{s(R_2, C_1)}{n_{R_2} n_{C_1}} \right] \quad (16)$$

where n_{R_1}, n_{R_2} are sizes of row clusters ($n_{R_1} + n_{R_2}$ = total row size); and n_{C_1}, n_{C_2} are sizes of column clusters ($n_{C_1} + n_{C_2}$ = total column size). Again, the first two terms represent the average within-cluster similarities which are maximized, and the last two terms represent average between-cluster similarities which are minimized.

We note that $\bar{J}_1$ is partially motivated by the classic scaling (also called principal coordinate analysis) in statistics [Borg and Gronen, 1997]. Suppose we are given pairwise distances d_{ij} for a dataset, define the pairwise similarity as $w_{ij} = -\frac{1}{2}d_{ij}^2$, follow through the same centering procedures and solve for the eigenvectors of $\bar{W}$. The classic scaling theorem proves that if d_{ij} are the Euclidean distances, the first K eigenvectors of $\bar{W}$ will recover the original coordinates. This theorem justifies the use of the K principal eigenvectors as the coordinates in multidimensional scaling. However, our clustering approach does not emphasize the recovery of original coordinates, and the objective function $\bar{J}_1$ is well-motivated for clustering only.

2.4 Cluster balance and MinMaxCut

Principal component clustering of §2.3 is motivated for balancing clusters by adding a panelity term to J_1 . Here we balance clusters through a divisive approach.

Maximizing J_1 of Eq.2 is equivalent to¹

$$\min \frac{s(c_1, c_2)}{s(c_1, c_1) + s(c_2, c_2)} \quad (17)$$

Although $s(c_1, c_1) + s(c_2, c_2)$ are maximized, it often occurs that one cluster is much larger than another, $s(c_1, c_1) \gg s(c_2, c_2)$ or vice versa.

¹Since $\exp(x)$ is monotonic, $\max J_1 \Rightarrow \max \exp(J_1)$, which is $\min \exp[s(c_1, c_2)] / \exp[s(c_1, c_1) + s(c_2, c_2)]$. This is approximately Eq.17.

To overcome this problem, we seek to prevent either $s(c_1, c_1)$ or $s(c_2, c_2)$ become very small. We optimize [Ding et al., 2001b]

$$\min J_3, \quad J_3 = \frac{s(c_1, c_2)}{s(c_1, c_1)} + \frac{s(c_1, c_2)}{s(c_2, c_2)} \quad (18)$$

One can show that

$$\min_{\mathbf{q}} J_3(\mathbf{q}) \Rightarrow \min_{\mathbf{q}} \frac{\mathbf{q}^T (D - W) \mathbf{q}}{\mathbf{q}^T D \mathbf{q}}, \quad (19)$$

subject to $\mathbf{q}^T W \mathbf{e} = \mathbf{q}^T D \mathbf{e} = 0$, where $D = (w_{ij})$ is a diagonal matrix. We relax $q(i)$ from discrete indicators to real values in $(-1, 1)$. The solution of $\mathbf{q}$ for minimizing the Rayleigh quotient of Eq.(19) is given by

$$(D - W) \mathbf{q} = \lambda D \mathbf{q}, \quad (20)$$

which can be written as

$$W \mathbf{q} = \zeta D \mathbf{q}, \quad \zeta = 1 - \lambda. \quad (21)$$

Again, the desired solution is the eigenvector associated with the second largest eigenvalue. Note that by comparing Eq.(21) with Eq.(4), the net effect for cluster balance is diagonal scaling. This diagonal scaling, however, leads to an important feature of self-aggregation (see §3.2).

2.5 Cluster balance for bipartite graphs

For balanced clustering of bipartite graphs, object function of Eq.(7) becomes

$$J_4 = \frac{s(R_1, C_2) + s(R_2, C_1)}{2s(R_1, C_1)} + \frac{s(R_1, C_2) + s(R_2, C_1)}{2s(R_2, C_2)} \quad (22)$$

Using the representation W and $\mathbf{q}$ of Eq.6, and similar derivation [Ding, 2003; Zha et al., 2001], the solution for optimization of J_4 is also given by Eq.21. Let $D_r = \text{diag}(B\mathbf{e})$ and $D_c = \text{diag}(B^T \mathbf{e})$, we have

$$D = \begin{pmatrix} D_r & 0 \\ 0 & D_c \end{pmatrix}, \quad \mathbf{z} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} D_r^{1/2} \mathbf{f} \\ D_c^{1/2} \mathbf{g} \end{pmatrix}. \quad (23)$$

Substituting into Eq.(21), we have

$$\begin{pmatrix} 0 & \hat{B} \\ \hat{B}^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \zeta \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}, \quad (24)$$

where

$$\hat{B} = D_r^{-1/2} B D_c^{-1/2}. \quad (25)$$

The solutions to Eq.(24) are SVD of $\hat{B}$: $\hat{B} = \sum_k \mathbf{u}_k \lambda_k \mathbf{v}_k^T$. The clustering indicator vectors are $\mathbf{f}_k = D_r^{-1/2} \mathbf{u}_k$ for row objects and $\mathbf{g}_k = D_c^{-1/2} \mathbf{v}_k$ for column objects. Note that Eq.(24) is identical Eq.(8), with the corresponding relationship $B \Rightarrow \hat{B}$, $\begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix} \Rightarrow \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}$. Therefore, the net effect of clustering balancing using MinMaxCut objective of Eq.(22) over the simple objective Eq.(7) is the scaling of the association matrix B in Eq.(25). However, with this scaling, the self-aggregation property emerges (see §3.4).

3 K -way clustering

The above mainly focus on 2-way clustering. Below we generalize to K -way clustering, $K \geq 2$. The generalization is based on the key observation that the solution for cluster indicator vectors (see §2.4 and §2.5) are scaled principal components, as we discuss next.

3.1 Scaled principal components

Associations among data objects are mostly quantified by a similarity metric. The scaled principal component approach starts with a nonlinear (non-uniform) scaling of W . Noting that $W = D^{1/2}(D^{-1/2}WD^{-1/2})D^{1/2}$, we apply spectral decomposition on the scaled matrix

$$\widehat{W} = D^{-1/2}WD^{-1/2}$$

instead of on W . This leads to

$$W = D^{1/2}\left(\sum_k \mathbf{z}_k \zeta_k \mathbf{z}_k^T\right)D^{1/2} = D \sum_k \mathbf{q}_k \zeta_k \mathbf{q}_k^T D \quad (26)$$

Here we call $\mathbf{q}_k = D^{-1/2}\mathbf{z}_k$ the *scaled principal components*. they are obtained by solving the eigenvalue system

$$\widehat{W}\mathbf{z} = (D^{-1/2}WD^{-1/2})\mathbf{z} = \zeta\mathbf{z}. \quad (27)$$

Note Eq.(27) is identical to Eq.(20), or Eq.(21), thus scaled principal components are cluster indicator vectors in Min-MaxCut (see §2.3).

3.2 Self-aggregation network

In Hopfield networks, a pattern $\mathbf{q}_1$ is encoded into the network as $\mathbf{q}_1\mathbf{q}_1^T$ (the Hebb rule); multiple patterns are encoded additively. In our problem, a pattern is a cluster partitioning indicator vector. We define a self-aggregation network with the connection weights

$$W_{SA} = \mathbf{q}_1\mathbf{q}_1^T + \cdots + \mathbf{q}_K\mathbf{q}_K^T = Q_K Q_K^T. \quad (28)$$

$Q_K = (\mathbf{q}_1, \dots, \mathbf{q}_K)$. Here we highlights several important properties of W_{SA} and provides several example applications. (Self-aggregation is first studied in [Ding *et al.*, 2002].)

W_{SA} has an interesting self-aggregation property enforced by within-cluster association (connectivity). To prove, we apply perturbation analysis by writing $\widehat{W} = \widehat{W}^{(0)} + \widehat{W}^{(1)}$, where $\widehat{W}^{(0)}$ is the similarity matrix when clusters are well-separated (zero-overlap) and $\widehat{W}^{(1)}$ accounts for the overlap among clusters and is treated as a perturbation [Ding *et al.*, 2001a]. This perturbation approach is standard in quantum physics [Mathews and Walker, 1971].

Well-separated clusters

First, we consider the case where clusters are well separated, i.e., no overlap (no connectivity) exist between different clusters. Let $W = (W_{pq})$, $p, q = 1, \dots, K$. be the subblock decomposition of W . No overlap is represented by the fact that off-diagonal blocks are zero: $W_{pq} = 0, p \neq q$. Now, $W - D = \text{diag}(W_{11} - D_{11}, \dots, W_{KK} - D_{KK})$. The eigenvalue problem decouples into K independent system $(W_{kk} - D_{kk})\mathbf{q} = \lambda D_{kk}\mathbf{q}$. Clearly, $\mathbf{e}_k = (1, \dots, 1)^T$ with the size of cluster G_k , is the eigenvector for G_k . Let $s_{pq} =$

$\sum_{i \in G_p} \sum_{j \in G_q} w_{ij}$, and $C = (c_{k\ell})$ be an arbitrary orthonormal matrix. Then

$$\mathbf{q}_k = (c_{1k}\mathbf{e}_1/s_{11}^{1/2}, \dots, c_{Kk}\mathbf{e}_K/s_{KK}^{1/2})^T, \quad k = 1, \dots, K, \quad (29)$$

are the K eigenvectors with eigenvalue $\zeta_k = 1$. $\mathbf{q}_k$ is a linear combination of K step functions, i.e., piece-wise constant function. Clearly, all data objects within the same cluster have *identical* elements in $\mathbf{q}$. The coordinate of object i in the K -dim SPCA space is $\mathbf{r}_i = (q_1(i), \dots, q_K(i))^T$. Thus objects within a cluster are located at (self-aggregate into) the same point in SPAC space.

Scaled principal components are not *unique*, since C could be any orthogonal matrix. However,

$$Q_K Q_K^T = \text{diag}(\mathbf{q}_1 \mathbf{q}_1^T / s_{11}, \dots, \mathbf{q}_K \mathbf{q}_K^T / s_{KK}), \quad (30)$$

is unique. Thus, self-aggregation of cluster member is equivalent to the fact that $Q_K Q_K^T$ has a block diagonal structure, where elements within the same diagonal block all have the same value.

It happens that two objects i, j within the same cluster are not initially connected, i.e., $w_{ij} = 0$. However, $(Q_K Q_K^T)_{ij}$ gets the same connection strength as any other pairs within the same cluster, i.e., SA network performs a transitive closure on each connected component. Thus, in SA network, the within-cluster connections are enhanced.

Clusters overlap

Consider the case when overlaps among different clusters exist. The overlaps are treated as a perturbation. **Theorem 3.** At the first order, the K scaled principal components and their eigenvalues have the form

$$\mathbf{q} = D^{-1/2} X_K \mathbf{y}, \quad \lambda = 1 - \zeta,$$

where $\mathbf{y}$ and ζ satisfy the eigensystem $\Gamma \mathbf{y} = \lambda \mathbf{y}$. The matrix Γ has the form $\Gamma = \Omega^{-1/2} \bar{\Gamma} \Omega^{-1/2}$, where

$$\bar{\Gamma} = \begin{pmatrix} h_{11} & -s_{12} & \cdots & -s_{1K} \\ -s_{21} & h_{22} & \cdots & -s_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ -s_{K1} & -s_{K2} & \cdots & h_{KK} \end{pmatrix} \quad (31)$$

$h_{kk} = \sum_{p, p \neq k} s_{kp}$ and $\Omega = \text{diag}(s_{11}, \dots, s_{KK})$. This analysis is accurate to order $\|\widehat{W}^{(1)}\|^2 / \|\widehat{W}^{(0)}\|^2$ for eigenvalues and to order $\|\widehat{W}^{(1)}\| / \|\widehat{W}^{(0)}\|$ for eigenvectors. $\square$

Several features of SPCA can be obtained from Theorem 3:

Corollary 3.1. SA network $Q_K Q_K^T$ has the same block diagonal form of Eq.(30), within the accuracy of Theorem 2. This assures that objects within the same cluster will self-aggregate as in Theorem 1.

Corollary 3.2. The first scaled principal component is $\mathbf{q}_1 = D^{-1/2} X_K \mathbf{y}_1 = (1, \dots, 1)^T$ with $\lambda_1 = 0, \zeta_1 = 1$. λ_1 and $\mathbf{q}_1$ are also the exact solutions to the original Eq.(20).

Corollary 3.3. When $K = 2$, the second principal component is

$$\mathbf{q}_2 = D^{-1/2} X_2 \mathbf{y}_2 = \sqrt{\frac{s_{22}}{s_{11}}} \mathbf{e}^{(1)} - \sqrt{\frac{s_{11}}{s_{22}}} \mathbf{e}^{(2)}. \quad (32)$$

Its eigenvalue is

$$\lambda_2 = \frac{s_{12}}{s_{11}} + \frac{s_{12}}{s_{22}}. \quad (33)$$

Note that this is precisely J_3 of Eq.18, the clustering objective we started with: our clustering framework is consistent.

Example 1. A dataset of 3 clusters with substantial random overlap between the clusters. All edge weights are 1. The similarity matrix and results are shown in Fig.1, where nonzero matrix elements are shown as dots. The exact λ_2 and approximate $\tilde{\lambda}_2$ from Theorem 3 are close:

$$\lambda_2 = 0.300, \tilde{\lambda}_2 = 0.268.$$

W_{SA} is much sharper than the original weight matrix W clearly due to self-aggregation: connections between different clusters are substantially suppressed while connections within same clusters are substantially enhanced.

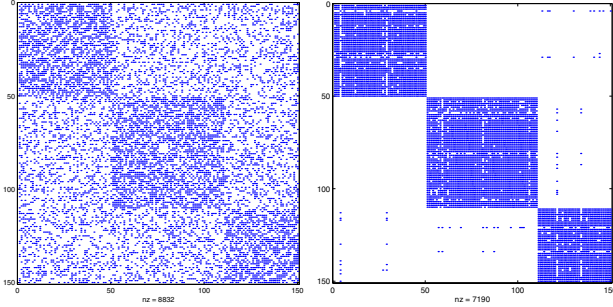


Figure 1: Left: similarity matrix W . Diagonal blocks represent weights inside clusters and off-diagonal blocks represent overlaps between clusters. Right: Computed W_{SA} .

Application 1. In DNA micro-array gene expression profiling, responses of thousands of genes from tumor tissues are simultaneously measured. We apply SPCA framework to gene expression profiles of lymphoma cancer samples [Alizadeh *et al.*, 2000]. Three cancer and three normal subtypes are shown in Fig.2. This is a difficult case due to the large variations of cluster sizes (the number of samples in each subtype are shown in parentheses in Fig.2B). Self-aggregation is evident in Figure 2B and 2C. The computed clusters correspond quite well to the normal and cancer subtypes identified by human experts.

3.3 Dynamic aggregation

The self-aggregation process can be repeated to obtain sharper clusters. W_K is the low-dimensional projection that contains the essential cluster structure. Combining this structure with the original similarity matrix, we obtain a new similarity matrix containing sharpened cluster information:

$$W^{(t+1)} = (1 - \alpha)W_K^{(t)} + \alpha W^{(t)}, \quad (34)$$

where $W_K^{(t)} = DW_{SA}D$ for $W^{(t)}$, the weight matrix at t -th iteration. Setting $\lambda_k = 1$ is crucial for enforcing the cluster structure. $W^{(1)} = W$ and $\alpha = 0.5$.

Applying SA net on $W^{(2)}$ leads to further aggregation (see Figure 2C). The eigenvalues of the 1st and 2nd SA net are shown in the insert in Figure 1C. As iteration proceeds, a clear gap is developed, indicating that clusters becoming more separated.

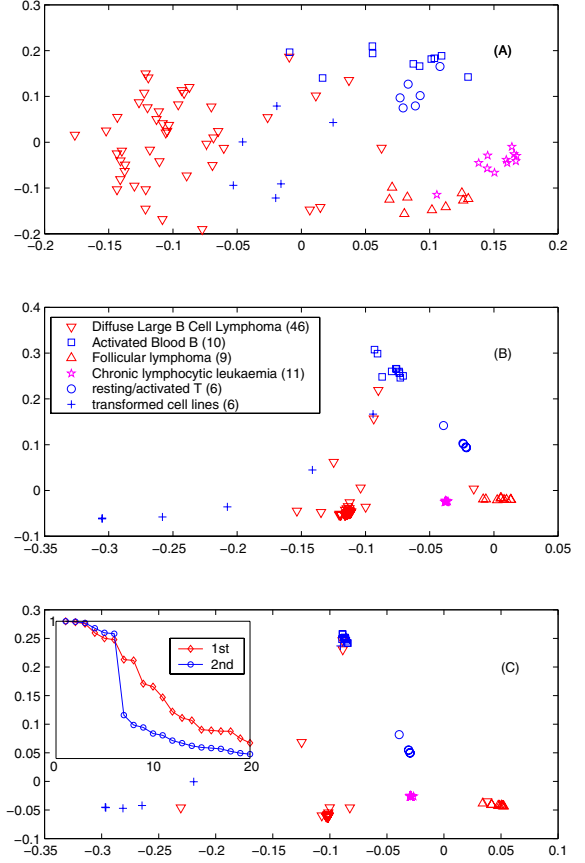


Figure 2: Gene expression profiles of cancerous and normal lymphoma tissues samples from Alizadeh *et al.* in original Euclidean space (A), in scaled PCA space (B), and in scaled PCA space after one iteration of Eq.34 (C). In all 3 panels, objects in original space are shown in 2D-view spanned by the first two PCA components. Cluster structures become clearer due to self-aggregation. The insert in (C) shows the eigenvalues of the 1st and 2nd SA network.

Noise reduction

SA net has noises. For example, W_{SA} has sometimes negative weights $(W_{SA})_{ij}$ whereas we expect them to be nonnegative. However, by Corollaries 2.1 and 3.1, W_{SA} has a diagonal block structure and every elements in the block are identical (Eq.30) even when overlaps exist. This property allows us to interpret $W_{SA} = QQ^T$ as the probability that two objects i, j belong to the same cluster:

$$p_{ij} = (W_{SA})_{ij} / (W_{SA})_{ii}^{1/2} (W_{SA})_{jj}^{1/2}.$$

To reduce noise in the dynamic aggregation, we set

$$(W_K)_{ij} = 0 \quad \text{if} \quad p_{ij} < \beta, \quad (35)$$

where $0 < \beta < 1$ and we chose $\beta = 0.8$. Noise reduction is an integral part of SA net. In our experiments, final results are insensitive to α, β . The above dynamic aggregation repeats self-aggregation process and forces data objects move towards the attractors, which are the desired clusters and their principal eigenvalues approach 1 (see insert in Fig.2C). Usually, after one or two iterations the cluster structure becomes evident.

3.4 Bipartite graphs

SPCA applies to bipartite graph problems as well. The non-linear scaling factors are D_r and D_c (cf. Eq.23). Let $B = D_r^{1/2}(\hat{B})D_c^{1/2}$, where $\hat{B}$ is given in Eq.(25). Applying SVD on $\hat{B}$, we obtain

$$B = D_r^{1/2} \left(\sum_k \mathbf{u}_k \zeta_k \mathbf{v}_k^T \right) D_c^{1/2} = D_r \sum_k \mathbf{f}_k \zeta_k \mathbf{g}_k^T D_c. \quad (36)$$

Scaled principal components are $\mathbf{f}_k = D_r^{-1/2} \mathbf{u}_k$ for row objects and $\mathbf{g}_k = D_c^{-1/2} \mathbf{v}_k$ for column objects. They have the same self-aggregation and related properties. We note that SPCA on bipartite graph leads to correspondence analysis [Greenacre, 1984] from a new perspective.

The structure of self-aggregation network $Q_K Q_K^T$ can be meaningfully further decomposed:

$$Q_K Q_K^T = \begin{pmatrix} F_K F_K^T & F_K G_K^T \\ G_K F_K^T & G_K G_K^T \end{pmatrix}.$$

where $F_K = (\mathbf{f}_1, \dots, \mathbf{f}_K)$ and $G_K = (\mathbf{g}_1, \dots, \mathbf{g}_K)$. $F_K F_K^T$ provides the cluster structure for row objects, while $G_K G_K^T$ provides the cluster structure for column objects. The off-diagonal block matrix $F_K G_K^T$ provides the sharpened association between row and column objects [Ding, 2003].

4 Discussions

In this paper, we present a data clustering framework based on properly motivated Hopfield network, and show the clustering indicators are principal components. Further motivated by cluster balance, we extend the framework to MinMaxCut that utilized the Laplacian matrix of a graph. The framework is generalized to multi-way clustering, by using scaled principal components, and self-aggregation networks are constructed. We prove the cluster member self-aggregation property of the network. This framework extends naturally to bipartite graphs which leads to row-row, column-column and row-column SA nets that simultaneously cluster the row and column objects.

In self-aggregation, data objects move towards each other guided by connectivity. This is similar to the self-organizing map [Kohonen, 1989], where feature vectors self-organize into a 2D feature map while data objects remain fixed. All these have a connection to recurrent networks [Hopfield, 1982; Haykin, 1998 2nd ed]. In Hopfield network, features are stored as associative memories. In more complicated networks, connection weights are dynamically adjusted to learn or discover the patterns. The self-aggregation network provides a new mechanism to realize this unsupervised learning.

References

[Alizadeh et al., 2000] A.A. Alizadeh, M.B. Eisen, et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000.

[Borg and Gronen, 1997] I. Borg and P. Gronen. *Modern multidimensional scaling: theory and applications*. Springer Verlag, 1997.

[Collins et al., 2001] M. Collins, S. Dasgupta, and R. Schapire. A generalization of principal component analysis to the exponential family. *Neural Info. Processing Systems (NIPS 2001)*, 2001.

[Deerwester et al., 1990] S. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman. Indexing by latent semantic analysis. *J. Amer. Soc. Info. Sci.*, 41:391–407, 1990.

[Ding et al., 2001a] C. Ding, X. He, and H. Zha. A spectral method to separate disconnected and nearly-disconnected web graph components. In *Proc. ACM Int'l Conf Knowledge Disc. Data Mining (KDD)*, pages 275–280, 2001.

[Ding et al., 2001b] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut algorithm for graph partitioning and data clustering. *Proc. IEEE Int'l Conf. Data Mining*, pages 107–114, 2001.

[Ding et al., 2002] C. Ding, X. He, H. Zha, and H. Simon. Unsupervised learning: self-aggregation in scaled principal component space. *Proc. 6th European Conf. Principles of Data Mining and Knowledge Discovery (PDKK 2002)*, pages 112–124, 2002.

[Ding, 2003] C. Ding. Document retrieval and clustering: from principal component analysis to self-aggregation networks. *Int'l Workshop on AI and Statistics*, pages 78–85, 2003.

[Fiedler, 1973] M. Fiedler. Algebraic connectivity of graphs. *Czech. Math. J.*, 23:298–305, 1973.

[Greenacre, 1984] M. J. Greenacre. *Theory and Applications of Correspondence Analysis*. Academic press, 1984.

[Hagen and Kahng, 1992] L. Hagen and A.B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE. Trans. on Computed Aided Design*, 11:1074–1085, 1992.

[Hastie and Stuetzle, 1989] T. Hastie and W. Stuetzle. Principal curves. *J. Amer. Stat. Assoc.*, 84:502–516, 1989.

[Haykin, 1998 2nd ed] S.S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998, 2nd ed.

[Hopfield, 1982] J.J. Hopfield. Neural networks and physical systems with emergent collective computation abilities. *Proc. Nat'l Acad Sci USA*, 79:2554–2558, 1982.

[Kohonen, 1989] T. Kohonen. *Self-organization and Associative Memory*. Springer-Verlag, 1989.

[Kramer, 1991] M.A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AICHE Journal*, 37:233–243, 1991.

[Lee and Seung, 1999] D. D. Lee and H. S. Seung. Learning the parts of objects with nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

[Mathews and Walker, 1971] J. Mathews and R.L. Walker. *Mathematical Methods of Physics*. Addison-Wesley, 1971.

[Meila and Shi, 2001] M. Meila and J. Shi. A random walks view of spectral segmentation. *AI-statistics Workshop*, 2001.

[Ng et al., 2001] A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Proc. Neural Info. Processing Systems (NIPS 2001)*, 2001.

[Pothen et al., 1990] A. Pothen, H. D. Simon, and K. P. Liou. Partitioning sparse matrices with eigenvectors of graph. *SIAM Journal of Matrix Anal. Appl.*, 11:430–452, 1990.

[Scholkopf et al., 1998] B. Scholkopf, A. Smola, and K. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[Shi and Malik, 2000] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE. Trans. on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.

[Zha et al., 2001] H. Zha, X. He, C. Ding, M. Gu, and H.D. Simon. Bipartite graph partitioning and data clustering. *Proc. 10th Int'l Conf. Information and Knowledge Management (CIKM 2001)*, pages 25–31, 2001.

Distributed Clustering Based on Sampling Local Density Estimates

Matthias Klusch

Deduction and Multiagent Systems
German Research Centre
for Artificial Intelligence
Stuhlsatzenhausweg 3
66123 Saarbruecken, Germany
klusch@dfki.de

Stefano Lodi

Department of Electronics,
Computer Science and Systems
University of Bologna
Viale Risorgimento 2
I-40136 Bologna BO, Italy
slodi@deis.unibo.it

Gianluca Moro

Department of Electronics,
Computer Science and Systems
University of Bologna
Via Rasi e Spinelli, 176
I-47023 Cesena FC, Italy
gmoro@deis.unibo.it

Abstract

Huge amounts of data are stored in autonomous, geographically distributed sources. The discovery of previously unknown, implicit and valuable knowledge is a key aspect of the exploitation of such sources. In recent years several approaches to knowledge discovery and data mining, and in particular to clustering, have been developed, but only a few of them are designed for distributed data sources. We propose a novel distributed clustering algorithm based on non-parametric kernel density estimation, which takes into account the issues of privacy and communication costs that arise in a distributed environment.

1 Introduction

Knowledge discovery is a process aiming at the extraction of previously unknown and implicit knowledge out of large databases, which may potentially be of added value for some given application [Fayyad *et al.*, 1996].

Data mining, which is devoted to the automated extraction of unknown patterns from given data, is a central element among the steps of the overall knowledge discovery process; the steps include preparation of the data to be analyzed as well as evaluation and visualization of the discovered knowledge. The large variety of data mining techniques which have been developed over the past decade include: methods for pattern-based similarity search, cluster analysis, decision-tree based classification, generalization taking the data cube or attribute-oriented induction approach, and mining of association rules [Chen *et al.*, 1996].

The increasing demand to scale up to massive data sets which are inherently distributed over networks with limited bandwidth and computational resources has led to methods for parallel and distributed knowledge discovery [Kargupta *et al.*, 2000]. The related pattern extraction problem in distributed knowledge discovery is referred to as *distributed data mining*. Distributed data mining is expected to perform partial analysis of data at individual sites and then to send the outcome as partial result to other sites where it is sometimes aggregated to the global result.

One of the most common approaches of business applications to perform distributed data mining is to centralize dis-

tributed data into a *data warehouse* on which to apply the usual data mining techniques. Data warehousing is a popular technology which integrates data from multiple data sources into a single repository in order to efficiently execute complex analysis queries [Moro and Sartori, 2001]. However, despite its commercial success, this approach may be impractical or even impossible for some business settings, for instance:

- when huge amounts of data are (frequently) produced at different sites and the cost for their centralization cannot scale in terms of communication, storage and computation;
- whenever data owners cannot or do not want to release information, for instance to protect privacy or because disclosing such information may result in a competitive advantage or a considerable commercial added value.

One of the most studied data mining techniques in centralized environments is *data clustering*. The goal of this technique is to decompose or partition a data set into groups such that both intra-group similarity and inter-group dissimilarity are maximized. Despite the success of data clustering in centralized environments, only a few approaches to the problem in a distributed environment are available to date.

In this work we present KDEC, a novel approach to distributed data clustering based on sampling density estimates. In KDEC each data source transmits an estimate of the probability density function of its local data to a helper site, and then executes a density based clustering algorithm that is driven by the overall density estimate, which is built by the helper from the samples of the local densities.

The paper is organized as follows. In Section 2 we describe related work and highlight differences with respect to our approach. Section 3 and 4 present the KDEC scheme to distributed data clustering. Finally, Section 5 concludes the paper and outlines ongoing and future research work.

2 Related work

In [Johnson and Kargupta, 1999] a tree clustering approach is taken to build a global dendrogram from individual dendrograms that are computed at local data sites subject to a given set of requirements. In contrast to the approach presented in this paper, the distributed data sets are assumed to be heterogeneous, therefore every site has access only to a subset of the features of an object. The proposed solution implements

a distributed version of the single-link clustering algorithm which generates clusters that are substantially different from the ones generated by density-based methods. In particular, it suffers from the so-called chaining effect, by which any of two well separated and internally homogeneous groups of objects connected only by a dense sequence of objects are regarded as a single cluster. [Kargupta *et al.*, 2001] proposes a technique for distributed principal component analysis, Collective PCA. It is shown that the technique satisfies efficiency and data security requirements and can be integrated with existing clustering methods in order to cluster distributed, high-dimensional heterogeneous data. Since the dimensionality of the data is reduced prior to clustering by applying PCA, the approach is orthogonal to ours. Another related research direction deals with incremental clustering algorithms. The BIRCH [Zhang *et al.*, 1996] and related BUBBLE method [Ganti *et al.*, 1999], compute the most accurate clustering, given the amount of memory available, while minimizing the number of I/O operations. It uses a dynamic index structure of nodes that store synthetic, constant-time maintainable summaries of sets of data objects. The method is sufficiently scalable requiring $O(N \log N)$ time and linear I/O. However, since it uses the centroid to incrementally aggregate objects, the method exhibits a strong bias towards globular clusters. IncrementalDBSCAN [Ester *et al.*, 1998] is a dynamic clustering method supporting both insertions and deletions, which is shown to be equivalent to the well-known static DBSCAN algorithm. Since in turn DBSCAN can be shown to be equivalent to a method based on density estimation when the kernel function is the square pulse and the clusters are density-based, IncrementalDBSCAN is less general than methods based on kernel density estimates. Its time complexity is $O(N \log N)$.

3 Data Clustering

3.1 The cluster analysis problem

Cluster analysis is a descriptive data mining task which aims at decomposing or partitioning a usually multivariate data set into groups such that the data objects in one group are similar to each other and are different as possible from those in other groups. Therefore, a clustering algorithm $\mathcal{A}(\cdot)$ is a mapping from any data set S of objects to a *clustering* of S , that is, a collection of pairwise disjoint subsets of S . Clustering techniques inherently hinge on the notion of distance between data objects to be grouped, and all we need to know is the set of interobject distances but not the values of any of the data object variables. Several techniques for data clustering are available but must be matched by the developer to the objectives of the considered clustering task [Grabmeier and Rudolph, 2002]. In partition-based clustering, for example, the task is to partition a given data set into multiple disjoint sets of data objects such that the objects within each set are as homogeneous as possible. Homogeneity here is captured by an appropriate cluster scoring function. Another option is based on the intuition that homogeneity is expected to be high in densely populated regions of the given data set. Consequently, searching for clusters may be reduced to searching for dense regions of the data space which are more likely to be populated by data objects. That leads us to the approach

of density estimation based clustering.

3.2 Density estimation based clustering

In *density estimation* (DE) based clustering the search for densely populated regions is accomplished by estimating a so-called probability density function from which the given data set is assumed to have arisen. Many techniques for DE-based clustering are available from the vast KDD literature [Ankerst *et al.*, 1999; Ester *et al.*, 1996; Schikuta, 1996; Hinneburg and Keim, 1998] and statistics [Silverman, 1986]. In both areas, the proposed clustering methods require the computation of a non-parametric estimation of the density function from the data. One important family of non-parametric estimates is known as *kernel estimators*. The idea is to estimate a density function by defining the density at any data object as being proportional to a weighted sum of all objects in the data set, where the weights are defined by an appropriately chosen kernel function. In the following we introduce kernel-based density estimation [Parzen, 1962; Silverman, 1986] and our approach to density estimation based clustering.

Let us assume a set $S = \{\vec{x}_i \mid i = 1, \dots, N\} \subseteq \mathbb{R}^n$ of data points or objects. Kernel estimators originate from the intuition that the higher the number of neighbouring data objects $\vec{x}_i$ of some given object $\vec{x} \in \mathbb{R}^n$, the higher the density at this object $\vec{x}$. However, there can be many ways of capturing and weighting the influence of data objects. When given the distance between one data object $\vec{x}$ and another $\vec{x}_i$ as an argument, the influence of $\vec{x}_i$ may be quantified by using a so called kernel function. A *kernel function* $K(x)$ is a real-valued, non-negative function on $\mathbb{R}$ which has finite integral over $\mathbb{R}$. When computing a kernel-based density estimation of the data set S , any element $\vec{x}_i$ in S is regarded as to exert more influence on some $\vec{x} \in \mathbb{R}^n$ than elements which are farther from $\vec{x}$ than the element. Accordingly, kernel functions are often non-increasing with $|x|$. Prominent examples of kernel functions are the square pulse function $\frac{1}{4}(\text{sign}(x+1) - \text{sign}(x-1))$, and the Gaussian function $\frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}x^2)$.¹

A *kernel-based density estimate* $\hat{\phi}_{K,h}[S](\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}_+$ is defined, modulo a normalization factor, as the sum over all data objects $\vec{x}_i$ in S of the distances between $\vec{x}_i$ and $\vec{x}$, scaled by a factor h , called *window width*, and weighted by the kernel function K :

$$\hat{\phi}_{K,h}[S](\vec{x}) = \sum_{i=1}^N K\left(\frac{d(\vec{x}, \vec{x}_i)}{h}\right). \quad (1)$$

The influence of data objects and the smoothness of the estimate is controlled by both the window width h and the shape of kernel K : h controls the smoothness of the estimate, whereas K determines the decay of the influence of a data object according to the distance. Even if the number N of data objects is very large, in practice it is not necessary to compute N distances for calculating the kernel density estimate at a given object $\vec{x}$. In fact, the value of commonly used kernel functions is negligible for distances larger than a few h units;

¹Where $\forall x \in \mathbb{R} \setminus \{0\} : \text{sign}(x) = x/|x|$, and $\text{sign}(0) = 0$.

it may even be zero if the kernel has bounded support, as it is the case, for example, for the square pulse. Using kernel-based density estimation, it is straightforward to decompose the clustering problem into three phases as follows.

1. Choose a window width h and a kernel function K .
2. Compute the kernel-based density estimate $\hat{\phi}_{K,h}[S](\vec{x})$ from the given data set.
3. Detect regions of the data space where the value of the estimate is high and group all data objects of space regions into corresponding clusters.

In the literature, many different definitions of cluster have been proposed formalizing the clusters referred to in step 3 above. A *density-based* cluster [Ester *et al.*, 1996] collects all data objects included in a region where density exceeds a threshold. *Center-defined* clusters [Hinneburg and Keim, 1998] are based on the idea that every local maximum of $\hat{\phi}$ corresponds to a cluster including all data objects which can be connected to the maximum by a continuous, uphill path in the graph of $\hat{\phi}$. Finally, an *arbitrary-shape* cluster [Hinneburg and Keim, 1998] is the union of center-defined clusters having their maxima connected by a continuous path whose density exceeds a threshold.

Algorithm 1 (DE-cluster) implements the computation of center-defined clusters by a climbing procedure driven by the density estimate. The main procedure is *DECluster*, taking as inputs an instance S of the class of data objects, the kernel function K , the window width h , and returning a clustering represented by C , which stores a mapping from each $\vec{x}_i$ to the unique integer label of $\vec{x}_i$'s cluster. It is assumed that S is an instance of a class which provides the following methods: *get(i)* to access object $\vec{x}_i$ given index i , *NQ(k, $\vec{x}$)* and *Radius(k, $\vec{x}$)* to retrieve, given $\vec{x} \in \mathbb{R}$, the indexes and maximum distance of $\vec{x}$'s k nearest neighbours. *Uphill* computes the steepest direction on the graph of the estimated density as the versor of its gradient, computed by function *DEGradient* (cf. [Hinneburg and Keim, 1998]). *Uphill* then moves in that direction a fraction $\delta < 1$ of the distance $S.Radius(k, \vec{x})$ of the k th nearest neighbour of $\vec{x}$, and finally returns the index of the nearest neighbour in S of the reached position. Every nested call to *FixedPoint* marks the current object $\vec{x}_i$ as visited and calls *Uphill* to get the index of the next data object $\vec{x}_j$. If such object has already been visited, the proximity of a local maximum has been reached and j is taken as new cluster label. Otherwise $\vec{x}_j$ is inductively assumed to lie at the bottom end of a path leading to the proximity of a local maximum, and to be already labeled accordingly. (If $\vec{x}_j$ is not marked as clustered, a recursive call ensures that the assumption holds.)

The complexity of the DE-cluster algorithm is that of calling $N = S.count$ times *FixedPoint*. At the beginning of every iteration in *DECluster*, the sets of clustered and visited objects are equal. *FixedPoint* is never called with a clustered object as argument, and visits unclustered objects at most once. Therefore, even if the number of visited data objects in one call of *FixedPoint* is bounded only by N , the number of visited data objects in all calls is still only N . For each visited object a single k -nearest neighbour query suffices to compute the gradient and the next uphill object. The methods *NQ($\cdot, \cdot$)* and *Radius($\cdot, \cdot$)* can be efficiently implemented by

Algorithm 1 DE-cluster: Clustering based on density estimation

```

func DEGradient( $\vec{x}, S, K(), h$ )  $\equiv$ 
  foreach  $i \in S.NQ(k, \vec{x})$  do
     $g := g + (\vec{x}_i - \vec{x}) * K(d(\vec{x}, \vec{x}_i)/h)$  od
   $g$ .
func Uphill( $i, S, K(), h$ )  $\equiv$ 
   $\vec{x} := S.get(i)$ 
   $\vec{v} := DEGradient(\vec{x}, S, K, h) / \|DEGradient(\vec{x}, S, K, h)\|$ 
   $S.NQ(1, \vec{x} + \delta * S.Radius(k, \vec{x}) * \vec{v})$ .
proc FixedPoint( $i, S, K(), h, C$ )
   $C.setVisited(i)$ 
   $j := Uphill(i, S, K, h)$ 
  if  $C.clustered(j)$ 
    then  $C.setLabel(i, C.getLabel(j))$ 
    else if  $C.visited(j)$  then  $C.setLabel(i, j)$ 
    else FixedPoint( $j, S, K, h, C$ )
     $C.setLabel(i, C.getLabel(j))$ 
  fi
   $C.setClustered(i)$ .
proc DECluster( $S, K(), h, k, C$ )  $\equiv$ 
  for  $i := 1$  to  $S.count$  do
    if  $\neg C.clustered(i)$  then FixedPoint( $i, S, K, h, C$ ) fi
  od.

```

equipping the class of S with a spatial access method like the KD-, or MVP-, or M-tree. Therefore, the time complexity of *DECluster* is $O(Nq(N))$, where $q(N)$ is the cost of a k nearest neighbour query in any such access method. Note that in many practical cases, $q(N)$ is very close to $\log N$.

4 Distributed Data Clustering

The body of work on applications of data clustering in distributed environments, the problem of so called *distributed data clustering* (DDC), is comparatively small. In this section we adopt the kernel density estimation based clustering approach presented above for the distributed case assuming homogeneous data, which means that a data object cannot be split across two sites.

4.1 The DDC Problem

We define the problem of *homogeneous distributed data clustering* for a clustering algorithm $\mathcal{A}$ as follows. Let $S = \{\vec{x}_i \mid i = 1, \dots, N\} \subseteq \mathbb{R}^n$ be a data set of objects. Let $L_j, j = 1, \dots, M$, be a finite set of *sites*. Each site L_j stores one data set D_j , and it will be assumed that $S = \bigcup_{j=1}^M D_j$. The DDC problem is to find a site clustering C_j residing in the data space of L_j , for $j = 1, \dots, M$, such that

- (i). $C_j = \{C \cap D_j : C \in \mathcal{A}(S)\}$ (*correctness requirement*)
- (ii). Time and communications costs are minimized (*efficiency requirement*)
- (iii). At the end of the computation, the size of the subset of S which has been transferred out of the data space of any site L_j is minimized (*privacy requirement*).

The traditional solution to the homogeneous distributed data clustering problem is to simply collect all the distributed data sets D_j into one centralized repository where their union S is computed, and the clustering C of the union S is computed and transmitted to the sites. Such approach, however, does not satisfy our problem's requirements both in terms of privacy and efficiency. We therefore propose a different approach yielding a kernel density estimation based clustering scheme, called KDEC.

4.2 The KDEC Scheme for DDC

The key idea of the KDEC scheme is based on the following observation: Although the density estimate computed on each local data set gives information on the distribution of the objects in the data set, it conceals the objects themselves. Moreover, the local density estimate can be coded to provide a more compact representation of the data set for the purpose of transmission. In the sequel, we tacitly assume that all sites L_j agree on using a global kernel function K and a global window width h . We will therefore omit K and h from our notation, and write $\hat{\phi}[S](\vec{x})$ for $\hat{\phi}_{K,h}[S](\vec{x})$. Density estimates in the form of Equation (1) are additive, i.e. the global density estimate $\hat{\phi}[S](\vec{x})$ can be decomposed into the sum of the site density estimates, one estimate for every data set D_j :

$$\hat{\phi}[S](\vec{x}) = \sum_{j=1}^M \sum_{\vec{x}_i \in D_j} K\left(\frac{d(\vec{x}, \vec{x}_i)}{h}\right) = \sum_{j=1}^M \hat{\phi}[D_j](\vec{x}). \quad (2)$$

Thus, the local density estimates can be transmitted to and summed up at a distinguished *helper site* yielding the global estimate which can be returned to all sites. Each site L_j may then apply, in its local data space, the hill-climbing technique of Algorithm 1 (DE-cluster) to assign clusters to the local data objects. There is nevertheless a weakness in such a plan: the definition of a density estimate explicitly refers to all the data objects $\vec{x}_i$. Hence, knowing how to manipulate the estimate entails knowing the data objects, which contradicts the privacy requirement. However, only an intensional, algebraic definition of the estimate includes knowledge of the data objects. Multidimensional sampling theory provides the basis for an alternative extensional representation of the estimate which makes no explicit reference to the data objects.

The theoretical idea of sampling is to represent a function f by a *sampling series*, that is, a summation of suitable expansion functions weighted by the values of f at a discrete subset of its domain [Higgins, 1996]. In the following, let the i th coordinate of $\vec{x} \in \mathbb{R}^n$ be denoted by $\vec{x}^{(i)}$, and let $\text{Diag}[\vec{u}]$, $\vec{u} = [u^{(1)}, \dots, u^{(n)}]^T \in \mathbb{R}^n$, denote the $n \times n$ diagonal matrix having diagonal $\vec{u}$, i.e., defined by $\text{Diag}[\vec{u}]_{ij} = 0$ if $i \neq j$, $\text{Diag}[\vec{u}]_{ii} = u^{(i)}$. Further let $\vec{\tau} = [\tau^{(1)}, \dots, \tau^{(n)}]^T \in \mathbb{R}^n$ a vector of *sampling periods*. The *sampled form* of $\hat{\phi}[S](\vec{x})$ at intervals $\vec{\tau}$ is the sequence $\{\hat{\phi}_{\vec{z}}[S]\}$, $\vec{z} \in \mathbb{Z}^n$, defined by

$$\hat{\phi}_{\vec{z}}[S] = \hat{\phi}[S](\text{Diag}[\vec{z}] \cdot \vec{\tau}), \quad (3)$$

where $\cdot$ is the inner product between vectors. Therefore, $\hat{\phi}_{\vec{z}}[S]$ is the sequence of the values of $\hat{\phi}[S](\vec{x})$ at all the real, n -dimensional vectors whose i th coordinates are spaced by a multiple of the i th sampling period $\tau^{(i)}$, $i = 1, \dots, n$. The sampled forms of the local density estimates are defined in a similar way by $\hat{\phi}_{\vec{z}}[D_j] = \hat{\phi}[D_j](\text{Diag}[\vec{z}] \cdot \vec{\tau})$, for $j = 1, \dots, M$. It

is immediate to see by (2) that additivity holds for the sampled forms:

$$\hat{\phi}_{\vec{z}}[S] = \sum_{j=1}^M \hat{\phi}_{\vec{z}}[D_j] \quad j = 1, \dots, M. \quad (4)$$

Therefore, after receiving the sampled forms $\hat{\phi}_{\vec{z}}[D_j]$ of the M density estimates, the helper site can compute by (4) the sampled form of the overall estimate and transmit it to the sites L_j . Sites L_j can then cluster local data with respect to the overall density estimate, using the gradient of the sampling series

$$\sum_{\vec{z} \in \mathbb{Z}^n} \hat{\phi}_{\vec{z}}[S] \text{Sinc}(\text{Diag}[\vec{\tau}]^{-1} \cdot (\vec{x} - \text{Diag}[\vec{z}] \cdot \vec{\tau})), \quad (5)$$

where $\text{Sinc}(\vec{x}) = \prod_{i=1}^n \text{sinc}(x^{(i)})$, and

$$\text{sinc}(x) = \begin{cases} 1 & \text{if } x = 0, \\ \frac{\sin \pi x}{\pi x} & \text{otherwise,} \end{cases}$$

as needed in the hill-climbing function.

We briefly discuss the extent to which the series (5) can be used to represent $\hat{\phi}[S](\vec{x})$. It is well known that, under mild conditions, sampling a function $g(\vec{x})$ is an invertible transformation if, for every coordinate $i = 1, \dots, n$, there is a frequency $f^{(i)}$ such that the Fourier transform of g differs from zero only in the interval $[-f^{(i)}, f^{(i)}]$, and the samples are computed with a period not greater than $\tau^{(i)} = 1/(2f^{(i)})$ (cf. [Higgins, 1996]). Under these assumptions, the value of the sampling series computed at $\vec{x}$ equals $g(\vec{x})$. Unfortunately, most popular kernel functions (hence summations of kernel functions) do not satisfy these hypotheses since the support of their Fourier transform is unbounded. Consequently sampling density estimates yields an information loss. However, it can be shown that the Fourier transform of a kernel density estimate is negligible everywhere except for $|f^{(i)}|$ not greater than $1/h$. Therefore, the global density estimate can be reconstructed from its samples by (5) introducing only a small error if $\tau^{(i)} < h/2$.

It is worth noting that the infinite series (5) need not be approximated, if it has finitely many nonzero terms. The latter case holds if the used kernel function has a bounded support, since the density estimate will also have bounded support. If, however, the kernel function has unbounded support, like the Gaussian kernel, then the density estimate can be approximated by regarding its value to be zero everywhere except inside an appropriately chosen bounded region.

According to this approach, we propose the following algorithmic KDEC scheme for computing the kernel density estimation based clusters for local data spaces at M distributed data sites L_j (see Algorithm 2). Every local site runs the procedure *SiteDECluster* whereas the helper site runs *Helper*. *SiteDECluster* is passed a reference H to the helper and the local data set D , and returns a clustering in a class instance C . *Helper* is passed a list of references L to the local sites. The procedure *SiteNegotiate* carries out a negotiation with the other local sites through *HelperNegotiate* at the helper site to determine the sampling periods $\vec{\tau}$, the bounding corners of the sampling rectangle $\vec{z}_1, \vec{z}_2 \in \mathbb{Z}^n$, the kernel K , and

Algorithm 2 KDEC: distributed clustering based on density estimation

```

funct Sample( $D, \vec{\tau}, \vec{z}_1, \vec{z}_2, K(\cdot), h$ )  $\equiv$ 
  s.setCorners( $\vec{z}_1, \vec{z}_2$ )
  foreach  $\vec{z} \in \text{Rectangle}(s.\text{getCorners}())$ 
    do s.Add( $\vec{z}, DE(\text{Diag}[\vec{z}] \cdot \vec{\tau}, D, K, h)$ ) od
  s.
funct Uphill( $\vec{x}, \vec{\tau}, \text{GlobalSam}$ )  $\equiv$ 
  if  $\| \text{SeriesGradient}(\vec{x}, \vec{\tau}, \text{GlobalSam}) \| > \epsilon$ 
    then  $\vec{x} + \delta * \text{SeriesGradient}(\vec{x}, \vec{\tau}, \text{GlobalSam})$ 
    else  $\vec{x}$  fi.
funct FixedPoint( $\vec{x}, D, \text{GlobalSam}, C$ )
   $\vec{y} := \text{Uphill}(\vec{x}, \vec{\tau}, \text{GlobalSam})$ 
   $q := D.\text{RangeQ}(\delta, \vec{y})$ 
  if  $\exists i \in q : C.\text{clustered}(i)$  then  $\text{Id} := C.\text{getLabel}(i)$ 
  elseif  $\vec{y} = \vec{x}$ 
    then  $\text{Id} := D.\text{Add}(\vec{x})$ 
    else  $\text{Id} := \text{FixedPoint}(\vec{y}, D, \text{GlobalSam}, C)$ 
  fi
foreach  $i \in D.\text{RangeQ}(\delta, \vec{x})$ 
  do C.setLabel( $i, \text{Id}$ ); C.setClustered( $i$ ) od
  Id.
proc SiteDECluster( $D, H, C$ )  $\equiv$ 
  SiteNegotiate( $H, \vec{\tau}, \vec{z}_1, \vec{z}_2, K, h$ )
  Send( $H, \text{Sample}(D, \vec{\tau}, \vec{z}_1, \vec{z}_2, K, h)$ )
  GlobalSam := Receive( $H$ )
  for  $i := 1$  to  $D.\text{count}$  do
    if  $\neg C.\text{clustered}(i)$ 
      then FixedPoint( $\vec{x}_i, D, \text{GlobalSam}, C$ ) fi
  od.
proc Helper( $L$ )  $\equiv$ 
  HelperNegotiate( $L$ )
  for  $j := 1$  to  $L.\text{count}$  do
    Sam := Receive( $L.\text{get}(j)$ )
    foreach  $\vec{z} \in \text{Rectangle}(\text{Sam}.\text{getCorners}())$ 
      do GlobalSam.Sum( $\vec{z}, \text{Sam}.\text{get}(\vec{z})$ ) od
  od
  for  $j := 1$  to  $L.\text{count}$  do Send(GlobalSam,  $L.\text{get}(j)$ ) od.

```

the window width h . The negotiation procedures contain appropriate handshaking primitives to ensure that all sites participate and exit negotiations only if an agreement has been reached. Each local site computes the sampled form of the estimate of D , by calling function *Sample*, and sends it to the helper. (Function *DE* computes the density estimate and is omitted for brevity.) The helper receives the sampled estimates, sums them by sampling indexes into a global sample, and returns it to all sites. Procedures *Send* and *Receive* implement appropriate blocking and handshaking to ensure the transmission takes place. Each local site uses the global sample in functions *FixedPoint* and *Uphill* to compute the values of the gradient of the global density estimate. Function *SeriesGradient* can be easily derived from (5). Local sites perform a DE-cluster algorithm to compute the corresponding local data clusters. The details of the hill-climbing strategy are however different from Algorithm 1 because the sites are allowed access to local data objects only. *Uphill* ad-

vances a fraction δ of the gradient in its direction, if the gradient's norm exceeds a threshold ϵ . If a δ -neighbourhood of the object returned by *Uphill* contains an already clustered data object, the current cluster label *Id* is set from that object's label. Otherwise, checking whether *Uphill* returned the same space object signals to *FixedPoint* that the proximity of the local maximum has been reached. The maximum is marked by adding the current space object $\vec{x}$ as a dummy object to the local data set; this ensures that subsequent paths converging to the same local maximum will use the same cluster label as the current path. Method *D.Add*($\cdot$) returns the identifier of the added object, which is used as current cluster label. If neither case holds, the label *Id* is obtained by a recursive call. Finally, all objects in a small neighbourhood of the current object are labeled by *Id*. Note that adding dummy objects has effect only on the range queries, and does not modify the density estimate.

4.3 Complexity of the KDEC scheme

In terms of the complexity in computation and communication one crucial point of the KDEC scheme is how many samples have to be computed and transferred among the sites. In most cases, to obtain good density estimates, h must not be less than a small multiple of the smallest object distance. As $\tau^{(i)} \simeq h/2$, the number of samples should rarely exceed the number of objects, if only space regions where the density estimate is not negligible are sampled. Since the size of a sample is usually much smaller than the size of a data object, the overall communication costs of our DDC approach will be in most cases significantly lower than in a centralized approach. Of course, the precise number of samples depends on the bounding region that is being sampled by every site. In Algorithm 2 the site L_j determines autonomously the rectangle that contains the computed samples.

The computational costs of the KDEC scheme in terms of used CPU cycles and I/O do not exceed the one in the centralized approach where clustering is performed on data collected in a single repository. The computational complexity is linear in the number of samples. The precise cost of computation of any KDEC-based DDC algorithm as an instance of the proposed scheme largely depends also on the used kernel function and local clustering algorithm. The DE-cluster algorithm we developed for the KDEC scheme in Section 3.2 is of complexity $O(Nq(N))$, where $q(N)$ is the cost of a nearest neighbour query (which in practical cases is close to $\log N$). Algorithm 2 implements a slightly different approach in the hill-climbing function than Algorithm 1, since the function does not use data objects to direct the uphill path. However, preliminary results of experiments conducted on a prototype implementation show good scalability of the approach, in terms of number of executed range queries.

5 Conclusion

Due to the explosion in the number of autonomous data sources there is a growing need for effective approaches to distributed knowledge discovery and data mining. In this paper we have presented KDEC, a novel scheme for distributed data clustering which computes the density estimation, to per-

form the clustering, from sampled forms of local densities at each data source site.

The approach exploits statistical density estimation and information theoretic sampling to minimize communications between sites. Moreover, the privacy of data is preserved to a large extent by never transmitting data values but kernel based density estimation samples outside the site of origin. The approach does not require CPU and I/O costs significantly higher than a similar centralized approach and its communication costs may be lower. Ongoing research focuses in particular on implementations of a multiagent system for KDEC-based distributed data clustering in a peer-to-peer network, and investigation on methods to mitigate the risk of security and privacy violations in distributed data mining environments.

References

- [Ankerst *et al.*, 1999] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 49–60, Philadelphia, PA, June 1999.
- [Chen *et al.*, 1996] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: an overview from a database perspective. *IEEE Trans. On Knowledge And Data Engineering*, 8:866–883, 1996.
- [Ester *et al.*, 1996] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, Portland, OR, 1996.
- [Ester *et al.*, 1998] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB'98)*, pages 323–333, New York City, NY, August 1998.
- [Fayyad *et al.*, 1996] Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI Press/MIT Press, 1996.
- [Ganti *et al.*, 1999] Venkatesh Ganti, Raghu Ramakrishnan, Johannes Gehrke, Allison L. Powell, and James C. French. Clustering large datasets in arbitrary metric spaces. In *Proceedings of the 15th International Conference on Data Engineering (ICDE 1999)*, pages 502–511, Sydney, Australia, March 1999.
- [Grabmeier and Rudolph, 2002] J. Grabmeier and A. Rudolph. Techniques of cluster algorithms in data mining. *Data Mining and Knowledge Discovery*, 6:303–360, 2002.
- [Higgins, 1996] J. R. Higgins. *Sampling Theory in Fourier and Signal Analysis*. Clarendon Press, Oxford, 1996.
- [Hinneburg and Keim, 1998] Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 58–65, New York City, New York, USA, 1998. AAAI Press.
- [Johnson and Kargupta, 1999] Erik Johnson and Hillol Kargupta. Collective, hierarchical clustering from distributed heterogeneous data. In M. Zaki and C. Ho, editors, *Large-Scale Parallel KDD Systems*, Lecture Notes in Computer Science, pages 221–244. Springer-Verlag, 1999.
- [Kargupta *et al.*, 2000] H. Kargupta, B. Park, D. Hersherberger, and E. Johnson. *Advances in Distributed and Parallel Knowledge Discovery*, chapter 5, Collective Data Mining: A New Perspective Toward Distributed Data Mining, pages 131–178. AAAI/MIT Press, 2000.
- [Kargupta *et al.*, 2001] H. Kargupta, W. Huang, S. Krishnamoorthy, and E. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems Journal*, 3(4):422–448, 2001.
- [Moro and Sartori, 2001] Gianluca Moro and Claudio Sartori. Incremental maintenance of multi-source views. In M. E. Orlowska and J. Roddick, editors, *Proceedings of the 12th Australasian conference on Database technologies, ADC 2001*, ACM International Conference Proceeding, pages 13–20, Queensland, Australia, 2001. IEEE Computer Society Press.
- [Parzen, 1962] E. Parzen. On estimation of a probability density function and mode. *Ann. Math. Statist.*, 33:1065–1076, 1962.
- [Schikuta, 1996] Erich Schikuta. Grid-clustering: An efficient hierarchical clustering method for very large data sets. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 101–105. IEEE, 1996.
- [Silverman, 1986] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.
- [Zhang *et al.*, 1996] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Canada, June 1996.

When Discriminative Learning of Bayesian Network Parameters Is Easy

Hannes Wettig*, Peter Grünwald[◦], Teemu Roos*, Petri Myllymäki*, and Henry Tirri*

* Complex Systems Computation Group (CoSCo)
Helsinki Institute for Information Technology (HIIT)
University of Helsinki & Helsinki University of Technology
P.O. Box 9800, FIN-02015 HUT, Finland
{Firstname}.{Lastname}@hiit.fi

[◦] Centrum voor Wiskunde en Informatica (CWI)
P.O. Box 94079
NL-1090 GB Amsterdam, The Netherlands.
Peter.Grunwald@cwi.nl

Abstract

Bayesian network models are widely used for discriminative prediction tasks such as classification. Usually their parameters are determined using ‘unsupervised’ methods such as maximization of the joint likelihood. The reason is often that it is unclear how to find the parameters maximizing the conditional (supervised) likelihood. We show how the discriminative learning problem can be solved efficiently for a large class of Bayesian network models, including the Naive Bayes (NB) and tree-augmented Naive Bayes (TAN) models. We do this by showing that under a certain general condition on the network structure, the discriminative learning problem is exactly equivalent to logistic regression with unconstrained convex parameter spaces. Hitherto this was known only for Naive Bayes models. Since logistic regression models have a concave log-likelihood surface, the global maximum can be easily found by local optimization methods.

1 Introduction

In recent years it has been recognized that for discriminative prediction tasks such as classification, we should use a ‘supervised’ learning algorithm such as conditional likelihood maximization [Friedman *et al.*, 1997; Ng and Jordan, 2001; Kontkanen *et al.*, 2001; Greiner and Zhou, 2002]. Nevertheless, for Bayesian network models the parameters are customarily determined using ordinary methods such as maximization of the joint (unsupervised) likelihood. One of the main reasons for this discrepancy is the difficulty in finding the global maximum of the conditional likelihood. In this paper, we show that this problem can be solved, as long as the underlying Bayesian network meets a particular additional condition, which is satisfied for many existing Bayesian-network based models including Naive Bayes (NB), TAN (tree-augmented NB) and ‘diagnostic’ models [Kontkanen *et al.*, 2001].

We consider domains of discrete-valued random variables. We find the maximum conditional likelihood parameters by logarithmic reparametrization. In this way, each conditional Bayesian network model is mapped to a logistic regression

model, for which the likelihood surface is known to be concave. However, in some cases the parameters of this logistic regression model are not allowed to vary freely. In other words, the Bayesian network model corresponds to a subset of a logistic regression model rather than the full model.

Our main result (Thm. 3 below) provides a general condition on the network structure under which, as we prove, the Bayesian network model is mapped to a full logistic regression model with freely varying parameters. Therefore, in the new parametrization the conditional log-likelihood becomes a concave function of the parameters that under our condition are allowed to vary freely over the convex set $\mathbb{R}^k$. Now we can find the global maximum in the conditional likelihood surface by simple local optimization techniques such as hill climbing.

The result still leaves open the possibility that there are *no* network structures for which the conditional likelihood surface has local, non-global maxima. This would make our condition superfluous. Our second result (Thm. 4 below) shows that this is not the case: there are very simple network structures that do not satisfy our condition, and for which the conditional likelihood can exhibit local, non-global maxima.

Viewing Bayesian network (BN) models as subsets of logistic regression models is not new; it was done earlier in papers such as [Heckerman and Meek, 1997a; Ng and Jordan, 2001; Greiner and Zhou, 2002]. Also, the concavity of the log-likelihood surface for logistic regression is known. Our main contribution is to supply the condition under which Bayesian network models correspond to logistic regression with *completely freely varying parameters*. Only then can we guarantee that there are no local maxima in the likelihood surface. As a direct consequence of our result, we show for the first time that the supervised likelihood of, for instance, the tree-augmented Naive Bayes (TAN) model has no local maxima.

This paper is organized as follows. In Section 2 we introduce Bayesian networks and an alternative, so-called *L*-parametrization. In Section 3 we show that this allows us to consider Bayesian network models as logistic regression models. Based on earlier results in logistic regression, we conclude that in the *L*-parametrization the supervised log-likelihood is a concave function. In Section 4 we present our main results, giving conditions under which the two parametrizations correspond to exactly the same conditional

distributions. Conclusions are summarized in Section 5; proofs of the main results are given in Appendix A.

2 Bayesian Networks and the L -model

We assume that the reader is familiar with the basics of the theory of Bayesian networks, see, e.g., [Pearl, 1988].

Consider a random vector $X = (X_0, X_1, \dots, X_{M'})$, where each variable X_i takes values in $\{1, \dots, n_i\}$. Let $\mathcal{B}$ be a Bayesian network structure over X , that factorizes $P(X)$ into

$$P(X) = \prod_{i=0}^{M'} P(X_i | Pa_i), \quad (1)$$

where $Pa_i \subseteq \{X_0, \dots, X_{M'}\}$ is the parent set of variable X_i in $\mathcal{B}$.

We are interested in predicting some *class variable* X_m for some $m \in \{0, \dots, M'\}$ conditioned on all $X_i, i \neq m$. Without loss of generality we may assume that $m = 0$ (i.e., X_0 is the class variable) and that the children of X_0 in $\mathcal{B}$ are $\{X_1, \dots, X_M\}$ for some $M \leq M'$. For instance, in the so-called Naive Bayes model we have $M = M'$ and the children of the class variable X_0 are independent given the value of X_0 . The Bayesian network model corresponding to $\mathcal{B}$ is the set of all distributions satisfying the conditional independencies encoded in $\mathcal{B}$. It is usually parametrized by vectors $\Theta^{\mathcal{B}}$ with components of the form $\theta_{x_i|pa_i}^{\mathcal{B}}$ defined by

$$\theta_{x_i|pa_i}^{\mathcal{B}} := P(X_i = x_i | Pa_i = pa_i), \quad (2)$$

where pa_i is any configuration (set of values) for the parents Pa_i of X_i . Whenever we want to emphasize that each pa_i is determined by the complete data vector $x = (x_0, \dots, x_{M'})$, we write $pa_i(x)$ to denote the configuration of Pa_i in $\mathcal{B}$ given by the vector x . For a given data vector $x = (x_0, x_1, \dots, x_{M'})$, we sometimes need to consider a modified vector where x_0 is replaced by x'_0 and the other entries remain the same. We then write $pa_i(x'_0, x)$ for the configuration of Pa_i given by $(x'_0, x_1, \dots, x_{M'})$.

We let $\mathcal{M}^{\mathcal{B}}$ be the set of *conditional* distributions $P(X_0 | X_1, \dots, X_{M'}, \Theta^{\mathcal{B}})$ corresponding to distributions $P(X_0, \dots, X_{M'} | \Theta^{\mathcal{B}})$ satisfying the conditional independencies encoded in $\mathcal{B}$. The conditional distributions in $\mathcal{M}^{\mathcal{B}}$ can be written as

$$P(x_0 | x_1, \dots, x_{M'}, \Theta^{\mathcal{B}}) = \frac{\theta_{x_0|pa_0(x)}^{\mathcal{B}} \prod_{i=1}^{M'} \theta_{x_i|pa_i(x)}^{\mathcal{B}}}{\sum_{x'_0=1}^{n_0} \theta_{x'_0|pa_0(x)}^{\mathcal{B}} \prod_{i=1}^{M'} \theta_{x_i|pa_i(x'_0, x)}^{\mathcal{B}}}, \quad (3)$$

extended to N outcomes by independence.

Given a complete data-matrix $D = (x^1, \dots, x^N)$, the *conditional log-likelihood*, $S^{\mathcal{B}}(D; \Theta^{\mathcal{B}})$, with parameters $\Theta^{\mathcal{B}}$ is given by

$$S^{\mathcal{B}}(D; \Theta^{\mathcal{B}}) := \sum_{j=1}^N S^{\mathcal{B}}(x^j; \Theta^{\mathcal{B}}), \quad (4)$$

where

$$S^{\mathcal{B}}(x; \Theta^{\mathcal{B}}) := \ln P(x_0 | x_1, \dots, x_{M'}, \Theta^{\mathcal{B}}). \quad (5)$$

Note that in (3), and hence also in (4), all $\theta_{x_i|pa_i}^{\mathcal{B}}$ with $i > M$ (standing for nodes that are neither the class variable

nor any of its children) cancel out, since for these terms we have $pa_i(x) \equiv pa_i(x'_0, x)$ for all x'_0 . Thus the only relevant parameters for determining the conditional likelihood are of the form $\theta_{x_i|pa_i}^{\mathcal{B}}$ with $i \in \{0, \dots, M\}$, $x_i \in \{1, \dots, n_i\}$ and pa_i any configuration of parents Pa_i . We order these parameters lexicographically and define $\Theta^{\mathcal{B}}$ to be the set of vectors constructed this way, with $\theta_{x_i|pa_i}^{\mathcal{B}} > 0$ and $\sum_{x_i=1}^{n_i} \theta_{x_i|pa_i}^{\mathcal{B}} = 1$ for all $i \in \{0, \dots, M\}$, x_i and all values (configurations) of pa_i . Note that we require all parameters to be *strictly* positive.

The model $\mathcal{M}^{\mathcal{B}}$ does not contain any notion of the joint distribution: Terms such as $P(X_i | Pa_i)$, where $0 < i \leq M'$, are undefined and neither are we interested in them. Our task is prediction of X_0 given $X_1, \dots, X_{M'}$. Heckerman and Meek call such models *Bayesian regression/classification* (BRC) models [Heckerman and Meek, 1997a; 1997b].

For an arbitrary conditional Bayesian network model $\mathcal{M}^{\mathcal{B}}$, we now define the so-called L -model, another set of conditional distributions $P(X_0 | X_1, \dots, X_{M'})$. This model, which we denote by $\mathcal{M}^L$, is parametrized by vectors Θ^L in some set Θ^L that closely resembles $\Theta^{\mathcal{B}}$. Each different $\mathcal{M}^{\mathcal{B}}$ gives rise to a corresponding $\mathcal{M}^L$, although we do not necessarily have $\mathcal{M}^{\mathcal{B}} = \mathcal{M}^L$. For each component $\theta_{x_i|pa_i}^{\mathcal{B}}$ of each vector $\Theta^{\mathcal{B}} \in \Theta^{\mathcal{B}}$, there is a corresponding component $\theta_{x_i|pa_i}^L$ of the vectors $\Theta^L \in \Theta^L$. The components $\theta_{x_i|pa_i}^L$ take values in the range $(-\infty, \infty)$ rather than $(0, 1)$. Each vector $\Theta^L \in \Theta^L$ defines the following conditional distribution:

$$P(x_0 | x_1, \dots, x_{M'}, \Theta^L) := \frac{(\exp \theta_{x_0|pa_0(x)}^L) \prod_{i=1}^M \exp \theta_{x_i|pa_i(x)}^L}{\sum_{x'_0=1}^{n_0} (\exp \theta_{x'_0|pa_0(x)}^L) \prod_{i=1}^M \exp \theta_{x_i|pa_i(x'_0, x)}^L}. \quad (6)$$

The model $\mathcal{M}^L$ is the set of conditional distributions $P(X_0 | X_1, \dots, X_{M'}, \Theta^L)$ indexed by $\Theta^L \in \Theta^L$, extended to N outcomes by independence. Given a data-matrix D , let $S^L(D; \Theta^L)$ be the conditional log-likelihood with parameters Θ^L , defined analogously to (4) with (6) in place of (3).

Theorem 1. $\mathcal{M}^{\mathcal{B}} \subseteq \mathcal{M}^L$.

Proof. From (3) and (6) we get that Θ^L defined by setting $\theta_{x_i|pa_i}^L = \ln \theta_{x_i|pa_i}^{\mathcal{B}}$ for all i, x_i and pa_i , indexes the same conditional distribution as $\Theta^{\mathcal{B}}$. ■

In words, all the conditional distributions that can be represented by parameters $\Theta^{\mathcal{B}} \in \Theta^{\mathcal{B}}$ can also be represented by parameters $\Theta^L \in \Theta^L$. The converse of Theorem 1, i.e., $\mathcal{M}^L \subseteq \mathcal{M}^{\mathcal{B}}$, is true only under some additional conditions on the network structure, as we explain in Section 4. First we take a closer look at the L -model.

3 The L -model Viewed as Logistic Regression

Although L -models are closely related to and in some cases formally identical to Bayesian network models, we can also think of them as predictors that combine the information of the attributes using the so-called *softmax* rule [Heckerman and Meek, 1997b; Ng and Jordan, 2001]. In statistics, such models have been extensively studied under the name of *logistic regression* models, see, e.g. [McLachlan, 1992, p.255].

More precisely, let $X_0 = \{1, \dots, n_0\}$ and let $Y_1, \dots, Y_k$ be real-valued random variables. The *multiple logistic regression model with dependent variable X_0 and covariates $Y_1, \dots, Y_k$* is defined as the set of conditional distributions

$$P(x_0 | y_1, \dots, y_k) := \frac{\exp \sum_{i=1}^k \beta_{x_0|i} y_i}{\sum_{x'_0=1}^{n_0} \exp \sum_{i=1}^k \beta_{x'_0|i} y_i} \quad (7)$$

where the $\beta_{x_0|i}$ are allowed to take on all values in $\mathbb{R}$. This defines a conditional model parameterized in $\mathbb{R}^{n_0 \cdot k}$. Now, for $i \in \{0, \dots, M\}$, $x_i \in \{1, \dots, n_i\}$ and pa_i in the set of parent configurations of X_i , let

$$Y_{(x_i, pa_i)} := \begin{cases} 1 & \text{if } X_i = x_i \text{ and } Pa_i = pa_i \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

The indicator random variables $Y_{(x_i, pa_i)}$ thus obtained can be lexicographically ordered and renamed $1, \dots, k$, which shows that each L -model corresponding to a Bayesian network structure $\mathcal{B}$ as in (6) is formally identical to the logistic model (7) with dependent variable X_0 and covariates given by (8). So, for all network structures $\mathcal{B}$, the corresponding L -model $\mathcal{M}^L$ is the standard multiple logistic model, where the input variables for the logistic model are transformations of the input variables to the L -model, the transformation being determined by the network structure $\mathcal{B}$.

It turns out that the conditional log-likelihood in the L -parameterization is a concave function of the parameters:

Theorem 2. *The parameter set Θ^L is convex, and the conditional log-likelihood $S^L(D; \Theta^L)$ is concave, though not strictly concave.*

Proof. The first part is obvious since each parameter can take values in $(-\infty, \infty)$. Concavity of $S^L(D; \Theta^L)$ is a direct consequence of the fact that multiple logistic regression models are exponential families; see, e.g., [McLachlan, 1992, p.260]. For an example showing that the conditional log-likelihood is not strictly concave, see [Wettig et al., 2002]. ■

Remark. Non-strictness of the proven concavity may pose a technical problem in optimization. This can be avoided by assigning a strictly concave prior on the model parameters and then maximizing the ‘conditional posterior’ [Grünwald et al., 2002; Wettig et al., 2002] instead of the likelihood. One may also prune the model of weakly supported parameters and/or add constraints to arrive at a strictly concave conditional likelihood surface. Our experiments [Wettig et al., 2002] suggest that for small data samples this should be done in any case, in order to avoid over-fitting; see also Section 5. Any constraint added should of course leave the parameter space a convex set, e.g. a subspace of the full Θ^L .

Corollary 1. *There are no local, non-global, maxima in the likelihood surface of an L -model.*

The conditions under which a global maximum exists are discussed in, e.g., [McLachlan, 1992] and references therein. A possible solution in cases where no maximum exists is to introduce a strictly concave prior as discussed above.

The global conditional maximum likelihood parameters obtained from training data can be used for prediction of

future data. In addition, as discussed in [Heckerman and Meek, 1997a], they can be used to perform model selection among several competing model structures using, e.g., the BIC or (approximate) MDL criteria. In [Heckerman and Meek, 1997a] it is stated that for general conditional Bayesian network models $\mathcal{M}^B$, “although it may be difficult to determine a global maximum, gradient-based methods [...] can be used to locate local maxima”. Theorem 2 shows that if the network structure $\mathcal{B}$ is such that the two models are equivalent, $\mathcal{M}^B = \mathcal{M}^L$, we can find even the *global* maximum of the conditional likelihood by reparametrizing $\mathcal{M}^B$ to the L -model, and using some local optimization method. Thus, the question under which condition $\mathcal{M}^B = \mathcal{M}^L$ becomes crucial. It is this question we address in the next section.

Remark. Because the log-transformation is continuous, it follows (with some calculus) that, if $\mathcal{M}^B = \mathcal{M}^L$, then all maxima of the (concave) conditional likelihood in the L -parameterization are global (and connected) maxima also in the original parametrization. Nevertheless, the likelihood surface as a function of $\Theta^B \in \Theta^B$ has some unpleasant properties (see [Wettig et al., 2002]): it is *not* concave in general and, worse, it can have ‘wrinkles’: by these we mean convex subsets Θ_0^B of Θ^B , such that, under the constraint that $\Theta^B \in \Theta_0^B$, the likelihood surface does exhibit local, non-global maxima. This suggests that it is computationally preferable to optimize over Θ^L rather than Θ^B . Empirical evidence for this is reported in [Greiner and Zhou, 2002].

4 Main Result

By setting $\theta_{x_i|pa_i}^L = \ln \theta_{x_i|pa_i}^B$, it follows that each distribution in $\mathcal{M}^B$ is also in $\mathcal{M}^L$ (Thm. 1). This suggests that, by doing the reverse transformation

$$\theta_{x_i|pa_i}^B = \exp \theta_{x_i|pa_i}^L, \quad (9)$$

we could also show that distributions in $\mathcal{M}^L$ are also in $\mathcal{M}^B$. However, Θ^L contains distributions that violate the ‘sum-up-to-one constraint’, i.e., for some $\Theta^L \in \Theta^L$ we have $\sum_{x_i=1}^{n_i} \exp \theta_{x_i|pa_i}^L \neq 1$ for some $i \in \{0, \dots, M'\}$ and pa_i . Then the corresponding Θ^B is *not* in Θ^B . But, since the L -parameterization is redundant (many different Θ^L index the same conditional distribution $P(\cdot | \cdot) \in \mathcal{M}^L$), it may still be the case that the distribution $P(\cdot | \cdot, \Theta^L)$ indexed by Θ^L is in $\mathcal{M}^B$. Indeed, it turns out that for *some* network structures $\mathcal{B}$, the corresponding $\mathcal{M}^L$ is such that *each* distribution in $\mathcal{M}^L$ can be expressed by a parameter vector Θ^L such that $\sum_{x_i=1}^{n_i} \exp \theta_{x_i|pa_i}^L = 1$ for all $i \in \{0, \dots, M'\}$ and pa_i . In that case, by (9), we *do* have $\mathcal{M}^B = \mathcal{M}^L$. Our main result is that this is the case if $\mathcal{B}$ satisfies the following condition:

Condition 1. For all $j \in \{1, \dots, M\}$, there exists $X_i \in Pa_j$ such that $Pa_j \subseteq Pa_i \cup \{X_i\}$.

Remark. Condition 1 implies that the class X_0 must be a ‘moral node’, i.e., it cannot have a common child with a node it is not directly connected with. But Condition 1 demands more than that; see Figures 1 and 2.

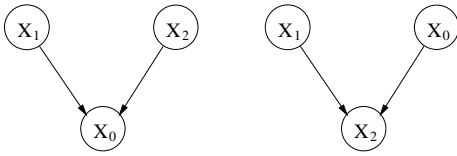


Figure 1: A simple Bayesian network (the class variable is denoted by X_0) satisfying Condition 1 (left); and a network that does not satisfy the condition (right).

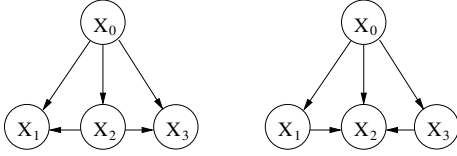


Figure 2: A tree-augmented Naive Bayes (TAN) model satisfying Condition 1 (left); and a network that is not TAN (right). Even though in both cases the class variable X_0 is a moral node, the network on the right does not satisfy Condition 1.

Example 1. Consider the Bayesian networks depicted in Figure 1. The leftmost network, B_1 , satisfies Condition 1, the rightmost network, B_2 , does not. Theorem 4 shows that the conditional likelihood surface of $\mathcal{M}^{B_2}$ can have local maxima, implying that in this case $\mathcal{M}^B \neq \mathcal{M}^L$. $\diamond$

Examples of network structures that satisfy Condition 1 are the Naive Bayes (NB) and the tree-augmented Naive Bayes (TAN) models [Friedman *et al.*, 1997]. The latter is a generalization of the former in which the children of the class variable are allowed to form tree-structures; see Figure 2.

Proposition 1. *Condition 1 is satisfied by the Naive Bayes and the tree-augmented Naive Bayes structures.*

Proof. For Naive Bayes, we have $Pa_j \subseteq \{X_0\}$ for all $j \in \{1, \dots, M\}$. For TAN models, all children of the class variable have either one or two parents. For children with only one parent (the class variable) we can use the same argument as in the NB case. For any child X_j with two parents, let X_i be the parent that is not the class variable. Because X_i is also a child of the class variable, we have $Pa_j \subseteq Pa_i \cup \{X_0\}$. $\blacksquare$

Condition 1 is also automatically satisfied if X_0 only has incoming arcs¹ (‘diagnostic’ models, see [Kontkanen *et al.*, 2001]). For Bayesian network structures B for which the condition does not hold, we can always add some arcs to arrive at a structure B' for which the condition does hold (for instance, add an arc from X_1 to X_3 in the rightmost network in Figure 2). Therefore, $\mathcal{M}^B$ is always a subset of a larger model $\mathcal{M}^{B'}$ for which the condition holds. We are now ready to present our main result (for proof see Appendix A):

Theorem 3. *If B satisfies Condition 1, then $\mathcal{M}^B = \mathcal{M}^L$.*

Together with Corollary 1, Theorem 3 shows that Condition 1 suffices to ensure that the conditional likelihood surface of $\mathcal{M}^B$ has no local (non-global) maxima. Proposition 1

¹It is easy to see that in that case the maximum conditional likelihood parameters may even be determined analytically.

now implies that, for example, the conditional likelihood surface of TAN models has no local maxima. Therefore, a global maximum can be found by local optimization techniques.

But what about the case in which Condition 1 does not hold? Our second result, Theorem 4 (proven in Appendix A) says that in this case, there can be local maxima:

Theorem 4. *Let $B_2 = X_1 \rightarrow X_2 \leftarrow X_0$ be the network structure depicted in Figure 1 (right). There exist data samples such that the conditional likelihood has local, non-global maxima over $\mathcal{M}^{B_2}$.*

The theorem implies that $\mathcal{M}^L \neq \mathcal{M}^{B_2}$. Thus, Condition 1 is not superfluous. We may now ask whether our condition is *necessary* for having $\mathcal{M}^L = \mathcal{M}^B$; that is, whether $\mathcal{M}^L \neq \mathcal{M}^B$ for all network structures that violate the condition. We plan to address this intriguing open question in future work.

5 Concluding Remarks

We showed that one can effectively find the parameters maximizing the conditional (supervised) likelihood of NB, TAN and many other Bayesian network models. We did this by showing that the network structure of these models satisfies our ‘Condition 1’, which ensures that the conditional distributions corresponding to such models are equivalent to a particular multiple logistic regression model with unconstrained parameters. An arbitrary network structure can always be made to satisfy Condition 1 by adding arcs. Thus, we can embed any Bayesian network model in a larger model (with less independence assumptions) that satisfies Condition 1.

Test runs for the Naive Bayes case in [Wettig *et al.*, 2002] have shown that maximizing the conditional likelihood in contrast to the usual practice of maximizing the joint (unsupervised) likelihood is feasible and yields greatly improved classification. Similar results are reported in [Greiner and Zhou, 2002]. Our conclusions are also supported by theoretical analysis in [Ng and Jordan, 2001]. Only on very small data sets we sometimes see that joint likelihood optimization outperforms conditional likelihood, the reason apparently being that the conditional method is more inclined to overfitting. We conjecture that in such cases, rather than resorting to maximizing the joint instead of the conditional likelihood, it may be preferable to use a simpler model or simplify (i.e. prune or restrict) the model at hand and still choose its parameters in a discriminative fashion. In our setting, this would amount to model selection using the L -parametrization. This is a subject of our future research.

References

- [Friedman *et al.*, 1997] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.
- [Greiner and Zhou, 2002] R. Greiner and W. Zhou. Structural extension to logistic regression: Discriminant parameter learning of belief net classifiers. In *Proceedings of the Eighteenth Annual National Conference on Artificial Intelligence (AAAI-02)*, pages 167–173, Edmonton, 2002.
- [Grünwald *et al.*, 2002] P. Grünwald, P. Kontkanen, P. Myllymäki, T. Roos, H. Tirri, and H. Wettig. Supervised posterior distri-

butions, 2002. Presented at the Seventh Valencia International Meeting on Bayesian Statistics, Tenerife, Spain.

[Heckerman and Meek, 1997a] D. Heckerman and C. Meek. Embedded bayesian network classifiers. Technical Report MSR-TR-97-06, Microsoft Research, 1997.

[Heckerman and Meek, 1997b] D. Heckerman and C. Meek. Models and selection criteria for regression and classification. In D. Geiger and P. Shenoy, editors, *Uncertainty in Artificial Intelligence 13*, pages 223–228. Morgan Kaufmann Publishers, San Mateo, CA, 1997.

[Kontkanen *et al.*, 2001] P. Kontkanen, P. Myllymäki, and H. Tirri. Classifier learning with supervised marginal likelihood. In J. Breese and D. Koller, editors, *Proceedings of the 17th International Conference on Uncertainty in Artificial Intelligence (UAI'01)*. Morgan Kaufmann Publishers, 2001.

[McLachlan, 1992] G.J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley & Sons, New York, 1992.

[Ng and Jordan, 2001] A.Y. Ng and M.I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *Advances in Neural Information Processing Systems*, 14:605–610, 2001.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[Wettig *et al.*, 2002] H. Wettig, P. Grünwald, T. Roos, P. Myllymäki, and H. Tirri. On supervised learning of Bayesian network parameters. Technical Report HIIT-2002-1, Helsinki Institute for Information Technology (HIIT), 2002. Available at <http://cosco.hiit.fi/Articles/hiit2002-1.ps>.

A Proofs

Proof of Theorem 3. We introduce some more notation. For $j \in \{1, \dots, M\}$, let m_j be the maximum number in $\{0, \dots, M\}$ such that $X_{m_j} \in Pa_j$, $Pa_j \subseteq Pa_{m_j} \cup \{X_{m_j}\}$. Such an m_j exists by Condition 1. To see this, note that the $X_i \in Pa_j$ mentioned in Condition 1 must lie in the set $\{X_0, X_1, \dots, X_M\}$ (otherwise $X_0 \in Pa_j$, $X_0 \notin Pa_i$, so $Pa_j \not\subseteq Pa_i \cup \{X_i\}$, contradiction).

Condition 1 implies that pa_j is completely determined by the pair (x_{m_j}, pa_{m_j}) . We can therefore introduce functions Q_j mapping (x_{m_j}, pa_{m_j}) to the corresponding pa_j . Hence, for all $x = (x_0, \dots, x_M)$ and $j \in \{1, \dots, M\}$ we have

$$pa_j = Q_j(x_{m_j}, pa_{m_j}). \quad (10)$$

We introduce, for all $i \in \{0, \dots, M\}$ and for each configuration pa_i of Pa_i , a constant $c_{i|pa_i}$ and define, for any $\Theta^L \in \Theta^L$,

$$\theta_{x_i|pa_i}^{(c)} := \theta_{x_i|pa_i}^L + c_{i|pa_i} - \sum_{j:m_j=i} c_{j|Q_j(x_i, pa_i)}. \quad (11)$$

The parameters $\theta_{x_i|pa_i}^{(c)}$ constructed this way are combined to a vector $\Theta^{(c)}$ which is clearly a member of Θ^L .

Having introduced this notation, we now show that no matter how we choose the constants $c_{i|pa_i}$, for all Θ^L and corresponding $\Theta^{(c)}$ we have $S^L(D; \Theta^{(c)}) = S^L(D; \Theta^L)$.

We first show that, for all possible vectors x and the corresponding parent configurations, no matter how the $c_{i|pa_i}$ are chosen, it holds that

$$\sum_{i=0}^M \theta_{x_i|pa_i}^{(c)} = \sum_{i=0}^M \theta_{x_i|pa_i}^L + c_{0|pa_0}. \quad (12)$$

To derive (12) we substitute all terms of $\sum_{i=0}^M \theta_{x_i|pa_i}^{(c)}$ by their definition (11). Clearly, for all $j \in \{1, \dots, M\}$, there is exactly one term of the form $c_{j|pa_j}$ that appears in the sum with a positive sign. Since for each $j \in \{1, \dots, M\}$ there exists exactly one $i \in \{0, \dots, M\}$ with $m_j = i$, it must be the case that for all $j \in \{1, \dots, M\}$, a term of the form $c_{j|Q_j(x_i, pa_i)}$ appears exactly once in the sum with a negative sign. By (10) we have $c_{j|Q_j(x_i, pa_i)} = c_{j|pa_j}$. Therefore all terms $c_{j|pa_j}$ that appear once with a positive sign also appear once with a negative sign. It follows that, except for $c_{0|pa_0}$, all terms $c_{j|pa_j}$ cancel. This establishes (12). By plugging in (12) into (6), it follows that $S^L(D; \Theta^{(c)}) = S^L(D; \Theta^L)$ for all D .

Now set, for all x_i and pa_i ,

$$\theta_{x_i|pa_i}^B = \exp \theta_{x_i|pa_i}^{(c)}. \quad (13)$$

We show that we can determine the constants $c_{i|pa_i}$ such that for all $i \in \{0, \dots, M\}$ and pa_i , the ‘sum up to one’ constraint is satisfied, i.e., we have

$$\sum_{x_i=1}^{n_i} \theta_{x_i|pa_i}^B = 1. \quad (14)$$

We achieve this by sequentially determining values for $c_{i|pa_i}$ in a particular order.

We need some additional terminology: we say ‘ c_i is determined’ if for all configurations pa_i of Pa_i , we have already determined $c_{i|pa_i}$. We say ‘ c_i is undetermined’ if we have determined $c_{i|pa_i}$ for no configuration pa_i of Pa_i . We say ‘ c_i is ready to be determined’ if c_i is undetermined and at the same time all c_j with $m_j = i$ have been determined.

We note that as long as some c_i with $i \in \{0, \dots, M\}$ are undetermined, there must exist $c_{i'}$ that are ready to be determined. To see this, first take any $i \in \{0, \dots, M\}$ with c_i undetermined. Either c_i itself is ready to be determined (in which case we are done), or there exists $j \in \{1, \dots, M\}$ with $m_j = i$ (and hence $X_i \in Pa_j$) such that c_j is undetermined. If c_j is ready to be determined, we are done. Otherwise we repeat the argument, move forward in B restricted to $\{X_0, \dots, X_M\}$ and (because B is acyclic) within M steps surely find a c_l that is ready to be determined.

We now describe an algorithm that sequentially assigns values to $c_{i|pa_i}$ such that (14) is satisfied. We start with all c_i undetermined and repeat the following steps:

WHILE there exists c_i , $i \in \{0, \dots, M\}$, that is undetermined DO

1. Pick the largest i such that c_i is ready to be determined.
2. Set, for all configurations pa_i of Pa_i , $c_{i|pa_i}$ such that $\sum_{x_i=1}^{n_i} \theta_{x_i|pa_i}^B = 1$ holds.

DONE

The algorithm loops $M + 1$ times and then halts. Step 2 does not affect the values of $c_{j|pa_j}$ for any j, pa_j such that $c_{j|pa_j}$ has already been determined. Therefore, after the algorithm halts, (14) holds.

Let $\Theta^L \in \Theta^L$. Each choice of constants $c_{i|pa_i}$ determines a corresponding vector $\Theta^{(c)}$ with components given by (11). This in turn determines a corresponding vector Θ^B with components given by (13). In Stage 2 we showed that we can take the $c_{i|pa_i}$ such that (14) holds. This is the choice of $c_{i|pa_i}$ which we adopt. With this particular choice, Θ^B indexes a distribution in $\mathcal{M}^B$. By applying the log-transformation to the components of Θ^B we find that for any D of any length, $S^B(D; \Theta^B) = S^L(D; \Theta^{(c)})$, where $S^B(D; \Theta^B)$ denotes the conditional log-likelihood of Θ^B as given by summing the logarithm of (3). The result of Stage 1 now implies that Θ^B indexes the same conditional distribution as Θ^L . Since $\Theta^L \in \Theta^L$ was chosen arbitrarily, this shows that $\mathcal{M}^L \subseteq \mathcal{M}^B$. Together with Theorem 1 this concludes the proof. ■

Proof (sketch) of Theorem 4. Use the rightmost network in Figure 1 with structure $X_0 \rightarrow X_2 \leftarrow X_1$. Let the data be $D = ((1, 1, 1), (1, 1, 2), (2, 2, 1), (2, 2, 2))$. Note that X_0 and X_1 always have the same value. We first show that with this data, there are four local, non-connected suprema of the conditional likelihood.

We are interested in predicting the value of X_0 given X_1 , and X_2 . The parameter defining the distribution of X_1 has no effect on conditional predictions and we can ignore it. For the remaining five parameters we use the following notation:

$$\begin{aligned} \theta_2 &:= P(X_0 = 2), \\ \theta_{2|1,1} &:= P(X_2 = 2 \mid X_0 = 1, X_1 = 1), \\ \theta_{2|1,2} &:= P(X_2 = 2 \mid X_0 = 1, X_1 = 2), \\ \theta_{2|2,1} &:= P(X_2 = 2 \mid X_0 = 2, X_1 = 1), \\ \theta_{2|2,2} &:= P(X_2 = 2 \mid X_0 = 2, X_1 = 2). \end{aligned} \quad (15)$$

The conditional log-likelihood can be written as

$$S^B(D; \Theta^B) = g(1 - \theta_2, \theta_{2|1,1}, \theta_{2|2,1}) + g(\theta_2, \theta_{2|2,2}, \theta_{2|1,2}), \quad (16)$$

where

$$g(x, y, z) := f(x, y, z) + f(x, 1 - y, 1 - z), \quad (17)$$

and

$$f(x, y, z) := \ln \frac{xy}{xy + (1 - x)z}. \quad (18)$$

Figure 3 illustrates functions $g(x, y, z)$ at $x = 0.5$. In (16) each parameter except θ_2 appears only once. Thus, for a fixed θ_2 we can maximize each term separately. From Lemma 1 below it follows that the supremum of the log-likelihood with θ_2 fixed is $\ln(1 - \theta_2) + \ln(\theta_2)$, which achieves its maximum value $-2 \ln 2$ at $\theta_2 = 0.5$. Furthermore, the lemma shows that the log-likelihood approaches its supremum when $\theta_{2|2,1} \in \{0, 1\}$, $\theta_{2|1,2} \in \{0, 1\}$, $\theta_{2|1,1} \rightarrow \theta_{2|2,1}$, and $\theta_{2|2,2} \rightarrow \theta_{2|1,2}$.

Setting $y = 0.5$ results in

$$\sup_{0 \leq z \leq 1} g(x, 0.5, z) = \ln \frac{x}{2 - x} < \ln x. \quad (19)$$

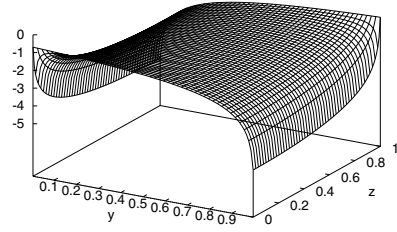


Figure 3: Function $g(x, y, z)$ given by (17) with $x = 0.5$.

Therefore setting either $\theta_{2|1,1}$ or $\theta_{2|2,2}$ to 0.5 results in a smaller supremum of the log-likelihood than the above choices. Consequently, the four suprema are separated by areas where the log-likelihood is smaller, i.e., the suprema are local and not connected.

To conclude the proof we still need to address two issues: (a) the four local suprema give the same conditional log-likelihood $-2 \ln 2$, and (b), they are suprema, not maxima (not achieved by any $\Theta^B \in \Theta^B$). To deal with (a), consider data D' consisting of n_1 repetitions of $(1, 1, 1)$, n_2 repetitions of $(1, 1, 2)$, n_3 repetitions of $(2, 2, 1)$ and n_4 repetitions of $(2, 2, 2)$. By doing a slightly more involved analysis, one can show that, for some choices of n_1, n_2, n_3, n_4 , the supervised log-likelihood still has four suprema, but they have different likelihood values. To deal with (b), let D'' be equal to D' but with four extra data vectors $(1, 2, 1), (2, 1, 1), (1, 2, 2), (2, 1, 2)$. If n_1, n_2, n_3 and n_4 are chosen large enough, the supervised likelihood for D'' has four maxima (rather than suprema), not all of which achieve the same supervised likelihood. We omit further details. ■

Lemma 1. *With $0 < x < 1$ fixed and y and z both varying between 0 and 1, the supremum of $g(x, y, z)$ defined by (17) is given by*

$$\sup_{0 \leq y, z \leq 1} g(x, y, z) = \ln(x). \quad (20)$$

The function approaches its supremum when $z \in \{0, 1\}$, and $y \rightarrow z$. That is, $\lim_{y \downarrow 0} g(x, y, 0) = \lim_{y \uparrow 1} g(x, y, 1) = \ln x$.

Proof. Differentiating twice wrt. z gives

$$\begin{aligned} \frac{\partial^2}{\partial^2 z} g(x, y, z) &= \frac{(1 - x)^2}{(xy + (1 - x)z)^2} \\ &\quad + \frac{(1 - x)^2}{(x(1 - y) + (1 - x)(1 - z))^2}, \end{aligned} \quad (21)$$

which is always positive and the function achieves its maximum values at $z \in \{0, 1\}$. At these two points derivating wrt. y yields

$$\begin{aligned} \frac{\partial}{\partial y} g(x, y, 0) &= \frac{x - 1}{(1 - y)(1 - xy)}, \\ \frac{\partial}{\partial y} g(x, y, 1) &= \frac{1 - x}{y(xy + 1 - x)}. \end{aligned} \quad (22)$$

Since in the first case the derivative is always negative, and in the second case the derivative is always positive, $g(x, y, 0)$ increases monotonically as $y \rightarrow 0$, and $g(x, y, 1)$ increases monotonically as $y \rightarrow 1$. In both cases the limiting value is $\ln(x)$. ■

LEARNING

ENSEMBLES

Monte Carlo Theory as an Explanation of Bagging and Boosting

Roberto Esposito
Università di Torino
C.so Svizzera 185, Torino, Italy
esposito@di.unito.it

Lorenza Saitta
Università del Piemonte Orientale
Spalto Marengo 33, Alessandria, Italy
saitta@mfn.unipmn.it

Abstract

In this paper we propose the framework of Monte Carlo algorithms as a useful one to analyze ensemble learning. In particular, this framework allows one to guess when bagging will be useful, explains why increasing the margin improves performances, and suggests a new way of performing ensemble learning and error estimation.

1 Introduction

Ensemble Learning aims at combining learned hypotheses for classification tasks instead of choosing one. It has proved to be a valuable learning approach, and it is actively investigated in the Machine Learning community [Freund 1995; Freund & Schapire 1997; Breiman, 1996; Wolpert, 1992].

In addition to the advantages pointed out by Dietterich [2000], what really attracts the great interest to ensemble learning is its amazing effectiveness and robustness to overfitting. In fact, ensemble classifiers, and specifically AdaBoost, often show a drop in generalization error far beyond the point in which the training error reaches zero.

In this paper we concentrate on two approaches, namely *Bagging* [Breiman, 1996], and *Boosting*, as implemented in the *AdaBoost* algorithm [Freund & Schapire, 1997], and we propose an analysis of these two approaches by exploiting links with Monte Carlo theory. This parallel allows a deeper understanding of the basic process underlying hypothesis combination. In particular:

- it suggests the conditions under which Bagging is likely to be better than single hypothesis learning,
- it explains why reducing the margin [Schapire *et al.*, 1998] increases the performances,
- it allows some reasons for robustness to overfitting to be conjectured.

The article is organized as follows: in Section 2 we briefly review Bagging and AdaBoost algorithms, and in Section 3 we introduce basic terminology and results

from Monte Carlo theory. Section 4 is the core of the paper: it introduces the link between Monte Carlo theory and Ensemble Learning algorithms, and analyzes the two algorithms in the ideal situation of having access to the whole example and hypothesis spaces. Section 5 relaxes the hypotheses of Section 4. In particular, it focuses on AdaBoost's case offering an intuitive accounting of the generalization performances of the algorithm. Finally, Section 6 concludes the paper and proposes future work.

2 Bagging and AdaBoost

Bagging [Breiman, 1996] is one among the most studied Ensemble Learning algorithms. It takes in input a learning set L_0 , an integer number T , and a (possibly weak) induction algorithm A . At each iteration step t the algorithm creates a bootstrap replicate L [Efron & Tibshirani, 1993] of the original learning set, and uses it in conjunction with A to form a new hypothesis h_t . When T hypotheses have been learned, they are combined using a simple majority voting rule.

Breiman [1996] found a significant increase of classification accuracy, when the hypotheses are bagged, on different datasets from the UCI repository, for both regression and classification tasks. Moreover, in the same paper, a theoretical analysis is carried on. The notion of *order correct* classifier is introduced, and it is shown that when A is order correct, then Bagging is expected to behave almost optimally (i.e., it behaves like the Bayes classifier). Throughout the paper it is claimed that “*bagging stable classifiers is not a good idea*”. A “stable” classifier is qualitatively defined as one that does not show large variations in the learned hypotheses when the training set undergoes small variations. Experimental analysis of Bagging's performances is provided by Bauer and Kohavi [1999], Quinlan [1996], and Dietterich [2000].

The simplest version of AdaBoost [Freund & Schapire, 1997] deals with binary classification problems, and this is the one we refer to. AdaBoost maintains a weight distribution over the training examples, and adjusts it at each iteration. The adjusted weights are chosen so that

higher values are associated to previously misclassified examples. As a main effect, the weighting scheme forces $\mathbf{A}$ to focus on “more difficult” examples, ensuring thus that all the examples will be, sooner or later, taken into account by the weak learner. The hypothesis combination rule is a weighted majority vote that puts more weight on successful hypotheses (i.e., hypotheses that incur a low weighted training error).

Despite its apparent simplicity AdaBoost solves a difficult problem: choosing example and hypothesis weights in such a way to ensure an exponential decrease in the training error committed by the boosted committee. Moreover, the choice is made without any knowledge or assumption on the weak learner itself (except that the weak learner is always able to outperform random guessing). In contrast, Bagging only works when “benign” weak learners are used. On the other hand, AdaBoost seems to be more sensitive to noise than Bagging [Quinlan, 1996].

As mentioned earlier, AdaBoost shows the characteristic of being robust with respect to overfitting [Schapire et al., 1998; Drucker & Cortes, 1996; Quinlan, 1996; Breiman, 1998; Freund & Schapire, 1998]. A rather convincing account of this behavior imputes to AdaBoost the ability to be very successful in increasing the final classifier’s margin over the training examples [Schapire et al., 1998].

In the effort of understanding and/or explaining AdaBoost’s behavior, boosting has been related to other existing theories, such as Game Theory [Schapire, 2001], logistic regression [Friedman et al., 1998; Collins et al., 2000], optimization [Rätsch et al., 2002], and Brownian motion [Freund, 1995].

3 Monte Carlo Algorithms

In this section we recall some basic notions from Monte Carlo algorithms theory. In the literature, the name Monte Carlo often denotes a generic stochastic algorithm; in this paper we will refer to the more restrictive definition given by Brassard and Bratley [1988]. Let Π be a class of problems, Y a finite set of answers for the problems in Π , and $c : \Pi \rightarrow 2^Y$ be a function that maps each problem into the set of its correct answers.

Definition 1 – A stochastic algorithm $\mathbf{A}$, mapping Π into Y , is said to be a *Monte Carlo* algorithm, if, when applied to an instance π of Π , it always terminates, providing an answer to π , which may be occasionally incorrect.

Two properties of Monte Carlo algorithms turn out to be relevant:

Definition 2 – A Monte Carlo algorithm is *consistent* if it never outputs two distinct *correct* answers when run two times on the same problem instance π .

Definition 3 – A Monte Carlo algorithm is *p-correct* if the probability that it gives a correct answer to π is at least p , independently of the specific problem instance π considered.

The great interest in Monte Carlo algorithms resides in their *amplification* ability: given a problem instance π , and a Monte Carlo algorithm MC , if MC is run multiple times on π and the majority answer is taken as the result, then, under mild assumptions, the probability of a correct answer exponentially increases. The conditions are that MC must be consistent for π and that p must be greater than random guess. We note that, whenever $c(\pi)$ reduces to a mapping from Π to Y (instead of 2^Y), the algorithm is consistent. Let us denote by MC_T the algorithm obtained by combining with a majority vote the answers given by multiple runs of a Monte Carlo MC .

4 Complete Information Case

In this section we provide a theoretical account of the links between Monte Carlo theory and bagging or boosting, in the case we have complete information on the learning/classification problem. More precisely, let $\mathbf{X}$ be a finite set of examples¹, $\square$ a target concept (binary classification), $\mathbf{A}$ a (weak) learner. Let $|\mathbf{X}| = N$ and $\mathbf{w}$ be a probability distribution over $\mathbf{X}$. Let moreover $\square$ be the power set of $\mathbf{X}$. Any intensional concept description, belonging to a given language $\mathcal{L}$, will be mapped to an element $\square$ of $\square$, i.e., $\square$ is the set of examples classified by that description as belonging to $\square$. In this way, we can reduce ourselves to a finite set of (extensional) hypotheses. Let $\square \subseteq \square$ be any subset of $\square$. Let $|\square| = K$. A probability distribution $\mathbf{q}$ is associated to the elements of $\square$. The set $\square$ can be thought of as the collection of the hypotheses generated by a learning algorithm $\mathbf{A}$ applied to a number of training sets, or a set of hypotheses provided by an oracle. Learning can then be simulated by extracting with replacement elements from $\square$ according to $\mathbf{q}$ (each extraction corresponds to a run of $\mathbf{A}$ on some training set).

Let us represent the described situation by means of Table 1. Here, let $p_j(x_k) \in \{1,0\}$ ² be the probability that hypothesis $\square_j$ correctly classifies example x_k . Let $p(x_k)$ be the average of such probabilities for x_k . Let r_j be the accuracy of hypothesis $\square_j$ over the whole $\mathbf{X}$.

By using the given probability distributions, we can compute the average accuracy for the whole table:

$$r = \sum_{k=1}^N w_k p(x_k) = \sum_{j=1}^K q_j r_j = \sum_{j=1}^K \sum_{k=1}^N q_j w_k p_j(x_k) \quad (1)$$

¹ The finiteness of $\mathbf{X}$ is not essential; it is only assumed for the sake of simplicity.

² This assumption implies that the Bayes error is zero.

Table 1 – Theoretical setting

	$\square_1$		$\square_j$		$\square_K$	
$\mathbf{x}_1$	$p_1(x_1)$		$p_j(x_1)$		$p_K(x_1)$	$p(x_1)$
....						
$\mathbf{x}_k$	$p_1(x_k)$		$p_j(x_k)$		$p_K(x_k)$	$p(x_k)$
....						
$\mathbf{x}_N$	$p_1(x_N)$		$p_j(x_N)$		$p_K(x_N)$	$p(x_N)$
	r_1		r_j		r_K	r

Given a subset $\mathbf{Z}$ of $\mathbf{X}$, let $\|\mathbf{Z}\|$ the measure of $\mathbf{Z}$ w.r.t. $\mathbf{w}$, i.e.:

$$\|\mathbf{Z}\| = \sum_{x_k \in \mathbf{Z}} w_k$$

Analogously, if $\square' \subseteq \square$, we define:

$$\|\square'\| = \sum_{\square_j \in \square'} q_j$$

Clearly the measures $\|\mathbf{Z}\|$ and $\|\square'\|$ reduce to $|\mathbf{Z}|/N$ and $|\square'|/K$, respectively, if $\mathbf{q}$ and $\mathbf{w}$ are uniform distributions.

4.1 Monte Carlo and Bagging

We will now establish a link between Monte Carlo algorithms and Bagging. Let L_0 be a training set, and T an integer; let L be a generic bootstrap replicate of L_0 . In this case, $\square$ is the set of all the different hypotheses that can be obtained by applying $\mathbf{A}$ to L_0 's replicates. The frequencies with which each $\square_j$ is generated determine $\mathbf{q}$.

If we look at Table 1, Bagging proceeds, so to say, *by columns*, by extracting T columns from $\square$ and using them to obtain a majority vote on each x_k . Considering the global average accuracy r , or even the exact distribution of the r_j , it is not easy to say how many of the x_k will be correctly classified by the majority voting procedure, and, then, whether to perform Bagging will be rewarding. It is generally said [Breiman, 1996] that bagging is useful when $\mathbf{A}$ is “unstable”. We will show later that this notion of “instability” can be made more precise.

Let us consider now a Monte Carlo interpretation of Table 1. To the contrary of Bagging, a Monte Carlo algorithm MC_T proceeds *by rows*: for each x_k (problem to be solved), it extracts T hypotheses $\square_j$ from $\square$ and classifies x_k by majority voting. This process is repeated for each row³. This procedure is necessary, because the components of the Monte Carlo voting must be independent, but the entries in each column are not. Monte Carlo theory tells us that the correctness of MC_T on x_k can be amplified if:

1. The T trials are independent
2. MC is consistent
3. $p(x_k) > 1/2$

Condition (1) is true by construction of the process (extraction with replacement of T hypotheses for each row).

³ Let us set apart, for the moment, the computational issues.

Condition (2) is true as long as the Bayes error is zero. Condition (3) may or may not be true, depending on the set $\square$.

Let $\mathbf{X}_a$ be the set of examples for which $p(x_k) > 1/2$. According to the Monte Carlo theory, all these examples will be “amplified”, i.e., they will be correctly classified in the limit (for $T \rightarrow \infty$). The application of MC_T to all the rows yields an asymptotic accuracy:

$$\square = \|\mathbf{X}_a\| \quad (2)$$

By denoting by $\square_\square$ the expected Monte Carlo accuracy when only a finite number T of extractions is performed, we obtain:

$$\lim_{T \rightarrow \infty} \Pr\{\square_T = \square\} = 1 \quad (3)$$

Let now $\square_b$ be the set of hypotheses whose accuracy is greater than $\square$. The value

$$\square = \|\square_b\| \quad (4)$$

is the probability that a hypothesis “better” than bagging is obtained in a single extraction. Then, in an infinite number of trials, such a hypothesis will occur with a frequency $\square$. It is clear that, in the limit, Bagging is surely convenient only when when $\square_b = \emptyset$, because, in this case, there is no single hypothesis better than Bagging. When $\square_b \neq \emptyset$ a better hypothesis will surely appear, soon or later. The interesting case, however, occurs when only a finite number T of extractions is performed. Before approaching this case, let us make some considerations on what has been described so far.

Let us first notice that ensemble learning claims that amplification occurs when the r_j 's (accuracy of each of the combined hypotheses) are greater than 1/2 (condition on the columns). Actually it may not be the case. What is really required is that, for each example x_k , more than 1/2 of the learned hypotheses are correct on it (condition on the rows). Then, Monte Carlo Theory suggests that Bagging would be successful when it is possible to find a subset of $\square$ such that $\mathbf{X}_a = \mathbf{X}$. Before looking with more details into the relationship between the r_j 's and the $p(x_k)$'s, let us consider some extreme cases.

Let $\mathbf{A}$ be a weak learner that always returns the same hypothesis $\square$, regardless of the training set. The lucky situation in which that single hypothesis $\square$ is quite accurate over the training set does not imply anything about the occurrence of Monte Carlo amplification. In fact, all the $p(x_k)$ are equal to 1 on those examples which are correctly classified by $\square$, and 0 otherwise. In other words, whatever the accuracy of $\square$, no amplification at all occurs. This is an example of a perfectly stable learner.

Let us consider a situation in which the global average r of the accuracy is a little more than 1/2. This value is at the same time the average of the last column and on the last row in Table 1, according to (1). For the sake of illustration, let $\mathbf{q}$ and $\mathbf{w}$ be uniform distributions. Let us consider the following three extreme situations (a de-

tailed analysis of several other interesting cases is provided by Esposito [2003]) :

- (a) The $p(x_k)$'s in the last column are distributed in such a way that one half of the examples have $p(x_k)$ values close to one, and one half of the examples have $p(x_k)$ values close to zero. The r_j 's in the last row, on the contrary, have all almost the same value around 1/2. (All hypotheses show the same performances and they cover almost the same examples)
- (b) The $p(x_k)$'s in the last column have all almost the same value around 1/2, whereas the r_j 's in the last row have, for one half, values close to one, and for the other half close to zero. (Some hypotheses are very good and other are very bad, but their coverage is uniformly distributed on the examples).
- (c) Both distributions are uniformly valued around 1/2 (All the hypotheses are uncertain on all examples).

In case (a) half of the examples are amplifiable, but Bagging is not useful, because any single hypothesis classifies correctly half of the examples, so that nothing can be gained by the Bagging process.

In case (b), the number of amplifiable examples depends on how many of them have actually values a little over 1/2. But, even in the best case ($\mathbf{X}_a = \mathbf{X}$, i.e., $\square = 1$), Bagging is again not convenient. In fact, half of the single hypotheses have $r_j = 1$; then, in one over two extractions (on average) a perfect hypothesis is extracted.

In case (c), if the $p(x_k)$ values in the last columns are all a little greater than 1/2, then $\square = 1$. This is the case in which Bagging is convenient, because the single hypothesis have all accuracy close to 1/2. Actually, case (c) is the ideal case for Bagging, because amplification can be obtained with the minimum effort (with "bad" hypotheses we may obtain a large amplification).

The following theorem summarizes the above considerations and establishes a link between Bagging and Monte Carlo amplification (proof is provided by Esposito [2003]):

Theorem 1 – The amplified accuracy $\square$ cannot be greater than $2r$, where r is the average accuracy of the bagged hypotheses. $\square$

Then, even by bagging single hypotheses with accuracy less than 1/2 it is possible to obtain higher accuracies by Monte Carlo amplification. However, in order to reach $\square = 1$, the value of r must be at least 1/2. Even in this case, some of the combined hypotheses are allowed to have a accuracy less than 1/2, without hindering amplification, provided that $\mathbf{X}_a = \mathbf{X}$ and that r does not go below 1/2.

4.2 Monte Carlo and AdaBoost

Let us notice that, if the set $\square$ of hypotheses coincides with the power set $2^{\mathbf{X}}$ (all possible hypotheses are considered), we are in the situation in which all $p(x_k) = 1/2$,

$r = 1/2$, and $\square = 1/2$, so that no amplification is possible. This means that the learning procedure was not able to make any useful selection inside the hypothesis space. On the contrary, we would like that the weak learner has the nice property of focusing on the best hypotheses. In the previous section we showed that the basic Monte Carlo approach, implemented by Bagging, may fail due to a number of reasons; this happens, for instance, in the cases (a) and (b) of Section 4.1. Even if not explicitly stated, the underling problem is that the weak learner may not be suited enough for this kind of procedure. The question we are now trying to investigate is what to do when such a situation occurs, in particular when we do not want or can modify the weak learner. Then, is it possible to convert case (a) or (b) into case (c)? Or, said in other words, is it possible to convert a learning problem in which the Monte Carlo assumptions for amplification do not hold into one in which they do?

The solution to this problem can be found by thinking again of the role of the weak learner in the Monte Carlo process. As mentioned earlier, the learner has the duty to modify the distribution from which the hypotheses are drawn (i.e., the q_j of the columns) in such a way that "better" hypotheses are more likely to be drawn. Moreover, we can actually change the behavior of the weak learner through its input, i.e., by modifying the training set. In particular, we have to force the weak learner to extract hypotheses that make the minimum value of the $p(x_k)$'s grow larger than 1/2 (in order to satisfy the Monte Carlo conditions of amplifiability for all the examples). Since the weak learner is influenced by the training set, the solution is to modify the training set in such a way that examples with low $p(x_k)$ values are more represented, in the attempt to make the weak learner try harder to be successful on them. What we obtain with this reasoning is an AdaBoost-like algorithm. In other words, when the weak learner is unable to satisfy Monte Carlo conditions for amplification, AdaBoost forces it to satisfy them by weighting the examples. More precisely, AdaBoost tries to increase the minimum $p(x_k)$ with the unconscious goal of extending $\mathbf{X}_a$. In fact, when the weak learner satisfies relatively mild conditions, it has been shown by Schapire et al. [1998] that AdaBoost is actually effective in increasing the minimal *margin*:

Definition 4 – For any particular example x_k , the classification *margin* $\mu(x_k)$ for x_k is the difference between the weight assigned to the correct label and the maximal weight assigned to any single incorrect label.

In the case of binary classification, the margin is the difference between the weight assigned to the correct label and the weight assigned to the incorrect one.

Theorem 2 – In the case of binary classification, the following relation holds between the margin and the Monte Carlo $p(x_k)$ -correctness value:

$$\mu(x_k) = p(x_k) - (1 - p(x_k)) = 2p(x_k) - 1 \quad (7)$$

The proof of Theorem 2 derives immediately from the definition of margin and $p(x_k)$. Formula (7) explains why increasing the margin improves the performances of boosting: in fact, increasing the minimum margin lets the cardinality of the set X_a increase, augmenting the number of examples for which the Monte Carlo conditions for amplifiability hold.

4.3 Non-Asymptotic Case

Let us consider a finite integer T . Let $\square_R$ denote the probability that a hypothesis “better” than Bagging is obtained in R extractions. In order to compare Bagging with hypothesis selection, let us consider a classification accuracy $\square$. Given Table 1, and given a number $\square$ close to 0, let $T(\square)$ be the minimum number of components of the bagged hypothesis such that:

$$\Pr\{\square_{T(\square)} \geq \square\} \geq 1 - \square \quad (8)$$

If $\square > \square$, then probability (8) is zero.

Analogously, let $\square_{\square}$ be the set of hypotheses with $r_j > \square$ and $R(\square)$ be the minimum number of hypothesis extractions such that:

$$\Pr\{\text{Extracting } \square_j \square \square_{\square}\} \geq 1 - \square \quad (9)$$

Again, if $\square_{\square} = \emptyset$, probability (9) is zero.

We can draw a graph of T (ordinate) vs. R (abscissa), parameterized by $\square$. The graph is problem dependent. What we can say in general is that points in the graph above the diagonal correspond to situation for which selection is better than Bagging (for the same accuracy $\square$, Bagging requires more trials), points below the diagonal correspond to situation for which selection is worse than Bagging (for the same accuracy $\square$, Bagging requires less trials), and points on the diagonal correspond to situations of indifference.

5 Problems with Partial Information

The setting discussed in Section 4 is useful for understanding the functioning of ensemble learning, but we need to extend it to the case of real learning problems. In concrete situation, neither the set $\square$ nor the example space X is available *a priori*. What we have is a training set L_0 ($|L_0| = M$) and a test set T_0 , both subsets of X , and a learning algorithm A for generating hypotheses. We learn hypotheses, possibly consistent with L_0 , and estimate the generalization error on T_0 ⁴.

If we consider the problem from the Monte Carlo perspective and we go back to Table 1, we can think to proceed as follows. At the beginning we have, on the rows, the training and the test examples, but the columns are empty. Let us consider the first example $x_1 \square L_0$. Let A run on T replicates of L_0 and let us compute $p_e(x_1)$, which

is an unbiased estimate of the Bernoulli probability $p(x_1)$. Even though the T hypotheses generated are only used to classify x_1 , and could then be discarded, nothing hinders from recording them in the first T (or less, if there is duplication) columns of Table 1. When we consider x_2 , we repeat the whole procedure, running again A on T new replicates of L_0 and classifying x_2 . Again we record the new generated hypotheses (or update the probability distribution q in case of repetitions). When we finish examining L_0 , we have also filled the part of the table corresponding to the test set, which allows estimates $p_e(x_i)$ for the examples in the test set to be computed. Each $p_e(x_i)$ is based on $T M$ experimental values, which is the number of runs necessary to estimate the generalization accuracy using the leave-one-out method on a number M of training examples.

What is interesting is that, during learning, we also *learn* whether it is convenient or not to exploit Bagging for classifying future unseen examples. It also turns out, experimentally⁵, that a much lower number of runs than $T \cdot M$ is very often sufficient to obtain a stable estimate of the sought probabilities.

5.1 AdaBoost’s generalization behavior

In this section we discuss AdaBoost’s generalization capabilities and their interpretation in the Monte Carlo framework. Having related the margin of an example x_k with the average probability that it is correctly classified by the hypotheses learned via the weak learner allows us to transfer results provided by Schapire et al. [1998] to the Monte Carlo framework. Interestingly enough, the link may be used to give a new, more intuitive, interpretation of the margin explanation of AdaBoost’s generalization capabilities. Moreover a nice explanation of why the test error is often observed to decrease even after the training error reaches zero can be suggested.

To understand how generalization error can be interpreted in the Monte Carlo framework it is necessary to deepen a little the discussion about the link between ensemble learning and Monte Carlo algorithms. Let us choose the randomization procedure (whatever it is). Once this is done, the choice of the weak learner determines the distribution of the $p(x_k)$ ’s. In other words, when the two major parameters of the ensemble algorithm are fixed, we can think of $p(x_k)$ as a theoretical property of the examples.

Once the ensemble process starts, we compute the estimations $p_e(x_k)$. The accuracy of the estimates increases with the number of iterations performed. Whether the final rule will classify correctly or not any example, is a matter of two things: the value of the underlying exact

⁴ In this paper we do not consider to use cross-validation or leave-one-out.

⁵ The datasets used, as well as part of the experimental results, can be found at the URL:
<http://www.di.unito.it/~esposito/mcandboost>

probability $p(x_k)$, and how lucky we were in the choice of the rules used to perform the actual estimation.

The above argument might explain the finding of the test error decreasing, for AdaBoost, beyond the point in which training error reaches zero. In fact, as an effect of the learning bias introduced by the weak algorithm, it is likely that the $p(x_k)$'s corresponding to examples in the training set would be greater than the ones corresponding to the other examples. It is then likely that those examples would be correctly classified with high probability after a moderate number of iterations. On the contrary, other examples would require a greater number of iterations, and thus they may start to be correctly labelled well beyond the point where the training error reaches zero. In fact, even though any example with $p(x_k) > 1/2$ will eventually be correctly classified, the number of iterations necessary to see this effect to emerge strongly depends on T . The observed effect of the test error decreasing is hence due to those examples for which $p(x_k)$ is only slightly greater than $1/2$. Initially, their $p(x_k)$ is possibly underestimated by the observed $p_e(x_k)$. For small values of T , $p_e(x_k)$ can be quite different from $p(x_k)$ and greater values of T are necessary to let the choice of hypotheses converge toward a set for which $p_e(x_k) \square p(x_k)$ and, hence, to obtain a stable correct classification of x_k in the test set.

6 Conclusions and Further Work

In this paper we have proposed to analyze ensemble learning (in particular Bagging and, partly, AdaBoost) from the perspective of Monte Carlo algorithm theory. We have shown that this theory can shed light on some basic aspects of Bagging and boosting, and, also offers a new way to actually perform ensemble learning and error estimation. More research is nevertheless needed to deepen our understanding of the presented framework. In particular, it would be very interesting to investigate the relationships between the Monte Carlo explanation of Ensemble Learning and other well known ones (for instance, the bias-variance decomposition). Moreover, it would be very interesting to investigate how a particularly aggressive class of Monte Carlo algorithms, namely the one of biased Monte Carlos, relates to ensemble learning.

References

- [Bauer and Kohavi, 1999] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105-139.
- [Brassard and Bratley, 1988] G. Brassard and P. Bratley. *Algorithmics: theory and practice*. Prentice-Hall, Inc., 1988.
- [Breiman, 1996] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123-140, 1996.
- [Breiman, 1998] L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801- 849, 1998.
- [Collins *et al.*, 2000] M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. In *Computational Learning Theory*, pages 158-169, 2000.
- [Dietterich, 2000] T. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1-15, 2000.
- [Drucker and Cortes, 1996] Drucker and Cortes, Boosting decision trees. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems, Vol. 8*, (pp. 479-485), The MIT Press, 1996.
- [Efron and Tibshirani, 1993] B. Efron and R. Tibshirani. *An introduction to the Bootstrap*. Chapman and Hall, 1993.
- [Esposito, 2003] R. Esposito. *Analyzing Ensemble Learning in the Framework of Monte Carlo Theory*. Ph.D. Thesis, University of Torino, Dept. of Informatica, Italy.
- [Freund, 1995]. Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 2(121):256-285, 1995.
- [Freund and Shapire, 1997] Y. Freund and R. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119-139, 1997.
- [Freund and Shapire, 1998] Y. Freund and R. Schapire. Discussion of the paper "arcing classifiers" by Leo Breiman. *The Annals of Statistics*, 26(3):824-832, 1998.
- [Friedman *et al.*, 1998] J. Friedman, T. Hastie, and R. Tibshirani. *Additive logistic regression: a statistical view of boosting*. Technical report, Department of Statistics, Sequoia Hall, Stanford University, July 1998.
- [Quinlan, 1996] R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 725-730, 1996. AAAI Press / MIT Press
- [Rätsch *et al.*, 2002] G. Rätsch, M. Warmuth, S. Mika, T. Onoda, S. Lemm_ and K. Müller. Barrier boosting. In *13th Conference on Computational Learning Theory*, 2000.
- [Schapire, 2001] R. Schapire. Drifting games. *Machine Learning* 43(3): 265-291, 2001.
- [Schapire *et al.*, 1998] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistic*, 26(5):1651-1686, 1998.
- [Wolpert, 1992] D. Wolpert. Stacked Generalization, *Neural Networks*, 5, 241-259, 1992.

Constructing Diverse Classifier Ensembles using Artificial Training Examples

Prem Melville and Raymond J. Mooney

Department of Computer Sciences

University of Texas

1 University Station, C0500

Austin, TX 78712

melville@cs.utexas.edu, mooney@cs.utexas.edu

Abstract

Ensemble methods like bagging and boosting that combine the decisions of multiple hypotheses are some of the strongest existing machine learning methods. The diversity of the members of an ensemble is known to be an important factor in determining its generalization error. This paper presents a new method for generating ensembles that directly constructs diverse hypotheses using additional artificially-constructed training examples. The technique is a simple, general meta-learner that can use any strong learner as a base classifier to build diverse committees. Experimental results using decision-tree induction as a base learner demonstrate that this approach consistently achieves higher predictive accuracy than both the base classifier and bagging (whereas boosting can occasionally decrease accuracy), and also obtains higher accuracy than boosting early in the learning curve when training data is limited.

1 Introduction

One of the major advances in inductive learning in the past decade was the development of *ensemble* or *committee* approaches that learn and retain multiple hypotheses and combine their decisions during classification [Dietterich, 2000]. For example, *boosting* [Freund and Schapire, 1996], an ensemble method that learns a series of “weak” classifiers each one focusing on correcting the errors made by the previous one, has been found to be one of the currently best generic inductive classification methods [Hastie *et al.*, 2001].

Constructing a *diverse* committee in which each hypothesis is as different as possible (decorrelated with other members of the ensemble) while still maintaining consistency with the training data is known to be a theoretically important property of a good committee [Krogh and Vedelsby, 1995]. Although all successful ensemble methods encourage diversity to some extent, few have focused directly on the goal of maximizing diversity. Existing methods that focus on achieving diversity [Opitz and Shavlik, 1996; Rosen, 1996] are fairly complex and are not general *meta-learners* like bagging [Breiman, 1996] and boosting that can be applied to any base learner to produce an effective committee [Witten and Frank, 1999].

We present a new meta-learner (DECORATE, Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples) that uses an existing “strong” learner (one that provides high accuracy on the training data) to build an effective diverse committee in a fairly simple, straightforward manner. This is accomplished by adding different randomly constructed examples to the training set when building new committee members. These artificially constructed examples are given category labels that *disagree* with the current decision of the committee, thereby easily and directly increasing diversity when a new classifier is trained on the augmented data and added to the committee.

Boosting and bagging provide diversity by sub-sampling or re-weighting the existing training examples. If the training set is small, this limits the amount of ensemble diversity that these methods can obtain. DECORATE ensures diversity on an arbitrarily large set of additional artificial examples. Therefore, one hypothesis is that it will result in higher generalization accuracy when the training set is small. This paper presents experimental results on a wide range of UCI data sets comparing boosting, bagging, and DECORATE, all using J48 decision-tree induction (a Java implementation of C4.5 [Quinlan, 1993] introduced in [Witten and Frank, 1999]) as a base learner. Cross-validated learning curves support the hypothesis that “DECORATED trees” generally result in greater classification accuracy for small training sets.

2 Ensembles and Diversity

In an ensemble, the combination of the output of several classifiers is only useful if they disagree on some inputs [Krogh and Vedelsby, 1995]. We refer to the measure of disagreement as the *diversity* of the ensemble. There have been several methods proposed to measure ensemble diversity [Kuncheva and Whitaker, 2002] — usually dependent on the measure of accuracy. For regression, where the mean squared error is commonly used to measure accuracy, variance can be used as a measure of diversity. So the diversity of the i^{th} classifier on example x can be defined as $d_i(x) = [C_i(x) - C^*(x)]^2$, where $C_i(x)$ and $C^*(x)$ are the predictions of the i^{th} classifier and the ensemble respectively. For this setting Krogh *et al* [1995] show that the generalization error, E , of the ensemble can be expressed as $E = \bar{E} - \bar{D}$, where $\bar{E}$ and $\bar{D}$ are the mean error and diversity of the ensemble respectively.

For classification problems, where the 0/1 loss function is most commonly used to measure accuracy, the diversity of the i^{th} classifier can be defined as:

$$d_i(x) = \begin{cases} 0 & \text{if } C_i(x) = C^*(x) \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

However, in this case the above simple linear relationship does not hold between E , $\bar{E}$ and $\bar{D}$. But there is still strong reason to believe that increasing diversity should decrease ensemble error [Zenobi and Cunningham, 2001]. The underlying principle of our approach is to build ensembles of classifiers that are consistent with the training data and maximize diversity as defined in (1).

3 DECORATE: Algorithm Definition

In DECORATE (see Algorithm 1), an ensemble is generated iteratively, learning a classifier at each iteration and adding it to the current ensemble. We initialize the ensemble to contain the classifier trained on the given training data. The classifiers in each successive iteration are trained on the original training data and also on some artificial data. In each iteration artificial training examples are generated from the data distribution; where the number of examples to be generated is specified as a fraction, R_{size} , of the training set size. The labels for these artificially generated training examples are chosen so as to differ maximally from the current ensemble's predictions. The construction of the artificial data is explained in greater detail in the following section. We refer to the labeled artificially generated training set as the *diversity data*. We train a new classifier on the union of the original training data and the diversity data. If adding this new classifier to the current ensemble increases the ensemble training error, then we reject this classifier, else we add it to the current ensemble. This process is repeated until we reach the desired committee size or exceed the maximum number of iterations.

To classify an unlabeled example, x , we employ the following method. Each base classifier, C_i , in the ensemble C^* provides probabilities for the class membership of x . If $P_{C_i,y}(x)$ is the probability of example x belonging to class y according to the classifier C_i , then we compute the class membership probabilities for the entire ensemble as:

$$P_y(x) = \frac{\sum_{C_i \in C^*} P_{C_i,y}(x)}{|C^*|}$$

where $P_y(x)$ is the probability of x belonging to class y . We then select the most probable class as the label for x i.e. $C^*(x) = \operatorname{argmax}_{y \in Y} P_y(x)$

3.1 Construction of Artificial Data

We generate artificial training data by randomly picking data points from an approximation of the training-data distribution. For a numeric attribute, we compute the mean and standard deviation from the training set and generate values from the Gaussian distribution defined by these. For a nominal attribute, we compute the probability of occurrence of each distinct value in its domain and generate values based on this distribution. We use Laplace smoothing so that nominal attribute values not represented in the training set still have a

non-zero probability of occurrence. In constructing artificial data points, we make the simplifying assumption that the attributes are independent. It is possible to more accurately estimate the joint probability distribution of the attributes; but this would be time consuming and require a lot of data.

In each iteration, the artificially generated examples are labeled based on the current ensemble. Given an example, we first find the class membership probabilities predicted by the ensemble, replacing zero probabilities with a small non-zero value. Labels are then selected, such that the probability of selection is inversely proportional to the current ensemble's predictions.

Algorithm 1 The DECORATE algorithm

Input:

BaseLearn - base learning algorithm

T - set of m training examples $\langle (x_1, y_1), \dots, (x_m, y_m) \rangle$ with labels $y_j \in Y$

C_{size} - desired ensemble size

I_{max} - maximum number of iterations to build an ensemble

R_{size} - factor that determines number of artificial examples to generate

1. $i = 1$
 2. $trials = 1$
 3. $C_i = \text{BaseLearn}(T)$
 4. Initialize ensemble, $C^* = \{C_i\}$
 5. Compute ensemble error, $\epsilon = \frac{\sum_{x_j \in T : C^*(x_j) \neq y_j} 1}{m}$
 6. While $i < C_{size}$ and $trials < I_{max}$
 7. Generate $R_{size} \times |T|$ training examples, R , based on distribution of training data
 8. Label examples in R with probability of class labels inversely proportional to C^* 's predictions
 9. $T = T \cup R$
 10. $C' = \text{BaseLearn}(T)$
 11. $C^* = C^* \cup \{C'\}$
 12. $T = T - R$, remove the artificial data
 13. Compute training error, ϵ' , of C^* as in step 5
 14. If $\epsilon' \leq \epsilon$
 15. $i = i + 1$
 16. $\epsilon = \epsilon'$
 17. otherwise,
 18. $C^* = C^* - \{C'\}$
 19. $trials = trials + 1$
-

4 Experimental Evaluation

4.1 Methodology

To evaluate the performance of DECORATE we ran experiments on 15 representative data sets from the UCI repository [Blake and Merz, 1998] used in similar studies [Webb, 2000;

Quinlan, 1996]. We compared the performance of DECORATE to that of Adaboost, Bagging and J48, using J48 as the base learner for the ensemble methods and using the Weka implementations of these methods [Witten and Frank, 1999]. For the ensemble methods, we set the ensemble size to 15. Note that in the case of DECORATE, we only specify a maximum ensemble size, the algorithm terminates if the number of iterations exceeds the maximum limit even if the desired ensemble size is not reached. For our experiments, we set the maximum number of iterations in DECORATE to 50. We ran experiments varying the amount of artificially generated data, R_{size} ; and found that the results do not vary much for the range 0.5 to 1. However, R_{size} values lower than 0.5 do adversely affect DECORATE, because there is insufficient artificial data to give rise to high diversity. The results we report are for R_{size} set to 1, i.e. the number of artificially generated examples is equal to the training set size.

The performance of each learning algorithm was evaluated using 10 complete 10-fold cross-validations. In each 10-fold cross-validation each data set is randomly split into 10 equal-size segments and results are averaged over 10 trials. For each trial, one segment is set aside for testing, while the remaining data is available for training. To test performance on varying amounts of training data, learning curves were generated by testing the system after training on increasing subsets of the overall training data. Since we would like to summarize results over several data sets of different sizes, we select different *percentages* of the total training-set size as the points on the learning curve.

To compare two learning algorithms across all domains we employ the statistics used in [Webb, 2000], namely the win/draw/loss record and the geometric mean error ratio. The win/draw/loss record presents three values, the number of data sets for which algorithm A obtained better, equal, or worse performance than algorithm B with respect to classification accuracy. We also report the *statistically significant* win/draw/loss record; where a win or loss is only counted if the difference in values is determined to be significant at the 0.05 level by a paired t -test. The geometric mean error ratio is defined as $\sqrt[n]{\prod_{i=1}^n \frac{E_A}{E_B}}$, where E_A and E_B are the mean errors of algorithm A and B on the same domain. If the geometric mean error ratio is less than one it implies that algorithm A performs better than B , and vice versa. We compute error ratios so as to capture the degree to which algorithms out-perform each other in win or loss outcomes.

4.2 Results

Our results are summarized in Tables 1-3. Each cell in the tables presents the accuracy of DECORATE versus another algorithm. If the difference is statistically significant, then the larger of the two is shown in bold. We varied the training set sizes from 1-100% of the total available data, with more points lower on the learning curve since this is where we expect to see the most difference between algorithms. The bottom of the tables provide summary statistics, as discussed above, for each of the points on the learning curve.

DECORATE has more *significant* wins to losses over Bagging for all points along the learning curve (see Table 2).

DECORATE also outperforms Bagging on the geometric mean ratio. This suggests that even in cases where Bagging beats DECORATE the improvement is less than DECORATE's improvement on Bagging on the rest of the cases.

DECORATE outperforms AdaBoost early on the learning curve both on significant wins/draw/loss record and geometric mean ratio; however, the trend is reversed when given 75% or more of the data. Note that even with large amounts of training data, DECORATE's performance is quite competitive with Adaboost - given 100% of the data DECORATE produces higher accuracies on 6 out of 15 data sets.

It has been observed in previous studies [Webb, 2000; Bauer and Kohavi, 1999] that while AdaBoost usually significantly reduces the error of the base learner, it occasionally increases it, often to a large extent. DECORATE does not have this problem as is clear from Table 1.

On many data sets, DECORATE achieves the same or higher accuracy as Bagging and AdaBoost with many fewer training examples. Figure 1 show learning curves that clearly demonstrate this point. Hence, in domains where little data is available or acquiring labels is expensive, DECORATE has an advantage over other ensemble methods.

We performed additional experiments to analyze the role that diversity plays in error reduction. We ran DECORATE at 10 different settings of R_{size} ranging from 0.1 to 1.0, thus varying the diversity of ensembles produced. We then compared the diversity of ensembles with the reduction in generalization error. Diversity of an ensemble is computed as the mean diversity of the ensemble members (as given by Eq. 1). We compared ensemble diversity with the *ensemble error reduction*, i.e. the difference between the average error of the ensemble members and the error of the entire ensemble (as in [Cunningham and Carney, 2000]). We found that the correlation coefficient between diversity and ensemble error reduction is 0.6225 ($p^1 \ll 10^{-50}$), which is fairly strong. Furthermore, we compared diversity with the *base error reduction*, i.e. the difference between the error of the base classifier and the ensemble error. The base error reduction gives a better indication of the improvement in performance of an ensemble over the base classifier. The correlation of diversity versus the base error reduction is 0.1552 ($p \ll 10^{-50}$). We note that even though this correlation is weak, it is still a *statistically significant* positive correlation. These results reinforce our belief that increasing ensemble diversity is a good approach to reducing generalization error.

To determine how the performance of DECORATE changes with ensemble size, we ran experiments with increasing sizes. We compared results for training on 20% of available data, since the advantage of DECORATE is most noticeable low on the learning curve. Due to lack of space, we do not include the results for all 15 datasets, but present five representative datasets (see Figure 2). The performance on other datasets is similar. We note, in general, that the accuracy of DECORATE increases with ensemble size; though on most datasets, the performance levels out with an ensemble size of 10 to 25.

¹The p -value is the probability of getting a correlation as large as the observed value by random chance, when the true correlation is zero.

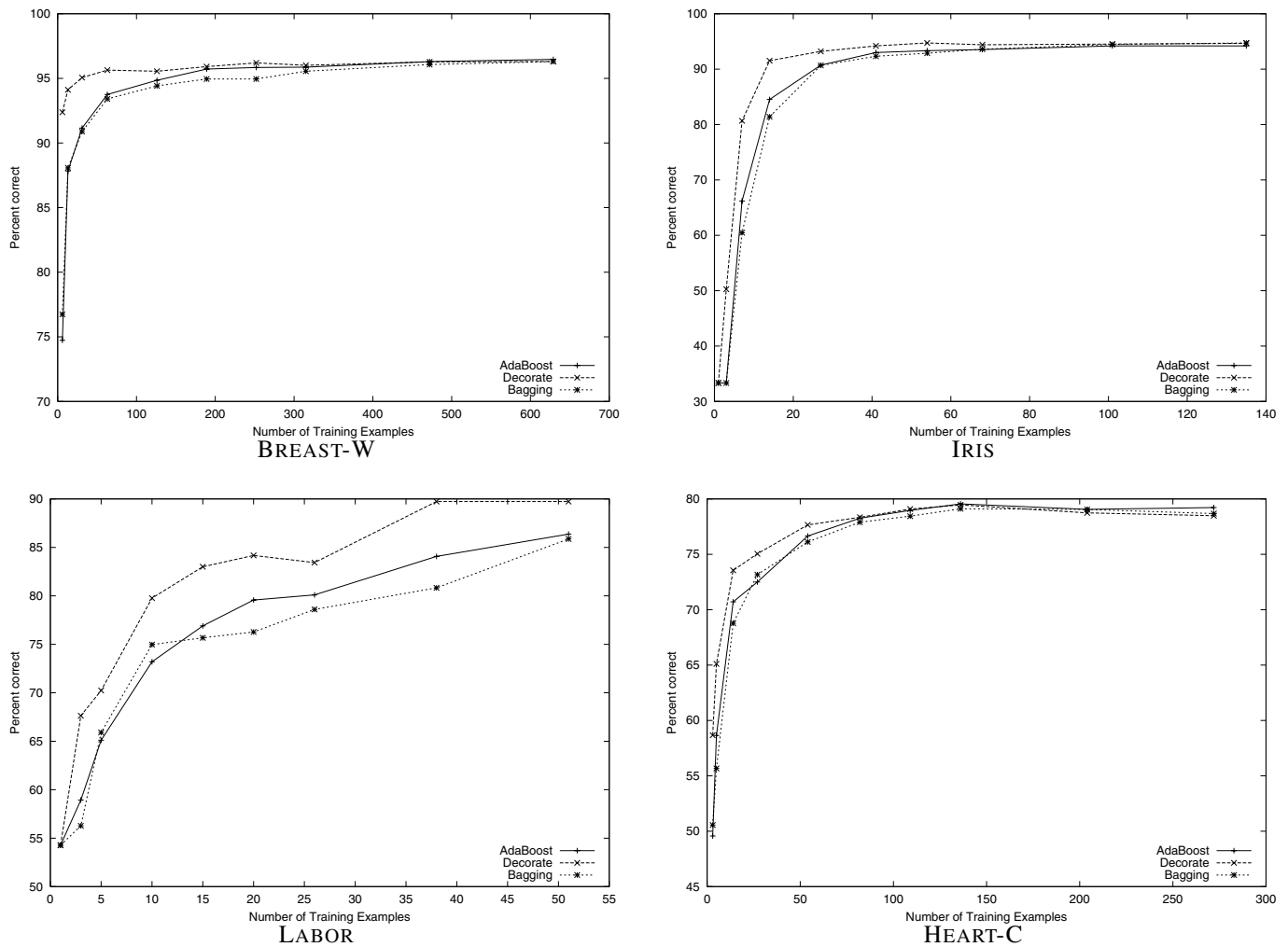


Figure 1: DECORATE compared to AdaBoost and Bagging

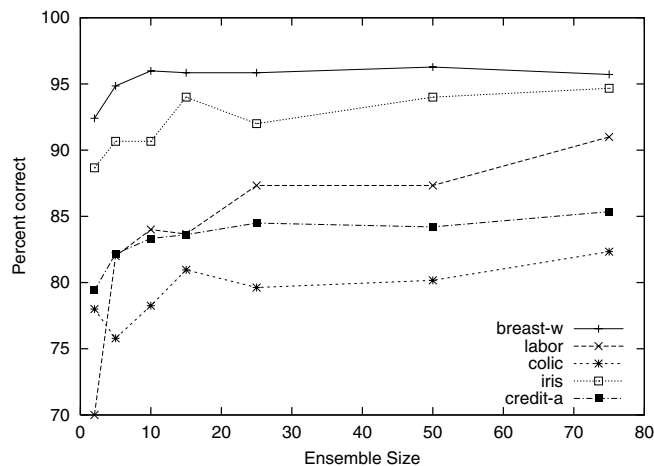


Figure 2: DECORATE at different ensemble sizes

5 Related Work

There have been some other attempts at building ensembles that focus on the issue of diversity. Liu et al [1999] and Rosen [1996] simultaneously train neural networks in an ensemble using a correlation penalty term in their error functions. Opitz and Shavlik [1996] use a genetic algorithm to search for a good ensemble of networks. To guide the search they use an objective function that incorporates both an accuracy and diversity term. Zenobi et al [2001] build ensembles based on different feature subsets; where feature selection is done using a hill-climbing strategy based on classifier error and diversity. A classifier is rejected if the improvement of one of the metrics lead to a “substantial” deterioration of the other; where “substantial” is defined by a pre-set threshold.

In all these approaches, ensembles are built attempting to simultaneously optimize the accuracy and diversity of individual ensemble members. However, in DECORATE, our goal is to minimize *ensemble error* by increasing diversity. At no point does the training accuracy of the ensemble go below

Table 1: DECORATE vs J48

Dataset	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
anneal	75.29/72.49	78.14/75.31	85.24/82.08	92.26/89.28	96.48/95.57	97.36/96.47	97.73/97.3	98.16/97.93	98.39/98.35	98.71/98.55
audio	16.66/16.66	23.73/23.07	41.72/41.17	55.42/51.67	64.09/60.59	67.62/64.84	70.46/68.11	72.82/70.77	77.8/75.15	82.1/77.22
autos	24.33/24.33	29.6/29.01	36.73/34.37	42.89/41.22	52.2/50.53	59.86/53.92	64.77/59.68	68.6/65.24	78/73.15	83.64/81.72
breast-w	92.38/74.73	94.12/87.34	95.06/89.42	95.64/92.21	95.55/93.09	95.91/93.36	96.2/93.85	96.01/94.24	96.28/94.65	96.31/95.01
credit-a	71.78/69.54	74.83/77.46	80.61/81.57	83.09/82.35	84.38/84.29	84.68/84.59	85.22/84.41	85.57/84.78	85.61/85.43	85.93/85.57
Glass	31.69/31.69	35.86/32.96	44.5/38.34	55.4/46.62	61.77/54.16	66.01/60.63	68.07/61.38	68.85/63.69	72.73/67.53	72.77/67.77
heart-c	58.66/49.57	65.11/58.03	73.55/67.71	75.05/70.15	77.66/73.44	78.34/74.61	79.09/74.78	79.46/75.62	78.74/76.7	78.48/77.17
hepatitis	52.33/52.33	71.95/65.93	76.59/72.75	78.85/78.25	80.28/78.61	81.14/78.63	81.53/79.35	81.68/79.57	82.37/79.04	82.43/79.22
colic	59.85/52.85	68.19/65.31	74.91/74.37	78.45/79.94	81.81/82.71	82.47/83.41	82.74/83.55	83.5/84.66	83.93/85.18	85.24/85.16
iris	33.33/33.33	50.87/33.33	80.67/59.33	91.27/84.33	93.07/91.33	94.4/92.73	95.07/93	94.07/93.33	94.67/94.07	94.93/94.73
labor	54.27/54.27	54.27/54.27	67.7/58.93	71.47/64.77	78.6/70.07	81.67/73.7	85.67/75.17	84.2/75.8	87.53/77.4	89.5/78.8
lymph	48.39/48.39	53.49/46.64	65.73/60.39	72.79/68.21	74.57/70.79	78.84/73.58	78.37/74.53	78.31/73.34	78.06/75.63	78.74/76.06
segment	67.94/52.43	80.75/73.26	89.52/85.41	92.87/89.34	94.99/92.22	95.82/93.37	96.54/94.34	96.93/94.77	97.56/95.94	98.02/96.79
soybean	19.37/13.69	32.12/22.32	55.55/42.94	73.51/59.04	84.63/74.49	88.52/81.59	90.37/84.78	91.35/86.89	92.85/89.44	93.81/91.76
splice	63.48/59.92	67.56/68.69	77.34/77.49	82.62/82.58	88.2/87.98	90.46/90.44	91.82/91.77	92.5/92.4	93.41/93.47	93.92/94.03
Win/Draw/Loss	15/0/0	13/0/2	13/0/2	14/0/1	14/0/1	14/0/1	14/0/1	14/0/1	13/0/2	14/0/1
Sig. W/D/L	7/8/0	10/3/2	11/4/0	10/5/0	11/4/0	12/3/0	12/2/0	12/2/1	10/4/1	10/4/1
GM error ratio	0.858	0.8649	0.8116	0.8098	0.8269	0.8103	0.7983	0.8305	0.8317	0.8293

Table 2: DECORATE vs Bagging

Dataset	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
anneal	75.29/74.57	78.14/76.42	85.24/82.88	92.26/89.87	96.48/95.67	97.36/96.89	97.73/97.34	98.16/97.78	98.39/98.53	98.71/98.83
audio	16.66/12.98	23.73/23.68	41.72/38.55	55.42/51.34	64.09/61.76	67.62/66.9	70.46/70.29	72.82/73.07	77.8/77.32	82.1/80.71
autos	24.33/22.16	29.6/28	36.73/35.88	42.89/44.65	52.2/54.32	59.86/59.67	64.77/65.6	68.6/69.88	78/77.97	83.64/83.12
breast-w	92.38/76.74	94.12/88.07	95.06/90.88	95.64/93.41	95.55/94.42	95.91/94.95	96.2/94.95	96.01/95.55	96.28/96.07	96.31/96.3
credit-a	71.78/69.54	74.83/77.99	80.61/82.58	83.09/83.9	84.38/85.13	84.68/85.78	85.22/85.59	85.57/85.64	85.61/86.12	85.93/85.96
Glass	31.69/24.85	35.86/31.47	44.5/40.87	55.4/49.6	61.77/58.9	66.01/64.35	68.07/66.3	68.85/68.44	72.73/72	72.77/74.67
heart-c	58.66/50.56	65.11/55.67	73.55/68.77	75.05/73.17	77.66/76.12	78.34/77.9	79.09/78.44	79.46/79.11	78.74/79.05	78.48/78.68
hepatitis	52.33/52.33	72.14/63.18	76.8/75.2	79.48/78.64	80.7/80.42	81.81/81.07	81.65/81.22	83.19/81.06	82.99/80.87	82.62/81.34
colic	58.37/53.14	66.58/63.83	75.85/76.44	79.54/80.06	81.33/83.04	82.47/83.58	83.02/83.98	83.1/84.47	84.02/85.4	84.69/85.34
iris	33.33/33.33	50.27/33.33	80.67/60.47	91.53/81.4	93.2/90.67	94.2/92.33	94.73/92.87	94.4/93.6	94.53/94.47	94.67/94.73
labor	54.27/54.27	54.27/54.27	67.63/56.27	70.23/65.9	79.77/74.97	83/75.67	84.17/76.27	83.43/78.6	89.73/80.83	89.73/85.87
lymph	48.39/48.39	53.62/47.11	65.06/60.12	71.2/69.68	76.74/73.6	78.84/76.58	78.17/77.68	78.99/76.98	79.14/76.8	79.08/77.97
segment	67.03/55.88	81.16/76.36	89.61/87.42	92.83/91.01	94.88/93.4	95.94/94.65	96.47/95.26	96.93/95.82	97.58/96.78	98.03/97.41
soybean	19.51/14.56	32.4/24.58	55.36/47.46	73.06/65.45	85.14/79.29	88.27/85.05	90.22/87.89	91.4/89.22	92.75/91.56	93.89/92.71
splice	62.77/62.52	67.8/72.36	77.37/80.5	82.55/85.44	88.24/89.5	90.47/91.44	91.84/92.4	92.41/93.07	93.44/94.06	93.92/94.53
Win/Draw/Loss	15/0/0	13/0/2	12/0/3	11/0/4	11/0/4	12/0/3	11/0/4	10/0/5	10/0/5	8/0/7
Sig. W/D/L	8/7/0	10/3/2	10/3/2	9/5/1	10/2/3	8/4/3	6/7/2	8/5/2	5/7/3	4/9/2
GM error ratio	0.8727	0.8785	0.8552	0.8655	0.8995	0.9036	0.8979	0.9214	0.9312	0.9570

that of the base classifier; however, this is a possibility with previous methods. Furthermore, none of the previous studies compared their methods with the standard ensemble approaches such as Boosting and Bagging (Opitz and Shavlik, 1996) compares with Bagging, but not Boosting).

Compared to boosting, which requires a “weak” base learner that does not completely fit the training data (boosting terminates once it constructs a hypothesis with zero training error), DECORATE requires a strong learner, otherwise the artificial diversity training data may prevent it from adequately fitting the real data. When applying boosting to strong base learners, they must first be appropriately weakened in order to benefit from boosting. Therefore, DECORATE may be a preferable ensemble meta-learner for strong learners.

To our knowledge, the only other ensemble approach to utilize artificial training data is the active learning method introduced in [Cohn *et al.*, 1994]. The goal of the committee here is to select good new training examples rather than to improve accuracy using the existing training data. Also, the labels of the artificial examples are selected to produce hypotheses that more faithfully represent the entire version space rather than to produce diversity. Cohn’s approach labels artificial data either all positive or all negative to encourage, respectively, the learning of more general or more specific hypotheses.

6 Future Work and Conclusion

In our current approach, we are encouraging diversity using artificial training examples. However, in many domains, a large amount of unlabeled data is already available. We could exploit these unlabeled examples and label them as diversity data. This would allow DECORATE to act as a form of *semi-supervised learning* that exploits both labeled and unlabeled data [Nigam *et al.*, 2000].

Our current study has used J48 as a base learner; however, we would expect similarly good results with other base learners. Decision-tree induction has been the most commonly used base learner in other ensemble studies, but there has been some work using neural networks and naive Bayes [Bauer and Kohavi, 1999; Opitz and Maclin, 1999]. Experiments on “DECORATING” other learners is another area for future work.

By manipulating artificial training examples, DECORATE is able to use a strong base learner to produce an effective, diverse ensemble. Experimental results demonstrate that the approach is particularly effective at producing highly accurate ensembles when training data is limited, outperforming both bagging and boosting low on the learning curve. The empirical success of DECORATE raises the issue of developing a sound theoretical understanding of its effectiveness. In gen-

Table 3: DECORATE vs AdaBoost

Dataset	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
anneal	75.29/73.02	78.14/77.12	85.24/87.51	92.26/94.16	96.48/97.13	97.36/97.95	97.73/98.54	98.16/98.8	98.39/99.23	98.71/99.68
audio	16.66/16.66	23.73/23.41	41.72/40.24	55.42/52.7	64.09/64.15	67.62/68.91	70.46/73.07	72.82/75.92	77.8/81.74	82.1/84.52
autos	24.33/24.33	29.6/29.71	36.73/34.2	42.89/43.28	52.2/56.13	59.86/62.2	64.77/69.14	68.6/72.03	78/80.28	83.64/85.28
breast-w	92.38/74.73	94.12/87.84	95.06/91.15	95.64/93.75	95.55/94.85	95.91/95.72	96.2/95.84	96.01/95.87	96.28/96.3	96.31/96.47
credit-a	71.78/68.8	74.83/75.3	80.61/79.68	83.09/81.14	84.38/83.04	84.68/84.22	85.22/84.13	85.57/84.58	85.61/84.93	85.93/85.42
Glass	31.69/31.69	35.86/32.93	44.5/40.71	55.4/49.78	61.77/58.03	66.01/64.33	68.07/66.93	68.85/68.69	72.73/74.69	72.77/76.06
heart-c	58.66/49.57	65.11/58.65	73.55/70.71	75.05/72.5	77.66/76.65	78.34/78.26	79.09/78.96	79.46/79.55	78.74/79.06	78.48/79.22
hepatitis	52.33/52.33	72.14/65.93	76.8/73.01	79.48/76.95	80.7/79.44	81.81/79.22	81.65/81.27	83.19/82.63	82.99/83.24	82.62/82.71
colic	58.37/52.85	66.58/67.18	75.85/72.85	79.54/77.17	81.33/79.36	82.47/79.24	83.02/79.51	83.1/80.22	84.02/80.59	84.69/81.93
iris	33.33/33.33	50.27/33.33	80.67/66.2	91.53/84.53	93.2/90.73	94.2/93	94.73/93.33	94.4/93.53	94.53/94.2	94.67/94.2
labor	54.27/54.27	54.27/54.27	67.63/58.93	70.23/65.1	79.77/73.2	83/76.9	84.17/79.57	83.43/80.1	89.73/84.07	89.73/86.37
lymph	48.39/48.39	53.62/46.64	65.06/60.54	71.2/69.57	76.74/74.16	78.84/78.62	78.17/80.35	78.99/79.88	79.14/80.96	79.08/81.75
segment	67.03/60.22	81.16/77.38	89.61/88.5	92.83/92.71	94.88/95.01	95.94/96.03	96.47/96.9	96.93/97.23	97.58/98	98.03/98.34
soybean	19.51/14.26	32.4/23.36	55.36/49.37	73.06/69.49	85.14/85.01	88.27/88.37	90.22/90.04	91.4/90.89	92.75/92.57	93.89/92.88
splice	62.77/65.11	67.8/73.9	77.37/82.22	82.55/86.13	88.24/88.27	90.47/89.82	91.84/90.8	92.41/90.78	93.44/92.63	93.92/93.59
Win/Draw/Loss	14/0/1	11/0/4	13/0/2	12/0/3	10/0/5	10/0/5	10/0/5	9/0/6	6/0/9	6/0/9
Sig. W/D/L	7/7/1	8/6/1	11/2/2	10/3/2	7/6/2	4/9/2	5/5/5	5/6/4	3/6/6	3/6/6
GM error ratio	0.8812	0.8937	0.8829	0.9104	0.9407	0.9598	0.9908	0.9957	1.0377	1.0964

eral, the idea of using artificial or unlabeled examples to aid the construction of effective ensembles seems to be a promising approach worthy of further study.

Acknowledgments

This work was supported by DARPA EELD Grant F30602-01-2-0571.

References

- [Bauer and Kohavi, 1999] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36, 1999.
- [Blake and Merz, 1998] C. L. Blake and C. J. Merz. UCI repository of machine learning databases. <http://www.ics.uci.edu/~mlearn/MLRepository.html>, 1998.
- [Breiman, 1996] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [Cohn *et al.*, 1994] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.
- [Cunningham and Carney, 2000] P. Cunningham and J. Carney. Diversity versus quality in classification ensembles based on feature selection. In *11th European Conference on Machine Learning*, pages 109–116, 2000.
- [Dietterich, 2000] T. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2000.
- [Freund and Schapire, 1996] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, July 1996.
- [Hastie *et al.*, 2001] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Verlag, New York, August 2001.
- [Krogh and Vedelsby, 1995] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation and active learning. In *Advances in Neural Information Processing Systems 7*, 1995.
- [Kuncheva and Whitaker, 2002] L. Kuncheva and C. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *submitted*, 2002.
- [Liu and Yao, 1999] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12, 1999.
- [Nigam *et al.*, 2000] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134, 2000.
- [Opitz and Maclin, 1999] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.
- [Opitz and Shavlik, 1996] D. Opitz and J. Shavlik. Actively searching for an effective neural-network ensemble. *Connection Science*, 8, 1996.
- [Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
- [Quinlan, 1996] J. Ross Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the 13th National Conference on Artificial Intelligence*, August 1996.
- [Rosen, 1996] B. Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8, 1996.
- [Webb, 2000] G. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40, 2000.
- [Witten and Frank, 1999] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, 1999.
- [Zenobi and Cunningham, 2001] G. Zenobi and P. Cunningham. Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In *Proceedings of the European Conference on Machine Learning*, 2001.

LEARNING

EVALUATING CLASSIFIERS

Evaluating Classifiers by Means of Test Data with Noisy Labels

Chuck P. Lam

Department of Electrical Engineering
Stanford University
Stanford, CA 94305
chucklam@stanford.edu

David G. Stork

Ricoh Innovations, Inc.
2882 Sand Hill Road, Suite 115
Menlo Park, CA 94025-7022
stork@rii.ricoh.com

Abstract

Often the most expensive and time-consuming task in building a pattern recognition system is collecting and accurately labeling training and testing data. In this paper, we explore the use of inexpensive noisy *testing* data for evaluating a classifier's performance. We assume 1) the (human) labeler provides category labels with a known mislabeling rate and 2) the trained classifier and the labeler are statistically independent. We then derive the number of "noisy" test samples that are, on average, equivalent to a single perfectly labeled test sample for the task of evaluating the classifier's performance. For practical and realistic error and mislabeling rates, this number of equivalent test patterns can be surprisingly low. We also derive an upper and lower bound for the true error rate when the labeler and the classifier are not independent.

1 Introduction

The overall construction of a modern classification system can be divided into four broad tasks: (1) specifying the classifier type, (2) collecting data, (3) training the classifier (i.e., learning), and (4) evaluating the classifier (i.e., testing) [Duda *et al.*, 2001]. The second stage, data collection, can further be divided into two tasks: gathering samples and labeling them. Recently, the machine learning community has realized that in many practical cases the most expensive part of the whole design process is the labeling of such samples. For example, there is an enormous number of text documents on the internet that can be obtained at very low cost; however, relatively few of these have been labeled — e.g., according to content topic, language, or style — in a consistent way that would facilitate training a classifier. Likewise, there are large databases of recorded speech, handwritten digits, and printed characters but these databases, too, are either not labeled accurately or not labeled at all [Stork, 1999]. To reduce the labeling expense, many researchers have sought ways to modify training algorithms so as to utilize both labeled and unlabeled data [Blum and Mitchell, 1998; Nigam *et al.*, 2000]. This approach has shown surprisingly encouraging results, in some cases reducing the number of

labeled samples by a few orders of magnitude [Nigam *et al.*, 2000].

In order to build up and extend this success in reducing the labeling cost, we turn to the problem of reducing the need for accurately labeled data in the classifier *evaluation* stage. In fact, most of the experiments for learning with labeled and unlabeled data use much more labels for testing than training [Nigam *et al.*, 2000]. Thus we now need to address the labeling cost for classifier evaluation.

As with many areas of commerce, the general economics of labeling is such that the higher the quality (accuracy) of labeling, the greater the associated cost. This greater cost may be due to greater expertise of the labeler, or the need for multiple passes of cross-checking, or both. There is thus an additional cost to "clean" or "truth" those data and labels. In some situations, such as marking a text corpus, the labeling task is complicated enough that even experts need several passes to reduce labeling errors [Eskin, 2000]. Furthermore, in some application domains, obtaining accurate labels is simply too cost prohibitive. For example, for some medical diagnostics, the true disease can only be known with expensive or invasive techniques. Similarly, in remote sensing, one must send measuring instruments to the ground location to obtain the "ground truth," and the transportation cost can be astronomical. (It is quite literal for remote sensing of other planets [Smyth, 1997].) For both situations in practice, one must rely on the imperfect judgements of experts [Smyth, 1997].

We propose to lower the labeling cost in classifier evaluation by using cheaper, noisy labels. This paper examines methodologies of estimating the error rate and classifier confusion matrix using test data with noisy labels. We shall see that even a slight labeling inaccuracy (say, 1%) can have a significant effect on the error rate estimate when the classifier performs well. In addition, when data sets used to be small and expensive to collect, it made sense to spend each additional labeling effort to increase label accuracy on that small data set. However, when data sets are large and cheap to collect, it is no longer obvious how one should spend each additional labeling effort. Should one spend it labeling the unlabeled data, or should one spend it increasing the accuracy of already labeled data? We present a preliminary analysis to this question.

2 Preliminaries and notation

Our formulation assumes an object $\mathbf{x}$ possessing a *true* label $y \in \Omega$, where $\Omega = \{\omega_1, \dots, \omega_c\}$ is the set of possible states of nature (e.g., category membership) for the object. The object is presented to a labeler, who marks it with a label $\tilde{y} \in \Omega$, as his guess of y . The situation that $y \neq \tilde{y}$ is call a *labeling error* (or a mislabeling). The classifier system, on the other hand, is presented with the feature vector $\mathbf{x}$ that represents certain aspects of the object, and the classifier outputs a label $\hat{y}(\mathbf{x}) \in \Omega$, as its guess of y . For notational convenience, we will call the classifier output $\hat{y}$, and its dependence on the feature vector $\mathbf{x}$ is implicit. The situation that $y \neq \hat{y}$ is call a *classification error* (or a misclassification).

The probability of the labeler making mistakes, $\Pr[y \neq \tilde{y}]$, is called the *mislabeling rate*. The probability of the classifier's label being different from the labeler's label, $\Pr[\hat{y} \neq \tilde{y}]$, is called the *apparent error rate*. Our goal is to estimate $\Pr[y \neq \hat{y}]$, which is called the *true error rate*. Note that it is possible to have a high apparent error rate even with a perfect classifier (with a true error rate of zero) simply because of a high mislabeling rate. That is, the classifier can classify all test data perfectly, but will often disagree with the test labels because those labels are incorrect. On the other hand, it is also possible to have a zero apparent error rate even with a high true error rate if the classifier and the labeler make the same kind of mistakes.

The *confusion matrix* for the human labeler is defined as

$$\mathbf{P}_{\tilde{y}|y} \equiv \begin{pmatrix} P(\tilde{y} = \omega_1 | y = \omega_1) & \cdots & P(\tilde{y} = \omega_1 | y = \omega_c) \\ \vdots & \ddots & \vdots \\ P(\tilde{y} = \omega_c | y = \omega_1) & \cdots & P(\tilde{y} = \omega_c | y = \omega_c) \end{pmatrix},$$

which is the identity matrix for a perfect labeler. Similarly the classifier's confusion matrix is defined as

$$\mathbf{P}_{\hat{y}|y} \equiv \begin{pmatrix} P(\hat{y} = \omega_1 | y = \omega_1) & \cdots & P(\hat{y} = \omega_1 | y = \omega_c) \\ \vdots & \ddots & \vdots \\ P(\hat{y} = \omega_c | y = \omega_1) & \cdots & P(\hat{y} = \omega_c | y = \omega_c) \end{pmatrix}.$$

For many two-class cases where one class has a much higher prior probability, the actual error rate is not a good measure of classifier usefulness. For example, in detecting email spams or network intrusions, the undesirable events are so rare that one can easily get an error rate less than 1% by classifying all events as "desirable." In those situations, then, one may want to compute the entire confusion matrix or metrics such as *precision* and *recall* [Frakes and Baeza-Yates, 1992]. We denote ω_2 as the "rare" class (e.g., spams or network intrusions). For the classifier, precision is defined as $P(y = \omega_2 | \hat{y} = \omega_2)$ and recall is defined as $P(\hat{y} = \omega_2 | y = \omega_2)$, and analogously for the labeler. Note that precision and recall can be derived from the confusion matrix and the class prior probabilities.

3 Obtaining the true error rate

In examining the relationship between true and apparent error rates, we make the constraint that we have a two-class problem, that is, $\Omega = \{\omega_1, \omega_2\}$. Thus $\{\hat{y} \neq \tilde{y} \cup y \neq \tilde{y}\} \Rightarrow \{y =$

$\hat{y}\}$. Then we can rewrite the apparent error rate as

$$\begin{aligned} \Pr[\hat{y} \neq \tilde{y}] &= \Pr[\hat{y} \neq \tilde{y}, y \neq \tilde{y}] + \Pr[\hat{y} \neq \tilde{y}, y = \tilde{y}] \\ &= \Pr[y = \hat{y}, y \neq \tilde{y}] + \Pr[y \neq \hat{y}, y = \tilde{y}] \\ &= \Pr[y = \hat{y}] \Pr[y \neq \tilde{y}] \\ &\quad + \Pr[y \neq \hat{y}] \Pr[y = \tilde{y}] \\ &= (1 - \Pr[y \neq \hat{y}]) \Pr[y \neq \tilde{y}] \\ &\quad + \Pr[y \neq \hat{y}] (1 - \Pr[y \neq \tilde{y}]) \\ &= \Pr[y \neq \tilde{y}] + \Pr[y \neq \hat{y}] (1 - 2 \Pr[y \neq \tilde{y}]) \end{aligned} \quad (1)$$

and thus

$$\Pr[y \neq \hat{y}] = \frac{\Pr[\hat{y} \neq \tilde{y}] - \Pr[y \neq \tilde{y}]}{1 - 2 \Pr[y \neq \tilde{y}]} \quad (2)$$

The above derivation assumed that $\Pr[y \neq \tilde{y}] \neq 1/2$; otherwise the noisy labels are meaningless. In practice, $\Pr[y \neq \tilde{y}]$ is always much less than $1/2$. More important is the independence assumption that the labeler and the classifier make errors independently, or stated succinctly, $\Pr[y \neq \hat{y}, y \neq \tilde{y}] = \Pr[y \neq \hat{y}] \Pr[y \neq \tilde{y}]$. That is, knowing that the labeler had made an error on a pattern does not change the probability that the classifier would also make an error, and vice versa. Section 6 will deal with some situations in which the independence assumption does not hold. In the meantime, we argue for this idealization and simplification based on the fact that human and computer generally classify samples using different methodologies, and thus they may not make similar kinds of mistakes.

3.1 Example: Apparent error rate for various true error rates and mislabeling rates

Equation 2 gives us a way to account for noisy labels when calculating the true error rate. A natural question, then, is how important is it to correct for the influence of noisy labels? Let's consider some classification systems with error rates between 2% and 10%¹ and testing data sets with 1% to 5% incorrect labels. Table 1 shows the apparent error rate for classifiers of different accuracy and testing data of different mislabeling rates. The percentage increase over the true error rate is also shown. For example, even when only 1% of the testing labels are wrong, a classifier with true error rate of 6% will have an apparent error rate 15% higher (at 6.88%). The percentage increase is even more dramatic with noisier labels or more accurate classifiers. A quick rule of thumb is that, when the labels have relatively few errors, the denominator in Eq. 2 is approximately 1.0 and can be ignored. The mislabeling rate of the testing labels ($\Pr[y \neq \tilde{y}]$) is then just an additive component to the true error rate. Continuing the previous example, a 1% mislabeling rate for a classifier with true error rate of 6% makes the apparent error rate approximately 7%, when the actual is 6.88%.

4 Noisy labels for estimating true error rate

Above we assumed knowledge of the apparent error rate, $\Pr[\hat{y} \neq \tilde{y}]$, but in practice, we must estimate this rate using test data. In this section, we analyze the effects of

¹We note that many classifiers on the UCI datasets have accuracy within this range [Kaynak and Alpaydin, 2000].

True Error Rate, $\Pr[y \neq \hat{y}]$	Mislabeling Rate, $\Pr[y \neq \tilde{y}]$									
	1%		2%		3%		4%		5%	
2%	2.96%	↑48%	3.92%	↑96%	4.88%	↑144%	5.84%	↑192%	6.80%	↑240%
4%	4.92%	↑23%	5.84%	↑46%	6.76%	↑69%	7.68%	↑92%	8.60%	↑115%
6%	6.88%	↑15%	7.76%	↑29%	8.64%	↑44%	9.52%	↑59%	10.40%	↑73%
8%	8.84%	↑11%	9.68%	↑21%	10.52%	↑32%	11.36%	↑42%	12.20%	↑53%
10%	10.80%	↑8%	11.60%	↑16%	12.40%	↑24%	13.20%	↑32%	14.00%	↑40%

Table 1: The left sub-columns of the table show the apparent error rates ($\Pr[\hat{y} \neq \tilde{y}]$) for different true error rates ($\Pr[y \neq \hat{y}]$) and different mislabeling rates ($\Pr[y \neq \tilde{y}]$), based on Eq. 2. It is assumed that the labeler and the classifier make errors independently. The right sub-columns of the table, with up-arrow (↑) signs, show the percentage increase of the apparent error rate over the true error rate (i.e., $(\Pr[\hat{y} \neq \tilde{y}] - \Pr[y \neq \hat{y}]) / \Pr[y \neq \hat{y}]$).

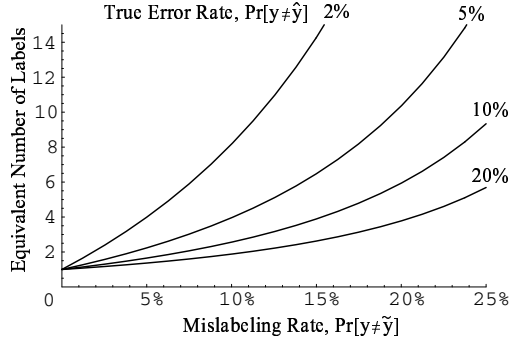


Figure 1: The figure shows the number of noisy labels needed to achieve the same variance in the true error rate estimate as a single perfect label (see Eq. 3). The four plots represent different true error rates. As the mislabeling rate increases, more noisy labels are needed to achieve the same confidence. Note that, in the ranges shown in the figure, when the mislabeling rate is smaller than the true error rate, a single perfect label is equivalent to less than four noisy labels.

such estimates. Assume we have l objects in the test set, each with a feature vector $\mathbf{x}_i$, true but unknown label y_i , noisy label $\tilde{y}_i$, and classification $\hat{y}_i$. Assume further that tuples $\{(\mathbf{x}_1, y_1, \tilde{y}_1, \hat{y}_1), \dots, (\mathbf{x}_l, y_l, \tilde{y}_l, \hat{y}_l)\}$ are independent and identically distributed as $(\mathbf{x}, y, \tilde{y}, \hat{y})$. The apparent error rate estimate is $\hat{\Pr}[\hat{y} \neq \tilde{y}] = \frac{1}{l} \sum_{i=1}^l I[\hat{y}_i \neq \tilde{y}_i]$, in which $I[\cdot]$ is the indicator function (i.e., $I[event] = 1$ if *event* is true and 0 otherwise). An estimate of the true error rate is

$$\hat{\Pr}[y \neq \hat{y}] \equiv \frac{\hat{\Pr}[\hat{y} \neq \tilde{y}] - \Pr[y \neq \tilde{y}]}{1 - 2\Pr[y \neq \tilde{y}]}$$

It is straightforward to verify that $\mathcal{E}[\hat{\Pr}[\hat{y} \neq \tilde{y}]] = \Pr[\hat{y} \neq \tilde{y}]$ and $\mathcal{E}[\hat{\Pr}[y \neq \hat{y}]] = \Pr[y \neq \hat{y}]$, thus the estimates are unbiased. Intuitively we know that we have less confidence when the error estimates are based on test data with noisy labels. To formalize this intuition, we examine the variance of the true error rate estimate,

$$\text{Var}[\hat{\Pr}[y \neq \hat{y}]] =$$

$$\frac{1}{l} \left(\frac{\Pr[y \neq \tilde{y}](1 - \Pr[y \neq \tilde{y}])}{(1 - 2\Pr[y \neq \tilde{y}])^2} + \Pr[y \neq \hat{y}](1 - \Pr[y \neq \hat{y}]) \right).$$

The variance of the error estimate given perfectly labeled data is $\Pr[y \neq \hat{y}](1 - \Pr[y \neq \hat{y}])/l$. Thus, to get the same variance, the ratio of noisy labels to perfect labels is

$$\frac{\Pr[y \neq \tilde{y}](1 - \Pr[y \neq \tilde{y}])}{(1 - 2\Pr[y \neq \tilde{y}])^2 \Pr[y \neq \hat{y}](1 - \Pr[y \neq \hat{y}])} + 1. \quad (3)$$

This ratio will help us understand the economic trade-offs between using perfect and noisy labels. Collecting perfect labels (or collecting noisy labels first and cleaning them) is often much more expensive than just collecting noisy labels itself. Therefore it may be economically justified to use noisy labels, as long as one does not need too many more of them.

Unfortunately, applying Eq. 3 requires us to know the true error rate of the classifier, which is exactly what one is trying to estimate. However, we often already have a good idea of a reasonable range for the true error rate. In any case, we examine the ratio for a wide range of true error rate and mislabeling rate, and we found the ratio to fall within a relatively narrow range, as shown in Fig. 1. Even for relatively noisy testing data with 10% incorrect labels, unless the classifier is much more accurate (with true error rate of less than 5%), cleaning the testing data to be perfectly labeled increases its value by less than a factor of four. In other words, one needs much less than four such noisy labels to achieve the same effect as one perfect label. Imagine that perfect labels need to be collected from a domain expert, whereas noisy labels can be collected from a non-expert, the high cost of a domain expert can often justify the use of noisy labels.

4.1 Example: Evaluating with many noisy labels or few reliable labels

In many labeling tasks, experts must make multiple passes through samples to ensure accurate labeling [Eskin, 2000]. We now question the wisdom of that policy when the samples are free but labeling cost is a constraint.

Consider a hypothetical labeling situation with two labelers, each paid to look at l samples, and both labelers have an error rate of ϵ . There are two choices in how to use these two labelers. One is to have them look at completely different samples, thus in the end we have a testing set of size $2l$ and mislabeling rate ϵ . Another choice is to have them look at the exact same samples. Assuming that they make independent

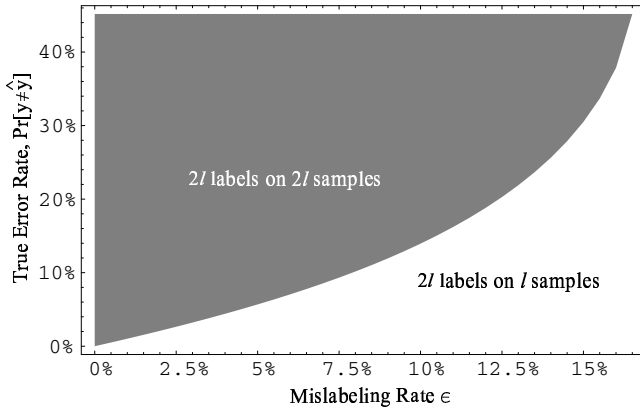


Figure 2: The figure shows which one of the two labeling policies is optimal for a range of mislabeling rate ϵ and true classifier error rate, based on Eq. 4. The problem is posed such that two labelers both have mislabeling rate ϵ and are paid to label l samples. One policy is that they label different samples, creating a test set of “ $2l$ labels on $2l$ samples,” with ϵ portion mislabeled. The other policy is that they both label the same samples, creating a testing set of “ $2l$ labels on l samples,” with ϵ^2 portion mislabeled (after various assumptions).

labeling errors, and optimistically assume that a sample has a wrong label only if both labelers err, then we have a testing set of size l and mislabeling rate ϵ^2 . Which is the better policy?

Based on the previous discussion, we can have an unbiased estimate of the true error rate from either testing set. We then should prefer one that gives us a lower variance estimate. That is, we go with the “ $2l$ labels on $2l$ samples” policy if its variance is lower than the “ $2l$ labels on l samples” policy,

$$\frac{1}{2l} \left(\frac{\epsilon(1-\epsilon)}{(1-2\epsilon)^2} + \Pr[y \neq \hat{y}](1 - \Pr[y \neq \hat{y}]) \right) < \frac{1}{l} \left(\frac{\epsilon^2(1-\epsilon^2)}{(1-2\epsilon^2)^2} + \Pr[y \neq \hat{y}](1 - \Pr[y \neq \hat{y}]) \right).$$

Which, after a little algebra, becomes

$$\left(\frac{2\epsilon^2(1-\epsilon^2)}{(1-2\epsilon^2)^2} - \frac{\epsilon(1-\epsilon)}{(1-2\epsilon)^2} + \frac{1}{4} \right) > \left(\Pr[y \neq \hat{y}] - \frac{1}{2} \right)^2. \quad (4)$$

An interesting observation is that in the realistic range of ϵ ($0 \leq \epsilon < 0.5$), the left hand side of Eq. 4 is negative for $\epsilon > 0.166$, which means that one should *always* choose the “ $2l$ labels on l samples” policy for such inaccurate labelers, and such high mislabeling rate does occur in practice [Smyth, 1997]. For other cases, we have plotted the policy boundary (i.e., $\frac{1}{2} - \sqrt{\frac{2\epsilon^2(1-\epsilon^2)}{(1-2\epsilon^2)^2} - \frac{\epsilon(1-\epsilon)}{(1-2\epsilon)^2} + \frac{1}{4}} = \Pr[y \neq \hat{y}]$) in Fig. 2.

For fairly accurate labelers (say, $\epsilon < 2.5\%$), Fig. 2 shows that one should prefer the “ $2l$ labels on $2l$ samples” policy unless the classifier error rate is very low. The hint for practitioners is that time spent cleaning labels is often not as effective as time spent labeling extra samples.

5 Obtaining True Confusion Matrix

In evaluating classification systems, we often need to know more than just the error rate. When the cost of different mis-

precision/recall (classifier)	precision/recall (labeler)		
	99%	98%	95%
70%	69.3%	68.7%	66.7%
80%	79.2%	78.4%	76.1%
90%	89.1%	88.2%	85.6%
100%	99%	98%	95%

Table 2: Apparent precision recall breakeven points (i.e., $P(\tilde{y} = \omega_2 | \hat{y} = \omega_2)$) for different actual classifier precision/recall and labeler precision/recall. The prior probabilities for ω_1 and ω_2 are 90% and 10% respectively.

classifications (e.g., false positives and false negatives) are not equal, we may want to know the full confusion matrix. In addition, in some domains, such as classifying text or spam, the distribution of classes is highly skewed, and the error rate can be misleadingly low. In those situations, we are more interested in precision and recall statistics, which can be estimated from the confusion matrix.

We define the *joint distribution matrix* between labeler and classifier as

$$\mathbf{P}_{\tilde{y}, \hat{y}} \equiv \begin{pmatrix} P(\tilde{y} = \omega_1, \hat{y} = \omega_1) & \cdots & P(\tilde{y} = \omega_1, \hat{y} = \omega_c) \\ \vdots & \ddots & \vdots \\ P(\tilde{y} = \omega_c, \hat{y} = \omega_1) & \cdots & P(\tilde{y} = \omega_c, \hat{y} = \omega_c) \end{pmatrix},$$

which can be estimated from data. Note that unlike our analysis of the error rates, it is not necessary to assume two-class problems.

Our goal is to recover the classifier’s confusion matrix given the labeler’s confusion matrix and the joint distribution matrix between labeler and classifier. If we make the independence assumption that $P(\tilde{y}, \hat{y} | y) = P(\tilde{y} | y)P(\hat{y} | y)$, then we have the decomposition $P(\tilde{y}, \hat{y}) = \sum_y P(y)P(\tilde{y} | y)P(\hat{y} | y)$. We can rewrite the decomposition in matrix form and solve for the classifier’s confusion matrix.

$$\begin{aligned} \mathbf{P}_{\tilde{y}, \hat{y}} &= \mathbf{P}_{\tilde{y} | y} \text{diag}(\mathbf{p}_y) \mathbf{P}_{\hat{y} | y}^t \\ \mathbf{P}_{\hat{y} | y} &= ((\mathbf{P}_{\tilde{y} | y} \text{diag}(\mathbf{p}_y))^{-1} \mathbf{P}_{\tilde{y}, \hat{y}})^t \\ &= (\text{diag}(\mathbf{p}_y)^{-1} \mathbf{P}_{\tilde{y} | y}^{-1} \mathbf{P}_{\tilde{y}, \hat{y}})^t, \end{aligned} \quad (5)$$

in which $\mathbf{p}_y$ is defined to be the column vector of prior probabilities, $(P(y = \omega_1), \dots, P(y = \omega_c))^t$.

When $\mathbf{p}_y$ is not given, it can be derived. To see this, define the probability vector $\mathbf{p}_{\tilde{y}} \equiv (P(\tilde{y} = \omega_1), \dots, P(\tilde{y} = \omega_c))^t$. It is the case that $\mathbf{p}_{\tilde{y}} = \mathbf{P}_{\tilde{y}, \hat{y}} \mathbf{1}_c$, in which $\mathbf{1}_c$ is a column vector of c 1’s. It is also the case that $\mathbf{p}_{\tilde{y}} = \mathbf{P}_{\tilde{y} | y} \mathbf{p}_y$. Combining those two equations we have

$$\mathbf{p}_y = \mathbf{P}_{\tilde{y} | y}^{-1} \mathbf{P}_{\tilde{y}, \hat{y}} \mathbf{1}_c \quad (6)$$

5.1 Example: Precision/recall breakeven points

As mentioned earlier, one benefit of being able to recover the confusion matrix is that one can then work with precision and recall measures. We analyze the following system to see some effects of noisy labels on those measures. To reduce the number of variables examined, we only look at precision

recall breakeven points, defined as the points where precision and recall are equal. They will simply be denoted as precision/recall. For a given $\mathbf{p}_y$ and precision/recall, the confusion matrix is uniquely determined. Table 2 shows the apparent precision/recall (i.e., $P(\tilde{y} = \omega_2 | \hat{y} = \omega_2)$ or $P(\hat{y} = \omega_2 | \tilde{y} = \omega_2)$) for different actual classifier precision/recall and labeler precision/recall. Table 2 assumes $\mathbf{p}_y = (.9, .1)^t$, although the values are almost exactly the same for both $\mathbf{p}_y = (.8, .2)^t$ and $\mathbf{p}_y = (.999, .001)^t$.

6 Bounds on true error rate when the classifier and labeler are not independent

In deriving the true error rate (Eq. 2), we have made the assumption that the labeler and the classifier make errors independently. We argue for this assumption because human and computer use different methodologies to classify samples. Even for algorithms inspired by human reasoning (e.g., neural networks), they still do not learn human intuition but they do avoid psychological biases. It is even harder to imagine algorithms based on more abstract models (e.g., support vector machine) to err in similar ways as humans. Furthermore, in many application domains (e.g., speech recognition), humans label the samples based on a full presentation of the object, whereas the feature vector $\mathbf{x}$ used for classification are mathematical notions (e.g., linear vector coefficients) that have little neurological basis. In many other application domains (e.g., statistical text classification), assumptions that blatantly violate how human reasons are often made (e.g., assume words in a text are independently generated, rather than in a grammatical way). Lastly, to make a stronger argument, we can require the training data to be labeled independently from the testing data (or better yet, be perfectly labeled), thus avoiding the possibility that the computer would learn biases and other “bad habits” from the training data that would correlate with labeling errors in the testing data.

However, even with the above reasoning for the independence assumption, it is still conceivable for one to be more conservative and assume some *non-negative* dependency,

$$\Pr[y \neq \hat{y}, y \neq \tilde{y}] \geq \Pr[y \neq \hat{y}] \Pr[y \neq \tilde{y}].$$

That is, the probability of a classifier misclassifying a sample is higher if the labeler has also mislabeled that sample, and vice versa. This can happen, for example, if the training data have been mislabeled in the same way as the testing data, and the classifier has learned to imitate those mislabelings. We have deliberately ignored the case of negative dependency, $\Pr[y \neq \hat{y}, y \neq \tilde{y}] < \Pr[y \neq \hat{y}] \Pr[y \neq \tilde{y}]$, as we are hard-pressed to find a justification for it in practice.

The non-negative dependency assumption is easily incorporated into Eq. 1 by changing the equal sign to a less-than-or-equal-to sign. Propagating that change through the derivation, we have a lower bound on the true error rate,

$$\begin{aligned} \Pr[y \neq \hat{y}] &\geq \frac{\Pr[\hat{y} \neq \tilde{y}] - \Pr[y \neq \tilde{y}]}{1 - 2\Pr[y \neq \tilde{y}]} \\ &\geq \Pr[\hat{y} \neq \tilde{y}] - \Pr[y \neq \tilde{y}] \end{aligned}$$

The second inequality can be tight if the mislabeling rate is small, as the denominator of the first inequality becomes approximately one.

Separately we derive an upper bound for the true error rate.

$$\begin{aligned} \Pr[\hat{y} = \tilde{y}] &= \Pr[\hat{y} = \tilde{y}, y \neq \tilde{y}] + \Pr[\hat{y} = \tilde{y}, y = \tilde{y}] \\ &\leq \Pr[y \neq \tilde{y}] + \Pr[y = \hat{y}] \\ \Pr[y \neq \hat{y}] &\leq \Pr[y \neq \tilde{y}] + \Pr[\hat{y} \neq \tilde{y}]. \end{aligned}$$

Note that no assumption is used in deriving the upper bound (not even limiting to two-class problems). One can easily verify that the bound is exact when the mislabeling rate is zero. The bound is also exact when the apparent error rate is zero, such that the true error rate is equal to the mislabeling rate. Thus with the looser assumption of non-negative dependency, the true error rate is in the range of $\Pr[\hat{y} \neq \tilde{y}] \pm \Pr[y \neq \tilde{y}]$.

6.1 Example: Simulation of non-negative dependency between classifier and labeler

In the above derivation, the lower bound is achieved exactly when the mislabeling rate is small and the independence assumption is true. We examine how tight the upper bound is by simulation. We have taken pairs of classes from UCI’s Opt-Digit dataset, which is a handwritten digit recognition dataset, and trained both a naive Bayes classifier and a nearest-neighbor classifier [Duda *et al.*, 2001] on the training set of each pair. The nearest-neighbor classifier is then used to simulate a labeler and labeled the testing set. The naive Bayes classifier is the classifier under evaluation. Since we have the actual labels for the testing set, both the mislabeling rate (of the nearest-neighbor “labeler”) and the true error rate (of the naive Bayes classifier) can be determined. The output of the naive Bayes classifier and the nearest-neighbor “labeler” are compared to determine the apparent error rate.

We have chosen the Opt-Digit dataset and the nearest-neighbor algorithm because we know that this combination can give very low error rate [Kaynak and Alpaydin, 2000], thus closely matching the accuracy of many human labelers. In fact, for most pairs of classes, the nearest-neighbor algorithm has zero error. The Opt-Digit dataset is also interesting because the handwritten digit recognition task is a classical example in which much human labeling effort has been applied. The naive Bayes classifier is chosen because it is a popular classifier and is sufficiently different from nearest-neighbor to give interesting results.

Table 3 shows the results for some pairs of classes where the nearest-neighbor “labeler” has non-zero error. Note that an insignificant positive dependence between the naive Bayes classifier and the nearest-neighbor “labeler” should be expected since they both are trained from the same dataset, use the same features, and assume independence of those features (explicitly in naive Bayes and implicitly in nearest-neighbor through its distance metric), even though they are different in other aspects (e.g., naive Bayes is generative while nearest-neighbor classifier is discriminative). The naive Bayes classifier’s true error rate is almost exactly the upper bound for the pairs (1,2) and (4,5), but it is much closer to the apparent error rate for the pairs (7,8) and (8,9). The simulation thus shows the upper bound to be tight in some non-trivial situations.

7 Discussion and Future Work

When designing classification systems there are frequently parameters that are not learned automatically from the train-

	classes to discriminate ($\omega_1 - \omega_2$)			
	1 - 2	4 - 5	7 - 8	8 - 9
Upper bound (apparent error rate + mislabeling rate)	9.20%	1.11%	2.27%	6.77%
True error rate (Naive Bayes)	9.19%	1.10%	1.70%	5.65%
Apparent error rate	8.64%	0.83%	1.70%	5.08%
Lower bound (apparent error rate - mislabeling rate)	8.08%	0.55%	1.13%	3.39%
Mislabeling rate (Nearest Neighbor)	0.56%	0.28%	0.57%	1.69%

Table 3: Error rates on pairs of digits from the UCI Opt-Digit dataset. The classifier under evaluation is a Naive Bayes classifier (NB), and a nearest-neighbor classifier (NN) is used to simulate the (human) labeler. The error rate of the NB and NN classifiers are considered to be the true error rate and mislabeling rate, respectively. The fraction of time the two classifiers disagree is the apparent error rate. The upper and lower bounds are derived in Sec. 6, which are simply the apparent error rate plus or minus the mislabeling rate. The true error rate does come very close to the upper bound in some cases (1 - 2 and 4 - 5).

ing data. Some examples are the number of hidden units in a feedforward neural network, the number k in a k -nearest-neighbor classifier, and the window width in a Parzen window classifier [Duda *et al.*, 2001]. *Validation* is one technique to estimate those parameters. In validation, one conceptually creates several classifiers with different values of the parameter and train them with the same training set. The trained classifiers are evaluated on the *validation* data set, and the best classifier is chosen. Our results for testing with noisy labels is directly applicable to validation. In fact, validation is not concerned with the actual value of the true error rates, but just their ordering. Therefore the *apparent error rate* $\Pr[\hat{y} \neq \tilde{y}]$ can work just as well, as long as the mislabeling rate $\Pr[y \neq \tilde{y}]$ is less than 0.5.

In a world where (unlabeled) data is cheap, noisy labels are easily obtained (e.g., the Open Mind Initiative [Stork, 1999; Stork and Lam, 2000]), but perfect labels are expensive, the findings in this paper allow one to confidently use noisy labels for testing and validating. An obvious area for future work is to use noisy labels for training as well. Although some works do allow for training with noisy labels, this has not been an active research area [Szummer and Jaakkola, 2000].

So far in our derivations we have assume either knowledge of the mislabeling rate $\Pr[y \neq \tilde{y}]$ or the labeler’s confusion matrix $\mathbf{P}_{\tilde{y}|y}$. In practice those information must be estimated and be treated as random. The effects of such estimates and the cost/benefit analysis of obtaining more accurate estimates are unknown. We hope to investigate them in the future.

8 Conclusion

Traditionally test data have been assumed to be perfectly labeled. Increasingly this assumption is becoming a burden. We advocate the use of noisy labels as a cheaper alternative. We have shown that, under the assumption in which the labeler and the classifier make mistakes independently, the true error rate and true confusion matrix can be derived exactly. We have also examined the number of noisy labels to achieve the equivalent estimation confidence as one perfect label, and we found that number to be less than four in many practical situations. Furthermore, if we loosen the independence assumption to the non-negative dependence assumption, the true error rate can be bounded to be between the apparent error rate plus or minus the mislabeling rate.

References

- [Blum and Mitchell, 1998] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT '98)*, pages 92–100, 1998.
- [Duda *et al.*, 2001] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2001.
- [Eskin, 2000] Eleazar Eskin. Detecting errors within a corpus using anomaly detection. In *Proceedings of the First Conference of the North American Association for Computational Linguistics (NAACL-2000)*, 2000.
- [Frakes and Baeza-Yates, 1992] William B. Frakes and Ricardo Baeza-Yates, editors. *Information Retrieval: Data Structures & Algorithms*. Prentice Hall, 1992.
- [Kaynak and Alpaydin, 2000] Cenk Kaynak and Ethem Alpaydin. Multistage cascading of multiple classifiers: One man’s noise is another man’s data. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 455–462, 2000.
- [Nigam *et al.*, 2000] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled examples. *Machine Learning*, 39(2):103–134, 2000.
- [Smyth, 1997] Padhraic Smyth. Bounds on the mean classification error rate of multiple experts. *Pattern Recognition Letters*, 1997.
- [Stork and Lam, 2000] David G. Stork and Chuck P. Lam. Open Mind Animals: Ensuring the quality of data openly contributed over the world wide web. In *AAAI Workshop on Learning with Imbalanced Data Sets*, pages 4–9, 2000.
- [Stork, 1999] David G. Stork. The Open Mind Initiative. *IEEE Intelligent Systems & Their Applications*, 14(3):19–20, 1999.
- [Szummer and Jaakkola, 2000] Martin Szummer and Tommi Jaakkola. Kernel expansions with unlabeled examples. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.

AUC: a Statistically Consistent and more Discriminating Measure than Accuracy

Charles X. Ling Jin Huang
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7
{ling, jhuang}@csd.uwo.ca

Harry Zhang
Faculty of Computer Science
University of New Brunswick
Fredericton, NB, Canada E3B 5A3
hzhang@unb.ca

Abstract

Predictive accuracy has been used as the main and often only evaluation criterion for the predictive performance of classification learning algorithms. In recent years, the area under the ROC (Receiver Operating Characteristics) curve, or simply AUC, has been proposed as an alternative single-number measure for evaluating learning algorithms. In this paper, we prove that AUC is a better measure than accuracy. More specifically, we present rigorous definitions on consistency and discriminancy in comparing two evaluation measures for learning algorithms. We then present empirical evaluations and a formal proof to establish that AUC is indeed statistically consistent and more discriminating than accuracy. Our result is quite significant since we formally prove that, for the first time, AUC is a better measure than accuracy in the evaluation of learning algorithms.

1 Introduction

The predictive ability of the classification algorithm is typically measured by its predictive accuracy (or error rate, which is 1 minus the accuracy) on the testing examples. However, most classifiers (including C4.5 and Naive Bayes) can also produce probability estimations or “confidence” of the class prediction. Unfortunately, this information is completely ignored in accuracy. This is often taken for granted since the true probability is unknown for the testing examples anyway.

In many applications, however, accuracy is not enough. For example, in direct marketing, we often need to promote the top $X\%$ of customers during gradual roll-out, or we often deploy different promotion strategies to customers with different likelihood of buying some products. To accomplish these tasks, we need more than a mere classification of buyers and non-buyers. We need (at least) a ranking of customers in terms of their likelihoods of buying. If we want to achieve a more accurate ranking from a classifier, one might naturally expect that we must need the true ranking in the training examples [Cohen *et al.*, 1999]. In most scenarios, however, that is not possible. Instead, what we are given is a dataset of examples with class labels only. Thus, given only classification labels

in training and testing sets, are there better methods than accuracy to evaluate classifiers that also produce rankings?

The ROC (Receiver Operating Characteristics) curve has been recently introduced to evaluate machine learning algorithms [Provost and Fawcett, 1997; Provost *et al.*, 1998]. It compares the classifiers’ performance across the entire range of class distributions and error costs. However, often there is no clear dominating relation between two ROC curves in the entire range; in those situations, the area under the ROC curve, or simply AUC, provides a single-number “summary” for the performance of the learning algorithms. Bradley [Bradley, 1997] has compared popular machine learning algorithms using AUC, and found that AUC exhibits several desirable properties compared to accuracy. For example, AUC has increased sensitivity in Analysis of Variance (ANOVA) tests, is independent to the decision threshold, and is invariant to *a priori* class probability distributions [Bradley, 1997]. However, no formal arguments or criteria have been established. How can we compare two evaluation measures for learning algorithms? How can we establish that one measure is better than another? In this paper, we give formal definitions on the consistency and discriminancy for comparing two measures. We show, empirically and formally, that AUC is indeed a statistically consistent and more discriminating measure than accuracy.

One might ask why we need to care about anything more than accuracy, since by definition, classifiers only classify examples (and do not care about ranking and probability). We answer this question from two aspects. First, as we discussed earlier, even with labelled training and testing examples, most classifiers do produce probability estimations that can rank training/testing examples. Ranking is very important in most real-world applications. As we establish that AUC is a better measure than accuracy, we can choose classifiers with better AUC, thus producing better ranking. Second, and more importantly, if we build classifiers that optimize AUC (instead of accuracy), it has been shown [Ling and Zhang, 2002] that such classifiers produce not only better AUC (a natural consequence), but also better accuracy (a surprising result), compared to classifiers that optimize the accuracy. To make an analogy, when we train workers on a more complex task, they will do better on a simple task than workers who are trained only on the simple task.

Our work is quite significant for several reasons. First,

we establish rigourously, for the first time, that even given only labelled examples, AUC is a better measure (defined in Section 2.2) than accuracy. Second, our result suggests that AUC should replace accuracy in comparing learning algorithms in the future. Third, our results prompt and allow us to re-evaluate well-established results in machine learning. For example, extensive experiments have been conducted and published on comparing, in terms of accuracy, decision tree classifiers to Naive Bayes classifiers. A well-established and accepted conclusion in the machine learning community is that those learning algorithms are very similar as measured by accuracy [Kononenko, 1990; Langley *et al.*, 1992; Domingos and Pazzani, 1996]. Since we have established that AUC is a better measure, are those learning algorithms still very similar as measured by AUC? We have shown [Ling *et al.*, 2003] that, surprisingly, this is not true: Naive Bayes is significantly better than decision tree in terms of AUC. These kinds of new conclusions are very useful to the machine learning community, as well as to machine learning applications (e.g., data mining). Fourth, as a new measure (such as AUC) is discovered and proved to be better than a previous measure (such as accuracy), we can re-design most learning algorithms to optimize the new measure [Ferri *et al.*, 2002; Ling and Zhang, 2002]. This would produce classifiers that not only perform well in the new measure, but also in the previous measure, compared to the classifiers that optimize the previous measure, as shown in [Ling and Zhang, 2002]. This would further improve the performance of our learning algorithms.

2 Criteria for Comparing Evaluation Measures for Classifiers

We start with some intuitions in comparing AUC and accuracy, and then we present formal definitions in comparing evaluation measures for learning algorithms.

2.1 AUC vs Accuracy

Hand and Till [Hand and Till, 2001] present a simple approach to calculating the AUC of a classifier G below.

$$\hat{A} = \frac{S_0 - n_0(n_0 + 1)/2}{n_0 n_1}, \quad (1)$$

where n_0 and n_1 are the numbers of positive and negative examples respectively, and $S_0 = \sum r_i$, where r_i is the rank of i_{th} positive example in the ranked list. Table 1 shows an example of how to calculate AUC from a ranked list with 5 positive examples and 5 negative examples. The AUC of the ranked list in Table 1 is $\frac{(5+7+8+9+10)-5 \times 6/2}{5 \times 5}$, which is 24/25. It is clear that AUC obtained by Equation 1 is a measure for the quality of ranking, as the more positive examples are ranked higher (to the right of the list), the larger the term $\sum r_i$. AUC is shown to be equivalent to the Wilcoxon statistic rank test [Bradley, 1997].

Intuitively, we can see why AUC is a better measure than accuracy from the following example. Let us consider two classifiers, Classifier 1 and Classifier 2, both producing probability estimates for a set of 10 testing examples. Assume that both classifiers classify 5 of the 10 examples as positive, and

	-	-	-	-	+	-	+	+	+	+
i					1		2	3	4	5
r_i					5		7	8	9	10

Table 1: An example for calculating AUC with r_i

Classifier 1	-	-	-	-	+	-	+	+	+	+
Classifier 2	+	-	-	-	-	+	+	+	+	-

Table 2: An Example in which two classifiers have the same classification accuracy, but different AUC values

the other 5 as negative. If we rank the testing examples according to increasing probability of being + (positive), we get the two ranked lists as in Table 2.

Clearly, both classifiers produce an error rate of 20% (one false positive and one false negative), and thus the two classifiers are equivalent in terms of error rate. However, intuition tells us that Classifier 1 is better than Classifier 2, since overall positive examples are ranked higher in Classifier 1 than 2. If we calculate AUC according to Equation 1, we obtain that the AUC of Classifier 1 is $\frac{24}{25}$ (as seen in Table 1), and the AUC of Classifier 2 is $\frac{16}{25}$. Clearly, AUC does tell us that Classifier 1 is indeed better than Classifier 2.

Unfortunately, “counter examples” do exist, as shown in Table 3 on two other classifiers: Classifier 3 and Classifier 4. It is easy to obtain that the AUC of Classifier 3 is $\frac{21}{25}$, and the AUC of Classifier 4 is $\frac{16}{25}$. However, the error rate of Classifier 3 is 40%, while the error rate of Classifier 4 is only 20% (again we assume that the threshold for accuracy is set at the middle so that 5 examples are predicted as positive and 5 as negative). Therefore, a larger AUC does not always imply a lower error rate.

Classifier 3	-	-	-	+	+	-	-	+	+	+
Classifier 4	+	-	-	-	-	+	+	+	+	-

Table 3: A counter example in which one classifier has higher AUC but lower classification accuracy

Another intuitive argument for why AUC is better than accuracy is that AUC is more discriminating than accuracy since it has more possible values. More specifically, given a dataset with n examples, there is a total of only $n + 1$ different classification accuracies ($0/n, 1/n, \dots, n/n$). On the other hand, assuming there are n_0 positive examples and n_1 negative examples ($n_0 + n_1 = n$), there are $n_0 n_1 + 1$ different AUC values ($0/n_0 n_1, 1/n_0 n_1, \dots, n_0 n_1/n_0 n_1$), generally more than $n + 1$. However, counter examples also exist in this regard. Table 4 illustrates that classifiers with the same AUC can have different accuracies. Here, we see that Classifier 5 and Classifier 6 have the same AUC ($\frac{3}{5}$) but different accuracies (60% and 40% respectively). In general, a measure with more values is not necessarily more discriminating. For example, the weight of a person (having infinitely many possible values) has nothing to do with the number of siblings (having only a small number of integer values) he or she has.

Classifier 5	- - + + -	+ + - - +
Classifier 6	- - + + +	- - + - +

Table 4: A counter example in which two classifiers have same AUC but different classification accuracies

How do we compare different evaluation measures for learning algorithms? Some general criteria must be established.

2.2 (Strict) Consistency and Discriminancy of Two Measures

Intuitively speaking, when we discuss two different measures f and g on evaluating two learning algorithms A and B, we want at least that f and g be *consistent* with each other. That is, when f stipulates that algorithm A is (strictly) better than B, then g will not say B is better than A. Further, if f is more *discriminating* than g , we would expect to see cases where f can tell the difference between algorithms A and B but g cannot.

This intuitive meaning of consistency and discriminancy can be made precise as the following definitions.

Definition 1 (Consistency) For two measures f, g on domain Ψ , f, g are (strictly) consistent if there exist no $a, b \in \Psi$, such that $f(a) > f(b)$ and $g(a) < g(b)$.

Definition 2 (Discriminancy) For two measures f, g on domain Ψ , f is (strictly) more discriminating than g if there exist $a, b \in \Psi$ such that $f(a) > f(b)$ and $g(a) = g(b)$, and there exist no $a, b \in \Psi$ such that $g(a) > g(b)$ and $f(a) = f(b)$.

As an example, let us think about numerical marks and letter marks that evaluate university students. A numerical mark gives 100, 99, 98, ..., 1, or 0 to students, while a letter mark gives A, B, C, D, or F to students. Obviously, we regard $A > B > C > D > F$. Clearly, numerical marks are consistent with letter marks (and vice versa). In addition, numerical marks are more discriminating than letter marks, since two students who receive 91 and 93 respectively receive different numerical marks but the same letter mark, but it is not possible to have students with different letter marks (such as A and B) but with the same numerical marks. This ideal example of a measure f (numerical marks) being strictly consistent and more discriminating than another g (letter marks) can be shown in the figure 1(a).

2.3 Statistical Consistency and Discriminancy of Two Measures

As we have already seen in Section 2.1, counter examples on consistency and discriminancy do exist for AUC and accuracy. Therefore, it is impossible to prove the consistency and discriminancy on AUC and accuracy based on Definitions 1 and 2. Figure 1(b) illustrates a situation where one measure f is not completely consistent with g , and is not strictly more discriminating than g . In this case, we must consider the probability of being consistent and degree of being more discriminating. What we will define and prove is the *probabilistic version* of the two definitions on strict consistency and discriminancy. That is, we extend the previous definitions to *degree of consistency* and *degree of discriminancy*, as follows:

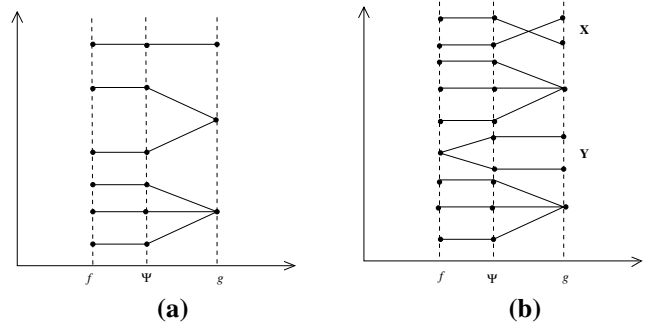


Figure 1: Illustrations of two measures f and g . A link between two points indicates that the function values are the same on domain Ψ . In (a), f is strictly consistent and more discriminating than g . In (b), f is not strictly consistent or more discriminating than g . Counter examples on consistency (denoted by X in the figure) and discriminancy (denoted by Y) exist here

Definition 3 (Degree of consistency) For two measures f and g on domain Ψ , let $R = \{(a, b) | a, b \in \Psi, f(a) > f(b), g(a) > g(b)\}$, $S = \{(a, b) | a, b \in \Psi, f(a) > f(b), g(a) < g(b)\}$. The degree of consistency¹ of f and g is C ($0 \leq C \leq 1$), where $C = \frac{|R|}{|R| + |S|}$.

Definition 4 (Degree of discriminancy) For two measures f and g on domain Ψ , let $P = \{(a, b) | a, b \in \Psi, f(a) > f(b), g(a) = g(b)\}$, $Q = \{(a, b) | a, b \in \Psi, g(a) > g(b), f(a) = f(b)\}$. The degree of discriminancy for f over g is $D = \frac{|P|}{|Q|}$.

There are clear and important implications of these definitions of measures f and g in evaluating two machine learning algorithms, say A and B. If f and g are consistent to degree C , then when f stipulates that A is better than B, there is a probability C that g will agree (stipulating A is better than B). If f is D times more discriminating than g , then it is D times more likely that f can tell the difference between A and B but g cannot than that g can tell the difference between A and B but f cannot. Clearly, we require that $C > 0.5$ and $D > 1$ if we want to conclude a measure f is “better” than a measure g . This leads to the following definition:

Definition 5 The measure f is statistically consistent and more discriminating than g if and only if $C > 0.5$ and $D > 1$. In this case, we say, intuitively, that f is a better measure than g .

The statistical consistency and discriminancy is a special case of the strict consistency and more discriminating. For the example of numerical and letter marks in the student evaluation discussed in Section 2.2, we can obtain that $C = 1.0$ and $D = \infty$, as the former is strictly consistent and more discriminating than the latter.

¹It is easy to prove that this definition is symmetric; that is, the degree of consistency of f and g is same as the degree of consistency of g and f .

To prove AUC is statistically consistent and more discriminating than accuracy, we substitute f by AUC and g by accuracy in the definition above. To simplify our notation, we will use AUC to represent AUC values, and acc for accuracy. The domain Ψ is the ranked lists of testing examples (with n_0 positive and n_1 negative examples). Since we require $C > 0.5$ and $D > 1$ we will essentially need to prove:

Theorem 1 *Given a domain Ψ , let $R = \{(a, b) | AUC(a) > AUC(b), acc(a) > acc(b), a, b \in \Psi\}$, $S = \{(a, b) | AUC(a) < AUC(b), acc(a) > acc(b), a, b \in \Psi\}$. Then $\frac{|R|}{|R|+|S|} > 0.5$ or $|R| > |S|$.*

Theorem 2 *Given a domain Ψ , let $P = \{(a, b) | AUC(a) > AUC(b), acc(a) = acc(b), a, b \in \Psi\}$, $Q = \{(a, b) | acc(a) > acc(b), AUC(a) = AUC(b), a, b \in \Psi\}$. Then $|P| > |Q|$.*

3 Empirical Verification on AUC and Accuracy

Before we present a formal proof that AUC is statistically consistent and more discriminating than accuracy, we first present an empirical verification. To simplify our notation, we will use AUC to represent AUC values, and acc for accuracy.

We conduct experiments with (artificial) testing sets to verify the statistical consistency and discriminancy between AUC and accuracy. The datasets are balanced with equal numbers of positive and negative examples. We test datasets with 4, 6, 8, 10, 12, 14, and 16 testing examples. For each number of examples, we enumerate all possible ranked lists of positive and negative examples. For the dataset with $2n$ examples, there are $\binom{2n}{n}$ such ranked lists.

We exhaustively compare all pairs of ranked lists to see how they satisfy the consistency and discriminating propositions probabilistically. To obtain degree of consistency, we count the number of pairs which satisfy “ $AUC(a) > AUC(b)$ and $acc(a) > acc(b)$ ”, and the number of pairs which satisfy “ $AUC(a) > AUC(b)$ and $acc(a) < acc(b)$ ”. We then calculate the percentage of those cases; that is, the degree of consistency. To obtain degree of discriminancy, we count the number of pairs which satisfy “ $AUC(a) > AUC(b)$ and $acc(a) = acc(b)$ ”, and the number of pairs which satisfy “ $AUC(a) = AUC(b)$ and $acc(a) > acc(b)$ ”.

Tables 5 and 6 show the experiment results for the balanced dataset. For consistency, we can see (Table 5) that for various numbers of balanced testing examples, given $AUC(a) > AUC(b)$, the number (and percentage) of cases that satisfy $acc(a) > acc(b)$ is much greater than those that satisfy $acc(a) < acc(b)$. When n increases, the degree of consistency (C) seems to approach 0.94, much larger than the required 0.5. For discriminancy, we can see clearly from Table 6 that the number of cases that satisfy $AUC(a) > AUC(b)$ and $acc(a) = acc(b)$ is much more (from 15.5 to 18.9 times more) than the number of cases that satisfy $acc(a) > acc(b)$ and $AUC(a) = AUC(b)$. When n increases, the degree of discriminancy (D) seems to approach 19, much larger than the required threshold 1.

These empirical experiments verify empirically that AUC is indeed a statistically consistent and more discriminating

#	$AUC(a) > AUC(b)$ & $acc(a) > acc(b)$	$AUC(a) > AUC(b)$ & $acc(a) < acc(b)$	C
4	9	0	1.0
6	113	1	0.991
8	1459	34	0.977
10	19742	766	0.963
12	273600	13997	0.951
14	3864673	237303	0.942
16	55370122	3868959	0.935

Table 5: Experimental results for verifying statistical consistency between AUC and accuracy for the balanced dataset

#	$AUC(a) > AUC(b)$ & $acc(a) = acc(b)$	$acc(a) > acc(b)$ & $AUC(a) = AUC(b)$	D
4	5	0	∞
6	62	4	15.5
8	762	52	14.7
10	9416	618	15.2
12	120374	7369	16.3
14	1578566	89828	17.6
16	21161143	1121120	18.9

Table 6: Experimental results for verifying AUC is statistically more discriminating than accuracy for the balanced dataset

measure than accuracy for the balanced datasets, as a measure for learning algorithms.

4 Formal Proof

In our following discussion, we assume that the domain Ψ is the set of all the ranked lists with n_0 positive and n_1 negative examples. In this paper, we only study the cases where a ranked list contains an equal number of positive and negative examples, and we assume that the cutoff for classification is at the exact middle of the ranked list (each classifier classifies exactly half examples into the positive class and the other half into the negative class). That is, $n_0 = n_1$, and we use n instead of n_0 and n_1 .

Lemma 1 gives the maximum and minimum AUC values for a ranked list with a fixed accuracy rate.

Lemma 1 *For a given accuracy rate α , let $AUC_{max}(\alpha) = \max\{AUC(r) | acc(r) = \alpha, r \in \Psi\}$, $AUC_{min}(\alpha) = \min\{AUC(r) | acc(r) = \alpha, r \in \Psi\}$. Then $AUC_{min}(\alpha) = \alpha^2$, $AUC_{max}(\alpha) = 1 - (1 - \alpha)^2$.*

Proof: Assume that there are n_p positive examples correctly classified. Then there are $n - n_p$ positive examples in negative section. According to Equation 1, AUC value is determined by S_0 , which is the sum of indexes of all positive examples. So when all $n - n_p$ positive examples are put on the highest positions in negative section (their indexes are $n, n-1, \dots, n_p+1$), and n_p positive examples are put on the highest positions in positive section (their indexes are $2n, 2n-1, \dots, 2n-n_p+1$),

AUC reaches the maximum value. So

$$AUC_{max}(\alpha) = \frac{\sum_{n_p+1}^n i + \sum_{2n-n_p+1}^{2n} i - n(n+1)/2}{n^2}.$$

Similarly when all positive examples are put on the lowest positions in both positive and negative sections, AUC reaches the minimum value. Thus

$$AUC_{min}(\alpha) = \frac{\sum_1^{n-n_p} i + \sum_{n+1}^{n+n_p} i - n(n+1)/2}{n^2}.$$

Since $\alpha = \frac{n_p}{n}$, from above two formula, we can obtain $AUC_{min}(\alpha) = \alpha^2$, $AUC_{max}(\alpha) = 1 - (1 - \alpha)^2$. $\square$

Lemma 2 For given accuracy rates $\alpha > \beta$ and a domain Ψ , $AUC_{max}(\alpha) > AUC_{max}(\beta)$, $AUC_{min}(\alpha) > AUC_{min}(\beta)$.

Proof: It is straightforward to prove it from Lemma 1. $\square$

Lemma 3 Let $\Gamma(\alpha) = \{AUC(r) | acc(r) = \alpha, r \in \Psi\}$, $R_1 = \{r | acc(r) = \alpha, AUC(r) = \beta, r \in \Psi\}$, $R_2 = \{r | acc(r) = \alpha, AUC(r) = AUC_{max}(\alpha) + AUC_{min}(\alpha) - \beta, r \in \Psi\}$. Then

$$(a) |\Gamma(\alpha)| = 2nn_p - 2n_p^2 + 1.$$

$$(b) |R_1| = |R_2|.$$

Proof: (a) It is straightforward to prove it from Lemma 1.

(b) For any ranked list $r \in R_1$ we can convert it to another unique ranked list $r' \in R_2$ by exchanging the positions of examples. In the negative section, for any positive example whose position is r_i , we exchange it with the example in the position of $n + 1 - r_i$. In the positive section, for any positive example in position r_i , we exchange it with the example in the position of $n - (r_i - n) + 1 + n$. It's easy to obtain that $AUC(r') = AUC_{max} + AUC_{min} - \beta$. So $r' \in R_2$. Thus, we have that $|R_1| \leq |R_2|$. Similarly, we can prove $|R_2| \leq |R_1|$. Therefore $|R_1| = |R_2|$. $\square$

Theorem 1 Given a domain Ψ , let $R = \{(a, b) | AUC(a) > AUC(b), acc(a) > acc(b), a, b \in \Psi\}$, $S = \{(a, b) | AUC(a) < AUC(b), acc(a) > acc(b), a, b \in \Psi\}$. Then $\frac{|R|}{|R|+|S|} > 0.5$ or $|R| > |S|$.

Proof: To prove $|R| > |S|$, first we prove for given accuracy rates $\alpha > \beta$, there are more ranked list pairs (a,b) to satisfy $AUC(a) > AUC(b)$ than that to satisfy $AUC(a) < AUC(b)$, where $acc(a) = \alpha, acc(b) = \beta$. Let $R_{\alpha\beta} = \{(a, b) | AUC(a) > AUC(b), acc(a) = \alpha, acc(b) = \beta, a, b \in \Psi\}$ and $S_{\alpha\beta} = \{(a, b) | AUC(a) < AUC(b), acc(a) = \alpha, acc(b) = \beta, a, b \in \Psi\}$. Then we should prove $|R_{\alpha\beta}| > |S_{\alpha\beta}|$.

1) If $AUC_{min}(\alpha) > AUC_{max}(\beta)$, then it is obvious $|R_{\alpha\beta}| > 0, |S_{\alpha\beta}| = 0$. So $|R_{\alpha\beta}| > |S_{\alpha\beta}|$.

2) Otherwise, $|R_{\alpha\beta}|$ can be computed by adding two parts together. $|R_{\alpha\beta}| = |P_1| + |P_2|$, where $P_1 = \{(a, b) | acc(a) = \alpha, acc(b) = \beta, AUC(a) > AUC_{max}(\beta), a, b \in \Psi\}$, and $P_2 = \{(a, b) | acc(a) = \alpha, acc(b) = \beta, AUC_{max}(\beta) \geq AUC(a) > AUC(b), a, b \in \Psi\}$.

For $S_{\alpha\beta}$, we have $S_{\alpha\beta} = \{(a, b) | acc(a) = \alpha, acc(b) = \beta, AUC_{max}(\beta) \geq AUC(b) > AUC(a) > AUC_{min}(\alpha), a, b \in \Psi\}$.

It can be proved that $|R_{\alpha\beta}| > |S_{\alpha\beta}|$. We omit the details due to the space limitation.

Since $R = \bigcup_{\alpha, \beta} R_{\alpha\beta}$, $S = \bigcup_{\alpha, \beta} S_{\alpha\beta}$, and $R_{\alpha_1\beta_1} \cap R_{\alpha_2\beta_2} = \emptyset$, $S_{\alpha_1\beta_1} \cap S_{\alpha_2\beta_2} = \emptyset$, for $\alpha_1 \neq \alpha_2$ or $\beta_1 \neq \beta_2$, where $\alpha > \beta$. So $|R| = \sum_{\alpha, \beta} |R_{\alpha\beta}|$, $|S| = \sum_{\alpha, \beta} |S_{\alpha\beta}|$. For any given accuracies $\alpha > \beta$, we have proved $|R_{\alpha\beta}| > |S_{\alpha\beta}|$, therefore, $|R| > |S|$. $\square$

Now let us explore the discriminancy of AUC and accuracy. Lemma 4 to Lemma 5 are proposed as a basis to prove Theorem 2. In the following lemmas, $|r|$ stands for the number of examples in ranked list r , and $|r|_+$ stands for the number of positive examples in r .

Lemma 4 Assume that R_1 and R_2 are two sets of ranked lists on a given domain Ψ , and that $R_1 = \{r | |r| = n, |r|_+ = m, 0 < m < \frac{n}{2}, AUC(r) = C, r \in \Psi\}$, $R_2 = \{r | |r| = n, |r|_+ = n - m, 0 < m < \frac{n}{2}, AUC(r) = C, r \in \Psi\}$. Then $|R_1| = |R_2|$.

Proof: For any ranked list $r \in R_1$, we switch the examples on position i and $n - i$, to obtain another unique ranked list s . Next we replace all positive examples in s by negative examples and all negative examples in s by positive examples to obtain ranked list s' . It is obvious that $AUC(s') = AUC(r)$. Since $|s'|_+ = n - m$, $s' \in R_2$. Thus, $|R_1| \leq |R_2|$. Similarly we can prove $|R_2| \leq |R_1|$. Therefore $|R_1| = |R_2|$. $\square$

Lemma 5 Given a domain Ψ , let α and β be accuracy rates $\alpha < \beta \leq \frac{1}{2}$. Consider a series of ranked list sets $R_i = \{r | acc(r) = \alpha, AUC(r) = \frac{i}{n^2}, r \in \Psi\}$, where $i = 0, \dots, n^2$; $S_j = \{s | acc(s) = \beta, AUC(s) = \frac{j}{n^2}, s \in \Psi\}$, where $j = 0, \dots, n^2$; Let $a_i = |R_i|$, $b_i = |S_i|$, $c_i = a_i + b_i$. Then

$$\sum_{i \neq j} c_i \bullet c_j \geq \sum_{i=0}^{n^2} a_i \bullet \sum_{i=0}^{n^2} b_i.$$

Proof: We omit the proof due to the limitation of space. $\square$

Theorem 2 Given a domain Ψ , let $P = \{(a, b) | AUC(a) > AUC(b), acc(a) = acc(b), a, b \in \Psi\}$, $Q = \{(a, b) | acc(a) > acc(b), AUC(a) = AUC(b), a, b \in \Psi\}$. Then $|P| > |Q|$.

Proof: Since Ψ is the set of all the ranked lists with equal number of positive and negative examples, Ψ can be partitioned into ranked list sets P_{ij} by different accuracy rates and AUC values. $P_{ij} = \{r | acc(r) = \frac{i}{n}, AUC(r) = \frac{j}{n^2}, 0 \leq i \leq n^2, 0 \leq j \leq n\}$. It is obvious $\Psi = \bigcup P_{ij}$. Let $t_{ij} = |P_{ij}|$. Then by Lemma 5 for any $i \neq j$ we have

$$\sum_{k \neq l}^{n^2} (t_{ki} + t_{kj})(t_{li} + t_{lj}) \geq (\sum_{k=0}^{n^2} t_{ki})(\sum_{l=0}^{n^2} t_{lj}).$$

After we sum all i and j , we have

$$\sum_{i \neq j} \sum_{k \neq l}^{n^2} (t_{ki} + t_{kj})(t_{li} + t_{lj}) \geq \sum_{i \neq j} (\sum_{k=0}^{n^2} t_{ki})(\sum_{l=0}^{n^2} t_{lj}). \quad (2)$$

Also for all $i \neq j$ and $k \neq l$, it's easy to obtain

$$\sum_{i \neq j} \sum_{k \neq l}^{n^2} (t_{ki} + t_{kj})(t_{li} + t_{lj}) = \sum_{i \neq k} t_{ij} t_{kl}. \quad (3)$$

Therefore, from inequality 2 and equation 3, we will get

$$\sum_{i \neq k} t_{ij} t_{kl} > \sum_{i \neq j} \left(\sum_{k=0}^{n^2} t_{ki} \right) \left(\sum_{k=0}^{n^2} t_{kj} \right). \quad (4)$$

On the other hand,

$$|D| = \sum_{i=0}^n \sum_{j=0}^n t_{ij} = \sum_{j=0}^n \sum_{i=0}^n t_{ij}.$$

So

$$\left(\sum_{j=0}^n \sum_{i=0}^n t_{ij} \right)^2 = \left(\sum_{i=0}^n \sum_{j=0}^n t_{ij} \right)^2.$$

The above equation can also be written as

$$\begin{aligned} \sum_{i=0}^{n^2} \left(\sum_{j=0}^n t_{ij} \right)^2 + \sum_{i \neq k} \left(\sum_{j=0}^n t_{ij} \sum_{j=0}^n t_{kj} \right) = \\ \sum_{j=0}^n \left(\sum_{i=0}^{n^2} t_{ij} \right)^2 + \sum_{j \neq k} \left(\sum_{i=0}^n t_{ij} \sum_{i=0}^n t_{ik} \right). \end{aligned} \quad (5)$$

According to formula 4, we have

$$\sum_{i \neq k} \left(\sum_{j=0}^n t_{ij} \sum_{j=0}^n t_{kj} \right) = \sum_{i \neq k} t_{ij} t_{kl} > \sum_{j \neq k} \left(\sum_{i=0}^{n^2} t_{ij} \sum_{i=0}^{n^2} t_{ik} \right). \quad (6)$$

Then by equation 5 and inequality 6, we have

$$\sum_{i=0}^{n^2} \left(\sum_{j=0}^n t_{ij} \right)^2 < \sum_{j=0}^n \left(\sum_{i=0}^{n^2} t_{ij} \right)^2. \quad (7)$$

Let

$$H_i = \sum_{j=0}^n t_{ij},$$

and

$$V_j = \sum_{i=0}^{n^2} t_{ij}.$$

Then

$$\begin{aligned} |R| &= \sum_{i=0}^n \binom{V_i}{2} = \sum_{i=0}^n \frac{V_i(V_i - 1)}{2} = \frac{1}{2} \sum_{i=0}^n V_i^2 - \frac{1}{2} |D| \\ |S| &= \sum_{i=0}^{n^2} \binom{H_i}{2} = \sum_{i=0}^{n^2} \frac{H_i(H_i - 1)}{2} = \frac{1}{2} \sum_{i=0}^{n^2} H_i^2 - \frac{1}{2} |D| \end{aligned}$$

Therefore, according to inequality 7, we have $|R| > |S|$. $\square$

5 Conclusions

In this paper, we give formal definitions of discriminancy and consistency in comparing evaluation measures for learning algorithms. We prove that AUC is statistically consistent and more discriminating than accuracy. Our result is quite significant since we have established rigorously, for the first time,

that AUC is a better measure than accuracy in evaluating and comparing classification learning algorithms. This conclusion has many important implications in evaluating, comparing, and designing learning algorithms. In our future research, we will extend the results in this paper to cases with unbalanced class distribution and multiple classes.

References

- [Bradley, 1997] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997.
- [Cohen et al., 1999] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.
- [Domingos and Pazzani, 1996] P. Domingos and M. Pazzani. Beyond independence: conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 105 – 112, 1996.
- [Ferri et al., 2002] C. Ferri, P. A. Flach, and J. Hernandez-Orallo. Learning decision trees using the area under the ROC curve. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002)*, pages 139–146, 2002.
- [Hand and Till, 2001] D. J. Hand and R. J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45:171–186, 2001.
- [Kononenko, 1990] I. Kononenko. Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition. In B. Wielinga, editor, *Current Trends in Knowledge Acquisition*. IOS Press, 1990.
- [Langley et al., 1992] P. Langley, W. Iba, and K. Thomas. An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference of Artificial Intelligence*, pages 223–228. AAAI Press, 1992.
- [Ling and Zhang, 2002] C. X. Ling and H. Zhang. Toward Bayesian classifiers with accurate probabilities. In *Proceedings of the Sixth Pacific-Asia Conference on KDD*. Springer, 2002.
- [Ling et al., 2003] C. X. Ling, J. Huang, and H. Zhang. AUC: A better measure than accuracy in comparing learning algorithms. In *Proceedings of 16th Canadian Conference on Artificial Intelligence*. Springer, 2003. To appear.
- [Provost and Fawcett, 1997] F. Provost and T. Fawcett. Analysis and visualization of classifier performance: comparison under imprecise class and cost distribution. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 43–48. AAAI Press, 1997.
- [Provost et al., 1998] F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453. Morgan Kaufmann, 1998.

LEARNING

INDUCTIVE LOGIC PROGRAMMING

Spaces of Theories with Ideal Refinement Operators

Nicola Fanizzi, Stefano Ferilli, Nicola Di Mauro, Teresa M.A. Basile

Dipartimento di Informatica, Università di Bari

Campus, Via Orabona 4, 70125 Bari, Italy

{fanizzi,ferilli,nicodimauro,basile}@di.uniba.it

Abstract

Refinement operators for theories avoid the problems related to the myopia of many relational learning algorithms based on the operators that refine single clauses. However, the non-existence of ideal refinement operators has been proven for the standard clausal search spaces based on θ -subsumption or logical implication, which scales up to the spaces of theories. By adopting different generalization models constrained by the assumption of object identity, we extend the theoretical results on the existence of ideal refinement operators for spaces of clauses to the case of spaces of theories.

1 Motivation

In the investigation of the algorithms for relational learning, regarding induction as a refinement process allows to decouple search from heuristics. Therefore, the choice of the generalization model for a search space plays a key role since it affects both its algebraic structure and the definition of refinement operators for that space.

Logical implication and θ -subsumption are the relationships that are commonly employed for inducing generalization models in relational learning (the latter turning out to be more tractable with respect to the former). Yet, they are not fully satisfactory because of the complexity issues that the resulting search spaces present, although subspaces have been found where the generalization model is more manageable. Indeed, the effectiveness and efficiency of learning as a refinement process strongly depends on the properties of the search space and, as a consequence, of the operators. In some cases the important property to be required to operators is *flexibility* [Badea, 2001], meaning that they should be capable of focussing dynamically on certain zones of the search space that may be more promising. Conversely, the property of *ideality* [Nienhuys-Cheng and de Wolf, 1997] has been recognized as particularly important for the efficiency of incremental algorithms in search spaces with *dense solutions*. It is also possible to derive *non-redundant* operators from ideal ones, the former being more suitable for spaces with *rare* solutions [Badea and Stanciu, 1999].

Most algorithms for relational learning, such as those employed in FOIL [Quinlan, 1990] and PROGOL [Mug-

leton, 1995], adopt iterative covering strategies, *separate-and-conquer* [Fürnkranz, 1999], based on the refinement of clauses. Alternative methods are based on *divide-and-conquer* strategies [Boström and Asker, 1999]. Although these refinements may turn out to be optimal with respect to a single clause, the result of assembling them in a theory is not guaranteed to be globally effective, since the interdependence of the clauses with respect to covering may lead to better theories made up of locally non-optimal clauses. Only some systems, e.g. MPL [De Raedt *et al.*, 1993] or HYPER [Bratko, 1999], cope with the problem of learning whole theories rather than constructing them clause by clause.

This urges more complex refinement operators to be adopted in algorithms obeying to a different strategy that is able to take into account the possible interactions between the single clausal refinements. Hence, the new problem is defining operators that refine whole theories rather than single clauses [Midelfart, 1999; Badea, 2001]. ALEPH is a system where ideas on *theory-level* induction has been rudimentarily implemented [Srinivasan, 2002]. Of course, heuristics are required to cope with the combinatorial complexity of the problem. The resulting extended setting would also take into account background knowledge that may be available, and then it is also comparable to *generalized* and *relative subsumption* [Plotkin, 1971; Buntine, 1988] or *implication* [Nienhuys-Cheng and de Wolf, 1997].

Weakening implication by assuming *object identity*, an extension of the *unique names assumption* [Reiter, 1980], as a semantic bias has led to the definition of θ_{OI} -subsumption and *OI-implication* [Esposito *et al.*, 2001b], clausal relationships which induce more manageable search spaces. The existence of ideal refinement operators in these generalization models is possible [Esposito *et al.*, 2001a], while this does not hold in clausal spaces ordered by θ -subsumption or implication [Nienhuys-Cheng and de Wolf, 1997]. The objective of this work is to extend the results obtained for spaces of clauses and prove the existence of ideal refinement operators for spaces of theories in those generalization models.

This paper is organized as follows. In Section 2, we recall semantics and proof-theory adopted in the framework. Section 3 deals with refinement operators and their properties. Then, in Section 4, the operators for the search space considered are defined and proven to be ideal. Section 5 summarizes the paper outlining possible developments.

2 Generalization Models and Object Identity

The representation language adopted in the proposed framework concerns logic theories (whose space is denoted 2^C) made up of clauses (space C). The background notions about clausal representations in Inductive Logic Programming can be found in [Nienhuys-Cheng and de Wolf, 1997].

The framework relies essentially on the following bias proposed in [Esposito *et al.*, 2001b]:

Assumption (Object Identity) *In a clause, terms denoted with different symbols must be distinct, i.e. they represent different entities of the domain.*

The intuition is the following: in spaces based on generalization models induced by θ -subsumption (or implication), considered the two clauses $C = q(x) \leftarrow p(X, X)$ and $D = q(x) \leftarrow p(X, X), p(X, Y), p(Y, Z), p(Z, X)$, they turn out to be equivalent (in fact C is the reduced clause of D); this is not so natural as it may appear, since more elements of the domain can be accounted for in D than in C (indeed in this framework C is more general than D). The expressive power is not diminished by this bias, since it is still possible to convey the same meaning of a clause, although employing more clauses, e.g. when object identity is assumed, $q(X) \leftarrow p(X, Y)$ is equivalent to the pair of clauses (a theory) $\{(q(X) \leftarrow p(X, X)), (q(X) \leftarrow p(X, Y))\}$.

Now it has to be specified how this bias affects the semantics and proof theory of a clausal representation.

2.1 Proof Theory and Semantics

Starting from substitutions, we recall the specification of the proof theory under object identity. Since a substitution can be regarded as a mapping from the variables to the terms of a language, we require these functions to satisfy additional properties to avoid the identification of terms:

Definition 2.1 *Given a set of terms T (omitted when obvious), a substitution σ is an OI-substitution w.r.t. T iff $\forall t_1, t_2 \in T: t_1 \neq t_2$ implies $t_1\sigma \neq t_2\sigma$.*

Based on OI-substitutions, it is possible to define related notions such as *ground* and *renaming* OI-substitutions, their composition and also unification:

Definition 2.2 *Given a finite set of clauses S , we say that θ is an OI-unifier iff $\exists E$ such that $\forall E_i \in S: E_i\theta = E$ and θ is an OI-substitution w.r.t. $\text{terms}(E_i)$.*

An OI-unifier θ for S is a most general OI-unifier for S iff for each OI-unifier σ of S there exists an OI-substitution τ such that $\sigma = \theta\tau$. This is denoted with $\text{mgu}_{\text{OI}}(S)$.

The following notions represent resolution and derivation when exclusively OI-unifiers are used.

Definition 2.3 *Given the clauses C and D that are supposed standardized apart, a clause R is an OI-resolvent of C and D iff there exist $M \subseteq C$ and $N \subseteq D$ such that $\{M, \bar{N}\}$ is unifiable through the mgu_{OI} and $R \in ((C \setminus M) \cup (D \setminus N))\theta$. $\mathcal{R}_{\text{OI}}(C, D)$ is the set of the OI-resolvents of C and D .*

An OI-derivation is obtained by successively chaining OI-resolutions.

Definition 2.4 *Given a theory T , the closure of OI-resolution is recursively defined: $\mathcal{R}_{\text{OI}}^*(T) = \bigcup_{n \geq 0} \mathcal{R}_{\text{OI}}^n(T)$, with $\mathcal{R}_{\text{OI}}^0(T) = T$ and $\mathcal{R}_{\text{OI}}^n(T) = \mathcal{R}_{\text{OI}}^{n-1}(T) \cup \{R = \mathcal{R}_{\text{OI}}(C, D) \mid C, D \in \mathcal{R}_{\text{OI}}^{n-1}(T)\}$. If $\exists C \in \mathcal{R}_{\text{OI}}^n(T)$ then there is an OI-derivation of C from T of length n .*

With respect to the model theory under object identity, more specific models are needed:

Definition 2.5 *Given a non-empty domain $\mathcal{D}$, a pre-interpretation J of the language $\mathcal{C}$ assigns each constant to a different element of $\mathcal{D}$ and each n -ary function symbol f to a mapping from $\mathcal{D}^n$ to $\mathcal{D}$.*

An OI-interpretation I based on J is a set of ground instances of atoms with arguments mapped in $\mathcal{D}$ through J .

Given a ground OI-substitution γ mapping $\text{vars}(\mathcal{C})$ to $\mathcal{D}$, an instance $A\gamma$ of an atom A is true in I iff $A\gamma \in I$ otherwise it is false in I . A negative literal $\neg A\gamma$ is true in I iff $A\gamma$ is not, otherwise it is false in I .

I is an OI-model for the clause C iff for all ground OI-substitutions γ there exists at least a literal in $C\gamma$ that is true in I , otherwise the clause is false in I .

Hence, the form of implication that is compliant with this semantics has been defined [Esposito *et al.*, 2001b] which, in turn, induces a quasi-order on spaces of clauses and theories.

Definition 2.6 *Let C, D be two clauses. C implies D under object identity (and then C is more general than D w.r.t. OI-implication) iff all OI-models for C are also OI-models for D . This relationship is denoted with $C \models_{\text{OI}} D$.*

Analogously, a theory T implies C under object identity, denoted with $T \models_{\text{OI}} C$, iff all OI-models for T are also OI-models for C . Finally, a theory T is more general than a theory T' w.r.t. OI-implication iff $\forall C' \in T': T \models_{\text{OI}} C'$.

OI-implication is a constrained form of logical implication biased by the object identity assumption, as shown in the following example:

Example 2.1 *Given the two theories $T = \{(p(X) \leftarrow q(f(X), Y), q(Y, f(X))), (q(f(X), f(X)) \leftarrow r(Z))\}$ and $T' = \{p(X) \leftarrow r(Z)\}$, observe that $T \models T'$ but $T \not\models_{\text{OI}} T'$. This depends on the disallowed binding $Y/f(X)$ that would identify terms within the same clause.*

The proof-procedure was proven sound in [Esposito *et al.*, 2001b], thus bridging the gap from the proof-theory to the model-theoretic definition of OI-implication.

2.2 θ_{OI} -subsumption and OI-implication

A simpler syntactic relationship, that is θ -subsumption biased by the object identity assumption, has been defined based the notion of OI-substitution.

Definition 2.7 *Given two clauses C and D , C θ_{OI} -subsumes D iff there exists an OI-substitution σ w.r.t. $\text{terms}(C)$ such that $C\sigma \subseteq D$. In this case, C is more general than D w.r.t. θ_{OI} -subsumption, denoted $C \geq_{\text{OI}} D$. If also $D \geq_{\text{OI}} C$ then they are equivalent w.r.t. θ_{OI} -subsumption, denoted $C \sim_{\text{OI}} D$.*

Analogously, given the theories T, T' , T is more general than T' w.r.t. θ_{OI} -subsumption iff $\forall D \in T' \exists C \in T: C \geq_{\text{OI}} D$, denoted $T \geq_{\text{OI}} T'$.

This quasi-order is weaker than OI-implication as proven by the following result [Esposito *et al.*, 2001b]:

Theorem 2.1 *Given a theory T and a non-tautological clause C , $T \models_{OI} C$ iff $\exists D \in \mathcal{R}_{OI}^*(T)$ such that $D \geq_{OI} C$.*

This result bridges the gap from model-theory to proof-theory. It also suggests the way to decompose OI-implication that is exploited for defining complete refinement operators.

Similarly to standard implication, it is nearly straightforward to demonstrate some consequences of Theorem 2.1 originally due to Gottlob [1987]. Given a clause C , let C^+ and C^- denote, respectively, the sets of its positive and negative literals. Then, it holds:

Proposition 2.1 *Let C and D be clauses. If $C \models_{OI} D$ then C^+ θ_{OI} -subsumes D^+ and C^- θ_{OI} -subsumes D^- .*

Since OI-substitutions map different literals of the subsuming clause onto different literals of the subsumed clause, equivalent clauses under θ_{OI} -subsumption have the same number of literals. Thus, a space ordered by θ_{OI} -subsumption is made up of non-redundant clauses. Indeed, it holds:

Proposition 2.2 *Let C and D be two clauses. If C θ_{OI} -subsumes D then $|C| \leq |D|$. Moreover, $C \sim_{OI} D$ iff they are alphabetic variants.*

As a consequence of the propositions above, it is possible to prove the following results giving lower bounds for the depth and cardinality of clauses in the generalization model based on OI-implication [Fanizzi and Ferilli, 2002].

Definition 2.8 *The depth of a term t is 1 when t is a variable or a constant. If $t = f(t_1, \dots, t_n)$, then $depth(t) = 1 + \max_{i=1}^n (depth(t_i))$. The depth of a clause C , denoted $depth(C)$, is the maximum depth among its terms.*

Proposition 2.3 *Given the clauses C and D , if $C \models_{OI} D$ then it holds that $depth(C) \leq depth(D)$ and $|C| \leq |D|$.*

3 Theory Refinement and Object Identity

A learning problem can be cast as a search problem [Mitchell, 1982] where theory refinement is triggered when new evidence made available is to be assimilated. The canonical inductive paradigm requires the fulfillment of the properties of *completeness* and *consistency* for the synthesized theory with respect to a set of input examples. When an inconsistent (respectively, incomplete) hypothesis is detected, a specialization (resp., generalization) of the hypothesis is required in order to restore this property of the theory. In the former case the refinement operators must search the space looking for more specific theories (downward refinements); in the latter, more general theories (upward refinements) are required.

The formal definition of the refinement operators for generic search spaces, is based on the algebraic notion of a *quasi-ordered set* $\mathcal{S}$ that is a set endowed with an ordering relationship (say $\preceq$) that is reflexive and transitive.

Definition 3.1 *Given a quasi-ordered set $(\mathcal{S}, \preceq)$, a refinement operator is a mapping from $\mathcal{S}$ to $2^{\mathcal{S}}$ such that:*

- $\forall C \in \mathcal{S} : \rho(C) \subseteq \{D \in \mathcal{S} \mid D \preceq C\}$
(downward refinement operator)

- $\forall C \in \mathcal{S} : \delta(C) \subseteq \{D \in \mathcal{S} \mid C \preceq D\}$
(upward refinement operator)

A notion of closure upon refinement operators is required when proving the completeness of the operators.

Definition 3.2 *In a quasi-ordered set $(\mathcal{S}, \preceq)$, let τ be a refinement operator. The closure of τ for $C \in \mathcal{S}$ is recursively defined: $\tau^*(C) = \bigcup_{n \geq 0} \tau^n(C)$, where $\tau^0(C) = \{C\}$ and $\tau^n(C) = \{D \in \mathcal{S} \mid \exists E \in \tau^{n-1}(C) : D \in \tau(E)\}$.*

Ultimately, refinement operators should construct chains of refinements, i.e. a sequence $C_0, C_1, \dots, C_n$ of elements of $\mathcal{S}$ such that $C_i \in \tau(C_{i-1})$, $1 \leq i \leq n$ going from the starting elements to target ones.

3.1 Properties of the Refinement Operators

As mentioned above, the properties of the refinement operators depend on the algebraic structure of the search space. A refinement operator traverses a *refinement graph* in the search space, that is a directed graph containing an edge from T to T' in $\mathcal{S}$ in case the operator τ is such that $T' \in \tau(T)$.

A major source of inefficiency may come from refinements that turn out to be equivalent to the starting ones. Depending on the search algorithm adopted, computing refinements that are equivalent to some element that has been already discarded may introduce a lot of useless computation. As to the effectiveness of the search, a refinement operator should be able to find a path between any two comparable elements of the search space (or their equivalent representatives). It is desirable that at least one path in the graph can lead to target elements. This means that a complete refinement operator can derive any comparable element in a finite number of steps. The following properties formally define these concepts:

Definition 3.3 *In a quasi-ordered set $(\mathcal{S}, \preceq)$, a refinement operator τ is locally finite iff $\forall C \in \mathcal{S} : \tau(C)$ is finite and computable.*

A downward (resp. upward) refinement operator ρ (resp. δ) is proper iff $\forall C \in \mathcal{S} : D \in \rho(C)$ implies $D \prec C$ (resp. $D \in \delta(C)$ implies $C \prec D$).

A downward (resp. upward) refinement operator ρ (resp. δ) is complete iff $\forall C, D \in \mathcal{S}, D \prec C$ implies $\exists E \in \mathcal{S} : E \in \rho^(C)$ and $E \sim D$ (resp. $C \prec D$ implies $\exists E \in \mathcal{S} : E \in \delta^*(C)$ and $E \sim D$).*

Let us observe that local finiteness and completeness ensure the existence of a computable refinement chain to a target element, and properness ensure a more efficient refinement process, by avoiding the search of equivalent clauses. Then, the combination of these properties confers more effectiveness and efficiency to an operator:

Definition 3.4 *In a quasi-ordered set $(\mathcal{S}, \preceq)$, a refinement operator is ideal iff it is locally finite, proper and complete.*

As mentioned in the introduction, other important properties of refinement operators have been defined, yet they go beyond the scope of this paper which focusses on ideality.

3.2 Minimal Refinements of Clauses

The existence of maximal specializations and minimal generalizations of clauses was proven for both the θ_{OI} -subsumption

and the OI-implication generalization model [Fanizzi and Ferilli, 2002]. These results are briefly recalled here for being used in the construction of ideal refinement operators presented in the following section.

As a consequence of Theorem 2.1, some limitations are provable as concerns depth and cardinality for a clause that implies (subsumes) another clause under object identity. This yields a bound to the proliferation of possible generalizations:

Proposition 3.1 *Let C and D be two clauses. The set of generalizations of C and D w.r.t. OI-implication is finite.*

The proof is straightforward since the depths and cardinalities of the generalizations are limited as a consequence of Proposition 2.3. Now, given two clauses C and D , let G be the set of generalizations of $\{C, D\}$ w.r.t. OI-implication. Observe that $G \neq \emptyset$ since $\square \in G$. Proposition 3.1 yields that G is finite. Thus, since the test of OI-implication between clauses is decidable [Fanizzi and Ferilli, 2002], it is theoretically possible to determine the minimal elements of G by comparing the clauses in G and eliminating those that are overly general.

For computing theories that are proper generalizations of the starting ones, an operator for inverting OI-resolutions is needed which is similar to the *V-operator* for the inversion of resolution [Muggleton, 1995]:

Definition 3.5 *Given a theory T , the operator for the inversion of the OI-resolution is defined:*

$$\mathcal{V}_{OI}(T) = \{D \in \mathcal{C} \mid \exists C \in T, D' \in \mathcal{C}: C \in \mathcal{R}_{OI}(D, D')\}$$

Note that there is a lot of indeterminacy in this definition. Yet it suffices for our theoretical purposes. In fact, the specification of an actual operator to be implemented in a learning system should consider also other information (such as examples, background knowledge, etc.), to define the underlying heuristic component.

With respect to maximal specializations, the major difficulty comes from the fact that under standard implication, $C \cup D$ is a clause that preserves the models of either clause, hence turning out to be a maximal specialization. In this setting, as expected, more clauses are needed than a single one; indeed the following operator has been defined:

Definition 3.6 *Let C_1 and C_2 be two clauses such that C_1 and C_2 are standardized apart and K a set of new constants such that: $|K| \geq |\text{vars}(C_1 \cup C_2)|$. A new set of clauses is defined $\mathcal{U}_{OI}(C_1, C_2) = \{C \mid C = (C_1\sigma_1 \cup C_2\sigma_2)\sigma_1^{-1}\sigma_2^{-1}\}$ where σ_1 and σ_2 are Skolem substitutions for, respectively, C_1 and C_2 with K as their term set¹.*

Example 3.1 *Given two clauses $C_1 = \{p(X, Y), q(X)\}$ and $C_2 = \{p(X', Y'), r(X')\}$, the OI-substitutions $\sigma_1 = \{X/a, Y/b\}$ and $\sigma_2 = \{X'/a, Y'/b\}$ yield the following clause: $F_1 = \{p(X, Y), q(X), r(X)\}$. Similarly $\sigma_3 = \{X/a, Y/b\}$ and $\sigma_4 = \{X'/b, Y'/a\}$ yield the clause: $F_2 = \{p(X, Y), p(Y, X), q(X), r(X)\}$ and so on.*

It is easy to see that, clauses $(C_1\sigma_1 \cup C_2\sigma_2)\sigma_1^{-1}\sigma_2^{-1}$ are equivalent to those in $(C_1\sigma_1 \cup C_2\sigma_2)\sigma_2^{-1}\sigma_1^{-1}$. Besides, the clauses in $\mathcal{U}_{OI}(C, D)$ preserve the OI-models of C and D :

¹The term set of a set of clauses T by the Skolem substitution σ is the set of all terms occurring in $T\sigma$.

Proposition 3.2 *Let C, D and E be clauses such that C and D are standardized apart. If $C \models_{OI} E$ and $D \models_{OI} E$ then $\forall F \in \mathcal{U}_{OI}(C, D): F \models_{OI} E$.*

This result implies that $\mathcal{U}_{OI}(C, D)$ contains maximal specializations of the two clauses w.r.t. OI-implication. For the proof of ideality given in the next section, it is important to point out that this set of specializations is finite. Moreover, the definition of $\mathcal{U}_{OI}$ and also Proposition 3.2 can be extended to the case of multiple clauses [Fanizzi and Ferilli, 2002].

4 Ideal Operators for Theories

Nonexistence conditions for ideal refinement operators for generic spaces are given in [Nienhuys-Cheng and de Wolf, 1997]. A close relationship has been recognized between ideality and the covers of elements in $(S, \preceq)$, a downward cover of C being a D such that $D \prec C$ and $\nexists E: D \prec E \prec C$ (resp. $C \prec D$ and $\nexists E: C \prec E \prec D$ for the upward case). A necessary condition for the ideality of refinement operators is that they return supersets of the sets of covers.

Theorem 4.1 *In the quasi-ordered space² $(\mathcal{C}, \leq_\theta)$ an ideal refinement operator does not exist.*

The non-existence of ideal refinement operators for spaces of clauses ordered by implication can be proven as a consequence of this result [Nienhuys-Cheng and de Wolf, 1997], since θ -subsumption is weaker than logical implication. Besides, this can be extended, proving the non-existence of refinement operators for search spaces of theories endowed with the ordering relationship induced by θ -subsumption [Midelfart, 1999] and the one induced by logical implication [Nienhuys-Cheng and de Wolf, 1997]. Conversely, within the framework we present, it is possible to exploit the definition of the refinement operators for clausal spaces for demonstrating the existence of ideal operators on spaces of theories.

4.1 Ideal Operators for θ_{OI} -subsumption

With respect to the spaces of clauses in the generalization model induced by θ_{OI} -subsumption, we exploit the ideality of the operators for clausal spaces [Esposito et al., 2001a].

Definition 4.1 *In the quasi-ordered space $(2^{\mathcal{C}}, \geq_{OI})$, given a theory T , let T_{nr} be a non redundant theory equivalent to T . The downward refinement operator ρ_{OI} is defined as follows:*

- $[(T_{nr} \setminus S) \cup \{D \in \rho_{OI}(C) \mid C \in S\}] \in \rho_{OI}(T)$ where $S \subseteq T_{nr}$
- $T_{nr} \setminus \{C\} \in \rho_{OI}(T)$ with $C \in T_{nr}$

The upward refinement operator δ_{OI} is defined as follows:

- $[(T_{nr} \setminus S) \cup \{D \in \delta_{OI}(C) \mid C \in S\}] \in \delta_{OI}(T)$ where $S \subseteq T_{nr}$
- $T_{nr} \cup \{C\} \in \delta_{OI}(T)$ with $C \notin T_{nr}$

The ideality of these operators is proven as follows:

Theorem 4.2 *In the search space $(2^{\mathcal{C}}, \geq_{OI})$, the refinement operators ρ_{OI} and δ_{OI} are ideal.*

Proof:

²here $\leq_\theta$ denotes the order induced by θ -subsumption and the language is supposed to contain at least a binary predicate.

ρ_{OI} : (locally finite) obvious.

(proper) by the properness of ρ_{OI} for clauses.

(complete) Suppose $T \geq_{OI} T'$ and $T' \not\geq_{OI} T$.

Let $T' = \{D_i \mid i = 1, \dots, n\}$. The theories can be supposed to be non redundant, otherwise the reduced equivalent theory can be computed by removing clauses by means of the second item of the operator.

By definition $T \geq_{OI} T'$ means that $\forall D_i \in T', i \in \{1, \dots, n\}, \exists C_i \in T: C_i \geq_{OI} D_i$.

By the completeness of the operator ρ_{OI} for clauses, it holds that $\forall i \in \{1, \dots, n\} \exists k_i: D_i \in \rho_{OI}^{k_i}(C_i)$.

Starting from $T_1 = T$, the first component of the operator is iterated, obtaining for each T_j a refinement T_{j+1} , by choosing S_j as the subset of T_j made up of the clauses that are not in the target theory T' while being strictly more general than clauses in T' , that is $S_j = \{C \in T_j \setminus T' \mid \exists D \in T': C \geq_{OI} D\}$.

Eventually it holds that $\exists T_k \in \rho_{OI}^k(T)$ for some $k \leq \max_{i=1}^n(k_i)$ such that $\forall i D_i \in T_k$.

T_k may be larger than T' . Thus the second component of ρ_{OI} for theories can be employed for deleting the exceeding clauses from T_k yielding T' .

Finally we have that $T' \in \rho_{OI}^*(T)$.

δ_{OI} : Analogously.

4.2 Ideal Operators for OI-implication

These operators above will be embedded in the definition of the refinement operators for spaces of theories in the stronger order induced by OI-implication. Besides, notions and results given in Section 3.2 are also exploited. In particular, these operators should be able to compute specializations and generalizations that are able to reach those clauses involved in OI-resolution steps (and their inverse).

Definition 4.2 In the quasi-ordered space $(2^C, \models_{OI})$, the downward refinement operator ρ_{OI} is defined as follows, given any $T \in 2^C$

- $\mathcal{U}_{OI}(C)_{C \in S} C \in \rho_{OI}(T)$ where $S \subseteq T$
- $\bigcup_{C \in S} ((T \setminus S) \cup \rho_{OI}'(C)) \in \rho_{OI}(T)$
where $S \subseteq T$ and ρ_{OI}' denotes the downward refinement operator for clauses wrt $\models_{OI}$
- $T' \in \rho_{OI}(T)$ if $T' \in \rho_{OI}''(T)$
where ρ_{OI}'' denotes the downward refinement operator for theories wrt $\geq_{OI}$

The upward refinement operator δ_{OI} is defined as follows:

- $T \cup \{C\} \in \delta_{OI}(T)$ where $\exists R \subseteq T$ and $C \in \mathcal{V}_{OI}(T)$
- $\bigcup_{C \in S} ((T \setminus S) \cup \delta_{OI}'(C)) \in \delta_{OI}(T)$
where $S \subseteq T$ and δ_{OI}' denotes the upward refinement operator for clauses wrt $\models_{OI}$
- $T' \in \delta_{OI}''(T)$ if $T' \in \delta_{OI}(T)$
where δ_{OI}'' denotes the upward refinement operator for theories wrt $\geq_{OI}$

The ideality of these operators is stated by the following result (Figure 1 depicts the related refinement graph):

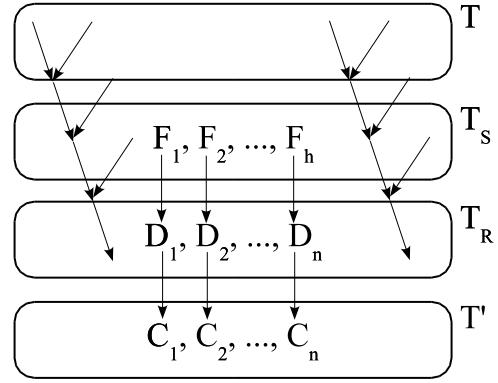


Figure 1: The Refinement Graph

Theorem 4.3 In the search space $(2^C, \models_{OI})$, the refinement operators ρ_{OI} and δ_{OI} are ideal.

Proof:

ρ_{OI} : (locally finite) by definition of the various operators and the finiteness of the theories.

(proper) by the properness of the refinement operators employed in the various items.

(complete) Suppose $T \models_{OI} T'$ and $T' \not\models_{OI} T$ with $n = |T'|$. Since redundancy can be eliminated by removing redundant clauses (and tautologies) through the last item of the operator, we will consider the theories as non redundant.

Let us observe that $T \models_{OI} T'$ is equivalent to $\forall C_i \in T': T \models_{OI} C_i$, for $i = 1, \dots, n$.

By Theorem 2.1, for each C_i , there exists a D_i such that $D_i \in \mathcal{R}_{OI}^{k_i}(T)$, for some k_i , and $D_i \theta_{OI}$ -subsumes C_i .

Observe that, by using the first item of the refinement operator, it is possible to produce the theory T_S containing the maximal specializations of the clauses in $S \subseteq T$ employed in the OI-derivation the of D_i 's (by the extension of Lemma 3.2 to the case of multiple clauses where each maximal specialization is more general wrt OI-implication than an OI-resolvent D_i): $T_S \in \rho_{OI}(T)$.

Observe that $\forall F \in T_S F \models_{OI} D_i$. Now, it is possible to iterate (at most n times for the D_i that are properly by some F) the second item of the operator, in order to compute the theory $T_R = \{D_i \mid i = 1, \dots, n\}$, thus we can write: $T_R \in \rho_{OI}^k(T_S), k \leq n$.

By construction $\forall i D_i \geq_{OI} C_i$, then $T_R \geq_{OI} T'$, then we can exploit the ideal operator for theories wrt θ_{OI} -subsumption (third item of the operator for theories wrt OI-implication) writing: $T' \in \rho_{OI}^*(T_R)$.

Finally, by chaining these steps, it is possible to conclude that: $T' \in \rho_{OI}^*(T)$.

δ_{OI} : In the same hypotheses the previous proof, it is possible to invert the θ_{OI} -subsumption of T_R wrt T' using the last item of the definition of δ_{OI} for theories wrt OI-implication: $T_R \in \delta_{OI}^*(T')$.

Then, a number of OI-resolutions are to be inverted by using the first item of δ_{OI} . This number is finite due to Proposition 2.3 and then can be done tentatively in a fi-

nite number of steps. Then $T \in \delta_{OI}^*(T_R)$.

Finally, by chaining these steps, it is possible to conclude that: $T \in \delta_{OI}^*(T')$.

Differently from the standard generalization models, in this framework the number of OI-resolution steps is bounded because, during OI-implication or θ_{OI} -subsumption steps, the sizes of the clauses increase (decrease) monotonically, as a consequence of Propositions 2.2 and 2.3.

5 Conclusions

In this work the existence of ideal refinement operators was proved in the search space of theories ordered by generalization models based on object identity. Coupled with some heuristics, this allows for the definition of efficient refinement algorithms that avoid the myopia of the traditional relational learning approaches.

We focussed on the effectiveness of the refinement operators, that is related to their static properties. In general this is not sufficient for defining a learning algorithm: efficiency plays a key role when dealing with first order logics. The successive step is to investigate the dynamic properties of these operators when they are to be guided by means of heuristics based on the available examples and/or other criteria.

We have also mentioned that in spaces with rare solutions it is more suitable to have an operator that is non redundant, because almost all of the paths that could be constructed would not lead to a target theory. It should be investigated how to define non redundant operators in this framework.

The object identity framework is currently implemented in the system INTHELEX [Esposito *et al.*, 2000], with a totally incremental strategy based on clause-level induction. We plan to investigate how to upgrade the system to theory-level, exploiting the theoretical operators presented in this paper together with suitable evaluation functions to mitigate the combinatorial complexity of the problem.

References

- [Badea and Stanciu, 1999] L. Badea and M. Stanciu. Refinement operators can be (weakly) perfect. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *LNAI*, pages 21–32. Springer, 1999.
- [Badea, 2001] L. Badea. A refinement operator for theories. In C. Rouveirol and M. Sebag, editors, *Proceedings of the 11th International Workshop on Inductive Logic Programming*, volume 2157 of *LNAI*, pages 1–14. Springer, 2001.
- [Boström and Asker, 1999] H. Boström and L. Asker. Combining divide-and-conquer and separate-and-conquer for efficient and effective rule induction. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *LNAI*, pages 33–43. Springer, 1999.
- [Bratko, 1999] I. Bratko. Refining complete hypotheses in ILP. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *LNAI*, pages 44–55. Springer, 1999.
- [Buntine, 1988] W. Buntine. Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence*, 36(2):149–176, 1988.
- [De Raedt *et al.*, 1993] L. De Raedt, N. Lavrač, and S. Džeroski. Multiple predicate learning. In R. Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, volume 2, pages 1037–1043. Morgan Kaufmann, 1993.
- [Esposito *et al.*, 2000] F. Esposito, G. Semeraro, N. Fanizzi, and S. Ferilli. Multistrategy theory revision: Induction and abduction in INTHELEX. *Machine Learning*, 38(1/2):133–156, 2000.
- [Esposito *et al.*, 2001a] F. Esposito, N. Fanizzi, S. Ferilli, and G. Semeraro. A generalization model based on OI-implication for ideal theory refinement. *Fundamenta Informaticae*, 47:15–33, 2001.
- [Esposito *et al.*, 2001b] F. Esposito, N. Fanizzi, S. Ferilli, and G. Semeraro. OI-implication: Soundness and refutation completeness. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 847–852, 2001.
- [Fanizzi and Ferilli, 2002] N. Fanizzi and S. Ferilli. Minimal generalizations under OI-implication. In M.-S. Hacid, Z. Raś, D. Zighed, and Y. Kodratoff, editors, *Proceedings of the 13th International Symposium on Methodologies for Intelligent Systems*, volume 2366 of *LNAI*. Springer, 2002.
- [Fürnkranz, 1999] J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence*, 13(1):3–54, 1999.
- [Gottlob, 1987] G. Gottlob. Subsumption and implication. *Information Processing Letters*, 24(2):109–111, 1987.
- [Midelfart, 1999] H. Midelfart. A bounded search space of clausal theories. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *LNAI*, pages 210–221. Springer, 1999.
- [Mitchell, 1982] T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.
- [Muggleton, 1995] S.H. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
- [Nienhuys-Cheng and de Wolf, 1997] S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *LNAI*. Springer, 1997.
- [Plotkin, 1971] G.D. Plotkin. A further note on inductive generalization. *Machine Intelligence*, 6:101–121, 1971.
- [Quinlan, 1990] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
- [Reiter, 1980] R. Reiter. Equality and domain closure in first order databases. *Journal of ACM*, 27:235–249, 1980.
- [Srinivasan, 2002] A. Srinivasan. The Aleph manual. Technical report, Oxford University Computing Laboratory, 2002. <http://www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph>.

Learning Minesweeper with Multirelational Learning

Lourdes Peña Castillo

Otto-von-Guericke-University
Magdeburg, Germany

Stefan Wrobel

Fraunhofer AIS, Sankt Augustin
and University Bonn, Germany

Abstract

Minesweeper is a one-person game which looks deceptively easy to play, but where average human performance is far from optimal. Playing the game requires logical, arithmetic and probabilistic reasoning based on spatial relationships on the board. Simply checking a board state for consistency is an NP-complete problem. Given the difficulty of hand-crafting strategies to play this and other games, AI researchers have always been interested in automatically learning such strategies from experience. In this paper, we show that when integrating certain techniques into a general purpose learning system (Mio), the resulting system is capable of inducing a Minesweeper playing strategy that beats the winning rate of average human players. In addition, we discuss the necessary background knowledge, present experimental results demonstrating the gain obtained with our techniques and show the strategy learned for the game.

1 Introduction

Minesweeper is a popular one-player computer game written by Robert Donner and Curt Johnson which was included in Microsoft Windows[©] in 1991. At the beginning of the game, the player is presented with a $p \times q$ board containing pq tiles or squares which are all blank. Hidden among the tiles are M mines distributed uniformly at random on the board. The task of the player is to uncover all the tiles which do not contain a mine. At each turn the player can select one of three actions (moves): to mark a tile as a mine; to unmark a tile; and to uncover a tile. In the last action, if the tile contains a mine, the player loses; otherwise, the number of mines around the tile is displayed. In the 4×4 board depicted in Fig. 1 left, the number 1 located on the second row from top indicates that there is one and only one mine hidden among the eight blank neighbouring tiles.

Although the simplicity of its rules makes Minesweeper look deceptively easy, playing the game well is indeed challenging: A player requires logic and arithmetic reasoning to perform certain moves given the board state, and probabilistic reasoning to minimize the risk of uncovering a mine when

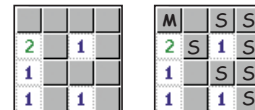


Figure 1: Left: Available information on a board. Right: Seven tiles can be determined safe (s) and one a mine(m)

a safe move cannot be done. Given the difficulty of hand-crafting playing strategies for this and other games, AI researchers have always been interested in the possibility of automatically learning such strategies from experience. However, with the exception of reinforcement learning [Tesauro, 1995], most of the playing strategies and heuristics used in game playing programs are coded and tuned per hand instead of automatically learned. In this work, we use a general purpose ILP system, Mio, to learn a playing strategy for Minesweeper. Multirelational learning or ILP consists in learning from examples usually within the framework provided by clausal logic.

The task of learning rules to deduce Minesweeper moves proved itself to be an arduous test for current multirelational learning systems. In this paper, we describe how recent optimizations make possible for Mio to discover a Minesweeper playing strategy. Experimental results obtained by playing Minesweeper using this strategy show a better performance than that obtained on average by non-expert human players.

The remainder of this paper is organized as follows. The next section discusses the complexity of the game and describes the learning task. Section 3 describes the background knowledge, the learning system and the new techniques used. Section 4 shows our empirical results on the effectiveness of the learning techniques, the strategy obtained and its performance at game playing. Related work is surveyed in Section 5 and Section 6 concludes.

2 Minesweeper

2.1 Why Is Minesweeper Interesting?

Minesweeper has been shown to be NP-complete by simulating boolean circuits as Minesweeper positions [Kaye, 2000]. Kaye describes the *Minesweeper consistency problem* as the problem of determining if there is some pattern of mines in the blank squares that give rise to the numbers seen in a given

board partially filled with numbers and marked mines, and thus determining that the data given is consistent.

One realizes the complexity of the game by calculating an estimate for the size of its search space. Consider an 8×8 board with $M = 10$ mines; in this case at the beginning of the game the player has $pq = 64$ tiles from which to choose a move (i.e., a tile to uncover) and in the last move, assuming the player does not uncover a mine, there are 11 tiles from which to choose one. This leads to $54! \approx 10^{71}$ possible move sequences to win a game. Alternatively, one can calculate the probability of a random player winning a game. In the first move the probability that the random player chooses a tile which does not contain a mine is $54/64$, and in the last move it has $1/11$ chance to choose the only tile without a mine. Then, the probability of a random player winning a game is $1/\binom{64}{54} \approx 10^{-12}$ and that is only for the easiest playing level!

Another measure of the complexity of Minesweeper is the number of games won on average by non-expert human players. To estimate the average human performance playing Minesweeper, we carried out an informal study. In the study, eleven persons who have played Minesweeper before were asked to play at least ten times in an 8×8 board with 10 mines. Every participant was told to aim for accuracy rather than for speed. In this study, a person won on average 35% of the games with a standard deviation of 8%.

2.2 The Learning Task

In Minesweeper there are situations that can be “solved” with nontrivial reasoning. For example, consider Fig. 1 left where the only available information about the board state are the numbers. After careful analysis one finds that the squares with an s (see Fig. 1 right) do not contain a mine, the square with an m is a mine, and the state of the blank tiles cannot be determined if we do not know how many mines are hidden in the board. There are other Minesweeper situations where the available information is not enough to identify a safe square or a mine, as in Fig. 2, and the best option available to the player is to make an informed guess, i.e., a guess that minimizes the risk of blowing up by uncovering a mine.

In this work, we consider the learning task in Minesweeper to be the induction of rules to identify all the *safe squares*¹ and squares with a mine which can be deduced given a board state. For instance, we want the system to learn rules to classify all the blank tiles in Fig. 1 either as *safe* or *mine*.

3 The Learning Tools

In machine learning it is possible to choose between a propositional representation (in the form of attribute-value tuples) and a multirelational representation (in the form of logic predicates). A multirelational representation has the expressiveness required to describe the concepts involved when reasoning about Minesweeper, and is thus more intuitive than the propositional one. For this reason we use multirelational learning for the learning task described above. Usually a multirelational learning system takes as input background knowledge B , positive (E^+) and negative (E^-) examples of a target

¹A safe square is a blank tile which given the current board state cannot contain a mine.

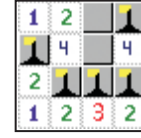


Figure 2: The position of the last mine cannot be determined

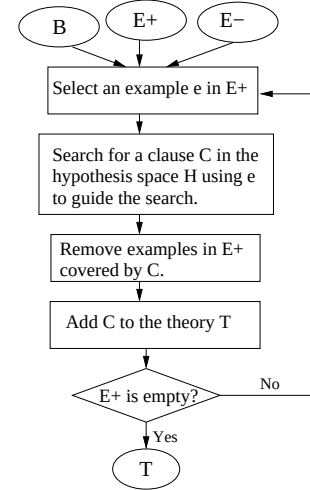


Figure 3: Example-driven covering algorithm

concept such that $E = E^+ \cup E^-$ and $E^+ \cap E^- = \emptyset$, and has to find a clausal theory T which minimizes the classification error on future instances. Next we describe the background knowledge and the system used.

3.1 Background Knowledge

The background knowledge provides the system with information about the domain and is given in the form of logic predicates (facts and rules, or clauses). A predicate is described as $name(arg_1, \dots, arg_n)$ and, in our case, arg_i indicates the argument’s type. The background knowledge provided to the learning system about Minesweeper is shown in Table 1. The predicates in the background knowledge were defined by trying to abstract the concepts used by humans when explaining their own Minesweeper playing strategies. These concepts were obtained from the first author’s Minesweeper playing experience and from Minesweeper pages on the web.

In the predicates listed in Table 1, a *TD* is a determined or uncovered tile, i.e., a number $0 \dots 8$ is shown on the tile; a *TU* is an undetermined or blank tile; *Board* is the board state given as a list of $p \times q$ characters $0 \dots 8, m, u$; *Zone* is a list of determined tiles, and *Set* is a set of undetermined tiles together with the number of mines hidden among those tiles. In addition, each symbol preceding an argument denotes how that argument should be instantiated when calling the predicate. A $+arg_i$ is an input argument and should be instantiated to a non-variable term; a $-arg_i$ is an output argument and should be uninstantiated, and a $\#arg$ indicates that the constant value of an output argument can appear in a rule.

Predicate	Description
zoneOfInterest(+TU, +Board, -Zone)	returns in Zone the tiles which are determined neighbours of TU and the determined tiles which share an undetermined neighbour with them (see Fig. 4 center).
totalMinesLeft(+Board, -Int)	returns how many mines remained to be marked.
allMinesInFringe(+Board, -Set)	gives the set of tiles in the fringe ³ where all the remaining mines are.
setHasXMines(-TD, +Board, +Zone, -Set)	gives in Set the undetermined neighbours of TD (TD is in Zone), and the number of mines hidden among them (see Fig. 4 right).
diffSetHasXMines(+Set1, +Set2, -Set) . . .	returns in Set all and only the tiles of Set1 which are not also in Set2 and the number of mines hidden among the tiles in Set.
inSet(+TU, +Set)	is true when TU is a member of Set.
lengthSet(+Set, +Int)	is true when Set contains Int tiles.
minesInSet(+Set, #Int)	returns the number of mines hidden among the tiles in Set.

Table 1: Minesweeper background knowledge

3.2 The System

Mio is an example-driven covering system (see Fig. 3) introduced by Peña Castillo and Wrobel [2002b] which uses a Progol-like declaration language and, the same as Progol [Muggleton, 1995], lower bounds the search space with a *most specific clause* $\perp$ (also called bottom clause). This $\perp$ is a maximally specific clause which entails (covers) a positive example e . Mio performs a general-to-specific (top-down) IDA* [Korf, 1985] search to find a clause to add to the theory. In addition, Mio selects stochastically the examples from which it learns, performs parallel search to increase the stability of the example-driven approach, and enforces type strictness. Three other techniques are implemented in Mio to allow the learning of Minesweeper rules: macro-operators (or macros, for short) to reduce the search space, greedy search with macros to speed up the learning process, and active learning to guide the exploration of the instance space.

Macros

Macros in multirelational learning [Peña Castillo and Wrobel, 2002a] are a formal technique to reduce the hypothesis space explored by a covering system. A macro is a sequence of literals, chosen from the bottom clause, which is created based on provider-consumer relationships between the literals. A literal is a provider if it has output arguments, and it is a consumer if it receives as an input argument a variable provided by another literal in the bottom clause. Peña and Wrobel show that by adding macros instead of literals to a clause, the number of clauses evaluated by the system is significantly reduced.

Greedy Search with Macros

In [Peña Castillo and Wrobel, 2002a] macros are used with IDA*. It is well known that greedy search explores on average less nodes than IDA*; however, greedy search could miss a solution because it underestimates the importance of provider literals without discriminative power which are nonetheless necessary to introduce new variables (this is known as the myopia problem). Since macros add several literals at once to a clause, they might reduce the myopia of greedy search allowing us to gain in efficiency without losing

too much in effectiveness. We implement a greedy search with macros which consists of a lookahead step where all the macros are combined with each other and the best evaluated clauses are selected. Then if the selected clauses can be extended (refined) the system tries to combine these clauses with all the macros available and selects the best candidates. This last step is repeated until there is no clause which can be extended and the best candidates are returned.

Active Inductive Learning

In the covering algorithm a clause is learned which covers (explains) some positive examples and none (or few) negative ones; however, in domains such as games and puzzles, thousands of examples are required to contain most of the possible game situations; on the other hand, considering thousands of examples when evaluating a rule slows down the learning process. Thus, to improve the efficiency of the exploration of the instance space, active learning [Cohn *et al.*, 1994] is included in Mio.

Active inductive learning consists of the following steps. At the beginning, Mio learns from few randomly drawn examples and when it has learned some clauses gives these clauses to an active learning server. The active learning server returns to Mio *counterexamples*⁴. These counterexamples are selected from examples given by a random example generator (or random sampler). While Mio iterates on the new examples received, the server tests the rules obtained against randomly drawn examples, discards all the rules below a user-defined accuracy value and collects new counterexamples. This validation step on the server side avoids overfitting⁵. These steps are repeated until a user-defined maximum number of iterations is reached or no counterexample is found.

Active inductive learning is similar in spirit to integrative windowing [Fürnkranz, 1998] with two main differences: in our approach random sampling is done dynamically and a client-server architecture is used which allows to treat testing and learning as separated processes.

⁴A counterexample is a positive example not covered by a set of clauses T or a negative example covered by at least one clause in T .

⁵Overfitting refers to obtaining results with high classification error over new data despite null or almost null training error.

³Fringe refers to all the blank tiles with a determined neighbour.

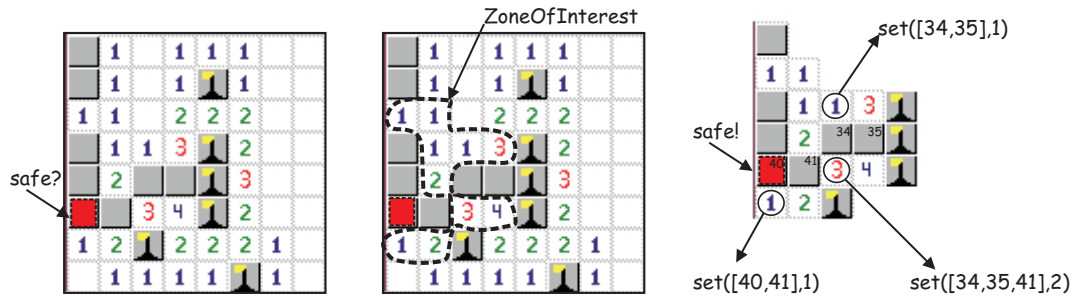


Figure 4: A rule learned by Mio. Left: Is the highlighted tile safe? Center: zoneOfInterest corresponding to the highlighted tile. Right: Applying difference operations to the sets determined by the tiles inside a circle is concluded that the tile 40 is safe

4 Empirical Results

4.1 Improvements Obtained with each Technique

Experiments were carried out to determine the effects on the rules obtained and on the system efficiency, of macros, greedy search with macros, and active inductive learning. To produce the training examples, we randomly generate board configurations and take all blank tiles with at least one determined neighbour as examples. If the blank tile does not contain a mine is labeled as safe, otherwise it is labeled as mine. Afterwards, contradictory examples are removed. In the experiments, the learning task was to learn rules to identify safe tiles. The rule to discover mines was learned using the best setting (i.e., using active learning, macros and greedy search).

For the completeness of this work, we ran Progol [Muggleton and Firth, 2001] and Foil [Quinlan and Cameron-Jones, 1995] on the same learning task as Mio; however, we failed to make the systems learn correct rules about safe tiles. Progol search was interrupted due to search limits implemented in the system (although the maximum stack depth and resolutions steps were set to 50000 and 10000, respectively), and Foil pruned determined literals which are needed. This might imply that the optimizations included in Mio are indeed necessary to learn a Minesweeper playing strategy. Table 2 shows the empirical results with four Mio settings.

All the experiments with active learning were performed with the same seeds which means that the same training examples are generated by the random sampler and that Mio selects the same examples to guide the search. For the experiments without active learning, we took five random samples from the set of examples used in the active learning experiments. The size of the sample is equal to the number of examples received by Mio when performing active learning (40 positive and 34 negative examples). We carried out an extra experiment where Mio was given the complete set of examples (2890 positive and 1306 negative) used by the active learning server to test Mio's rules and select counterexamples; however, this experiment was stopped after Mio ran for 10 days. To reduce the running time of the experiments, we set the maximum number of clauses explored per search to 4000 clauses.

Table 2 shows that each optimization added to Mio reduces the average number of rules (nodes) explored per search and

the number of times the search is interrupted because of the search limit. Without active learning, overfitting occurs and erroneous rules are obtained. The performance of the rules obtained with IDA*+M is worse than that of the IDA* rules because Mio with macros explores a larger part of the hypothesis space and thus the IDA* + M setting overfits more the training data. However, if no limit in the maximum number of clauses explored per search is set, both settings (IDA* and IDA*+M) obtain the same rules. The running time of the fastest setting (AL+GS+M) is 56hrs.

4.2 Rules Learned

Table 3 shows the rules with the highest winning rate which were obtained by using both AL+GS+M and AL+IDA*+M. An extra rule was obtained with the latter setting; however, this extra rule does not improve the playing performance. One important feature of the rules learned by Mio is that they can be applied independently of the size of the board and the number of mines. The rules vary in complexity. Rule S-1 and Rule M-1 correspond to the trivial situations where a determined tile needs k mines and k mines are already marked, and where a determined tile needs k mines and it has k blank neighbours, respectively.

On the other hand, Rule S-3 can be seen as one of the most complex rules because it involves three determined tiles to deduce a safe tile. Fig. 4 left shows a board state where Rule S-3 is the only one which allows to identify a safe tile. The rule obtains the zoneOfInterest corresponding to the undetermined tile considered (Fig. 4 center). Then by applying difference operations on the sets determined by three uncovered tiles from the zoneOfInterest (see Fig. 4 right), the set $([40], 0)$ is obtained and thus it is deduced that tile 40 is safe.

4.3 Game Playing

To evaluate the performance at game playing of each set of rules obtained, we used each set of rules as the playing strategy of an automatic Minesweeper player and calculated the percentage of games won by the player in 1000 random games (see Table 2). The playing conditions were the same as the ones presented to the human players; i.e., at the beginning the player is presented with an empty 8×8 board with $M = 10$ mines and can uncover a mine in the first move. Note that in most Minesweeper implementations, one never hits a mine in the first move.

	Mio			
	IDA*	IDA*+M	AL+IDA*+M	AL+GS+M
Ave. No. clauses explored per search	3135	1462	1008	566
% of searches which reached 4000-clause limit	70%	23%	4%	3%
Ave. No. of rules obtained	2.8	4	4	3
Ave. % of games won with rules	42.6%	34.0%	52.4 %	52.0%

Table 2: Performance of various Mio settings used to learn rules about safe tiles (AL = Active Learning, GS = Greedy Search, IDA* = Iterative Deepening A*, M = Macros)

Rules about Safe Tiles
<pre> %% RULE S-1 %% safe(TILEUK, BOARD):- zoneOfInterest(TILEUK, BOARD, ZONE), setHasXMines(TILEK, BOARD, ZONE, SET), inSet(TILEUK, SET), minesInSet(SET, 0). %% RULE S-2 %% safe(TILEUK, BOARD):- zoneOfInterest(TILEUK, BOARD, ZONE), setHasXMines(TILEK0, BOARD, ZONE, SET0), setHasXMines(TILEK1, BOARD, ZONE, SET1), diffSetHasXMines(SET1, SET0, SET3), inSet(TILEUK, SET3), minesInSet(SET3, 0). %% RULE S-3 %% safe(TILEUK, BOARD):- zoneOfInterest(TILEUK, BOARD, ZONE), setHasXMines(TILEK0, BOARD, ZONE, SET0), setHasXMines(TILEK1, BOARD, ZONE, SET1), setHasXMines(TILEK2, BOARD, ZONE, SET2), diffSetHasXMines(SET1, SET0, SET3), diffSetHasXMines(SET2, SET3, SET4), inSet(TILEUK, SET4), minesInSet(SET4, 0). </pre>
Rules about Mines
<pre> %% RULE M-1 %% mine(TILEUK, BOARD):- zoneOfInterest(TILEUK, BOARD, ZONE), setHasXMines(TILEK, BOARD, ZONE, SET), inSet(TILEUK, SET), minesInSet(SET, INT), lengthSet(SET, INT). </pre>

Table 3: Minesweeper rules learned by Mio

Let us analyze the performance of the best rule set (see Table 3). In 1000 games, the player made 15481 moves from which 2762 were random guesses, 169 used Rule S-1, 10285 used Rule S-2, 54 used Rule S-3, and 2211 used Rule M-1.

In addition, we examined the effect of adding probabilistic reasoning. In the experiment, we instructed the player using the rules shown in Table 3 to select a tile which minimizes the probability $P(T_u)$ that an undetermined tile T_u is a mine when none of the rules can be applied. $P(T_u)$ is equal to $\max_{T_d} \left(\frac{f_m(T_d)}{f_n(T_d)} \right)$ where T_d is a determined neighbour of T_u , $f_m(T_d)$ returns the number of mines needed by T_d and $f_n(T_d)$ returns the number of blank neighbours of T_d . Every time the player has to guess, it selects the tile which minimizes $P(T_u)$. This player wins 579 of 1000 random games.

5 Related Work

5.1 Minesweeper Playing Programs

There are several Minesweeper programs available on the web. These programs are not learning programs but playing programs where the authors have embedded their own game playing strategy. Among these programs, John D. Ramsdell's PGMS is quite successful winning 60% of 10000 random games in a 8×8 board with 10 mines.

PGMS plays using the *Equation Strategy* based on finding approximate solutions to derived integer linear equations, and probabilities. As mentioned by Ramsdell [2002], PGMS represents the information available on the board as a set of integer linear equations. Associated with an undetermined tile is a variable x that has the value 1 if the tile hides a mine, or 0 otherwise. An equation is generated for each uncovered tile with an adjacent undetermined tile. Each equation has the form $c = \sum_{i \in S} x_i$, where S is a set of undetermined tiles, and c is the number of mines hidden among S . To simplify notation, this equation is written as $c \doteq S$. Since the total number of hidden mines is known, an additional equation simply equates this number with the sum of all of the undetermined tiles.

Every time a tile t is determined safe or a mine, the board changes are propagated to all the equations containing t and a new equation for the undetermined neighbours of t is added. In addition, if $c_0 \doteq S_0$ and $c_1 \doteq S_1$ are two equations such that S_0 is a proper subset of S_1 , the equation $c_1 - c_0 \doteq S_1 \setminus S_0$ is added. To determine whether a tile is safe or a mine, PGMS iteratively applies the following rules until none are applicable⁷:

⁷In these rules $|S|$ is the cardinality of S and $S_0 \setminus S_1$ is the difference between S_0 and S_1 .

- If $0 \doteq S$, all tiles in S are safe.
- If $c \doteq S$ and $c = |S|$, all tiles in S are a mine.
- Let $c_0 \doteq S_0$ and $c_1 \doteq S_1$ be two equations and t_u be an undetermined tile such that $c_0 < c_1$, and $t_u \in S_0$ and $t_u \in S_1$. If $c_1 - c_0 = |S_1 \setminus S_0|$, all the tiles in $S_1 \setminus S_0$ are a mine and all the tiles in $S_0 \setminus S_1$ are safe.

PGMS must guess when presented with a board to which none of the rules apply. For each tile t it computes the value $P(t)$ as follows. Given an equation $c \doteq S$, define its single equation probability to be $c/|S|$. $P(t)$ is equal to $\max_{t \in S} (c/|S|)$. PGMS picks the tile t that minimizes $P(t)$. A random choice is made when there is more than one tile that minimizes $P(t)$.

We were surprised to notice that although Mio was only given general background knowledge about Minesweeper, the rules it learned are similar to the rules programmed in PGMS. For example, Mio's Rule S-1 and Rule S-2 correspond to the first and third rule in PGMS, respectively; and Mio's Rule M-1 is similar to PGMS second rule. To compare PGMS performance with the performance of Mio's best playing strategy, we let our best player (i.e., the player using the rules in Table 3 and probabilities) play 10000 random games in a 8×8 board with 10 mines. Its winning rate is also 60%.

5.2 Multirelational Learning for Games

Other work has been done which applies ILP systems to learn heuristics or playing strategies for games. Ramon et al. [2001] used Tilde [Blockeel and Raedt, 1998] to learn a theory that predicts the value of a move in Go. Morales [1996] applied the system PAL to learn chess patterns for constructing chess playing strategies. Nakano et al. [1998] presented an approach to generate an evaluation function for Shogi mating problems using ILP.

6 Conclusions and Future Work

In this paper we described how the use of new ILP techniques such as macros, greedy search with macros, and active inductive learning allow Mio to learn a Minesweeper playing strategy. This learning task proved itself to be a challenging testbed for general purpose multirelational learning systems.

The best rules obtained by Mio win 52% of the games in a 8×8 board with 10 mines, while on average a non-expert human player wins 35% of the games. The performance of the playing program using these rules as playing strategy improves to 60% when adding the use of probabilities.

By examining the games played using Mio's rules, we notice that there are still situations where the player guesses without need (i.e., a sure move can be deduced). As future work, we want to use other ILP systems (e.g., Tilde), and other machine learning approaches to learn Minesweeper playing strategies and compare their performance.

Acknowledgments

We thank all the persons who participated as guinea pigs playing Minesweeper, the anonymous reviewers for their comments, and Oscar Meruvia for proofreading. The second author was partially supported by DFG (German Science Foundation), projects WR40/1-3 and WR40/2-1.

References

- [Blockeel and Raedt, 1998] Hendrik Blockeel and Luc De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.
- [Cohn et al., 1994] David Cohn, Les Atlas, and Richard Lader. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.
- [Fürnkranz, 1998] Johannes Fürnkranz. Integrative windowing. *Journal of Artificial Intelligence Research*, 8:129–164, 1998.
- [Kaye, 2000] Richard Kaye. Minesweeper is NP-complete. *The Mathematical Intelligencer*, 22(2):9–15, Spring 2000.
- [Korf, 1985] Richard E. Korf. Iterative-deepening A*: An optimal admissible tree search. In *Proc. of the 9th IJCAI*, pages 1034–1036, 1985.
- [Morales, 1996] Eduardo Morales. Learning playing strategies in chess. *Computational Intelligence*, 12(1):65–87, 1996.
- [Muggleton and Firth, 2001] Stephen Muggleton and John Firth. Relational rule induction with CProlog4.4: a tutorial introduction. In *Relational Data Mining*, pages 160–187, 2001.
- [Muggleton, 1995] Stephen Muggleton. Inverse entailment and Prolog. *New Generation Computing Journal*, 13:245–286, 1995.
- [Nakano et al., 1998] Tomofumi Nakano, Nobuhiro Inuzuka, Hirohisa Seki, and Hidenori Itoh. Inducing Shogi heuristics using inductive logic programming. In *Proc. of the 8th Int. Conf. on ILP*, pages 155–164, 1998.
- [Peña Castillo and Wrobel, 2002a] Lourdes Peña Castillo and Stefan Wrobel. Macro-operators in multirelational learning: a search-space reduction technique. In *Proc. of ECML'2002*, pages 357–368, 2002.
- [Peña Castillo and Wrobel, 2002b] Lourdes Peña Castillo and Stefan Wrobel. On the stability of example-driven learning systems: a case study in multirelational learning. In *Proc. of MICAI'2002*, pages 321–330, 2002.
- [Quinlan and Cameron-Jones, 1995] J. Ross Quinlan and R. Michael Cameron-Jones. Induction of logic programs: FOIL and related systems. *New Generation Computing, Special issue on ILP*, 13(3-4):287–312, 1995.
- [Ramon et al., 2001] Jan Ramon, Tom Francis, and Hendrik Blockeel. Learning a tsume-go heuristic with TILDE. In *Proc. of the 2nd Int. Conf. Computers and Games*, pages 151–169, 2001.
- [Ramsdell, 2002] John D. Ramsdell, November 25, 2002. Personal communication.
- [Tesauro, 1995] Gerald Tesauro. Temporal-difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

LEARNING

KERNEL METHODS

Multi-prototype Support Vector Machine

Fabio Aioli

Dept. of Computer Science
University of Pisa, Italy
aioli@di.unipi.it

Alessandro Sperduti

Dept. of Pure and Applied Mathematics
University of Padova, Italy
sperduti@math.unipd.it

Abstract

We extend multiclass SVM to multiple prototypes per class. For this framework, we give a compact constrained quadratic problem and we suggest an efficient algorithm for its optimization that guarantees a local minimum of the objective function. An annealed process is also proposed that helps to escape from local minima. Finally, we report experiments where the performance obtained using linear models is almost comparable to that obtained by state-of-art kernel-based methods but with a significant reduction (of one or two orders) in response time.

1 Introduction

Automatic multiclass classification, i.e. the process to automatically assign exactly one from a prefixed set of labels to a stream of input instances, is a central task for many real world problems like speech recognition, OCR, text categorization etc. Many supervised learning methods have been studied that help on these tasks. Recently, kernel-based methods, like SVM, have been well studied especially for binary settings, and they yielded state-of-art performance in many different tasks. SVM searches for a high margin linear discriminant model in a very high dimensional space (feature space) where examples are implicitly mapped via a function $\Phi(x)$. Since kernel-based algorithms need only of dot products in feature space, it is possible to resort to the so called kernel trick if these dot products can be computed efficiently with a kernel function $K(x, y) = \Phi(x) \cdot \Phi(y)$. When the number of examples (or support vectors) is large, these approaches tend to be less efficient w.r.t. the time spent for classifying new vectors since they require to work with the implicit characterization of the discriminant model.

One of the most effective SVM extension able to deal with general multiclass problems has been recently provided [Crammer and Singer, 2000] and it has shown very good results. In this paper, we extend the multiclass SVM to the multi-prototype setting where for each class there can be more than one prototype vector. This allowed us to get very expressive decision functions by using simple models without necessarily requiring the use of kernels.

1.1 Preliminaries and notation

Let $S = \{(x_1, c_1), \dots, (x_n, c_n)\}$ be a set of n training examples, $x_i \in X \subseteq \mathbb{R}^d$ and $c_i \in C = \{1, \dots, m\}$. A single-label multi-class classifier is a function $h : X \rightarrow C$ that maps instances $x \in X$ to labels $c \in C$. We focus on the class of *winner-take-all* linear classifiers with form:

$$h_M(x) = \mathcal{C} \left(\arg \max_{r \in R} M_r \cdot x \right) \quad (1)$$

where M_r is the r -th prototype vector, R is the set of prototype indexes and $\mathcal{C}(r)$ the function returning the class associated to the prototype indexed r . Given a classifier $h_M(x)$ and a training example (x_i, c_i) we will say that $h_M(x)$ misclassifies x_i if $h_M(x_i) \neq c_i$.

In the following, we denote by y_i^r the constant that is equal to 1 if $\mathcal{C}(r) = c_i$ and -1 otherwise. For a given example x_i , we denote by $P_i = \{r \in R : y_i^r = 1\}$ the set of "positive" prototypes indexes and by $N_i = \{r \in R : y_i^r = -1\}$ the set of "negative" prototypes indexes. Finally, the real value $s_r(x) = M_r \cdot x$ is referred to as *similarity score* (or simply *score*) of the r -th prototype on the instance x .

2 Single-prototype multi-class SVM

An effective multi-class extension to SVM has been already proposed in [Crammer and Singer, 2000]. The resulting classifier is of the same form of (1) where each class has associated exactly one prototype, i.e. $R \equiv C$ and $\mathcal{C}(r) = r$. The solution is obtained through the minimization of a convex quadratic constrained problem. In this section, we present a simpler derivation of an equivalent formulation that will be extended in the next section to the multi-prototype framework.

In the multiclass setting, in order to have a correct classification of the instance x_i , the prototype of the correct class should have a score that is greater than the maximum among the scores of the prototypes associated to incorrect classes. We can formally write the constraints for a correct classification of the example x_i with a margin greater or equal to 1 by requiring:

$$M_{y_i} \cdot x_i \geq \theta_i + 1 \text{ and } \theta_i = \max_{r \neq y_i} M_r \cdot x_i,$$

where $y_i \equiv c_i$ is the index of the prototype vector associated to the correct class for the example x_i . To allow for margin

violations, for each example, we introduce soft margin slack variables $\xi_i \geq 0$, s.t.

$$\xi_i = [\theta_i + 1 - M_{y_i} \cdot x_i]_+,$$

where $[x]_+$ denotes the hinge loss equal to x if $x > 0$ and 0 otherwise. Notice that the value ξ_i is an upper bound to the $0 - 1$ loss for the example x_i , and consequently its average value over the training set will be an upper bound for the empirical error.

Motivated by the structural risk minimization (SRM) principle in [Vapnik, 1998], we want to search for a set of prototype vectors $M = \{M_1, \dots, M_{|C|}\}$ with small norm such to minimize the empirical error on the training set. For this, we can formulate the problem in a SVM-style by requiring a set of prototypes of minimal norm that fulfill the soft constraints given by the classification requirements. Thus, the single-prototype multiclass SVM (SProtSVM) will result in:

$$\begin{aligned} \min_{M, \xi, \theta} \quad & \frac{1}{2} \|M\|^2 + \gamma \sum_i \xi_i \\ \text{subject to:} \quad & \begin{cases} \forall i, r \neq y_i, M_r \cdot x_i \leq \theta_i, \\ \forall i, M_{y_i} \cdot x_i \geq \theta_i + 1 - \xi_i, \\ \forall i, \xi_i \geq 0 \end{cases} \end{aligned} \quad (2)$$

Notice that, as desired, the optimal solution for θ_i will be the maximum value among the negative scores for the instance x_i . This problem is convex and it can be solved in the standard way by resorting to the optimization of the Wolfe dual problem. In this case, the lagrangian is:

$$\begin{aligned} L(M, \xi, \theta, \alpha, \lambda) &= \frac{1}{2} \|M\|^2 + \gamma \sum_i \xi_i + \\ &\quad \sum_{i, r \neq y_i} \alpha_i^r (M_r \cdot x_i - \theta_i) + \\ &\quad \sum_i \alpha_i^{y_i} (\theta_i + 1 - \xi_i - M_{y_i} \cdot x_i) - \\ &\quad \sum_i \lambda_i \xi_i \\ &= \frac{1}{2} \|M\|^2 - \sum_{i, r} y_i^r \alpha_i^r (M_r \cdot x_i - \theta_i) + \\ &\quad \sum_i \alpha_i^{y_i} + \sum_i (\gamma - \alpha_i^{y_i} - \lambda_i) \xi_i, \end{aligned} \quad (3)$$

subject to the constraints $\alpha_i^r, \lambda_i \geq 0$. By differentiating the lagrangian with respect to the primal variables and imposing the optimality conditions we obtain the set of constraints (KKT conditions) that the variables must fulfill in order to be an optimal solution:

$$\begin{aligned} \frac{\partial L(M, \xi, \theta, \alpha, \lambda)}{\partial M_r} = 0 &\Leftrightarrow M_r = \sum_i y_i^r \alpha_i^r x_i \\ \frac{\partial L(M, \xi, \theta, \alpha, \lambda)}{\partial \theta_i} = 0 &\Leftrightarrow \gamma - \alpha_i^{y_i} - \lambda_i = 0 \Leftrightarrow \alpha_i^{y_i} \leq \gamma \\ \frac{\partial L(M, \xi, \theta, \alpha, \lambda)}{\partial \xi_i} = 0 &\Leftrightarrow \alpha_i^{y_i} = \sum_{r \neq y_i} \alpha_i^r \end{aligned} \quad (4)$$

By using the fact $\alpha_i^{y_i} = \frac{1}{2} \sum_r \alpha_i^r$ and substituting conditions (4) in (3) and omitting constants that do not change the solution, the problem can be restated as:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i, r} \alpha_i^r - \|M(\alpha)\|^2 \\ \text{subject to:} \quad & \begin{cases} \forall i, r, \alpha_i^r \geq 0 \\ \forall i, \alpha_i^{y_i} = \sum_{r \neq y_i} \alpha_i^r \leq \gamma \end{cases} \end{aligned} \quad (5)$$

The next section includes an efficient optimization procedure for the more general multi-prototype setting that includes the single-prototype case as a particular instance.

3 Multi-prototype multi-class SVM

In this section, the SProtSVM model previously defined will be extended to the case of multiple prototypes per class. The basic idea is that, given multiple prototype vectors, we have a correct classification iff *at least* one of the prototypes associated to the correct class has a score higher than the maximum of the scores of the prototypes associated to incorrect classes.

In this case, we write the constraints for a correct classification of the example x_i with a margin greater or equal to 1 requiring:

$$\exists r \in P_i : M_r \cdot x_i \geq \theta_i + 1 \text{ and } \theta_i = \max_{r \in N_i} M_r \cdot x_i.$$

To allow for margin violations, for each example x_i , we introduce soft margin slack variables $\xi_i^r \geq 0$, one for each positive prototype, s.t.

$$\forall r \in P_i, \xi_i^r = [\theta_i + 1 - M_r \cdot x_i]_+.$$

Given a pattern x_i , we arrange the soft margin slack variables ξ_i^r in a vector $\xi_i \in \mathbb{R}^{|P_i|}$. Let us now introduce, for each example x_i , a new vector $\pi_i \in \{0, 1\}^{|P_i|}$, whose components are all zero except one component that is 1. In the following, we refer to π_i as the assignment of the pattern x_i . Notice that the dot product $\pi_i \cdot \xi_i$ is always an upper bound to the $0 - 1$ loss for the example x_i independently from its assignment.

Now, we are ready to formulate the multi-prototype problem by requiring a set of prototypes of small norm and the best assignment for the examples able to fulfill the soft constraints given by the classification requirements.

Thus, the MProtSVM formulation will result in:

$$\begin{aligned} \min_{M, \xi, \theta, \pi} \quad & \frac{1}{2} \|M\|^2 + \gamma \sum_i \pi_i \cdot \xi_i \\ \text{subject to:} \quad & \begin{cases} \forall i, r \in N_i, M_r \cdot x_i \leq \theta_i, \\ \forall i, r \in P_i, M_r \cdot x_i \geq \theta_i + 1 - \xi_i^r, \\ \forall i, r \in P_i, \xi_i^r \geq 0 \end{cases} \end{aligned} \quad (6)$$

Unfortunately, this is a mixed integer problem that is not convex and it is a difficult problem in general but, as we will see in the following, it is prone to an efficient optimization procedure that approximates the global optimum. At this point, it is worth noticing that, since the effective formulation is itself an (heuristic) approximation to the structural risk minimization principle, a good solution, although not optimal, can anyway give good results. As we see after, this claim is confirmed by the results obtained in our experimental work.

Let suppose now that the assignments are kept fixed. In this case the reduced problem becomes convex and can be solved as above by resorting to the optimization of the Wolfe dual problem. In this case, the lagrangian is:

$$\begin{aligned} L_{\pi}(M, \xi, \theta, \alpha, \lambda) &= \frac{1}{2} \|M\|^2 + \gamma \sum_i \pi_i \cdot \xi_i + \\ &\quad \sum_{i, r \in P_i} \alpha_i^r (\theta_i + 1 - \xi_i^r - M_r \cdot x_i) - \\ &\quad \sum_{i, r \in P_i} \lambda_i^r \xi_i^r + \\ &\quad \sum_{i, r \in N_i} \alpha_i^r (M_r \cdot x_i - \theta_i), \end{aligned} \quad (7)$$

subject to the constraints $\alpha_i^r, \lambda_i^r \geq 0$. As above, by differentiating the lagrangian of the reduced problem and imposing

the optimality conditions, we obtain:

$$\begin{aligned} \frac{\partial L_\pi(M, \xi, \theta, \alpha, \lambda)}{\partial M_r} = 0 &\Leftrightarrow M_r = \sum_i y_i^r \alpha_i^r x_i \\ \frac{\partial L_\pi(M, \xi, \theta, \alpha, \lambda)}{\partial \xi_i^r} = 0 &\Leftrightarrow \gamma \pi_i^r - \alpha_i^r - \lambda_i^r = 0 \Leftrightarrow \alpha_i^r \leq \gamma \pi_i^r \\ \frac{\partial L_\pi(M, \xi, \theta, \alpha, \lambda)}{\partial \theta_i} = 0 &\Leftrightarrow \sum_{r \in P_i} \alpha_i^r = \sum_{r \in N_i} \alpha_i^r \end{aligned} \quad (8)$$

Notice that the second condition requires the dual variables associated to positive prototypes and not assigned to the associated pattern to be 0. By denoting now as y_i the unique index $r \in P_i$ such that $\pi_i^r = 1$, once using the conditions (8) in (7) and omitting constants that do not change the obtained solution, the reduced problem can be restated as:

$$\begin{aligned} \max_{\alpha} \sum_{i,r} \alpha_i^r - \|M(\alpha)\|^2 \\ \text{subject to: } \begin{cases} \forall i, r, \alpha_i^r \geq 0 \\ \forall i, \alpha_i^{y_i} = \sum_{r \in N_i} \alpha_i^r \leq \gamma \\ \forall i, r \in P_i \setminus \{y_i\}, \alpha_i^r = 0 \end{cases} \end{aligned} \quad (9)$$

It can be trivially shown that this formulation is consistent with the formulation for SPotSVM's given above.

3.1 Optimization with static assignments

When patterns are statically assigned to the prototypes via constant vectors π_i , the convexity of the associated MProtSVM problem implies that the optimal solution for the primal problem in (6) can be found through the maximization of the lagrangian in (9).

Assuming an equal number q of prototypes per class, the dual involves $n \times m \times q$ variables which lead to a very large scale problem. Anyway, the independence of constraints among the different patterns allows for the separation of the variables in n disjoint sets of $m \times q$ variables.

The algorithm we propose for the optimization of the problem in (9) is inspired by the ones already presented in [Cramer and Singer, 2000; Aiolfi and Sperduti, 2002a] and consists in iteratively selecting patterns from the training set and optimizing with respect to the variables associated to that pattern. From the convexity, each iteration leads to an increase of the lagrangian until no more improvement is possible since the optimal solution for the lagrangian has been found.

After selecting a pattern x_i , to enforce the constraint $\sum_{r \in N_i} \alpha_i^r + \lambda_i = \gamma$, $\lambda_i \geq 0$, two elements from the set of variables $\{\alpha_i^r, r \in N_i\} \cup \{\lambda_i\}$ will be optimized in pair while keeping the solution inside the feasible region. In particular, let ω_1 and ω_2 be the two selected variables, we restrict the updates to the form $\omega_1 \leftarrow \omega_1 + \rho$ and $\omega_2 \leftarrow \omega_2 - \rho$ with optimal choices for ρ . Iterating the application of this basic step over different pairs and then doing the same for different patterns will guarantee to reach the optimum for the overall problem.

Let now compute the optimal value for ρ . First of all, we observe that the update will affect the squared norm of the prototype vector M_r of an amount

$$\Delta \|M_r\|^2 = \|\Delta M_r\|^2 + 2M_r \cdot \Delta M_r.$$

Now, we show how to analytically solve the associated problem with respect to an update involving a single variable α_p^r , $r \in N_i$ and the variable λ_p . Since λ_p does not influence

the value of the objective function, it is possible to solve the associated problem with respect to the variable α_p^r and $\alpha_p^{y_p}$ to keep the constraint $\alpha_p^{y_p} = \sum_{r \in N_p} \alpha_p^r$ satisfied and then to enforce the constraints $\lambda_p = \gamma - \sum_{s \in N_i} \alpha_p^s \geq 0$. Thus, in this case we have:

$$\alpha_p^r \leftarrow \alpha_p^r + \rho \text{ and } \alpha_p^{y_p} \leftarrow \alpha_p^{y_p} + \rho.$$

Since $\Delta M_r = -\rho x_p$, $\Delta M_{y_i} = \rho x_p$ and $\Delta M_s = 0$ for $s \notin \{r, y_i\}$, we obtain $\Delta \|M\|^2 = \Delta \|M_r\|^2 + \Delta \|M_{y_i}\|^2 = 2\rho^2 \|x_p\|^2 + 2\rho(s_{y_p}(x_p) - s_r(x_p))$ and $\Delta L(\alpha) = 2\rho(1 - s_{y_p}(x_p) + s_r(x_p) - \rho \|x_p\|^2)$. Since this last formula is concave on ρ , it is possible to find the optimal value when the first derivative is null, i.e.

$$\hat{\rho} = \arg \max_{\rho} \Delta L(\alpha) = \frac{1 - s_{y_p}(x_p) + s_r(x_p)}{2\|x_p\|^2}$$

If the values of α_p^r and $\alpha_p^{y_p}$, after being updated, would turn out to be not feasible for the constraints $\alpha_p^r \geq 0$ and $\alpha_p^{y_p} \leq \gamma$, we select the value for ρ such to fulfill the violated constraint bounds at the limit.

Now, we show how to analytically solve the associated problem with respect to an update involving a pair of variables $\alpha_p^{r_1}, \alpha_p^{r_2}$ such that $r_1, r_2 \in N_i$ and $r_1 \neq r_2$. Since, in this case, the update must have zero sum, we have:

$$\alpha_p^{r_1} \leftarrow \alpha_p^{r_1} + \rho \text{ and } \alpha_p^{r_2} \leftarrow \alpha_p^{r_2} - \rho$$

In this case, $\Delta M_{r_1} = -\rho x_p$, $\Delta M_{r_2} = \rho x_p$ and $\Delta M_s = 0$ for $s \notin \{r_1, r_2\}$, we obtain $\Delta \|M\|^2 = \Delta \|M_{r_1}\|^2 + \Delta \|M_{r_2}\|^2 = 2\rho^2 \|x_p\|^2 + 2\rho(s_{r_2}(x_p) - s_{r_1}(x_p))$ and $\Delta L(\alpha) = 2\rho(s_{r_1}(x_p) - s_{r_2}(x_p) - \rho \|x_p\|^2)$. Since also this last formula is concave on ρ , it is possible to find the optimal value

$$\hat{\rho} = \arg \max_{\rho} \Delta L(\alpha) = \frac{s_{r_1}(x_p) - s_{r_2}(x_p)}{2\|x_p\|^2}$$

Similarly to the previous case, if the values of the $\alpha_p^{r_1}$ and $\alpha_p^{r_2}$, after being updated, would turn out to be not feasible for the constraints $\alpha_p^{r_1} \geq 0$ and $\alpha_p^{r_2} \geq 0$ we select the value for ρ such to fulfill the violated constraint bounds at the limit.

Iterating multiple times the basic step described above on pairs of variables chosen among that associated to a given pattern it is guaranteed to find the optimality condition for the pattern. This has been exploited in [Aiolfi and Sperduti, 2002a] to devise an incremental algorithm that uses the method on the reduced problem of a single example and then iterates over different examples. This optimization procedure can be considered incremental in the sense that the solution previously found for a given pattern forms the initial condition when the pattern is selected again for optimization. In this version, the basic step of optimization of the reduced problem can require the optimization over all the $((m-1) \times q + 1)((m-1) \times q)/2$ pairs of variables not constrained to 0 associated with the selected pattern. Thus the complexity of the optimization of the reduced problem is $O(I \times (m \times q)^2)$ where I is the number of iterations.

Now, we present a further step by giving an algorithm that at each iteration has a complexity $O(m \times q)$. For this, we observe that for the variables $\{\alpha_p^r, \lambda_p\}$ associated to the pattern

x_p to be optimal, the feasible analytic solution ρ must be 0 for each pair.

In particular, for the first case above, in order to be able to apply the step, it is required that, one of the following two conditions are verified, i.e.:

$$\begin{aligned} & (s_{y_i}(x_p) < \max_{r \in N_p} s_r(x_p) + 1) \wedge (\alpha_p^{y_i} < \gamma) \\ & \vee \\ & (s_{y_i}(x_p) > \max_{r \in N_p, \alpha_p^r > 0} s_r(x_p) + 1) \wedge (\alpha_p^{y_i} > 0) \end{aligned}$$

while for the second case we must have:

$$\max_{r \in N_p} s_r(x_p) > \min_{r \in N_p, \alpha_p^r > 0} s_r(x_p).$$

Notice that these facts can be checked in linear time. It is easy to show that the solution obtained at a certain step can be improved iff one of these facts are verified. This suggests an efficient procedure, shown in Figure 1(Top), that tries to greedily fulfill the previous condition of optimality.

3.2 Optimization of general MProtSVM

In this section, we present an efficient procedure that guarantees to reach a local minimum of the objective function of the problem in (6) associated to MProtSVM. This procedure will try to optimize also with respect to the assignments π_i .

Let suppose to start with a given assignment $\pi(1)$ for the patterns. In this case, as we have already seen, the associated problem is convex and can be efficiently solved by using the algorithm given in Section 3.1. Once that the optimal value for the primal $P_{\pi(1)}^*$ has been reached, we observe that the solution can be further improved by updating the assignments in such a way to assign each pattern x_i to a positive prototype with minimal slack value, i.e. by setting the vector $\pi_i(2)$ such to have the unique 1 corresponding to the best performing positive prototype. However, with this new assignment $\pi(2)$, the variables α might not fulfill the second KKT condition in eq. (8) anymore. If it is the case, it simply means that the current solution α is not optimal for the new assignment. Thus, a lagrangian optimization done by satisfying constraints dictated by KKT conditions for the new assignment is guaranteed to obtain a new α with a better optimal primal value $P_{\pi(2)}^*$. For the optimization algorithm to succeed, however, the KKT conditions on the α must be restored in order to return back to a feasible solution and then finally resuming the lagrangian optimization with the new assignment $\pi(2)$. We restore the KKT conditions by setting $\alpha_i = 0$ whenever there exists $r \in P_i$ such that $\alpha_i^r > 0 \wedge \pi_i^r = 0$.

Performing the same procedure over different assignments, each one obtained from the previous one by the procedure just mentioned, implies the convergence of the algorithm to a local solution for the primal problem when no improvements are possible and the KKT conditions are all fulfilled by the current solution. This is due to the fact that every step induces an improvement on the primal value.

The first problem with this procedure is that it results onerous when dealing with many prototypes since we must perform many lagrangian optimizations. For this, we observe that for the procedure to work, at each step, it is sufficient to stop the optimization of the lagrangian when we find a value

```

OptimizeOnPattern( $x_p, \epsilon$ )
 $\forall r, s_r := s_r(x_p), K_{pp} = ||x_p||^2;$ 
do
  if ( $\alpha_p^{y_p} = 0$ ) then {
     $r_1 := \arg \max_{r \in N_p} s_r; \rho_1 := \min\{\frac{1-s_{y_p}+s_{r_1}}{2K_{pp}}, \gamma\};$ 
     $V_1 := 2\rho_1(1-s_{y_p}+s_{r_1}-\rho_1 K_{pp}); k := 1\}$ 
  else {
     $r_1 := \arg \max_{r \in N_p} s_r;$ 
     $r_2 := \arg \max_{r \in N_p, \alpha_p^r > 0} s_r;$ 
     $r_3 := \arg \min_{r \in N_p, \alpha_p^r > 0} s_r;$ 
     $\rho_1 := \min\{\frac{1-s_{y_p}+s_{r_1}}{2K(x_p, x_p)}, \gamma - \alpha_p^{y_p}\};$ 
     $\rho_2 := \max\{\frac{1-s_{y_p}+s_{r_2}}{2K(x_p, x_p)}, -\alpha_p^{r_2}\};$ 
     $\rho_3 := \min\{\frac{s_{r_1}-s_{r_3}}{2K(x_p, x_p)}, \alpha_p^{r_3}\};$ 
     $V_1 := 2\rho_1(1-s_{y_p}+s_{r_1}-\rho_1 K_{pp});$ 
     $V_2 := 2\rho_2(1-s_{y_p}+s_{r_2}-\rho_2 K_{pp});$ 
     $V_3 := 2\rho_3(s_{r_1}-s_{r_3}-\rho_3 K_{pp});$ 
     $k := \arg \max_j V_j \};$ 
  case  $k$  of {
    1:  $\alpha_p^{y_p} := \alpha_p^{y_p} + \rho_1; \alpha_p^{r_1} := \alpha_p^{r_1} + \rho_1;$ 
        $s_{y_p} := s_{y_p} + \rho_1 K_{pp}; s_{r_1} := s_{r_1} - \rho_1 K_{pp};$ 
    2:  $\alpha_p^{y_p} := \alpha_p^{y_p} + \rho_2; \alpha_p^{r_2} := \alpha_p^{r_2} + \rho_2;$ 
        $s_{y_p} := s_{y_p} + \rho_2 K_{pp}; s_{r_2} := s_{r_2} - \rho_2 K_{pp};$ 
    3:  $\alpha_p^{r_1} := \alpha_p^{r_1} + \rho_3; \alpha_p^{r_3} := \alpha_p^{r_3} - \rho_3;$ 
        $s_{r_1} := s_{r_1} - \rho_3 K_{pp}; s_{r_3} := s_{r_3} + \rho_3 K_{pp}\}$ 
  until ( $V_k \leq \epsilon$ );
return  $\{\alpha\}$ 

```

```

AnnealedMProtSVM()
 $T := T_0$ ; randomly initialize  $\pi(1)$ ;
compute the primal  $E(1) := P_{\pi(1)}(0)$ ;
for  $t = 1, \dots, t_{max}$ 
  do for all the examples  $x_p \in S$ 
     $\alpha_p = \text{OptimizeOnPattern}(x_p, \epsilon);$ 
  until  $P_{\pi(t)}(\alpha) < E(t);$ 
  compute a new assignment  $\pi(t+1)$  using  $T(t, T_0)$ ;
  compute the new primal  $E(t+1) := P_{\pi(t+1)}(\alpha);$ 
  restore KKT conditions on  $\alpha$ ; /*see Section 3.2*/
end;

```

Figure 1: *Top*: algorithm for the optimization of the variables associated with a given pattern x_p and a tolerance ϵ . *Bottom*: algorithm for the optimization of MProtSVM.

for the primal better than the last found and this is going to happen for sure since the last solution has been found not optimal. This requires only a periodic check of the primal value when optimizing the lagrangian.

The second and more stringent problem is that the procedure will lead to a local minimum that can be very far from the best possible. For this, we suggest to update the assignments on the basis of a stochastic annealed function instead of the hard decision function described above.

Specifically, let us view the value of the primal as an energy function

$$E(\pi) = \frac{1}{2} \|M\|^2 + \gamma \sum_{i \in S} \pi_i \cdot \xi_i.$$

Let suppose to have a pattern x_i having slack variables ξ_i^r , $r \in P_i$ and suppose that the probability for the assignment to be in the state s (i.e. with the s -th component set to 1) follows the law

$$p_i(s) \propto \exp(-\Delta E_s / T)$$

where T is the temperature of the system and $\Delta E_s = \gamma(\xi_i^s - \xi_i^{y_i})$ the variation of the system energy when the pattern x_i is assigned to the s -th prototype. By multiplying every term $p_i(r)$ by the normalization term $\exp(\gamma(\xi_i^{y_i} - \xi_i^0))$ where $\xi_i^0 = \min_{r \in P_i} \xi_i^r$ and considering that probabilities over alternative states must sum to one, i.e. $\sum_{r \in P_i} p_i(r) = 1$, we obtain

$$p_i(s) = \exp(-\frac{\gamma(\xi_i^s - \xi_i^0)}{T}) / Z$$

with $Z = \sum_{r \in P_i} \exp(-\gamma(\xi_i^r - \xi_i^0)/T)$ the partition function.

Thus, when assigning the pattern x_i , each positive prototype s will be selected with probability $p_i(s)$. Notice that, when the temperature of the system is low, the probability for a pattern to be assigned to a prototype different from the one having minimal slack value tends to 0 and we obtain a behavior similar to the not annealed version of the algorithm. The simulated annealing is typically implemented by decreasing the temperature as the number of iterations increases by a monotonic decreasing function $T = T(t, T_0)$. The full algorithm is depicted in Figure 1 (Bottom).

4 Experimental Results

We tested our model against three datasets that we briefly describe in the following:

NIST: it consists of a 10-class task of 10705 digits randomly taken from the NIST-3 dataset. The training set consists of 5000 randomly chosen digits, while the remaining 5705 digits are used in the test set.

USPS: it consists of a 10-class OCR task (digits from 0 to 9) whose input are the pixels of a scaled digit image. There are 7291 training examples and 2007 test examples.

LETTER: it consists of a 26-class task (alphabetic letters A-Z). Inputs are measures of the printed font glyph. The first 16000 examples are used for training and the last 4000 for testing.

Even if any kernel function can be in principle used, for the following preliminary experiments, we used the linear kernel $K(x, y) = (x \cdot y + 1)$. This allowed us to write an optimized

code for training and classification that works directly with the explicit (compact) version of the model M .

Since we are primarily interested into the evaluation of MProtSVM w.r.t. SProtSVM, for each dataset, we performed validation on the parameter γ for the SProtSVM model and we re-used the obtained value for training every MProtSVM generated for the same dataset. This approach seems us anyway a pessimistic estimate of the performance of MProtSVM.

The annealing process required by MProtSVM has been implemented by decreasing the temperature of the system with the exponential law:

$$T(t, T_0) = T_0(1 - \tau)^t$$

where $0 < \tau < 1$ and $T_0 > 0$ are external parameters. We used $T_0 = 10$ for all the following experiments.

q	LVQ2.1 Error %	MProtSVM Error %
1	7.43	6.45
5	4.68	3.70
10	4.35	3.28
15	3.52	3.24

Table 1: Comparison of generalization performances between LVQ and MProtSVM increasing the number of codebooks/prototypes on the NIST dataset ($\beta = 0.2 \times q$).

q	USPS Error (%)	q	LETTER Error (%)
1	8.12	1	21.36
3	6.13	2	13.42
5	5.83	3	9.64
10	5.48	5	6.42
15	5.33	10	4.84
20	5.38	15	3.16

Table 2: (a) Test error of MProtSVM on the USPS dataset ($\gamma = 0.01$, $\tau = .05$, $\beta = 0.137 \times q$), with an increasing number of prototypes; (b) Test error of MProtSVM on the LETTER dataset ($\gamma = 10$, $\tau = 0.1$, $\beta = 0.043 \times q$), with an increasing number of prototypes.

A set of experiments have been performed to compare the generalization performance of our (linear) model versus LVQ [Kohonen *et al.*, 1996] which seemed to us the most comparable model. For this, we have reported the best results that have been obtained by LVQ on the NIST dataset. Specifically, they have been obtained with the LVQ2.1 version of the algorithm (see [Sona and Sperduti, 2000]). As it is possible to see in Table 1, MProtSVM performs significantly better when few prototypes per class are used, while the difference gets lower when the number of prototypes per class increases. This can be due to the more effective control of the margin for SVM w.r.t. LVQ models. On the same dataset, the tangent-distance based TVQ algorithm [Aiolfi and Sperduti, 2002b] has obtained the best result, a remarkable 2.1% test error, and polynomial SVM's have obtained a 2.82% test error. In

q	$\tau = 0.2$	$\tau = 0.1$	$\tau = 0.05$	$\tau = 0.03$
3	7.44626 (6.33%)	7.28049 (6.03%)	7.08138 (6.13%)	7.04274 (6.48%)
5	7.49136 (6.08%)	7.27318 (5.63%)	7.10498 (5.83%)	7.00946 (5.58%)
10	7.82233 (5.58%)	7.51780 (5.88%)	7.27596 (5.48%)	7.12517 (5.23%)
15	7.82222 (5.33%)	7.57009 (5.73%)	7.38722 (5.33%)	7.22250 (5.53%)
20	7.78410 (5.48%)	7.79388 (5.72%)	7.49125 (5.38%)	7.21303 (5.53%)

Table 3: Primal values and generalization error obtained with different configurations varying the parameter τ (USPS dataset).

addition, we tested the MProtSVM on the UCI Irvine USPS and LETTER datasets. As it is possible to see in Table 2, by combining a reasonably high number of linear prototypes, we have obtained performances almost comparable with the ones obtained in literature by using non-linear models. In fact, on the USPS dataset, we have been able to get a 4.63% error, using a SProtSVM with polynomial kernel of degree 3 and without preprocessing the data, while for LETTER, we refer to the 1.95% obtained in [Crammer and Singer, 2001] by SProtSVM with exponential kernel. Although obtained with a slightly different split of the LETTER dataset (15000 examples for training and 5000 for test), we would like to mention the results reported in [Michie *et al.*, 1994] where LVQ and k-NN yielded a 7.9% and 6.8% error, respectively.

Notice that MProtSVM returns far more compact models with respect to state of the art non-linear kernel methods, thus allowing a (one or two order) reduced response time in classification. Defining a sort of model complexity factor $\beta = 100 \times (m \times q)/n$, i.e. the number of prototypes produced as a fraction of the cardinality of the training set, the above experiments have shown very low value for β (e.g. 15×26 prototypes in the LETTER dataset gives $\beta = 0.65\%$ and 20×10 prototypes for USPS gives $\beta = 2.74\%$). Notice that β can be directly compared with the fraction of support vectors in kernel machines. Thus, MProtSVMs also give us a way to decide (before training) the complexity of the model.

Finally, in Table 3 we have reported the values of the objective function of the primal problem in (6) along with their corresponding test errors obtained with the USPS dataset using different configurations and lowering the parameter τ . As expected, fixed a row in the table, better values for the primal can be obtained with lower values of τ . Moreover, as the number of prototypes per class increases, the choice of small τ tends to be more crucial. Anyway, higher values for τ , and thus not optimal values for the primal, can nevertheless lead to good generalization performances. This can be due to the fact that the primal value is just a way to approximate the theoretical SRM principle.

5 Conclusion

We have proposed an extension of multiclass SVM able to deal with several prototypes per class. This extension defines a non-convex problem. We suggested to solve this problem by using a novel efficient optimization procedure within an annealing framework where the energy function corresponds to the primal of the problem. Experimental results on some popular benchmarks demonstrated that it is possible to reach very competitive performances by using few linear models

per class instead of a single model per class with kernel. This allows the user to get very compact models which are very fast in classifying new patterns. Thus, according to the computational constraints, the user may decide how to balance the trade-off between better accuracy and speed of classification. Finally, it should be noted that the proposed approach compares favorably versus LVQ, a learning procedure that, similarly to the proposed approach, returns a set of linear models per class.

References

- [Aioli and Sperduti, 2002a] Fabio Aioli and Alessandro Sperduti. An efficient smo-like algorithm for multiclass svm. In *Proceedings of IEEE workshop on Neural Networks for Signal Processing*, 2002.
- [Aioli and Sperduti, 2002b] Fabio Aioli and Alessandro Sperduti. A re-weighting strategy for improving margins. *Artificial Intelligence Journal*, 137/1-2:197–216, 2002.
- [Crammer and Singer, 2000] Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 35–46, 2000.
- [Crammer and Singer, 2001] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based machines. *Journal of Machine Learning Research*, 2(Dec):265–292, 2001.
- [Kohonen *et al.*, 1996] Teuvo Kohonen, Jussi Hynninen, Jari Kangas, Jorma Laaksonen, and Kari Torkkola. Lvq-pak: The learning vector quantization program package. Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, Rakentajanaukio 2 C, SF-02150 Espoo, Finland, January 1996. <http://www.cis.hut.fi/nnrc/nnrc-programs.html>.
- [Michie *et al.*, 1994] D. Michie, D. Spiegelhalter, and C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.
- [Sona and Sperduti, 2000] Diego Sona and Alessandro Sperduti. Discriminant pattern recognition using transformation-invariant neurons. *Neural Computation*, 12:1355–1370, 2000.
- [Vapnik, 1998] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Continuous nonlinear dimensionality reduction by kernel eigenmaps

Matthew Brand

Mitsubishi Electric Research Laboratories
Cambridge, MA 02460 USA

Abstract

We equate nonlinear dimensionality reduction (NLDR) to graph embedding with side information about the vertices, and derive a solution to either problem in the form of a kernel-based mixture of affine maps from the ambient space to the target space. Unlike most spectral NLDR methods, the central eigenproblem can be made relatively small, and the result is a continuous mapping defined over the entire space, not just the datapoints. A demonstration is made to visualizing the distribution of word usages (as a proxy to word meanings) in a sample of the machine learning literature.

1 Background: Graph embeddings

Consider a connected graph with weighted undirected edges specified by edge matrix $\mathbf{W}$. Let $W_{ij} = W_{ji}$ be the positive edge weight between connected vertices i and j , zero otherwise. Let $\mathbf{D} \doteq \text{diag}(\mathbf{W}\mathbf{1})$ be a diagonal matrix where $D_{ii} = \sum_j W_{ij}$, the cumulative edge weights into vertex i . The following points are well known or easily derived in spectral graph theory [Fiedler, 1975; Chung, 1997]:

1. The generalized eigenvalue decomposition (EVD)

$$\mathbf{W}\mathbf{V} = \mathbf{D}\mathbf{V}\mathbf{\Lambda} \quad (1)$$

has real eigenvectors $\mathbf{V} \doteq [\mathbf{v}_1, \dots, \mathbf{v}_N]$ and eigenvalues $\mathbf{\Lambda} \doteq \text{diag}([\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N])$.

2. Premultiplying equation (1) by $\mathbf{D}^{-1}$ makes the generalized eigenproblem into a *stochastic* eigenproblem

$$(\mathbf{D}^{-1}\mathbf{W})\mathbf{V} = \mathbf{V}\mathbf{\Lambda}, \quad (2)$$

where $\mathbf{D}^{-1}\mathbf{W}$ is a stochastic transition matrix having nonnegative rows that sum to one. The largest eigenvalue of equation (1) is therefore *stochastic* ($\lambda_1 = 1$) and its paired eigenvector is *uniform* ($\mathbf{v}_1 = \mathbf{1}/\sqrt{N}$).

3. Expanding and collecting terms in W_{ij} reveals the geometric meaning of the eigenvalues:

$$\lambda_k = 1 - \sum_{ij} (v_{ik} - v_{jk})^2 W_{ij} / 2. \quad (3)$$

The d eigenvectors paired to eigenvalues λ_2 through λ_{d+1} therefore give an embedding of the vertices in

$\mathbb{R}^d$ with minimal distortion vis-a-vis the weights, in the sense that a larger W_{ij} stipulates a shorter embedding distance. Formally, the embedding

$$\mathbf{Y}_{1:d} \doteq [\mathbf{v}_2, \dots, \mathbf{v}_{d+1}]^\top = \arg \max_{\mathbf{Y}\mathbf{D}\mathbf{Y}^\top = \mathbf{I}} \text{trace}(\mathbf{Y}\mathbf{W}\mathbf{Y}^\top) \quad (4)$$

minimizes the distortion $d - \text{trace}(\mathbf{Y}\mathbf{W}\mathbf{Y}^\top)$

$$= \sum_{k=2}^{d+1} (1 - \lambda_k) = \sum_{k=2}^{d+1} \sum_{ij} (v_{ik} - v_{jk})^2 W_{ij} / 2 \quad (5)$$

for any integer $d \in [1, N]$. The norm constraint $\mathbf{Y}\mathbf{D}\mathbf{Y}^\top = \mathbf{I}$ sets the scale of the embedding and causes vertices of high cumulative weight to be embedded nearer to the origin.

4. $\mathbf{Y}$ can be rigidly rotated in $\mathbb{R}^d$ without changing its distortion. The distortion measure is also invariant to rigid translations, but the eigenproblem is not, thus there is an unwanted degree of freedom (DOF) in the solution. Due to stochasticity, this DOF is isolated in the uniform eigenvector $\mathbf{v}_1$, which is suppressed from the embedding without error (because $1 - \lambda_1 = 0$). Adding $\mathbf{t}\mathbf{v}_1^\top$ to $\mathbf{Y}$ rigidly translates the embedding by $\mathbf{t} \in \mathbb{R}^d$.
5. Premultiplying by $\mathbf{V}^\top$ and rearranging equates equation 1 to the EVD of the graph Laplacian $\mathbf{D} - \mathbf{W}$:

$$\mathbf{V}^\top (\mathbf{D} - \mathbf{W}) \mathbf{V} = \mathbf{I} - \mathbf{\Lambda}. \quad (6)$$

6. Premultiplying by $\mathbf{D}^{-1/2}$ connects equation 1 to the (symmetric) EVD of the normalized Laplacian:

$$(\mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2}) \mathbf{V}' = \mathbf{V}' \mathbf{\Lambda} \quad (7)$$

with $\mathbf{V}' \doteq \mathbf{D}^{1/2} \mathbf{V}$.

In summary: Equation 1 gives an optimal embedding of a graph in $\mathbb{R}^d$ via eigenvectors $\mathbf{v}_2 \dots \mathbf{v}_{d+1}$; eigenvalue λ_1 is stochastic and the corresponding eigenvector $\mathbf{v}_1$ is uniform; this is an important property of the EVD solution because it isolates the problem's unwanted translational degree of freedom in a single eigenvector, leaving the remaining eigenvectors unpolluted.

Many embedding algorithms can be derived from this analysis, including the Fiedler vector [Fiedler, 1975], locally linear embeddings (LLE) [Roweis and Saul, 2000], and Laplacian eigenmaps [Belkin and Niyogi, 2002]. For example, direct solution of equation 1 gives the Laplacian eigenmap; as

a historical note, the symmetrized formulation was proposed by Fiedler in the 1970s and has been used for heuristic graph layout since the 1980s [Mohar, 1991].

2 Transformational embeddings

Now consider a more general problem: We are given some information about the vertices in a matrix $\mathbf{Z} \doteq [\mathbf{z}_1, \dots, \mathbf{z}_N] \in \mathbb{R}^{d \times N}$, whose columns are generated by applying a vector-valued function $z(\cdot) \rightarrow \mathbf{z} \in \mathbb{R}^d$ to each vertex of the graph. We seek a linear operator which transforms $\mathbf{Z}$ to the optimal graph embedding: $G(\mathbf{Z}) \rightarrow \mathbf{Y}$. We will call this the “transformational embedding,” to distinguish it from the “direct embedding” discussed above.

A natural candidate for the algebraic statement of the transformational embedding problem is the generalized EVD

$$(\mathbf{Z}\mathbf{W}\mathbf{Z}^\top)\mathbf{V} = (\mathbf{Z}\mathbf{D}\mathbf{Z}^\top)\mathbf{V}\mathbf{\Lambda}, \quad (8)$$

because setting $\mathbf{Y} = \mathbf{V}^\top \mathbf{Z}$ makes this equivalent to the original direct embedding problem. Again, there is an equivalent symmetric eigenproblem: Make Cholesky¹ decomposition $\mathbf{R}^\top \mathbf{R} \leftarrow \mathbf{Z}\mathbf{D}\mathbf{Z}^\top$ into upper-triangular $\mathbf{R} \in \mathbb{R}^{d \times d}$ and let

$$\mathbf{B} \doteq (\mathbf{R}^{-\top} \mathbf{Z}\mathbf{W}\mathbf{Z}^\top \mathbf{R}^{-1}) \in \mathbb{R}^{d \times d}. \quad (9)$$

Then

$$\mathbf{B}\mathbf{V}' = \mathbf{V}'\mathbf{\Lambda} \quad (10)$$

with

$$\mathbf{V}' \doteq \mathbf{R}\mathbf{V}, \quad \mathbf{V} = \mathbf{R}^{-1}\mathbf{V}'. \quad (11)$$

This gives an embedding $[\mathbf{v}_2, \mathbf{v}_3, \dots]^\top \mathbf{Z}$, and a computational advantage: If $\mathbf{Z} \in \mathbb{R}^{d \times N}$ is a short matrix ($d \ll N$), the original $N \times N$ eigenproblem can be reduced to a very small $d \times d$ problem, and the matrix multiplications also scale as $O(d^2 N)$ rather than $O(N^3)$, due to the sparsity of $\mathbf{W}$ and $\mathbf{D}$.

2.1 Correcting problematic eigenstructure

It is generally the case that $\mathbf{Y}^\top \notin \text{range}(\mathbf{Z}^\top)$ —there is no linear combination of the rows of $\mathbf{Z}$ giving $\mathbf{Y}$, so the desired linear mapping $G(\mathbf{Z}) \rightarrow \mathbf{Y}$ does not exist. Equations 8–11 give the *optimal least-squares approximation* $G(\mathbf{Z}) = \mathbf{V}^\top \mathbf{Z} \approx \mathbf{Y}$. This approximation can have a serious flaw: If $\mathbf{1} \notin \text{range}(\mathbf{Z}^\top)$ then the first eigenvector $\mathbf{v}_1$ is *not* uniform; it cannot be discarded as the unwanted translational DOF. Worse, all the other eigenvectors will be variously contaminated by the unwanted DOF, resulting in an embedding polluted with artifacts. For this reason, we call direct solution of equation 8 a *raw* approximation.

Our options for remedy are limited to those that modify the row-space of $\mathbf{Z}$ to reintroduce the uniform eigenvector. For reasons that will become obvious below, we will restrict ourselves to operations that can be applied to any column of $\mathbf{Z}$ without knowing any other column.

The simplest such operation is to append a uniform row to $\mathbf{Z}$, so that $\mathbf{z}_i \rightarrow [\mathbf{z}_i^\top, 1]^\top$. This makes the relation between $\mathbf{Z}$

¹Any gram-like factorization will work. For example, given EVD $\mathbf{A}\mathbf{\Omega}\mathbf{A}^\top \leftarrow \mathbf{Z}\mathbf{D}\mathbf{Z}^\top$, $\mathbf{R} = \mathbf{\Omega}^{1/2}\mathbf{A}^\top$. The Cholesky is especially attractive for its numerical stability, sparsity, and easy invertibility.

and $\mathbf{Y}$ *affine* and guarantees that $\mathbf{v}_1^\top \mathbf{Z}$ is uniform, but it can also force the eigenvectors to model additional variance that is not part of the problem.

Working backward from the desiderata that the leading column of $\mathbf{V}^\top \mathbf{Z}$ should be uniform, let $\mathbf{K} \doteq \text{diag}(\mathbf{v}_1^\top \mathbf{Z})^{-1}$ such that $\mathbf{Z}\mathbf{K}$ is a modified representation of the vertices with values of $z(\cdot)$ *reweighted* on a per-vertex basis: $\mathbf{z}_i \rightarrow \mathbf{z}_i / (\mathbf{v}_1^\top \mathbf{z}_i)$. Clearly $(\mathbf{V}^\top \mathbf{Z}\mathbf{K})^\top$ has a uniform first column, since each row is divided by its first element.

It follows immediately that the related eigenproblem

$$(\mathbf{Z}\mathbf{K}\mathbf{W}\mathbf{K}^\top)\mathbf{V}'' = (\mathbf{Z}\mathbf{K}\mathbf{D}\mathbf{K}^\top)\mathbf{V}''\mathbf{\Lambda}'' \quad (12)$$

is *stochastic*, and $\mathbf{Y}'' \doteq [\mathbf{v}_2'', \mathbf{v}_3'', \dots]^\top \mathbf{Z}\mathbf{K}$ is an embedding with the unwanted translational degree of freedom totally removed. Note that the raw and stochastic approximations are orthogonal (under metric $\mathbf{D}$): $\mathbf{Y}''^\top \mathbf{D}\mathbf{Y}''^\top$ is a diagonal matrix; the other methods are not.

It should be noted that—when scaled to have equal norm $\text{trace}(\mathbf{Y}\mathbf{D}\mathbf{Y}^\top)$ —none of these approximations has uniformly superior distortion scores; but in Monte Carlo trials with random graphs, we find a clear ordering from lowest to highest distortion: *reweighted*, *affine*, *stochastic*, *raw* (see figure 1).

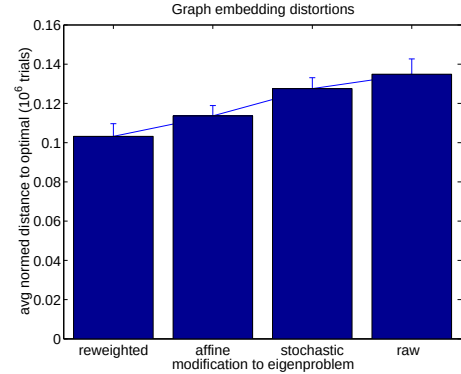


Figure 1: Comparison of methods for modifying the row-space of $\mathbf{Z}$. The graph shows distortion from the optimal embedding, averaged over 10^6 trials with 50-node matrices having random edge weights and random $\mathbf{Z} \in \mathbb{R}^{4 \times 50}$.

The raw approximation is suboptimal because information about the d -dimensional embedding is spread over $d + 1$ eigenvectors, no subset of which is optimal. The stochastic approximation is also suboptimal—it optimizes a different measure implied by equation 12. In practice, when computing embeddings of graphs whose embedding structure is known *a priori*, we find that the reweighted and stochastic approximations give results that are clearly very similar, and superior to the other approximations.

The need for *any* such correction stems from the fact that—the literatures of spectral graph theory and NLDR notwithstanding—equation 1 is *not* a completely correct statement of the embedding problem. We will show in a forthcoming paper that, as a statement of the embedding problem, equation 1 is both algebraically underconstrained and numerically ill-conditioned. In particular, point #2 is

not strictly true: *The stochastic eigenvalue is not always paired to a uniform eigenvector.* This leads to pathologies that can ruin the embedding, whether obtained from the basic or derived formulations. NLDR algorithms that can be derived from equation 1 (e.g., [Roweis and Saul, 2000; Belkin and Niyogi, 2002; Teh and Roweis, 2003]) do not mediate the problem.

A forthcoming paper makes a full analysis of these issues, identifies the correct problem statements for both equations 1 & 8, and gives closed-form optimal solutions to both problems. The approximation methods discussed in this section are still useful in that they are faster and give reasonably high-quality embeddings. For the NLDR method and datasets considered below, the result of the reweighted approximation is almost numerically indistinguishable from the optimal embedding, and requires substantially less calculation. The reweighting method can also be justified as a Padé approximation of the optimal solution.

3 Nonlinear dimensionality reduction

Let $\mathbf{X} \doteq [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^D$ be a set of points sampled from a low-dimensional manifold embedded in a high-dimensional ambient space. A reduced-dimension *embedding* $\mathbf{Y} \doteq [\mathbf{y}_1, \dots, \mathbf{y}_N] \in \mathbb{R}^d$ with $d < D \leq N$ is a set of low-dimensional points with the same local neighborhood structure. We desire instead a *mapping* $G: \mathbb{R}^D \rightarrow \mathbb{R}^d$, which will generalize the correspondence to the whole continuum, with reasonable interpolation and extrapolation to be expected in the neighborhood of the data. Spectral methods for NLDR typically require solution of many and/or very large eigenvalue or generalized eigenvalue problems [Kruskal and Wish, 1978; Kambhatla and Leen, 1997; Tenenbaum *et al.*, 2000; Roweis and Saul, 2000; Belkin and Niyogi, 2002], and with the exception of [Teh and Roweis, 2003; Brand, 2003], offer embeddings of points rather than mappings between spaces.

Here we show how to leverage the transformational embedding of section 2 into a continuous NLDR algorithm, specifically a kernel-based mixture of affine maps from the ambient space to the target space. To do so, we must show how the edge weight matrix $\mathbf{W}$ and vertex matrix $\mathbf{Z}$ are specified. Let $W_{ij} \doteq f(\mathbf{x}_i, \mathbf{x}_j)$ iff $\mathbf{x}_i$ and $\mathbf{x}_j$ satisfy some locality criterion, e.g., $\|\mathbf{x}_i - \mathbf{x}_j\| < \epsilon$, otherwise $W_{ij} = 0$. As stated above, an embedding $\mathbf{Y}$ of $\mathbf{X}$ should satisfy

$$\mathbf{Y} = \arg \min_{\mathbf{Y}} \sum_{i \neq j} \|\mathbf{y}_i - \mathbf{y}_j\|^2 W_{ij} \quad (13)$$

where larger W_{ij} penalize large distances between $\mathbf{y}_i$ and $\mathbf{y}_j$.

How should W_{ij} be computed? f is a measure of similarity: The graph-theoretic literature usually takes $f(\cdot, \cdot) = 1$, while NLDR methods typically take $f(\mathbf{x}_i, \mathbf{x}_j) \propto \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2)$ to be a Gaussian kernel, on analogy to heat diffusion models [Belkin and Niyogi, 2002]. The uninformative setting $W_{ij} = 1$ is only usable when there is a very large number of points (and edges), so that connectivity information alone suffices to determine metric properties of the embedding. The Gaussian setting has a complementary weakness: It can be very sensitive to small variations in distance to neighbors (that may be introduced by the curvature of the data manifold or measurement noise in the ambient space).

f should be monotonically decreasing, relatively insensitive to noise (df should be small), and it should lead to exact reconstructions of data sampled from manifolds that are already flat. Straightforward calculus shows that equation 13 has the desired minimum when $f(\mathbf{x}_i, \mathbf{x}_j) \propto \|\mathbf{x}_i - \mathbf{x}_j\|^{-1}$, or more generally, the multiplicative inverse of whatever distance measure is appropriate in the ambient space². (By contrast, the LLE weightings are not correlated with distances.) To make the problem scale invariant, we scale $\mathbf{W}$ such that its largest nonzero off-diagonal value is 1 (consequently $df \leq 1$ everywhere f is computed).

Let us now situate some Gaussian kernels $p_k(\mathbf{x}) \doteq \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k)$ on the manifold. In this paper, we will take a random subset of data points as kernel centers, and set all $\Sigma_k = \sigma^2 \mathbf{I}$; these kernels are radial basis functions. Let vector

$$\mathbf{z}_{ik} \doteq \begin{bmatrix} \mathbf{K}_i(\mathbf{x}_i - \mu_k) \\ 1 \end{bmatrix} \frac{p_k(\mathbf{x}_i)}{\sum_k p_k(\mathbf{x}_i)} \quad (14)$$

be the k th local homogeneous coordinate of $\mathbf{x}_i$ scaled by the posterior of the k th kernel. $\mathbf{K}_i$ is an optional local dimensionality-reducing linear projection. Let representation vector

$$\mathbf{z}(\mathbf{x}_i) \doteq [\downarrow_k \mathbf{z}_{ik}] = [\mathbf{z}_{i1}^\top, \dots, \mathbf{z}_{iK}^\top]^\top \quad (15)$$

be the vertical concatenation of all such local coordinate vectors. Collect all such column vectors into a *basis* matrix $\mathbf{Z} \doteq [\mathbf{z}(\mathbf{x}_1), \dots, \mathbf{z}(\mathbf{x}_N)]$.

To summarize thus far, our goal now is to find a linear transform $\mathbf{y}_i \doteq G(\mathbf{z}(\mathbf{x}_i))$ of the basis (kernel-weighted coordinates) that is maximally consistent with our local distance constraints, specifically

$$\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N] = \arg \min_{\mathbf{Y}} \sum_{ij} \frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{\|\mathbf{x}_i - \mathbf{x}_j\|} \quad (16)$$

This is isomorphic to the graph embeddings of section 2; the methods developed there apply directly to $\mathbf{W}$ and $\mathbf{Z}$. The continuous mapping from ambient to embedding space immediately follows from the continuity and smoothness of $\mathbf{z}(\cdot)$:

$$G(\mathbf{x}) = \mathbf{G}\mathbf{z}(\mathbf{x}),$$

where the EVD determines the transformation $\mathbf{G} \doteq [\mathbf{v}_2, \dots, \mathbf{v}_{d+1}]^\top$ of the continuous kernel representation defined over the entire ambient space:

$$\mathbf{z}(\mathbf{x}) \doteq \left[\downarrow_k \begin{bmatrix} \mathbf{K}_i(\mathbf{x} - \mu_k) \\ 1 \end{bmatrix} \frac{p_k(\mathbf{x})}{\sum_k p_k(\mathbf{x})} \right]. \quad (17)$$

²Proof: Consider three points $\{x_1 = 0, x_2 = \lambda, x_3 = 1\}$ on a 1D manifold. What similarity measure $W_{ab} = f(\|x_a - x_b\|)$ causes the distortion $(y_2 - y_1)^2 W_{12} + (y_2 - y_3)^2 W_{23}$ to have a global minimum at $y_2 = \lambda$? Without loss of generality, we fix the global location and scale of the embedding by fixing the endpoints: $\{y_1 = 0, y_3 = 1\}$. Solving for the unique zero of the distortion's first derivative, we obtain the optimum at $y_2 = W_{23} / (W_{12} + W_{23})$. Since this is a harmonic relation, the unique continuous satisfying measure is $W_{ab} = (\|x_a - x_b\|)^{-1}$. This sets $W_{12} = 1/\lambda$ and $W_{23} = 1/(1 - \lambda)$; some simple algebra confirms that indeed $y_2 = \lambda$ at the optimum. The induction to multiple points in multiple dimensions is direct.

As a matter of numerical prudence, we recommend using the reweighted approximation:

$$G(\mathbf{x}) = \frac{\mathbf{G}\mathbf{z}(\mathbf{x})}{\mathbf{v}_1^\top \mathbf{z}(\mathbf{x})}. \quad (18)$$

At first blush, it would seem that reweighting should not be necessary: By construction, $\mathbf{1} \in \text{range}(\mathbf{Z}^\top)$, thus $\mathbf{v}_1^\top \mathbf{z}(\mathbf{x})$ —and the denominator—should be uniform at the datapoints. However, as mentioned above, even when the algebra predicts this structure, numerical eigensolvers may not find it.

To obtain an approximate inverse mapping, we map the means and covariances of each kernel $p_k(\cdot)$ into the target space to obtain kernels $p'_k(\mathbf{y}) \doteq \mathcal{N}(\mathbf{y}|\mu'_k, \Sigma')$ there. Then, breaking $\mathbf{G} = [\mathbf{G}_1, \dots, \mathbf{G}_K]$ into blocks corresponding to each kernel, take the Moore-Penrose pseudo-inverse of each, and set $\mathbf{G}^+ \doteq [\mathbf{G}_1^+, \dots, \mathbf{G}_K^+]$. If using the reweighted map, the approximate inverse map is

$$\mathbf{G}^+(\mathbf{y}) \approx \mathbf{G}^+ \mathbf{z}^+(\mathbf{y}) \cdot (\mathbf{v}_1^\top \mathbf{G}^+ \mathbf{z}^+(\mathbf{y})), \quad (19)$$

$$\text{where } \mathbf{z}^+(\mathbf{y}) \doteq \left[\downarrow_k \begin{bmatrix} \mathbf{y} - \mu'_k \\ 1 \end{bmatrix} \frac{p'_k(\mathbf{y})}{\sum_k p'_k(\mathbf{y})} \right].$$

4 Illustrative example

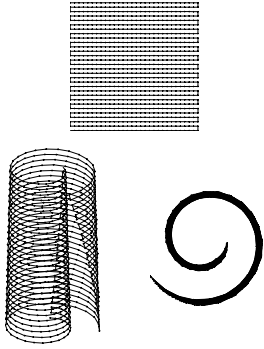


Figure 2: The swiss roll.

We will use a variant of the “swiss roll”, a standard test manifold in the NLDR community, to illustrate the arguments and methods developed in this paper. We sampled a twisted version of the manifold regularly on a 30×30 grid and added a small amount of Gaussian isotropic noise. Figure 2 shows the ideal $\mathbb{R}^2$ parameterization and two views of the ambient $\mathbb{R}^3$ embedding. Points are shown joined into a line to aid visual interpretation of the embeddings. All experiments in this

section use a $\mathbf{W}$ matrix that was generated using the 12 nearest neighbors to each point and the inverse distance function.

The Laplacian eigenmap embedding (figure 3) shows the embedding specified by the $\mathbf{W}$ matrix. Note that it exhibits some folding at the corners and top and bottom edges, due partly to problems with the uniform eigenvector and exacerbated by the fact that spectral embeddings tend to compress near the boundaries. The Laplacian eigenmap requires solution of a large 900×900 eigenproblem, and offers no mapping off the points. Kernel eigenmaps will be approximations to this embedding.

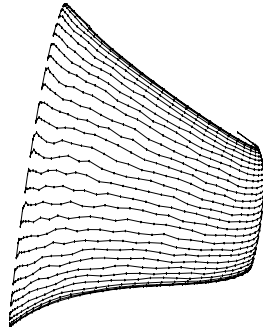


Figure 3: Laplacian eigenmap embedding.

We now show some kernel eigenmaps computed using the transformational embedding of section 2. All embedding methods are given the same inputs.

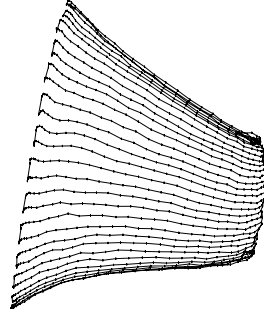


Figure 4: Kernel eigenmap embedding, raw result.

Figure 4 shows a raw kernel eigenmap embedding computed using a basis ($\mathbf{Z}$ matrix) created from 64 Gaussian unit- σ kernels placed on random points. This required solving a much more manageable 256×256 eigenproblem. 100 trials were performed with different sets of randomly placed kernels. In all trials, the reweighted and stochastic maps gave the best reconstructions, while

the raw and affine maps exhibited substantial folding at the edges and corners of the embedding.

Figure 5 shows a *reweighted* kernel eigenmap computed from the same $\mathbf{W}$ and $\mathbf{Z}$ as figures 3 & 4. The result is smoother and actually exhibits *less* folding than the original Laplacian eigenmap.

The problem can be regularized by putting positive mass on the diagonal of $\mathbf{W}$ (e.g., $\mathbf{W} \rightarrow \mathbf{W} + \mathbf{I}$), thereby making the recovered kernel eigenmap more isometric (bottom figure 5). This regularization is appropriate when it is believed that all neighborhoods are roughly the same size.

The recently proposed Locality Preserving Projection (LPP) [He and Niyogi, 2002], is essentially the raw approximation (direct solution of equation 8) with $W_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/t}$ and $\mathbf{Z} = \mathbf{X}$, thereby giving a linear projection from the ambient space to the target space that best preserves local relationships.

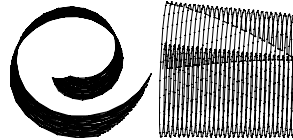


Figure 6: LPP embedding and our affine upgrade.

the swiss roll produced by LPP and by an affine modification of it that is equivalent to our method with a trivial single uniform-

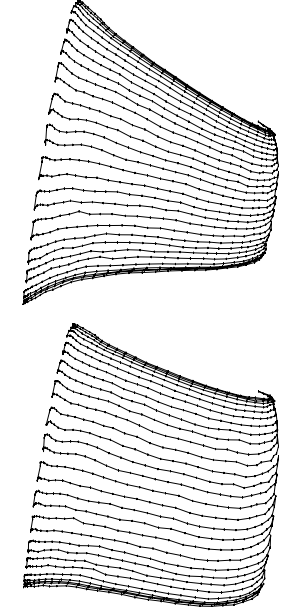


Figure 5: Kernel eigenmap embedding, reweighted and regularized results.

LPP is admirably simple, but it can be shown that the affine approximation from section 2 will always have less distortion. LPP can also suffer from loss of the uniform eigenvector. Figure 6 shows embeddings of the swiss roll produced by LPP and by an affine modification of it that is equivalent to our method with a trivial single uniform-

density kernel. Upgrading LPP to an affine projection captures more of the data's structure. Even so, there is no affine "view" of this manifold that avoids folding.

5 Visualizing word usages

In statistical analyses of natural language, similar usage patterns for two words are taken to indicate that they have similar meanings or strongly related meanings. Latent semantic analysis (LSA) is a linear dimensionality reduction of a term-document co-occurrence matrix. The principal components of this matrix give an embedding in which similarly used words are similarly located. Literally, co-location is a proxy for collocation (the propensity of words to be used together) and synonymy. We may expect that the kernel eigenmap offers a more powerful nonlinear analysis:

The NIPS12 corpus³ features a matrix counting occurrences of 13000+ words in 1700+ documents. We modeled the first 1000 words and the last 200 documents in the matrix. This roughly corresponds to one year's papers, a reasonable "snapshot" of the ever-changing terminology of the field. We stemmed the words and combined counts for the same roots, then determined distance between two word roots as the cosines of the angles between their log-domain-transformed occurrence vectors ($x_{ij} \rightarrow \log_2(1 + x_{ij})$). The $\mathbf{W}$ matrix was generated by adding an edge from each word to its 30 closest neighbors in cosine-space. The representation $\mathbf{Z}$ was made using 4 random words as kernel centers. Figure 7 discusses the resulting 2D embedding, in which technical terms are clearly grouped by field and many of the more common English words are tightly clustered by common semantics. The first two LSA dimensions (also shown in figure 7) of the same data are reveal significantly less semantic structure.

6 Discussion

The kernel eigenmap generates continuous nonlinear mapping functions for dimensionality reduction and manifold reconstruction. Suitable choices of kernels can reproduce the behavior of several other NLDR methods. One could put a kernel at every local group of points, perform local dimensionality reduction (e.g., a PCA) at each kernel, and thereby obtain from equations 8 and 17 an NLDR algorithm much like charting [Brand, 2003] or automatic alignment [Teh and Roweis, 2003]. Or, as in the demonstrations above, the kernel eigenmap can simultaneously determine the local dimensionality reductions and their global merger.

The kernel eigenmap typically substitutes a small dense EVD for the the large sparse EVD of graph embedding problems. In the sparse case, a specialized power method can compute the desired eigenvectors in significantly less than the $O(N^3)$ time required for a full EVD. In the kernel setting, similar efficiencies apply because both $\mathbf{W}$ and $\mathbf{Z}$ are typically sparse, allowing fast construction of the reduced EVD problem $\mathbf{ZWZ}^T$; this too is amenable to fast power methods. Of course, the most important efficiency of the kernel method is its ability to embed new points quickly via the function

$G(\mathbf{x})$ —there is no need to compute a new global embedding or revise the EVD.

The reweighting scheme, although theoretically mooted by our subsequent discovery of a better problem formulation and closed-form solution, is still practically viable as a fast approximation for large problems, and as a post-conditioning step for unavoidable numerical error of any NLDR algorithm based on eigenvalue decompositions.

In this paper we have used random kernels. There are numerous avenues to discovering stronger methods by investigating placement and tuning of the kernels, stability of the embedding and its topological structure, and sample complexity. In short, all the issues that proved fertile ground for research in classification and regression can be studied anew in the context of estimating the geometry and topology of manifolds.

References

- [Belkin and Niyogi, 2002] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. Technical Report TR-2002-01, University of Chicago Computer Science, 2002.
- [Brand, 2003] Matthew Brand. Charting a manifold. In *Proc. NIPS-15*, 2003.
- [Chung, 1997] Fan R.K. Chung. *Spectral graph theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.
- [Fiedler, 1975] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czech. Math. Journal*, 25:619–633, 1975.
- [He and Niyogi, 2002] Xiafei He and Partha Niyogi. Locality preserving projections. Technical Report TR-2002-09, University of Chicago Computer Science, October 2002.
- [Kambhatla and Leen, 1997] N. Kambhatla and Todd Leen. Dimensionality reduction by local principal component analysis. *Neural Computation*, 9, 1997.
- [Kruskal and Wish, 1978] J. B. Kruskal and M. Wish. *Multidimensional Scaling*. Sage Publications, Beverly Hills, CA, 1978.
- [Mohar, 1991] B. Mohar. The laplacian spectrum of graphs. In Y. Alavi, editor, *Graph Theory, Combinatorics and Applications*, pages 871–898. J. Wiley, New York, 1991.
- [Roweis and Saul, 2000] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, December 22 2000.
- [Teh and Roweis, 2003] Yee Whye Teh and Sam T. Roweis. Automatic alignment of hidden representations. In *Proc. NIPS-15*, 2003.
- [Tenenbaum et al., 2000] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, December 22 2000.

³Courtesy S. Roweis, available from the U. Toronto website.

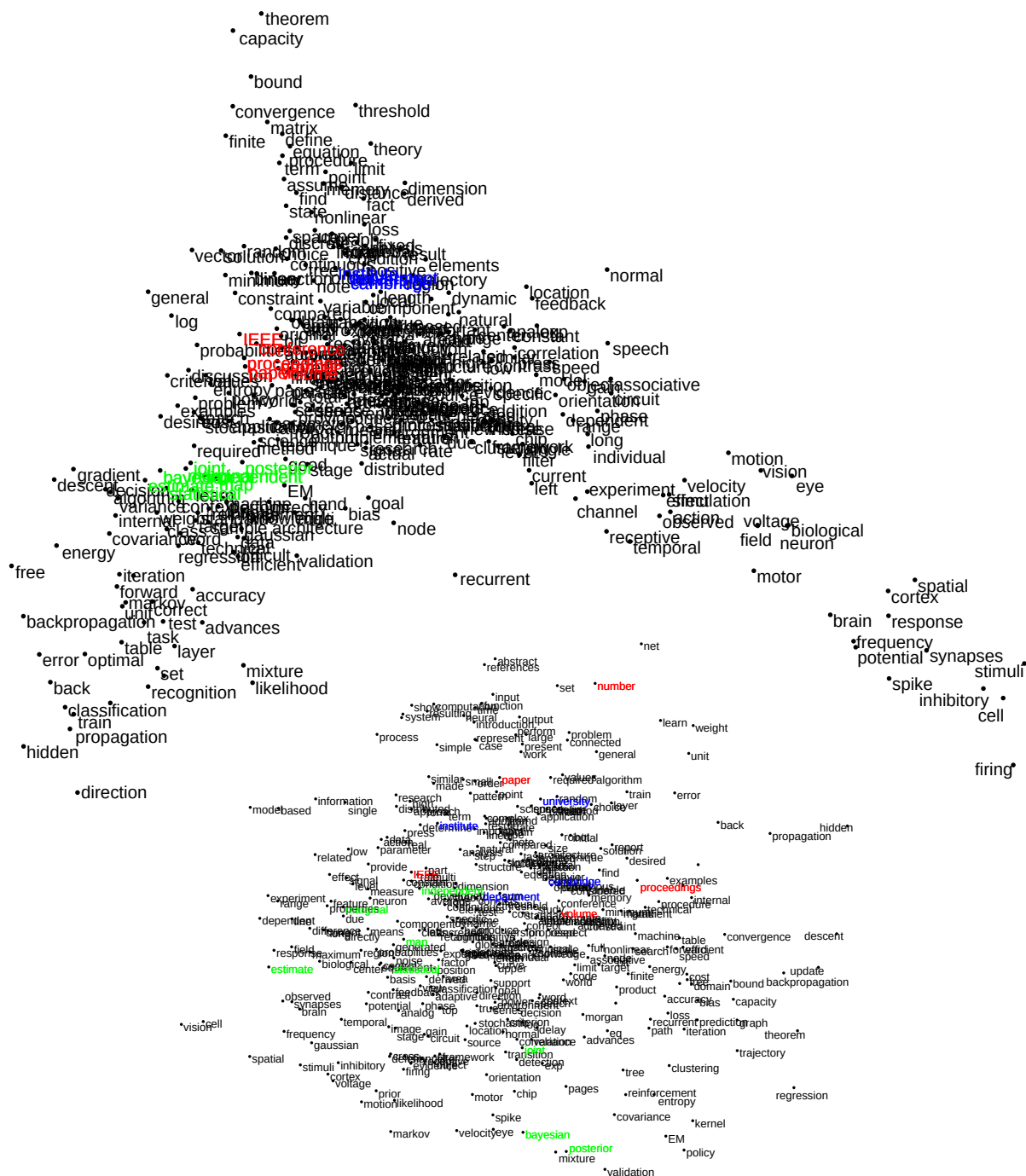


Figure 7: ABOVE: A 2D kernel eigenmap of word usages in recent NIPS papers. To improve legibility we show just a subset of the data; some labels have been shifted slightly to reduce overlap. Word roots are shown in their first occurring unstemmed variant. The three lobes of the distribution roughly correspond to favored terminology in the submission areas of Algorithms & Architectures (left), Neuroscience (right), and Theory (top). Words with broader usage are more tightly distributed in the center (presumably because they are more likely to co-occur in general discourse), with several clusters of words having strongly related meanings. Three of these clusters have been colored: red for publishing terms (IEEE, conference, number, paper, proceedings, volume), green for probability terms (bayesian, estimate, independent, map, marginal, posterior, joint, statistical), and blue for locations (cambridge, department, institute, university). BELOW, SMALLER: A linear embedding obtained from a latent semantic analysis of the same data. Though collocated words are often co-located, when compared with the kernel eigenmap result, semantic structures are far less obvious.

LEARNING

PARTIALLY LABELED DATA

Semi-Supervised Learning with Explicit Misclassification Modeling

Massih-Reza Amini, Patrick Gallinari

University of Pierre and Marie Curie

Computer Science Laboratory of Paris 6

8, rue du capitaine Scott, F-75015 Paris, France

{amini, gallinari}@poleia.lip6.fr

Abstract

This paper investigates a new approach for training discriminant classifiers when only a small set of labeled data is available together with a large set of unlabeled data. This algorithm optimizes the classification maximum likelihood of a set of labeled-unlabeled data, using a variant form of the Classification Expectation Maximization (CEM) algorithm. Its originality is that it makes use of both unlabeled data and of a probabilistic misclassification model for these data. The parameters of the label-error model are learned together with the classifier parameters. We demonstrate the effectiveness of the approach on four data-sets and show the advantages of this method over a previously developed semi-supervised algorithm which does not consider imperfections in the labeling process.

1 Introduction

In many real-life applications, labeling training data for learning is costly, sometimes not realistic and often prone to error. For example, for many rapidly evolving data bases available via the web, there is not enough time to label data for different information needs. In some cases, like medical diagnosis or biological data analysis, labeling data may require very expensive tests so that only small labeled data sets may be available. In other cases, like object identification in images, noise is inherent in the labeling process.

The statistician and pattern recognition communities were the first to consider the problem of forming discriminant rules using either partially classified or random misclassified training data in order to cope with this type of situation.

More recently this idea has motivated the interest of the machine learning community and many papers now deal with this subject. The use of partially classified data for training, known as semi-supervised learning, has been the subject of intense studies since 1998 and more recently there has been a resurgence of interest for training with misclassified data also called learning in presence of label noise.

We consider here semi-supervised learning for classification. Most approaches to this problem make use of a mixture density model where mixture components are identified as classes. Labeled data are known to belong to exactly one

mixture component whereas unlabeled data may belong to any components. Using the Expectation Maximization (EM) algorithm [Dempster et al., 1977], proposed approaches usually attempt to optimize the likelihood of the whole labeled-unlabeled data. Starting from an initial labeling, these approaches proceed by computing at each E-step, tentative labels for unlabeled data using the current parameters of the model and update in the M-step, these parameters using the estimated labels. Our departure point here is the work of [Amini and Gallinari-a, 2002] who proposed a semi-supervised discriminant algorithm using a variant form of the CEM algorithm [Celeux and Govaert, 1992]. Discriminative approaches which attempt to estimate directly posterior class probabilities are often considered superior to generative models which compute these posteriors after learning class conditional densities. Tests on different datasets in [Amini and Gallinari-b, 2002] led to the same conclusion for semi-supervised learning. Like for other methods, at each step of the algorithm, the model computes tentative labels for unlabeled data. We extend here the system by incorporating a model which takes into account label errors. This provides an unifying framework for semi-supervised learning and learning with label noise. To our knowledge, this form of model has not been studied yet. We detail the algorithm for the case of a logistic classifier and give a convergence proof for the general case. We then show experimentally that modeling the stochastic labeling noise, increases notably the performance, especially when only small labeled datasets are available. This paper is organized as follows; we first make a brief review of work on semi-supervised learning and learning in the presence of label noise (section 2). In section 3 we present the formal framework of our model and describe in section 4 the semi-supervised approach we propose. Finally we present a series of experiments on four data sets.

2 Related work

2.1 Learning with both labeled and unlabeled data

The idea of using partially labeled data for learning started in the statistician community at the end of 60's. The seminal paper by [Day, 1969] presents an iterative EM-like approach for learning the parameters of a mixture density under the assumption of multivariate normal components with a common covariance matrix for two group-conditional distributions.

Other iterative algorithms for building maximum likelihood classifiers from labeled and unlabeled data based on the same type of assumption followed [O'Neill, 1978] [McLachlan and Ganesalingam, 1982]. Some other authors have suggested updating procedures for no-normal group conditional densities using for example kernel methods for modeling mixture components [Murray and Titterton, 1978]. There has been considerably fewer work on discriminative approaches. In his fundamental paper on logistic regression, Anderson suggests modifying a logistic regression classifier to incorporate unlabeled data in order to maximize the likelihood function [Anderson, 1979].

The semi-supervised paradigm has been recently rediscovered by the machine learning community. Most papers propose mixture density models for combining labeled and unlabeled data for classification. [Miller and Uyar, 1996] consider a mixture density model where each class is described by several component densities. [Roth and Steinhage, 1999] propose kernel discriminant analysis as an extension to classical linear discriminant analysis. This framework can be used also for semi-supervised learning. [Nigam et al., 2000] propose a semi-supervised EM algorithm which is essentially similar to the one in [McLachlan, 1992] but makes use of naive Bayes estimator for modeling the different densities. They present empirical evaluation for text classification tasks. Some authors make use of discriminant classifiers instead of modeling conditional densities. For example [Joachims, 1999] propose a transductive support vector machine which finds parameters for a linear separator using both the labeled data in the training set and the current test data whose class is unknown. [Blum and Mitchell, 1998] introduce the co-training paradigm where each sample x is supposed to be described by two modalities. Two classifiers are then used, one for each modality, operating alternatively as teacher and student. This framework can be used for unsupervised and semi-supervised learning. Based on the multi-modal framework introduced by [Blum and Mitchell, 1998], [Muslea et al., 2002] propose to combine both active and semi-supervised learning. Since that, different authors have proposed semi-supervised learning schemes, they usually follow one of the above ideas.

2.2 Learning with imperfectly labeled data

Practical applications of pattern recognition, like eg image classification problems, have motivated in the early 80's some work on the problem of learning in presence of mislabeled data for fully supervised learning. [Chittineni, 1980] obtained error bounds on the performance of the Bayes and nearest neighbor classifiers with imperfect labels. [Krishnan, 1988] considered a 2-class classification problem for two group multivariate normal mixture when training samples are subject to random misclassification and derived the likelihood estimation of parameters. [Titterton, 1989] proposed a logistic-normal distribution model and worked out an EM algorithm for the estimation of its parameters. More recently, [Lawrence and Schölkopf, 2001] proposed an algorithm for constructing a kernel Fisher discriminant from training examples in the presence of label noise.

3 A semi-supervised probabilistic model for mislabeling

We consider here the problem of semi-supervised learning. We start from the Logistic-CEM algorithm described in [Amini and Gallinari-a, 2002]. It is a generic scheme in the sense that it can be used with any discriminant classifier estimating the posterior class probabilities. The classifier is first trained on labeled data, it then alternates two steps until convergence to a local maximum of the Classification Maximum Likelihood (CML) criterion [Symons, 1981]. Unlabeled data are first labeled using the output of the current classifier. Classifier parameters are then learned by maximizing the CML function computed using the known labels of labeled data and the current estimated labels for unlabeled data. In this algorithm, at each iteration, labels computed for unlabeled data are considered as desired outputs.

We suppose here that labels from the labeled dataset are correct and that at each step of this algorithm, labels computed for unlabeled data are subject to error. We propose to model the imperfection of these labels using a probabilistic formalism and to learn the semi-supervised classifier by taking into account its labeling errors according to this error model. Parameters of the error model and of the classifier will be learned simultaneously in this new algorithm. Compared to the baseline algorithm [Amini and Gallinari-a, 2002], we explicitly take into account the fact that the current classifier is not optimally trained at each step, since many data labels are missing and are only estimated.

In the following we present the general framework and the learning criterion of our model.

3.1 General framework

We suppose that each example belongs to one and only one class and that there are available a set of n labeled examples, D_l and a set of m unlabeled examples, D_u . A discriminant classifier is to be trained on the basis of these $n + m$, d -dimensional feature vectors¹ $x \in \mathbb{R}^d$. For each labeled example x_i in D_l , let c_i and $t_i = \{t_{ki}\}_k$ be respectively the class label and the indicator vector class associated to x_i :

$$\forall i \in \{1, \dots, n\}, \forall k, c_i = k \Leftrightarrow t_{ki} = 1 \text{ and } \forall h \neq k, t_{hi} = 0$$

We suppose that for each unlabeled example x_i in D_u , there exists a perfect and an imperfect label respectively denoted c_i and $\tilde{c}_i$. We propose to model the imperfections in the labels by the following probabilities:

$$\forall k, \forall h, \alpha_{kh} = p(\tilde{c} = k \mid c = h) \quad (1)$$

Which are subject to the constraint:

$$\forall h, \sum_k \alpha_{kh} = 1 \quad (2)$$

In order to simplify the presentation, we consider in the following a two-class classification problem.

$$\forall i \in \{1, \dots, n\}, \forall i' \in \{n+1, \dots, n+m\}, c_i, \tilde{c}_{i'} \in \{1, 2\}$$

This is not restrictive since we can easily extend the analysis to multi-class cases.

¹Components of x could also be discrete

3.2 Classification Maximum Likelihood estimation in presence of label noise

The baseline semi-supervised discriminant algorithm has been conceived as an extension to classification problems of the CEM algorithm proposed by [Celeux and Govaert, 1992] for clustering. CEM has been first proposed for learning the parameters of gaussian mixtures using as training criterion the classification maximum likelihood criterion initially described in [Symons, 1981].

McLachlan has extended CML and CEM to the case where both labeled and unlabeled data are used for learning ([McLachlan, 1992], page 39). The CML criterion in this case is the complete-data likelihood and writes:

$$V_c = \prod_{i=1}^n \prod_{k=1}^2 \{p(x_i, c = k)\}^{t_{ki}} \times \prod_{i=n+1}^{n+m} \prod_{k=1}^2 \{p(x_i, \tilde{c} = k)\}^{\tilde{t}_{ki}}$$

The algorithm proposed by [McLachlan, 1992] optimizes this criterion for a mixture of normal densities. In their algorithm for training discriminant classifiers on labeled and unlabeled data, [Amini and Gallinari-a, 2002] make use of the following form of the log-classification-likelihood:

$$L_c = \sum_{i=1}^n \sum_{k=1}^2 t_{ki} \log p(c = k | x_i) + \sum_{i=n+1}^{n+m} \sum_{k=1}^2 \tilde{t}_{ki} \log p(\tilde{c} = k | x_i) + \sum_{i=1}^{n+m} \log p(x_i)$$

This writing makes apparent the posterior probabilities which are directly estimated in their algorithm instead of the conditional densities in [McLachlan, 1992]. As no assumptions are made on the distributional nature of data, maximizing L_c is equivalent to the maximization of L'_c ([McLachlan, 1992], page 261).

$$L'_c = \sum_{i=1}^n \sum_{k=1}^2 t_{ki} \log p(c = k | x_i) + \sum_{i=n+1}^{n+m} \sum_{k=1}^2 \tilde{t}_{ki} \log p(\tilde{c} = k | x_i) \quad (3)$$

Let us now introduce our misclassification model. For that, we will express the $p(\tilde{c} = k | x_i)$ in the second summation in (3) as a function of the mislabeling probabilities and of the posterior probability of correct labels $p(c = k | x_i)$. Consider

$$p(x_i, \tilde{c} = k) = \sum_{h=1}^2 p(x_i, \tilde{c} = k, c = h)$$

$p(x_i, \tilde{c} = k, c = h)$ can be decomposed with regard to the conditional class probabilities:

$$p(x_i, \tilde{c} = k, c = h) = p(x_i | \tilde{c} = k, c = h) \times p(\tilde{c} = k, c = h)$$

Following [Chittineni, 1980] we make the assumption that the density of an example, given its true label, does not depend on its imperfect label:

$$p(x_i | \tilde{c} = k, c = h) = p(x_i | c = h)$$

It then comes:

$$p(x_i, \tilde{c} = k) = \sum_{h=1}^2 p(x_i | c = h) \times p(c = h) \times p(\tilde{c} = k | c = h)$$

Using Bayes rule and (1), we get:

$$p(x_i, \tilde{c} = k) = p(x_i) \times \sum_{h=1}^2 \{\alpha_{kh} p(c = h | x_i)\} \quad (4)$$

$p(\tilde{c} = k | x_i)$ can then be derived from $p(c = k | x_i)$

$$\forall x_i \in D_u, p(\tilde{c} = k | x_i) = \sum_{h=1}^2 \{\alpha_{kh} p(c = h | x_i)\} \quad (5)$$

From (5), (3) becomes

$$L'_c = \sum_{i=1}^n \sum_{k=1}^2 t_{ki} \log p(c = k | x_i) + \sum_{i=n+1}^{n+m} \sum_{k=1}^2 \left\{ \tilde{t}_{ki} \log \left(\sum_{h=1}^2 \alpha_{kh} p(c = h | x_i) \right) \right\} \quad (6)$$

4 Updating a discriminant function on the basis of labeled and imperfect labeled data

We now present an iterative discriminant CEM algorithm for learning a classifier for semi-supervised learning, which incorporates the mislabeling error model. The training criterion is (6). For simplification, we consider a simple logistic classifier [Anderson, 1982], but the algorithm can be easily adapted for training any discriminant classifier. Consider C_β a logistic classifier with parameters $\{\beta_j\}_{j \in \{0, \dots, d\}}$. The output of C_β

for input $x \in \mathbb{R}^d$ is $G(x) = \frac{1}{1 + \exp(-(\beta_0 + \beta^t x))}$. After β have been learned, $G(x)$ and $1 - G(x)$ are respectively used to estimate $p(c = 1 | x)$ and $p(c = 2 | x)$. Let, $\pi^{(p)}$ be the current partition for the unlabeled data, $\alpha^{(p)}$, $\beta^{(p)}$ the parameters for the misclassification model and the logistic classifier at iteration p of the algorithm. The learning criterion (6) is a function of α , β and π . An iterative approach is then adopted for its maximization (algorithm 1). Parameters β are first initialized by training the classifier on the labeled dataset D_l . Two steps are then iterated until the convergence of criterion L'_c . In the first step, the classifier is considered as an imperfect supervisor for the unlabeled data. Its outputs, $G(x)$ and $1 - G(x)$, are used to compute the posterior imperfect class probabilities $p(\tilde{c} = k | x)$ for each x in D_u and $k \in \{1, 2\}$, x is then labeled according to the maximum imperfect output. In the second step, the parameters of the error model and of the classifier are updated using the imperfect labels obtained in the previous step as well as the labeled data. We adopted in this step, a gradient algorithm to maximize (6). An advantage of this method is that it only requires the first order derivatives at each iteration. In the following lemma, we provide a proof of convergence of the algorithm to a local maximum of the likelihood function for semi-supervised training.

Algorithm 1 Semi-supervised learning with imperfect labels

Initialization : Train a discriminant logistic classifier $C_{\beta^{(0)}}$ over D_l

For $p \geq 0$, iterate until the convergence of L'_c

1. Apply $C_{\beta^{(p)}}$ on D_u , estimate the imperfect class posterior probabilities using the output $G^{(p)}$ of the classifier and get an imperfect label for each $x_i \in D_u$:

$$\forall x_i \in D_u, \tilde{c}_i^{(p+1)} = \underset{k}{\operatorname{argmax}} p(\tilde{c}^{(p)} = k \mid x_i)$$

Let $\pi^{(p+1)}$ be the new partition obtained from this classifier for the unlabeled data.

2. a) Maximize $L'_c(\pi^{(p+1)}, \alpha^{(p)}, \beta^{(p)})$ with respect to $\alpha^{(p)}$ subject to constraints $\forall k, \forall h, \alpha_{kh}^{(p+1)} \in [0, 1]$ and $\forall h, \sum_k \alpha_{kh}^{(p+1)} = 1$
b) Maximize $L'_c(\pi^{(p+1)}, \alpha^{(p+1)}, \beta^{(p)})$ with respect to $\beta^{(p)}$

Lemma 1. *The modified CML criterion, L'_c , increases for every sequence $(\pi^{(p)}, \alpha^{(p)}, \beta^{(p)})$ of Algorithm 1 and the sequence $L'_c(\pi^{(p)}, \alpha^{(p)}, \beta^{(p)})$ converges to a stationary point.*

Proof : we first show that L'_c is increasing.

- Since $\forall x_i \in D_u, \tilde{c}_i^{(p+1)} = k \Leftrightarrow p(\tilde{c}^{(p+1)} = k \mid x_i) \geq p(\tilde{c}^{(p+1)} = k' \mid x_i)$ for $k \neq k'$ (step 1), we have

$$L'_c(\pi^{(p+1)}, \alpha^{(p)}, \beta^{(p)}) \geq L'_c(\pi^{(p)}, \alpha^{(p)}, \beta^{(p)})$$

- and, since $(\pi^{(p+1)}, \alpha^{(p+1)}, \beta^{(p)})$ and $(\pi^{(p+1)}, \alpha^{(p+1)}, \beta^{(p+1)})$ maximize iteratively L'_c (step 2), we have

$$L'_c(\pi^{(p+1)}, \alpha^{(p+1)}, \beta^{(p+1)}) \geq L'_c(\pi^{(p+1)}, \alpha^{(p)}, \beta^{(p)})$$

Finally as there is a finite number of partitions of the example into 2-classes, the increasing sequence $L'_c(\pi^{(p)}, \alpha^{(p)}, \beta^{(p)})$ takes a finite number of values and thus, converges to a stationary value. ■

In the following section we will present results on four datasets, using a baseline logistic classifier trained with the updating scheme presented above.

5 Experiments

5.1 Data sets

In our experiments we used the Spambase, Credit screening and Mushroom collections from the UCI repository² as well as the Computation and Language (Cmp_lg) collection of TIPSTER SUMMAC³ for text summarization. Table 1 summarizes the characteristics of these datasets. We removed 37 samples with missing attributes from the Credit data set. For summarization, we

²ftp://ftp.icsS.uci.edu/pub/machine-learning-databases/

³http://www.itl.nist.gov/iaui/894.02/related_projects/tipster_summac/

Table 1: Characteristics of the datasets and percentage of the smallest of the two classes in the datasets.

Data set	size	Attributes	Smallest class (%)
Credit	690(-37)	15	44.5%
Email spam	4601	57	39.4%
Mushroom	8124	22	48.2%
Cmp_lg	28985	5	10%

adopt the text-span extraction paradigm which amounts at selecting a subset of representative document sentences. This is a classification problem where sentences are to be classified as relevant or non relevant for the extracted summary. All four problems are two-class classification tasks, Email Spam and Cmp_lg have continuous attributes, Credit vectors are mixed continuous and qualitative, Mushroom attributes are qualitative.

For each dataset, we ran 3 algorithms - the semi-supervised learning algorithm using the label imperfections, the baseline semi-supervised algorithm [Amini and Gallinari-a, 2002] and a fully supervised logistic classifier trained only on the available labeled data. We compared the performance of these algorithms on 20 runs of arbitrary training and test splits, by varying the proportion of labeled-unlabeled data on the training set.

For text summarization, we represent each sentence using a continuous version of features proposed by [Kupiec et al., 1995]. This characterization has given good results in previous work [Amini and Gallinari-b, 2002]. Each sentence i , with length $l(i)$ is characterized by $\vec{x}_i = (\varphi_1^i, \varphi_2^i, \varphi_3^i, \varphi_4^i, \varphi_5^i)$, where $\varphi_1^i = \frac{l(i)}{\sum_j l(j)}$ is the normalized sentence length, $\varphi_2^i = \frac{\text{frequency of cue words in } i}{l(i)}$ is the normalized number of cue words in sentence i , $\varphi_3^i = \frac{\text{frequency of acronyms in } i}{l(i)}$ is the normalized number of acronyms in i , φ_4^i is the position indicator feature (beginning, middle or end of the document) and φ_5^i is the normalized number of terms within a generic query q and i . q corresponds to the most frequent words in the training set.

5.2 Evaluation measures

For the three UCI datasets there is approximately the same proportion of examples for each class (table 1). We used as performance criterion the percentage of good classification (PGC) defined as:

$$\text{PGC} = \frac{\# \text{ of examples in the test set well classified by the system}}{\# \text{ of examples in the test set}}$$

For text summarization, we followed the SUMMAC evaluation by using a 10% compression ratio. Hence, for each document in the test set we have formed its summary by selecting the top 10% sentences having higher score with respect to the output of the classifier. For evaluation we compared these extractive sentences with the desired summary of each document. The desired extractive summaries were generated from the abstract of each article using the text-span alignment method described in [Banko et al., 1999]. Since the collection is not well balanced between positive and negative

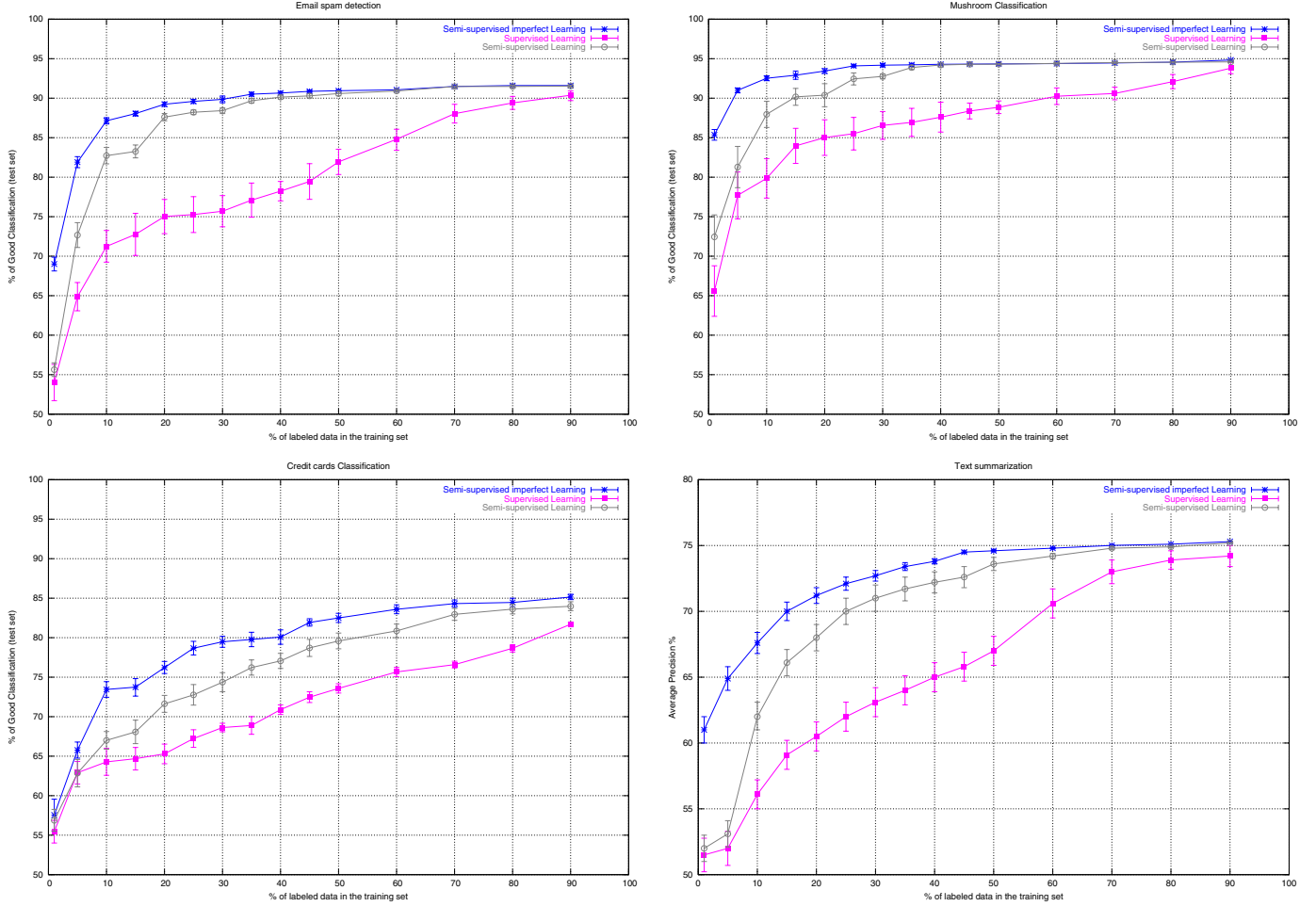


Figure 1: Performance curves for the 4 datasets showing, for a classical logistic classifier trained in a fully supervised scheme (square), the baseline semi-supervised Logistic-CEM algorithm (circle) and our semi-supervised algorithm with label imperfections (star). Each point represents the mean performance for 20 arbitrary runs. The error bars show standard deviations for the estimated performance.

examples, PGC is meaningless, we used the average precision (AP) measure for the evaluation. Let ξ , be the number of sentences extracted by the system which are in the target summaries, and σ , be the total number of sentences extracted by the system. The precision is defined as the ratio ξ/σ .

5.3 Results

For each dataset and each cross validation, 25% of examples are held aside as a test set. We vary the percentage of labeled-unlabeled data in the remaining 75% training set of the collections. Figure 1, shows the performance on the test sets for the four datasets as a function of the proportion of labeled data in the training set. On the x -axis, 5% means that 5% of data in the training set were labeled for training, the 95% remaining being used as unlabeled training data. Each experiment was carried on twenty paired trials of randomly selected training-test splits. On the y -axis, each point represents the mean performance for the 20 runs and the error bars correspond to the standard deviation for the estimated performance [Tibshirani,

1996].

All figures exhibit the same behavior. In all four datasets, semi-supervised algorithms are over the baseline classifier trained only on the labeled training data. For example, if we consider text summarization, using only 5% of labeled sentences, our algorithm allows to increase performance by 15% compared to a fully supervised algorithm trained on the 5%. 15% labeled data are needed to reach the same performance with the baseline method. Our model is uniformly better than the two reference models used in our experiments. It provides an important performance increase especially when there are only few labeled data available which is the most interesting situation in semi-supervised learning. For the Credit card problem, the dataset is small and semi-supervised learning allows for a smaller increase of performance than for other datasets where there are a lot of unlabeled data available. Semi-supervised learning allows to reach 'optimal' performance with about 50% of labeled data (only 30% for the Mushroom set).

6 Conclusion

We have described how to incorporate a label error model into an iterative semi-supervised discriminant algorithm. We have detailed a version of this general algorithm in the case of a simple logistic classifier, shown its convergence and proved empirically its efficiency on four datasets with different characteristics. The algorithm allows for an important performance increase compared to a reference efficient semi-supervised algorithm without mislabeling model. The main contribution of the paper is to provide a general framework for handling simultaneously semi-supervised learning and learning in the presence of label noise. The noise model we have used is simple and allows for efficient estimations in the semi-supervised setting. More sophisticated models have still to be investigated. However, our experience is that simple models do often perform better when only few labeled data are available.

References

- [Amini and Gallinari-a, 2002] M. R. Amini and P. Gallinari. Semi-supervised logistic regression. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 390–394, 2002. ECAI.
- [Amini and Gallinari-b, 2002] M. R. Amini and P. Gallinari. The use of unlabeled data to improve supervised learning for text summarization. In *Proceedings of the 25th International ACM SIGIR Conference*, pages 105–112, 2002. SIGIR.
- [Anderson, 1979] J. A. Anderson. Multivariate logistic compounds. *Biometrika*, 66(1):17–26, 1979.
- [Anderson, 1982] J. A. Anderson. Logistic discrimination. In *Handbook of Statistics*, P.R. Krishnaiah and L. Kanal (Eds.), 2:169–191, 1982.
- [Banko et al., 1999] M. Banko, V. Mittal, M. Kantrowitz, J. Goldstein. Generating extraction-based summaries from hand-written done by text alignment. In *Proceedings of the Pacific Association for Computational Linguistics*, 1999. PACLING.
- [Blum and Mitchell, 1998] A. Blum and T. Mitchell. Combining labeled and unlabeled data with Co-training. In *Proceedings of the Workshop on Computational Learning Theory*, pages 92–100, 1998. COLT.
- [Celeux and Govaert, 1992] G. Celeux and G. Govaert. A Classification EM algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*, 14(3):315–332, 1992.
- [Chittineni, 1980] C. B. Chittineni. Learning with imperfect labeled patterns. *Pattern Recognition*, 12(5):281–291, 1980.
- [Day, 1969] N. E. Day. Estimating the components of a mixture of normal distributions. *Biometrika*, 56(3):463–474, 1969.
- [Dempster et al., 1977] A. Dempster, N. Laird and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society, Series B*, 39(1):1–38, 1977.
- [Joachims, 1999] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209, 1999. ICML.
- [Krishnan, 1988] T. Krishnan. Efficiency of learning with imperfect supervision. *Pattern Recognition*, 21(2):183–188, 1988.
- [Kupiec et al., 1995] J. Kupiec, J. Pederson, F. A. Chen. Trainable Document Summarizer. In *Proceedings of the 18th International ACM SIGIR Conference*, pages 68–73, 1995. SIGIR.
- [Lawrence and Schölkopf, 2001] N. D. Lawrence and B. Schölkopf. Estimating a kernel Fisher discriminant in the presence of label noise. In *Proceedings of the 18th International Conference on Machine Learning*, pages 306–313, 2001. ICML.
- [McLachlan, 1992] G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley & Sons, Inc. New York, 1992.
- [McLachlan and Ganesalingam, 1982] G. J. McLachlan and S. Ganesalingam. Updating a discriminant function in basis of unclassified data. *Communication on Statistics-Simulation and Computation*, 11(6):753–767, 1982.
- [Miller and Uyar, 1996] D. Miller and H. Uyar. A mixture of experts classifier with learning based on both labeled and unlabeled data. *Advances in Neural Information Processing Systems* 9, pages 571–577, 1996. NIPS.
- [Murray and Titterton, 1978] G. D. Murray and D. M. Titterton. Estimation problems with data from a mixture. *Applied Statistics*, 27(3):325–334, 1978.
- [Muslea et al., 2002] I. Muslea, S. Minton and C. Knoblock. Active + semi-supervised learning = robust multi-view learning. In *Proceedings of the 19th International Conference on Machine Learning*, pages 435–442, 2002. ICML.
- [Nigam et al., 2000] K. Nigam, A. K. McCallum, S. Thrun, T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.
- [O’Neill, 1978] T. J. O’Neill. Normal discrimination with unclassified observations. *Journal of the American Statistical Association*, 73(364):821–826, 1978.
- [Roth and Steinhage, 1999] V. Roth and V. Steinhage. Nonlinear discriminant analysis using kernel functions. *Advances in Neural Information Processing Systems* 12, pages 568–574, 1999. NIPS.
- [Symons, 1981] M. J. Symons. Clustering criteria and multivariate normal mixture. *Biometrics*, 37(1):35–43, 1981.
- [Tibshirani, 1996] R. Tibshirani. A Comparison of Some Error Estimates for Neural Network Models *Neural Computation*, 8:152–163, 1996.
- [Titterton, 1989] D. M. Titterton. An alternative stochastic supervisor in discriminant analysis *Pattern Recognition*, 22(1):91–95, 1989.

Spectral Learning

Sepandar D. Kamvar

SCCM

Stanford University

Stanford, CA 94305-9040

sdkamvar@cs.stanford.edu

Dan Klein

Computer Science Dept.

Stanford University

Stanford, CA 94305-9040

klein@cs.stanford.edu

Christopher D. Manning

Computer Science Dept.

Stanford University

Stanford, CA 94305-9040

manning@cs.stanford.edu

Abstract

We present a simple, easily implemented spectral learning algorithm which applies equally whether we have no supervisory information, pairwise link constraints, or labeled examples. In the unsupervised case, it performs consistently with other spectral clustering algorithms. In the supervised case, our approach achieves high accuracy on the categorization of thousands of documents given only a few dozen labeled training documents for the 20 Newsgroups data set. Furthermore, its classification accuracy increases with the addition of unlabeled documents, demonstrating effective use of unlabeled data. By using normalized affinity matrices which are both symmetric and stochastic, we also obtain both a probabilistic interpretation of our method and certain guarantees of performance.

1 Introduction

Spectral algorithms use information contained in the eigenvectors of a data affinity (i.e., item-item similarity) matrix to detect structure. Such an approach has proven effective on many tasks, including information retrieval [Deerwester *et al.*, 1990], web search [Page *et al.*, 1998; Kleinberg, 1998], image segmentation [Meila and Shi, 2000], word class detection [Brew and Schulte im Walde, 2002] and data clustering [Ng *et al.*, 2002]. But while spectral algorithms have been very useful in unsupervised learning (clustering), little work has been done in developing spectral algorithms for supervised learning (classification).

In this work, we consider the adaptation of spectral clustering methods to classification. We first present a method for combining item similarities with supervision information to produce a Markov transition process between data items. We call this Markov process the “interested reader” model by appeal to the special case of text clustering/classification. Our algorithm incorporates supervisory information whenever it is available, either in the form of pairwise constraints or labeled data (or both). Empirically, our algorithm achieves high accuracy when supplied either small amounts of labeled data (Section 4) or small numbers of pairwise constraints (Section 5).

2 The “Interested Reader” Model

We propose a Markov chain model similar in spirit to the “random surfer” model of [Page *et al.*, 1998].¹ This description is motivated in the context of text categorization, but the model depends only on notions of pairwise data similarity and is completely general. In the model, there is a collection of documents, each of which has some (possibly unknown) topic. A reader begins with some document of interest and continues to read successive documents. When she chooses the next document to read, she tries to read another document on the same topic, and hence will prefer other documents which are similar to her current document. Some mapping between similarities and transition probabilities must be chosen; we describe a specific choice in Section 3.

These transition probabilities define a Markov chain among the documents in the collection. If there exist distinct topic areas in the document set (or, generally, if there are clusters in the data), this Markov chain will be composed of subsets that have high intra-set transition probabilities, and low inter-set transition probabilities. We will refer to these subsets as *cliques*. Each of the cliques corresponds to a topic in the text clustering problem.

Of course, the natural clusters in the data need not be perfectly compatible with document labels, and we have said nothing about the use of supervision information. In Section 4, we use supervision to override the similarity-based transition probabilities. For example, we will disallow transition between two documents which are known to be differently-labeled, regardless of their pairwise similarity.

3 Spectral Clustering Algorithms

In this section, we discuss the process of turning an *affinity matrix* A of pairwise document similarities into a normalized Markov transition process N . The eigenvectors of N are then used to detect blocks or near-blocks in N , which will correspond to clusters of the data.

¹Note that there is an important difference between the way these two models are used; the random surfer model is used for the first left eigenvector of the transition matrix, which indicates the relative amount of time the process spends at each data item. On the other hand, we are interested in right eigenvectors of our transition matrix, which more straightforwardly relate to (near-)block structure in the transition matrix.

Form spectral representation:

1. Given data B , form the affinity matrix $A \in \mathbb{R}^{n \times n} = f(B)$.
2. Define D to be the diagonal matrix with $D_{ii} = \sum_j A_{ij}$.
3. Normalize: $N = (A + d_{\max}I - D)/d_{\max}$.
4. Find $x_1, \dots, x_k$, the k largest eigenvectors of N and form the matrix $X = [x_1, \dots, x_k] \in \mathbb{R}^{n \times k-1}$.
5. Normalize the rows of X to be unit length.

For clustering:

6. Treating each row of X as a point in $\mathbb{R}^k$, cluster into k clusters using k-means or any other sensible clustering algorithm.
7. Assign the original point x_i to cluster j if and only if row i of X was assigned to cluster j .

For classification:

6. Represent each data point i by the row X_i of X .
7. Classify these rows as points in $\mathbb{R}^k$ using any reasonable classifier, trained on the labeled points.
8. Assign the data point i the class c that X_i was assigned.

Figure 1: Spectral Learning Algorithm.

Algorithm	Normalization	$v(A, D)$
MNCUT	Divisive	$N = D^{-1}A$
NJW	Symmetric Divisive	$N = D^{-1/2}AD^{-1/2}$
LSA	None	$N = A$
SL	Normalized Additive	$N = (A + d_{\max}I - D)/d_{\max}$

Table 1: Normalizations used by spectral methods.

3.1 Calculating the Transition Matrix

In order to fully specify the data-to-data Markov transition matrix, we must map document similarities to transition probabilities. Let A be the affinity matrix over documents whose elements A_{ij} are the similarities between documents i and j . When we are given documents i as points x_i and a distance function $d(x_i, x_j)$, a common definition is $A_{ij} = e^{-d(x_i, x_j)/2\sigma^2}$, where σ is a free scale parameter. In LSA [Deerwester *et al.*, 1990], we are given a row-normalized term-document matrix B , and A is defined to be $B^T B$ (the cosine similarity matrix [Salton, 1989]).

We may map document similarities to transition probabilities in several of ways. We can define $N = D^{-1}A$ [Meilă and Shi, 2001], where D is the diagonal matrix whose elements $D_{ii} = \sum_j A_{ij}$. This corresponds to transitioning with probability proportional to relative similarity values. Alternatively, we can define $N = (A + d_{\max}I - D)/d_{\max}$ [Fiedler, 1975; Chung, 1997], where $d_{\max}$ is the maximum rowsum of A . Here, transition probabilities are sensitive to the absolute similarity values. For example, if a given document is similar to very few others, the interested reader may keep reading that document repeatedly, rather than move on to another document. While either of these normalizations are plausible, we chose the latter, since it had slight empirical performance benefits for our data.

In [Meilă and Shi, 2001], it is shown that a probability transition matrix N for a Markov chain with k strong cliques will have k piecewise constant eigenvectors, and they suggest clustering by finding approximately equal segments in the top k eigenvectors. Our algorithm uses this general method as

	Spectral Learning	k-means
3 NEWS	0.84	0.20
20 NEWS	0.36	0.07
LYMPHOMA	0.50	0.10
SOYBEAN	0.41	0.34

Table 2: A comparison of Spectral Learning and k-means.

well, but some of the details differ; our algorithm is shown in Figure 1. This algorithm is most similar to the algorithm presented in [Ng *et al.*, 2002], which we call NJW after its authors. In fact, the only difference is the type of normalization used. There are two differences between our algorithm and MNCUT from [Meilă and Shi, 2001]; the normalization of A is again different, and, additionally, MNCUT does not row normalize X (step 5). Table 1 describes the different types of normalizations and mentions some algorithms that use them.

It should be noted that for data sets where there are distant outliers, additive normalization can lead to very poor performance. This is because, with additive normalization, the outliers become their own clique. Therefore, the clusters will represent outliers rather than true clusters. In a dataset where there are distant outliers, divisive normalization is likely to lead to better performance.

3.2 Parameter Selection

The importance of parameter selection is often overlooked in the presentation of standard spectral clustering methods. With different values of σ , the results of spectral clustering can be vastly different. In [Ng *et al.*, 2002], the parameter σ is chosen based on that value of σ that gives the least distorted clusters.

In our text experiments, the data B was a term-document matrix, and the similarity function f gave the pairwise cosine similarities, with an entry A_{ij} set to zero if neither i was one of the top k nearest-neighbors of j nor the reverse. Thresholding the affinity matrix in this manner is very useful, as spectral methods empirically work much better when there are zeros in the affinity matrix for pairs of items that are not in the same class. For our experiments, we chose $k = 20$; however, one may learn the optimal k in the same manner that [Ng *et al.*, 2002] learn the optimal scale factor σ .

3.3 Empirical Results

We compared the spectral learning algorithm in Figure 1 to k-means on 4 data sets:

- 20 NEWSGROUPS a collection of approximately 1000 postings from each of 20 usenet newsgroups.²
- 3 NEWSGROUPS 3 of the 20 newsgroups: sci.crypt, talk.politics.mideast, and soc.religion.christian.
- LYMPHOMA gene expression profiles of 96 normal and malignant lymphocyte samples. There are 2 classes: Diffuse Large B-Cell Lymphoma (42 samples), and Non-DLCBL (54 samples) [Alizadeh, 2000].

²From <http://www.ai.mit.edu/~jrennie/20Newsgroups/>; a total of 18828 documents. Documents were stripped of headers, stopwords, and converted to lowercase. All numbers were discarded. All words that occur in more than 150 or less than 2 documents were removed.

- SOYBEAN is the SOYBEAN-LARGE data set from the UCI repository. 15 classes.³

The results are shown in Table 2. The numbers reported are adjusted Rand Index values [Hubert and Arabie, 1985] for the clusters output by the algorithms. The Rand Index is frequently used for evaluating clusters, and is based on whether pairs are placed in the same or different clusters in two partitionings. The Adjusted Rand Index ranges from -1 to 1 , and its key property is that the expected value for a random clustering is 0 . The result that spectral methods generally perform better than k -means is consistent with the results in [Ng *et al.*, 2002; Brew and Schulte im Walde, 2002]. In some cases, the poor performance of k -means reflects its inability to cope with noise dimensions (especially in the case of the text data) and highly non-spherical clusters (in the case of the composite negative cluster for LYMPHOMA).⁴ However, spectral learning outperforms k -means on the SOYBEAN dataset as well, which is a low-dimensional, multi-class data set.

4 Spectral Classification

In the previous section, we described *clustering* a data set by creating a Markov chain based on the similarities of the data items with one another, and analyzing the dominant eigenvectors of the resulting Markov matrix. In this section, we show how to *classify* a data set by making two changes. First, we modify the Markov chain itself by using class labels, when known, to override the underlying similarities. Second, we use a classification algorithm in the spectral space rather than a clustering algorithm.

4.1 Modifying the “Interested Reader” Model

The model described in Section 2 can be modified to incorporate labeled data in the following simple manner. If the interested reader happens to be at a labeled document, the probability that she will choose another labeled document of the same category is high, while the probability that she will choose a labeled document of a different category is low (or zero). Transition probabilities to unlabeled documents are still proportional to their similarity to the current source document, whether the current document is labeled or not.

We wish to create a Markov matrix that reflects this modified model. We propose doing this in the following manner, using the normalization introduced in Section 3. For most similarity functions, the maximum pairwise similarity value is 1 , and the minimum similarity is 0 . Therefore, we would like to say that two points in the same class are maximally similar, and two points in different classes are minimally similar:

³There are 562 instances, 35 features, and 15 different classes. It is nominal; Hamming distance was used.

⁴For some of these sets, the k -means numbers are low. This is partially illusory, due to the zeroed expectation of the adjusted Rand index, and partially a real consequence of the sparse high-dimensionality of the text data. Better k -means results on text typically require some kind of aggressive dimensionality reduction, (usually LSA, another spectral method) or careful feature selection (or both).

1. Define the affinity matrix A as in the previous algorithms.
2. First, for each pair of points (i, j) that are in the same class, assign the values $A_{ij} = A_{ji} = 1$.
3. Likewise, for each pair of points (i, j) that are in different classes, assign the values $A_{ij} = A_{ji} = 0$.
4. Normalize $N = \frac{1}{d_{\max}}(A + d_{\max}I - D)$.

This gives us a symmetric Markov matrix describing the “interested reader” process which uses supervisory information when present, and data similarities otherwise. A strength of this model lies in the fact that it incorporates unlabeled data, whereas the majority of classification models deal strictly with the labeled data. A benefit of additive normalization is that, after the affinities are adjusted, same-labeled pairs will always have a higher (or equal) mutual transition probability than unlabeled pairs. This will not necessarily be the case with other normalization schemes.

4.2 A Spectral Classification Algorithm

Again, if natural classes occur in the data, the Markov chain described above should have cliques. Furthermore, the cliques will become stronger as the number of labeled documents increases. Given this model, we wish to categorize documents by assigning them to the appropriate clique in the Markov chain. The spectral clustering methods given in Section 3 can be adapted to do classification by replacing the final few steps (clustering in spectral space) with the steps shown in Figure 1 (which classify in spectral space).

The key differences between the spectral classifier and the clustering algorithm are (a) that our transition matrix A incorporates labeling information, and (b) we use a classifier in the spectral space rather than a clustering method. What is novel here is that this algorithm is able to classify documents by the similarity of their transition probabilities to known subsets of B . Because the model incorporates both labeled and unlabeled data, it should improve not only with the addition of labeled data, but also with the addition of unlabeled data. We observe this empirically in Section 4.3.

4.3 Empirical Results

Cliques

It was suggested in Section 4.2 that the Markov chain described above will have cliques, that is, subsets of nodes in the graph that are internally fast-mixing, but are not mutually fast-mixing. Figure 4 shows the thresholded sparsity pattern for the affinity matrices for the 3 Newsgroups data set, as labeled data is added. The left matrix is the affinity matrix for 1% labeled data. Even the underlying similarities show block-like behavior, if weakly. To the extent that the unlabeled data gives a block-like affinity matrix, clusters naturally exist in the data; this is the basis for spectral clustering. The subsequent matrices have increasing fractions of data labeled. The effect of adding labeled data is to sharpen and coerce the natural clusters into the desired classes. As more labels are added, the blocks become clearer, the cliques become stronger, and, in the limit of 100% labeled data, the interested reader will never accidentally jump from a document of one topic to one of another.

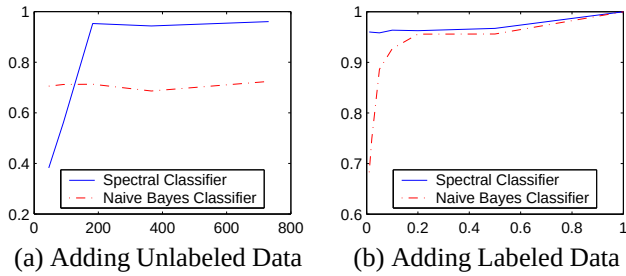


Figure 2: Categorization accuracy on the 3 NEWSGROUPS task as the number of (a) unlabeled and (b) labeled points increases. In (a), 12 labeled documents and the given number of unlabeled documents were used as a training set. In (b), the training set is all of 3 NEWSGROUPS, with the given fraction labeled. In both cases, the test set for a given run consisted of all documents in 3 NEWSGROUPS whose labels were not known during training for that run.

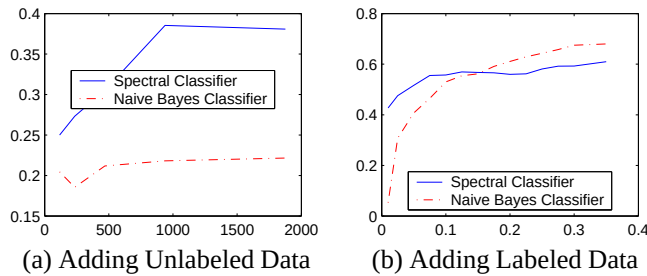


Figure 3: Categorization accuracy on the 20 NEWSGROUPS task as the (a) amount of unlabeled data and (b) fraction of labeled data increases.

Accuracies

To validate the utility of spectral classification, we performed the following experiments on the 20 NEWSGROUPS data set.

We built two batch classifiers. The first was a standard multinomial Naive Bayes (NB) classifier with standard add-one smoothing. The second was a spectral classifier as described above, which used a single-nearest neighbor classifier to perform the final classification in the spectral space. The affinity matrix A was the thresholded cosine similarity between documents.⁵ We note here that the thresholding is important, since it weakens the effect of outliers. Furthermore, it saves space and computation time, since the resulting affinity matrix is sparse.

We split the data into a labeled training set and an unlabeled test set. For classification, the spectral classifier processed both the training and test set, but was evaluated on the test set only.

Figure 2(a) shows the effect of using a sample of 12 documents from the 3 NEWSGROUPS data as a labeled training set, with an increasing number of unlabeled documents as a test set. The accuracy of the NB classifier is, of course, constant up to sampling variation, since it discards unlabeled data. The spectral classifier is more accurate than the NB classifier when given sufficiently many additional unlabeled

⁵For each document, we take the most similar 20 documents, and put those similarities in the appropriate row and column. All other entries are 0.

documents to incorporate.

Figure 2(b) shows the effect of supplying increasingly large fractions of the 3 NEWSGROUPS data set as labeled training instances, and using the remainder of the data set as test instances. The spectral classifier outperforms Naive Bayes, more substantially so when there is little labeled data. Figures 3(a) and (b) show the same graphs for the 20 NEWSGROUPS data set. Again, spectral classification performs well, especially when less than 10% of the data is labeled. It should be noticed that, for this data set, Naive Bayes outperforms the spectral classifier in the strongly supervised case ($>15\%$ labeled data). The strength of Spectral Learning lies in incorporating unlabeled data, and, for the strongly supervised case, this is of less value.

Spectral Space

To further investigate the behavior of the spectral classifier, we performed the following experiment. We took the 3 NEWSGROUPS data and labeled various fractions of each of the 3 classes. We then plotted each document’s position in the resulting 3-dimensional spectral space (the space of the rows of the matrix X as defined by our spectral learning algorithm). Figure 4 shows dimensions 2 and 3 of these plots. With no labels, the data does not tightly cluster. As we add labels (circled points), two things happen. First, the labeled points move close to same-labeled points, away from different-labeled points, and generally towards the outside, since they are “hubs” of the Markov process. Second, they pull the unlabeled points out radially along with them. This is effective in that it seems to pull the classes apart, even though the classes were not naturally very strong clusters in the unlabeled data.

4.4 Related Work

In [Yu and Shi, 2001], a spectral grouping method, which they call “Grouping with Bias”, is presented that allows for top-level bias, as in labeled data. They formulate the problem as a constrained optimization problem, where the optimal partition is sought, subject to the constraint that the normalized cut values of any two nodes that are preassigned to the same class should be the same.

The main drawback with the algorithm in [Yu and Shi, 2001] is that it only constrains same-labeled data points. The algorithm we present here benefits from the zeros in the sparsity pattern introduced by differently-labeled pairs. Furthermore, it should be noted that, in the multi-class case, labeled sets combinatorially tend to embody more differently-labeled pairs than same-labeled pairs. The other drawback to not using the information given by differently-labeled points is that the trivial partition (all points in one cluster) will satisfy the constraints, even when many points are labeled. In fact, when all the data is labeled, it is likely that the partition found by the Grouping with Bias algorithm will be the trivial partition. Figure 6(a) shows that our Spectral Classifier outperforms the Grouping with Bias algorithm for the 3 NEWSGROUPS data set. In fact, Grouping with Bias started performing slightly worse when a large fraction of the data was labeled.

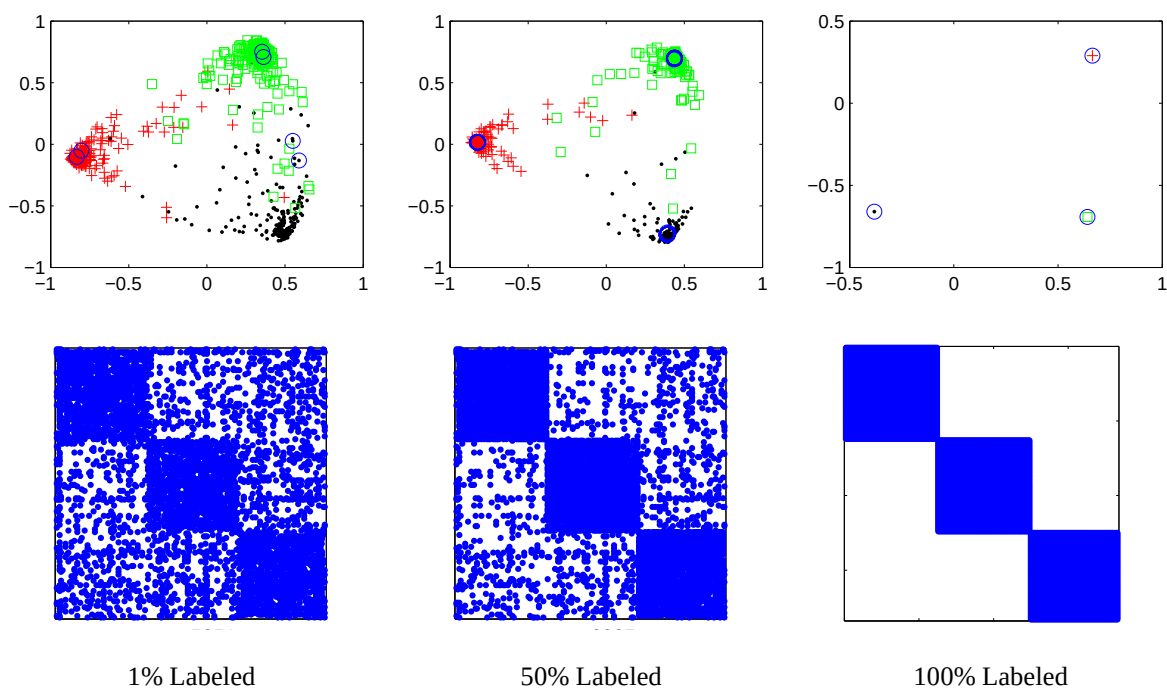


Figure 4: Three classes of the 20 NEWSGROUPS data set in spectral space with increasing amounts of *labeled* data. The classes are sci.crypt (pluses), talk.politics.mideast (dots), and soc.religion.christian (squares). The labeled points are circled. The bottom graphs show the sparsity patterns of the associated affinity matrices.

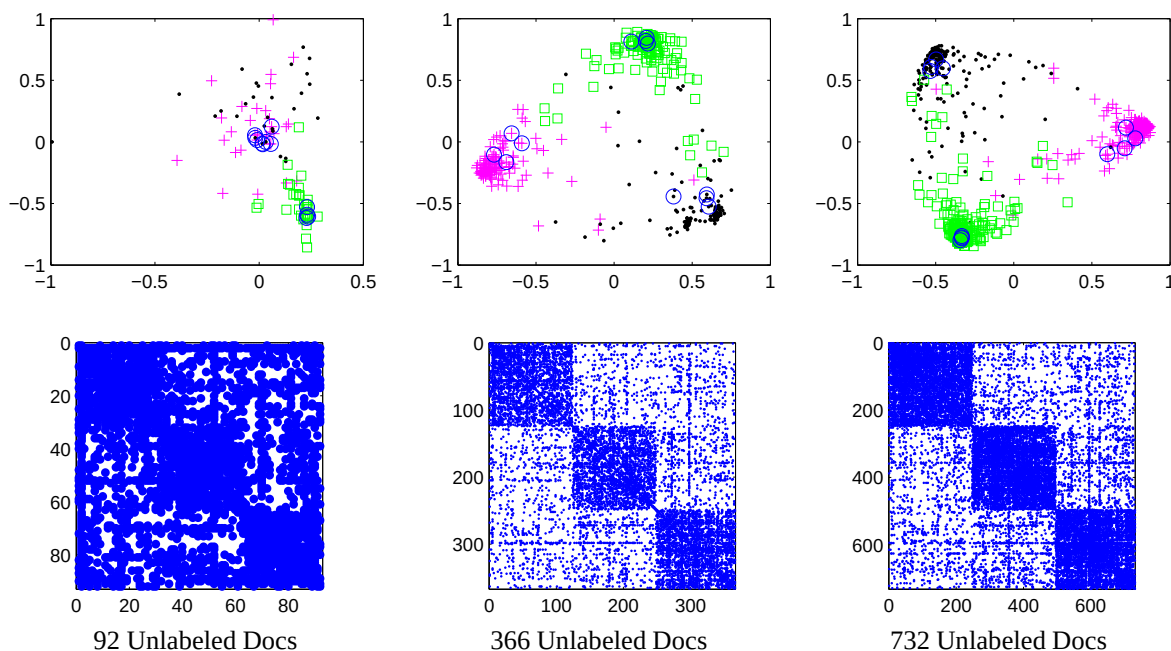


Figure 5: Three classes of the 20 NEWSGROUPS data set in spectral space with increasing amounts of *unlabeled* data. The classes are sci.crypt (pluses), talk.politics.mideast (dots), and soc.religion.christian (squares). There are 12 labeled documents (circled).

5 Constrained Spectral Clustering

Recently, there has been interest in *constrained clustering* [Wagstaff and Cardie, 2000; Klein *et al.*, 2002], which involves clustering with two types of pairwise constraints:

1. *Must-links*: two items are known to be in the same class.
2. *Cannot-links*: two items are in different classes.

Constrained clustering allows one to do exploratory data analysis when one has some prior knowledge, but not class labels. The classifier presented in section 4 can be easily modi-

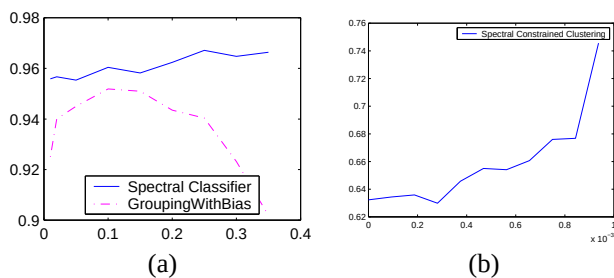


Figure 6: (a) Classification accuracy vs. fraction of pairs constrained for spectral classifier vs. grouping with bias. (b) Rand Index of spectral constrained clustering vs. fraction of pairs constrained (shown up to 0.1% constrained). Both are on 3 NEWSGROUPS data.

fied for this kind of prior knowledge: we have been reducing labeling information to pairwise information during affinity modification all along.

Specifically, the affinity matrix is now defined as follows:

1. Define the affinity matrix A as before.
2. For each pair of must-linked points (i, j) assign the values $A_{ij} = A_{ji} = 1$.
3. For each pair of cannot-linked points (i, j) that are in different classes, assign the values $A_{ij} = A_{ji} = 0$.
4. Normalize $N = \frac{1}{d_{\max}}(A + d_{\max}I - D)$.

This is equivalent to the *imposing constraints* step in [Klein *et al.*, 2002]. In this step, must-linked points are made more similar than any other pair of points in the data set, and cannot-linked points are made more dissimilar than any pair of points in the data set.

The spectral constrained clustering algorithm proceeds just as the other spectral clustering algorithms presented here; the only difference is that it uses the modified normalized affinity matrix presented in this section.

Figure 6(b) shows a plot of accuracy vs. number of constraints for the 3 NEWSGROUPS data set using spectral constrained clustering. The accuracy is measured by the constrained Rand index [Klein *et al.*, 2002; Wagstaff and Cardie, 2000]. The accuracy increases with number of constraints, showing that the spectral constrained clustering can effectively use constraints to better cluster data.

6 Conclusion

We present here a probabilistic model and an associated spectral learning algorithm that is able to work for the unsupervised, semi-supervised, and fully-supervised learning problems. We show that this algorithm is able to cluster well, and further is able to effectively utilize prior knowledge, either given by pairwise constraints or by labeled examples.

Acknowledgments

This paper is based on research supported in part by the National Science Foundation under Grant No. IIS-0085896, and by the Research Collaboration between NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation and CSLI, Stanford University (research project on Concept Bases for Lexical Acquisition and Intelligently Reasoning with Meaning).

References

- [Alizadeh, 2000] A.A. Alizadeh. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–11, 2000.
- [Brew and Schulte im Walde, 2002] C. Brew and S. Schulte im Walde. Spectral clustering for german verbs. In *Proceedings of EMNLP-2002*, 2002.
- [Chung, 1997] F. Chung. *Spectral Graph Theory*. AMS, Providence, RI, 1997.
- [Deerwester *et al.*, 1990] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
- [Fiedler, 1975] M. Fiedler. A property of eigenvectors of non-negative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25:619–672, 1975.
- [Hubert and Arabie, 1985] L. J. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.
- [Klein *et al.*, 2002] Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *The Nineteenth International Conference on Machine Learning*, 2002.
- [Kleinberg, 1998] J. Kleinberg. Authoritive sources in a hyper-linked environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.
- [Meila and Shi, 2000] Marina Meila and Jianbo Shi. Learning segmentation by random walks. In *Advances in Neural Information Processing Systems*, pages 873–879, 2000.
- [Meilă and Shi, 2001] Marina Meilă and Jianbo Shi. A random walks view of spectral segmentation. In *AI and Statistics (AISTATS)*, 2001.
- [Ng *et al.*, 2002] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of Advances in Neural Information Processing Systems (NIPS 14)*, 2002.
- [Page *et al.*, 1998] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
- [Salton, 1989] G. Salton. *Automatic Text Processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley, 1989.
- [Wagstaff and Cardie, 2000] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *The Seventeenth International Conference on Machine Learning*, pages 1103–1110, 2000.
- [Yu and Shi, 2001] Stella Yu and Jianbo Shi. Grouping with bias. Technical Report CMU-RI-TR-01-22, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2001.

SVMC: Single-Class Classification With Support Vector Machines

Hwanjo Yu

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801 USA
hwanjoyu@uiuc.edu

Abstract

Single-Class Classification (SCC) seeks to distinguish one class of data from the universal set of multiple classes. We present a new SCC algorithm that efficiently computes an accurate boundary of the target class from positive and unlabeled data (without labeled negative data).

1 Introduction

Single-Class Classification (SCC) seeks to distinguish one class of data from the universal set of multiple classes. (e.g., distinguishing apples from fruits, identifying “waterfall” pictures from image databases, or classifying personal homepages from the Web) (Throughout the paper, we call the target class *positive* and the complement set of samples *negative*.)

Since it is not natural to collect the “non-interesting” objects (i.e., negative data) to train the concept of the “interesting” objects (i.e., positive data), SCC problems are prevalent in real world where positive and unlabeled data are widely available but negative data are hard or expensive to acquire [Yu *et al.*, 2002; Letouzey *et al.*, 2000; DeComite *et al.*, 1999]. For example, in text or Web page classification (e.g., personal homepage classification), collecting negative training data (e.g., a sample of “non-homepages”) is delicate and arduous because manually collected negative data could be easily biased because of a person’s unintentional prejudice, which could be detrimental to classification accuracy. In an example of diagnosis of a disease, positive data are easy to access (e.g., all patients who have the disease) and unlabeled data are abundant (e.g., all patients), but negative data are expensive if detection tests for the disease are expensive since all patients in the database cannot be assumed to be negative samples if they have never been tested. Further applications can be also found in pattern recognition, image retrieval, classification for data mining, rare class classification, etc. In this paper, we focus on this SCC problem from positive and unlabeled data (without labeled negative data).

1.1 Previous Approaches for SCC

Traditional (semi-)supervised learning schemes are not suitable for SCC without labeled negative data because: (1) the portions of positive and negative spaces are seriously unbalanced without being known (i.e., $Pr(P) \ll Pr(\bar{P})$), and

(2) the absence of negative samples in the labeled data set makes unfair the initial parameters of the model and thus it leads to unfair guesses for the unlabeled data.

Active learning methods also try to minimize the labeling labors to construct an accurate classification function by a different approach that involves an interactive process between the learning system and users [Tong and Koller, 2000].

Valiant in 1984 [Valiant, 1984] pioneered *learning theory from positive examples* based on rule learning. In 1998, F. Denis defined the Probably Approximately Correct (PAC) learning model for positive and unlabeled examples, and showed that k -DNF (Disjunctive Normal Form) is learnable from positive and unlabeled examples [Denis, 1998]. After that, some experimental attempts to learn using positive and unlabeled data have been tried using k -DNF or C4.5 [Letouzey *et al.*, 2000; DeComite *et al.*, 1999]. Rule learning methods are simple and efficient for learning nominal features but tricky to use for the problems of continuous features, high dimensions, or sparse instance spaces.

Positive Example-Based Learning (PEBL) framework was proposed for Web page classification [Yu *et al.*, 2002]. Their method is limited to the Web domain with binary features, and its training efficiency is poor because of using SVM iteratively whose training time is already at least quadratic to the size of training data set. This problem becomes critical when the size of unlabeled data set is large.

A probabilistic method for the SCC problem has been recently proposed for the text domain [Liu *et al.*, 2002]. As they specified in the paper, their method – a revision of the EM algorithm – performs badly on “hard” problems due to the fundamental limitations of the generative model assumption, the attribute independence assumption which results in linear separation, and the requirement of good estimation of prior probabilities.

OSVM (One-Class SVM) also distinguishes one class of data from the rest of the feature space given only a positive data set [Tax and Duin, 2001; Manevitz and Yousef, 2001]. Based on a strong mathematical foundation, OSVM draws a nonlinear boundary of the positive data set in the feature space using two parameters – ν (to control the noise in the training data) and γ (to control the “smoothness” of the boundary). They have the same advantages as SVM, such as efficient handling of high dimensional spaces and systematic nonlinear classification using advanced kernel functions.

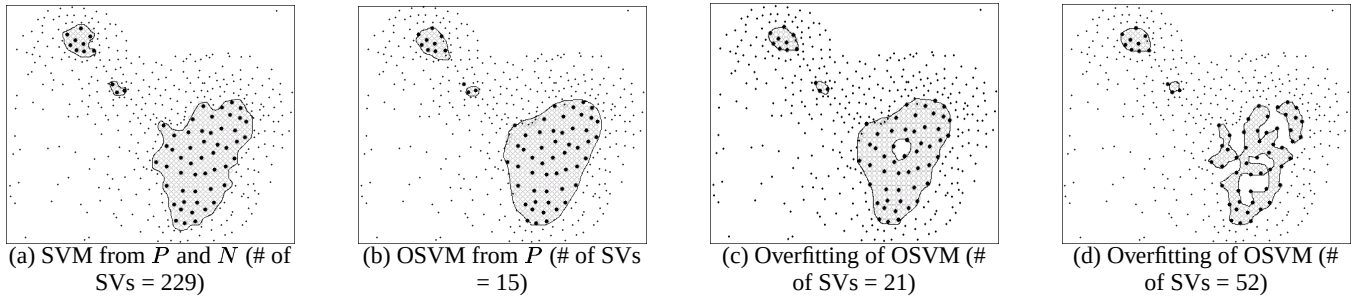


Figure 1: Boundaries of SVM and OSVM on a synthetic data set. *big dots: positive data, small dots: negative data*

However, OSVM requires a much larger amount of positive training data to induce an accurate class boundary because its support vectors (SVs) of the boundary only comes from the positive data set and thus the small number of positive SVs can hardly cover the major directions of the boundary especially in high dimensional spaces. Due to the SVs coming only from positive data, OSVM tends to overfit and underfit easily. Tax proposed a sophisticated method which uses artificially generated unlabeled data to optimize the OSVM's parameters that "balance" between overfitting and underfitting [Tax and Duin, 2001]. However, their optimization method is infeasibly inefficient in high dimensional spaces, and even with the best parameter setting, its performance still lags far behind the original SVM with negative data due to the shortage of SVs which makes "incomplete" the boundary description. Figure 1(a) and (b) show the boundaries of SVM and OSVM on a synthetic data set in a two-dimensional space. (We used LIBSVM version 2.33¹ for SVM implementation.) In this low-dimensional space with "enough" data, the ostensibly "smooth" boundary of OSVM is not the result of the good generalization but instead is from the poor expressibility caused by the "incomplete" SVs, which will become much worse in high-dimensional spaces where more SVs around the boundary are needed to cover major directions in the high-dimensional spaces. When we increase the number of SVs in OSVM, it overfits rather than being more accurate as shown in Figure 1(c) and (d).

1.2 Contributions and Paper Layout

We first discuss the "optimal" SCC boundary, which motivates our new SCC framework *Mapping-Convergence (MC)*, where the algorithms under the MC framework generate the boundary close to the optimum (Section 2). In Section 3, we present an efficient SCC algorithm *Support Vector Mapping Convergence (SVMC)* under the MC framework. We prove that although SVMC iterates under the MC framework for the "near-optimal" result, its training time is independent of the number of iterations, which is asymptotically equal to that of a SVM. We empirically verify our analysis of SVMC by extensive experiments on various domains of real data sets such as text classification (e.g., Web page classification), pattern recognition (e.g., letter recognition), and bioinformatics (e.g., diagnosis of breast cancer), which shows the outstanding performance of SVMC in a wide spectrum of SCC prob-

lems (with nominal or continuous attributes, linear or nonlinear separation, and low or high dimensions) (Section 4).

1.3 Notation

We use the following notation throughout this paper.

- x is a data instance such that $x \in \mathcal{U}$.
- $\mathcal{P}$ is a subspace for positive class within $\mathcal{U}$, from which positive data set P is sampled.
- $\mathcal{U}$ (unlabeled data set) is a uniform sample of the universal set.
- $\mathcal{U}$ is the feature space for the universal set such that $\mathcal{U} \subseteq \mathbb{R}^m$ where m is the number of dimensions.

For an example of Web page classification, the universal set is the entire Web, $\mathcal{U}$ is a uniform sample of the Web, P is a collection of Web pages of interest, and $x \in \mathbb{R}^m$ is an instance of a Web page.

2 Mapping Convergence (MC) Framework

2.1 Motivation

In machine learning theory, the "optimal" class boundary function (or hypothesis) $h(x)$ given a limited number of training data set $\{(x, l)\}$ (l is the label of x) is considered the one that gives the best *generalization* performance which denotes the performance on "unseen" examples rather than on the training data. The performance on the training data is not regarded as a good evaluation measure for a hypothesis because the hypothesis ends up *overfitting* when it tries to fit the training data too hard. (When a problem is easy (to classify) and the boundary function is complicated more than it needs to be, the boundary is likely overfitted. When a problem is hard and the classifier is not powerful enough, the boundary becomes underfit.) SVM is an excellent example of supervised learning that tries to maximize the generalization by maximizing the *margin* and also supports nonlinear separation using advanced kernels, by which SVM tries to avoid overfitting and underfitting [Burges, 1998].

The "optimal" SCC classifier without labeled negative data also needs to maximize the generalization somehow with highly expressive power to avoid overfitting and underfitting. To illustrate an example of the "near-optimal" SCC boundary without labeled negative data, consider the synthetic data set (Figure 2) simulating a real situation where within $\mathcal{U}$, (1) the universal set is composed of multiple groups of data, (2)

¹<http://www.csie.ntu.edu.tw/~cjlin/libsvm>

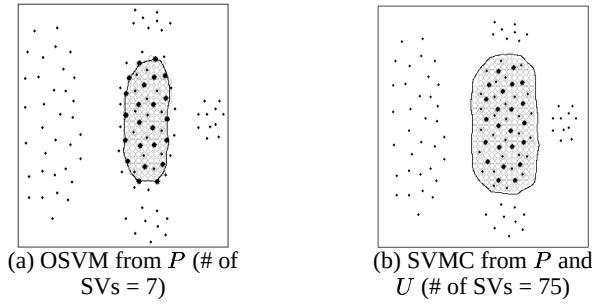


Figure 2: Synthetic data set simulating a real situation. P : big dots, U : all dots (big and small dots)

the positive class $\mathcal{P}$ is one of them (supposing $\mathcal{P}$ is the data group in the center), and (3) the positive data set P is a sample from $\mathcal{P}$ (assuming that the big dots are the sample P). OSVM draws $\hat{\mathcal{P}}$, a tight boundary around P , as shown in Figure 2(a), which overfits the true class area $\mathcal{P}$ due to the absence of the knowledge of the distribution of U . However, the “near-optimal” SCC classifiers must locate the boundary between $\mathcal{P}$ and U outside $\mathcal{P}$ (Figure 2(b)) and thus maximize the generalization. The MC framework using U systematically draws the boundary of Figure 2(b).

2.2 Negative Strength

Let $h(x)$ be the boundary function of the positive class in $\mathcal{U}$, which outputs the distance from the boundary to the instance x in $\mathcal{U}$ such that

$$\begin{aligned} h(x) &> 0 && \text{if } x \text{ is a positive instance,} \\ h(x) &< 0 && \text{if } x \text{ is a negative instance,} \\ |h(x)| &> |h(x')| && \text{if } x \text{ is located farther than } x' \\ &&& \text{from the boundary in } \mathcal{U}. \end{aligned}$$

Definition 1 (Strength of negative instances). For two negative instances x and x' such that $h(x) < 0$ and $h(x') < 0$, if $|h(x)| > |h(x')|$, then x is **stronger** than x' .

Example 1. Consider a resume page classification function $h(x)$ from the Web ($\mathcal{U}$). Suppose there are two negative data objects x and x' (non-resume pages) in $\mathcal{U}$ such that $h(x) < 0$ and $h(x') < 0$: x is “how to write a resume” page, and x' is “how to write an article” page. In $\mathcal{U}$, x' is considered more distant from the boundary of the resume class because x has more relevant features to the resume class (e.g., the word “resume” in text) though it is not a true resume page.

2.3 MC Framework

The MC framework is composed of two stages: the *mapping stage* and the *convergence stage*. In the mapping stage, the algorithm uses a weak classifier Ψ_1 , which draws an initial approximation of “strong negatives” – the negative data located far from the boundary of the positive class in $\mathcal{U}$ (steps 1 and 2 in Figure 4). Based on the initial approximation, the convergence stage runs in iteration using a second base classifier Ψ_2 , which maximizes the margin to make a progressively better approximation of negative data (steps 3 through 5 in Figure 4). Thus the class boundary eventually converges to

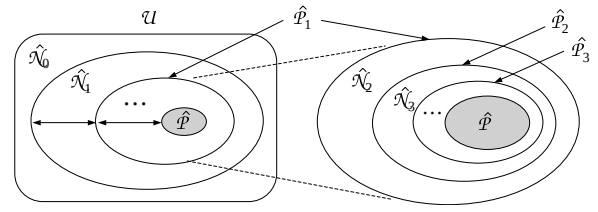


Figure 3: Example of the spaces of the MC framework in $\mathcal{U}$

the boundary around the positive data set in the feature space which also maximizes the margin.

Input: - positive data set P , unlabeled data set U
Output: - a boundary function h_i

Ψ_1 : an algorithm identifying “strong negatives” from U
 Ψ_2 : a supervised learning algorithm that maximizes the margin

Algorithm:

1. Use Ψ_1 to construct a classifier h_0 from P and U which classifies only “strong negatives” as negative and the others as positive
2. Classify U by h_0
 - * $\hat{N}_0 :=$ examples from U classified as negative by h_0
 - * $\hat{P}_0 :=$ examples from U classified as positive by h_0
3. Set $N := \emptyset$ and $i := 0$
4. Do loop
 - 4.1. $N := N \cup \hat{N}_i$
 - 4.2. Use Ψ_2 to construct h_{i+1} from P and N
 - 4.3. Classify $\hat{P}_i$ by h_{i+1}
 - * $\hat{N}_{i+1} :=$ examples from $\hat{P}_i$ classified as negative by h_{i+1}
 - * $\hat{P}_{i+1} :=$ examples from $\hat{P}_i$ classified as positive by h_{i+1}
 - 4.4. $i := i + 1$
 - 4.5. Repeat until $\hat{N}_i = \emptyset$
5. return h_i

Figure 4: MC framework

Assume that $\hat{\mathcal{P}}$ is a subspace tightly subsuming P within $\mathcal{U}$ where the class of the boundary function for $\hat{\mathcal{P}}$ is from the algorithm Ψ_2 (e.g., SVM). In Figure 4, let $\hat{N}_0$ be the negative space and $\hat{P}_0$ be the positive space within $\mathcal{U}$ divided by h_0 (a boundary drawn by Ψ_1), and let $\hat{N}_i$ be the negative space and $\hat{P}_i$ be the positive space within $\hat{P}_{i-1}$ divided by h_i (a boundary drawn by Ψ_2). Then, we can induce the following formulae from the MC framework of Figure 4. (Figure 3 illustrates an example of the spaces of the framework in $\mathcal{U}$.)

$$\mathcal{U} = \hat{\mathcal{P}} + \bigcup_{k=0}^i \hat{N}_k \quad (1)$$

$$\hat{\mathcal{P}}_i = \hat{\mathcal{P}} + \bigcup_{k=i+1}^I \hat{N}_k \quad (2)$$

where I is the number of iterations in the MC framework.

Theorem 1 (Boundary Convergence). Suppose U is uniformly distributed in $\mathcal{U}$. If algorithm Ψ_1 does not generate

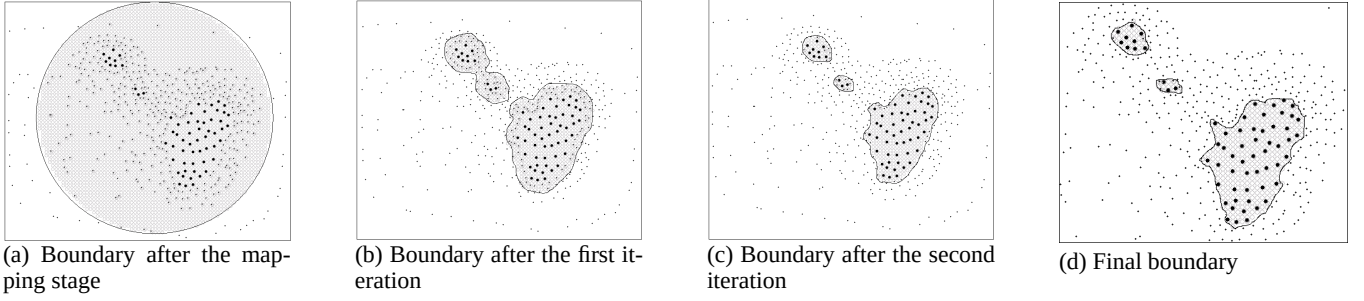


Figure 5: Intermediate results of SVMC

false negatives, and algorithm Ψ_2 maximizes margin, then (1) the class boundary of the MC framework converges into the boundary that maximally separates P and U outside $\hat{\mathcal{P}}$, and (2) I (the number of iterations) is logarithmic in the margin between $\hat{\mathcal{N}}_0$ and $\hat{\mathcal{P}}$.

Proof. $\hat{\mathcal{N}}_0 \cap \hat{\mathcal{P}} = \emptyset$ because a classifier h_0 constructed by the algorithm Ψ_1 does not generate false negative. A classifier h_1 constructed by the algorithm Ψ_2 , trained from the separated space $\hat{\mathcal{N}}_0$ and $\hat{\mathcal{P}}$, divides the rest of the space ($\mathcal{U} - (\hat{\mathcal{N}}_0 + \hat{\mathcal{P}})$) which is equal to $\cup_{k=1}^I \hat{\mathcal{N}}_k$ into two classes with a boundary that maximizes the margin between $\hat{\mathcal{N}}_0$ and $\hat{\mathcal{P}}$. The first part becomes $\hat{\mathcal{N}}_1$ and the other becomes $\cup_{k=2}^I \hat{\mathcal{N}}_k$. Repeatedly, a classifier h_{i+1} constructed by the same algorithm Ψ_2 , trained from the separated space $\cup_{k=0}^i \hat{\mathcal{N}}_k$ and $\hat{\mathcal{P}}$, divides the rest of the space $\cup_{k=i+1}^I \hat{\mathcal{N}}_k$ into $\hat{\mathcal{N}}_{i+1}$ and $\cup_{k=i+2}^I \hat{\mathcal{N}}_k$ with equal margins. Thus, $\hat{\mathcal{N}}_{i+1}$ always has the margin of half of $\hat{\mathcal{N}}_i$ (for $i \geq 1$). Therefore, I will be logarithmic in the margin between $\hat{\mathcal{N}}_0$ and $\hat{\mathcal{P}}$.

The iteration stops when $\hat{\mathcal{N}}_i = \emptyset$, where there exists no sample of U outside $\hat{\mathcal{P}}$. Therefore, the final boundary will be located between P and U outside $\hat{\mathcal{P}}$ while maximizing the margin between them. $\square$

Theorem 1 proves that under certain conditions, the final boundary will be located between P and U outside $\hat{\mathcal{P}}$. However, in the example of Figure 2(b), our framework generates the “better” boundary located between P and U outside $\mathcal{P}$ because in theorem 1, we made a somewhat strong assumption, i.e., U is uniformly distributed, to guarantee the boundary convergence. In a more realistic situation where there is some distance δ between classes—Figure 2 shows some gaps between classes—if the margin between h_{i+1} and $\hat{\mathcal{N}}_i$ becomes smaller than δ at some iteration, the convergence stops because $\hat{\mathcal{N}}_{i+1}$ becomes empty. The margin between h_{i+1} and $\hat{\mathcal{N}}_i$ reduces by half at each iteration as the boundary h_{i+1} approaches to $\hat{\mathcal{P}}$ and thus the boundary is not likely to stop converging when it is far from $\hat{\mathcal{P}}$ unless U is severely sparse. Thus, we have the following claim:

Claim 1. *The boundary of MC is located between P and U outside $\mathcal{P}$ if U and P are not severely sparse and there exists visible gaps between $\mathcal{P}$ and U .*

Validity of the component algorithms Ψ_1 and Ψ_2

1. Ψ_1 must not generate false negatives.

Most classification methods have a threshold to control the trade-off between precision and recall. We can adjust the threshold of Ψ_1 so that it makes near 100% recall by sacrificing precision. (Some violations of this can be handled by the soft constraint of Ψ_2 (e.g., SVM).) Determining the threshold can be intuitive or automatic when not concerning the precision quality much. The precision quality of Ψ_1 does not affect the accuracy of the final boundary as far as it approximates a certain amount of negative data because the boundary will converge eventually. Figure 5 visualizes the boundary after each iteration of SVMC. The mapping stage only identifies very strong negatives by covering a wide area around the positive data (Figure 5(a)). (We used OSVM for the algorithm Ψ_1 of the mapping stage. We intuitively set the parameters of OSVM such that it covers all the positive data without much concern for false positives.) Although the precision quality of mapping is poor, the boundary at each iteration converges (Figures 5(b) and (c)), and the final boundary is very close to the true boundary drawn by SVM with P and N (Figure 1(a) and 5(d)). Our experiments in Section 4 also show that the final boundary becomes very accurate although the initial boundary of the mapping stage is very rough by the “loose” setting of the threshold of Ψ_1 .

2. Ψ_2 must maximize margin.

SVM and Boosting are currently the most popular supervised learning algorithms that maximize the margin. With a strong mathematical foundation, SVM automatically finds the optimal boundary without a validation process and without many parameters to tune. The small numbers of theoretically motivated parameters also work well for an intuitive setting. For these reasons, we use SVM for Ψ_2 for our research. In practice, the soft constraint of SVM is necessary to cope with noise or outliers. The soft constraint of SVM can affect I and the accuracy of the final boundary. However, P is not likely to have a lot of noise in practice because it is usually carefully collected by users. In our experiments, a low setting (i.e., $\nu = 0.01$) of ν (the parameter to control the rate of noise in the training data) performs well for all cases for this reason. (We used ν -SVM for the semantically meaningful parameter [Chang and Lin, 2001].)

3 Support Vector Mapping Convergence (SVMC)

3.1 Motivation

The classification time of the final boundary of SMC (“Simple” MC with $\Psi_2 = SVM$) is equal to that of SVM because the final boundary is a boundary function of Ψ_2 . The training time of SMC can be very long if $|U|$ is very large because the training time of SVM highly depends on the size of data set n ($\approx |U|$), and SMC runs iteratively. $t_{SMC} = O(|U|^2 * \log|U|)$ assuming the number of iterations $\approx \log|U|$ and $t_{SVM} = O(|U|^2)$ where t_Ψ is the training time of a classifier Ψ . (t_{SVM} is known to be at least quadratic to n and linear to the number of dimensions). Refer to [Chang and Lin, 2001] for more discussion on the complexity of SVM. However, decreasing the sampling density of U to reduce the training time hurts the accuracy of the final boundary because the density of U will directly affect the quality of the SVs of the final boundary.

3.2 SVMC

SVMC prevents the training time from increasing dramatically as the sample size grows. We prove that although SVMC iterates under the MC framework for the “near-optimal” result, its training time is independent of the number of iterations, and thus its training time is asymptotically equal to that of a SVM.

The approach of SVMC is to use minimally required data set at each iteration such that the data set does not degrade the accuracy of the boundary while it saves the training time of each SVM maximally. To illustrate how SVMC achieves this, consider the point of starting the third iteration (when $i = 2$) in SMC. (See step 4.1 in Figure 4.) After we merge $\hat{N}_2$ into N , we may not need all the data from N in order to construct h_3 because the data far from h_3 may not contribute to the SVs. The set of negative SVs of h_2 is the representative data set for $\hat{N}_0$ and $\hat{N}_1$, so we only keep the negative SVs of h_2 and the newly induced data set $\hat{N}_2$ to support the negative side of h_3 .

Claim 2 (Minimally required negative data). *Minimally required negative data at $(i + 1)$ th iteration ($\forall i \geq 1$) that makes h_{i+1} as accurate as the boundary constructed from $\cup_{j=0}^i \hat{N}_j$ and P , is $\hat{N}_i \cup$ negative support vectors of h_i .*

Rationale. The negative SVs of h_{i+1} will be from $\hat{N}_i$ and the negative SVs of h_i because $\hat{N}_i$ is the closest data set to h_{i+1} and because the directions not supported by $\hat{N}_i$ in the feature space will be supported by the negative SVs of h_i , which are the representing data set for $\cup_{j=0}^{i-1} \hat{N}_j$. However, if any of the negative SVs of h_i is excluded in constructing h_{i+1} , h_{i+1} might suffer because the negative SVs of h_i need to support the direction that $\hat{N}_i$ does not support in the feature space. Thus, $\hat{N}_i \cup$ negative support vectors of h_i are the minimally required negative data set at $(i + 1)$ th iteration. $\square$

For the minimally required data set for the *positive* side, we cannot definitely exclude any data object from P at each

iteration because positive SVs are determined depending on negative SVs, and it is hard to determine the positive data that cannot be SVs independent of negative SVs or SVM parameters.

Surprisingly, adding the following statement between step 4.4 and 4.5 of the original MC framework of Figure 4 completes the SVMC algorithm.

Reset N with negative SVs of Ψ_2

Theorem 2 (Training time of SVMC). *Suppose $t_{SVM} = O(n^2)$, and $|\hat{N}_{i+1}| = \frac{|\hat{N}_i|}{2}$, $\forall i \geq 1$. Then, $t_{SVMC} = O(n^2)$.*

Proof. For simplicity of the proof, we approximate each value as follows.

$$\begin{aligned} n &= |P| + |\cup_{i=0}^I \hat{N}_i| \approx |U| \\ |\hat{N}_0 \cup \hat{N}_1| &\approx \frac{|U|}{2} \end{aligned}$$

$$\begin{aligned} t_{SVMC} &= \left(\frac{|U|}{2}\right)^2 + \left(\frac{|U|}{4}\right)^2 + \dots + \left(\frac{|U|}{2^I}\right)^2 \\ &= \sum_{i=1}^I \left(\frac{|U|}{2^i}\right)^2 = |U|^2 \frac{1}{3} \left(1 - \frac{1}{4^I}\right) \approx \frac{|U|^2}{3} \\ &= O(n^2) \end{aligned}$$

$\square$

Theorem 2 states that the training complexity of SVMC is asymptotically equal to that of SVM. Our experiments in Section 4 also show that SVMC trains much faster than SMC while it remains the same accuracy. Figure 5 visualizes the boundary after each iteration of SVMC on the same data set of Figure 1.

4 Empirical Results

In this section, we show the empirical verification of our analysis on SVMC by extensive experiments on various domains of real data sets – Web page classification, letter recognition, and diagnosis of breast cancer – which show the outstanding performance of SVMC in a wide spectrum of SCC problems (with nominal or continuous attributes, linear or nonlinear separation, and low or high dimensions).

4.1 Datasets and Methodology

Due to space limitations, we reports only the main results. Our evaluation is based on the $F1$ measure ($F1 = 2pr/(p + r)$, p is precision and r is recall) as was used in [Liu et al., 2002] – one of the most recent works on SCC from positive and unlabeled data². We also report the accuracy.

We used the letter recognition and breast cancer data sets from the UCI machine learning repository³ for direct comparisons with OSVM. (OSVM is often used for letter or digit

²Refer to [Liu et al., 2002] for the justification of using the $F1$ measure for SCC.

³<http://www.ics.uci.edu/~mllearn/MLRepository.html>

Class	P	U	P _U	F1, Accuracy (%)					T-Time (sec.)	
				TSVM	SMC	SVMC	OSVM	SVM_NN	SMC	SVMC
letter 'A'	521	10007	384	0.9929, 99.96	0.9840, 99.91	0.9840, 99.91	0.8457, 99.22	0.0811, 97.26	171.77	45.37
'B'	521	10004	377	0.9651, 99.83	0.9046, 99.50	0.9204, 99.59	0.7207, 98.69	0.0834, 97.58	75.61	14.34
'C'	485	9996	371	0.9860, 99.23	0.9641, 99.82	0.9641, 99.82	0.7354, 98.82	0.0758, 97.55	155.70	29.93
'D'	525	10050	402	0.9820, 99.91	0.9300, 99.63	0.9300, 99.63	0.6921, 98.60	0.0902, 97.52	80.47	16.06
'E'	518	10026	392	0.9798, 99.90	0.9419, 99.70	0.9396, 99.69	0.7112, 98.78	0.1333, 97.64	98.13	21.57
b-cancer	135	259	77	0.9628, 98.87	0.9585, 98.75	0.9585, 98.75	0.6315, 83.32	0.2434, 36.06	0.131	0.025
course	482	4179	448	0.8969, 97.82	0.8259, 96.65	0.8434, 96.89	0.2028, 36.59	0.0880, 89.30	636.70	143.29
faculty	532	4209	592	0.9032, 97.68	0.8420, 95.89	0.8705, 96.58	0.2621, 50.11	0.0168, 86.05	749.63	217.85
student	816	4154	825	0.9240, 97.27	0.8495, 94.29	0.8505, 94.15	0.3384, 32.52	0.0000, 80.14	1181.59	296.50

Table 1: **Performance results.** |P_U|: # of positives in U; T-Time: Training Time

recognition [Tax and Duin, 2001].) We also used Webkb⁴ for Web page classification as used in [Liu *et al.*, 2002] for the indirect comparison with it. We set up the experiment environment in the same way as [Liu *et al.*, 2002], except our setup is more realistic: In [Liu *et al.*, 2002], U is composed of $b\%$ of positives (e.g., student) and samples from another class (e.g., course). Our U is $b\%$ of positives (e.g., student) and the remainder is from all other classes. (Refer to [Liu *et al.*, 2002] for the rest of the data set description.)

4.2 Results

Table 4 shows the performance results. TSVM (Traditional SVM) shows the ideal performance using SVM from P and manually classified N from U . SVM_NN (SVM with Noisy Negatives) is SVM from P , with U as a substitute for N . (U can be thought of as a good approximation of N .) Note that for TSVM, SVM_NN, and MC (SMC and SVMC), we used theoretically motivated fixed parameters without performing explicit optimization or validation process. For OSVM, we thoroughly searched for the best parameters based on the testing data set since optimizing parameters as specified in [Tax and Duin, 2001] is infeasibly inefficient especially in high dimensional spaces.

MC (SMC and SVMC) without labeled negative data show performance close to that of TSVM. SVMC trains much faster than SMC for most data sets. The performance of SMC and SVMC is comparable. (They differ a little because of the soft constraints of SVM and noise in the data.) OSVM performs fairly well on letter recognition and breast cancer (of low dimensionality with large amounts of data) but poor on Webkb (of high dimensionality). SVM_NN suffers from very low $F1$ scores because negative prediction dominates due to many false positives in the training data.

5 Conclusion

We present the MC framework and its instance algorithm SVMC, a new SCC method from positive and unlabeled data (without labeled negative data). SVMC without labeled negative data computes an accurate classification boundary around the positive data using the distribution of unlabeled data in a systematic way.

⁴<http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwwkb/>

References

- [Burges, 1998] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.
- [Chang and Lin, 2001] C.-C. Chang and C.-J. Lin. Training nu-support vector classifiers: Theory and algorithms. *Neural Computation*, 13:2119–2147, 2001.
- [DeComite *et al.*, 1999] F. DeComite, F. Denis, and R. Gilleron. Positive and unlabeled examples help learning. In *Proc. 11th Int. Conf. Algorithmic Learning Theory (ALT'99)*, pages 219–230, Tokyo, Japan, 1999.
- [Denis, 1998] F. Denis. PAC learning from positive statistical queries. In *Proc. 10th Int. Conf. Algorithmic Learning Theory (ALT'99)*, pages 112–126, Otzenhausen, Germany, 1998.
- [Letouzey *et al.*, 2000] F. Letouzey, F. Denis, and R. Gilleron. Learning from positive and unlabeled examples. In *Proc. 11th Int. Conf. Algorithmic Learning Theory (ALT'00)*, pages 11–30, Sydney, Australia, 2000.
- [Liu *et al.*, 2002] B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. In *Proc. 19th Int. Conf. Machine Learning (ICML'02)*, pages 387–394, Sydney, Australia, 2002.
- [Manevitz and Yousef, 2001] L. M. Manevitz and M. Yousef. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.
- [Tax and Duin, 2001] D. M. J. Tax and R. P. W. Duin. Uniform object generation for optimizing one-class classifiers. *Journal of Machine Learning Research*, 2:155–173, 2001.
- [Tong and Koller, 2000] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proc. 17th Int. Conf. Machine Learning (ICML'00)*, pages 999–1006, Stanford, CA, 2000.
- [Valiant, 1984] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.
- [Yu *et al.*, 2002] H. Yu, J. Han, and K. C. Chang. PEBL: Positive-example based learning for Web page classification using SVM. In *Proc. 8th Int. Conf. Knowledge Discovery and Data Mining (KDD'02)*, pages 239–248, Edmonton, Canada, 2002.

LEARNING

TEXT AND WEB

A Learning Algorithm for Web Page Scoring Systems

Michelangelo Diligenti, Marco Gori, Marco Maggini

Dipartimento di Ingegneria dell'Informazione, Università di Siena, Italy
{michi, marco, maggini}@dii.unisi.it

Abstract

Hyperlink analysis is a successful approach to define algorithms which compute the relevance of a document on the basis of the citation graph. In this paper we propose a technique to learn the parameters of the page ranking model using a set of pages labeled as relevant or not relevant by a supervisor. In particular we describe a learning algorithm applied to a scheme similar to PageRank. The ranking algorithm is based on a probabilistic Web surfer model and its parameters are optimized in order to increase the probability of the surfer to visit a page labeled as relevant and to reduce it for the pages labeled as not relevant. The experimental results show the effectiveness of the proposed technique in reorganizing the page ordering in the ranking list accordingly to the examples provided in the learning set.

1 Introduction

PageRank [Page *et al.*, 1998] is the most popular algorithm to compute the page relevance in a collection of hyperlinked documents, like the Web. This algorithm exploits the topology of the citation graph in order to determine the authority of each page using the intuitive idea that authoritative documents are linked by many authoritative documents. Thus, the original algorithm computes a ranking which depends only on the graph connectivity without considering the page contents. However, focused versions of PageRank have been proposed in the literature [Haveliwala, 2002; Richardson and Domingos, 2002; Diligenti *et al.*, 2002] in order to define more selective rankings for topic-specific search engines. These approaches allow a limited degree of adaptation related to the choice of the specific topic. A text classifier is trained using examples of pages on the topic of interest but no further level of learning is considered in the ranking models. In particular, the classifier is trained independently on the ranking algorithm.

We propose a more general approach in which the learning algorithm is defined directly on the ranking model. In this case there is not a predefined topic, but a supervisor provides examples of pages to be promoted and pages which are considered as not relevant. Thus, the learning algorithm is able

to manage both positive examples (relevant pages) and negative examples (not relevant pages). The ranking function is based on the Web surfer model [Diligenti *et al.*, 2002]. The relevance of a page is defined by the probability of the surfer to visit the page when considering the stationary probability distribution of the Markov chain which describes the surfer behavior. Basically the learning algorithm computes the parameters which define the surfer model in order to reduce the probability of visiting a negative example and to increase the probability of being in a positive example. In order to reduce the number of free parameters, the surfer model is simplified by dividing the pages among a set of categories and assuming that the surfer behavior depends only on the category of the page where it is located. In particular, the set of categories does not need to be predefined but it is determined by a clustering algorithm based on the document contents. Moreover, whereas the focused versions of the PageRank simply direct the Web surfer to the most promising page (i.e. the page whose content is the most similar to the content of a target page), the proposed algorithm can exploit hierarchies of topics, allowing the surfer to follow complex paths, composed by many steps, leading to relevant pages.

The paper is organized as follows. In the next section the web surfer model is reviewed. Then, in section 3 the learning algorithm is described. Section 4 reports some experiments which show the effectiveness of the proposed learning algorithm. Finally, in section 5 the conclusions are drawn.

2 Random Walks on the Web graph

Random walk theory has been widely used to compute the absolute relevance of a page in the Web [Page *et al.*, 1998; Lempel and Moran, 2000]. The Web is represented as a graph G , where each Web page is a node and a link between two nodes represents a hyperlink between the associated pages. A common assumption is that the relevance x_p of page p is represented by the probability of ending up in that page during a walk on this graph.

We consider the model of a Web surfer who can perform one out of two atomic actions while visiting the Web graph: jumping to a node of the graph (action J) or following a hyperlink from the current page (action I). In general, the action taken by the surfer will depend on the page contents and the links it contains. We can model the user's behavior by a set of probabilities which depend on the current page:

$x(l|q)$ is the probability of following one hyperlink from page q , and $x(J|q)$ is the probability of jumping from page q . These values must satisfy the normalization constraint, $x(J|q) + x(l|q) = 1$.

The two actions need to specify their targets. Assuming that surfer behavior is time-invariant, then the targets are specified by the probability $x(p|q, J)$ of jumping from page q to page p , and the probability $x(p|q, l)$ of selecting a hyperlink from page q to page p . $x(p|q, l)$ is not null only for the pages p linked directly by page q , i.e. $p \in ch(q)$, being $ch(q)$ the set of the children of node q in the graph G . These sets of values must satisfy the probability normalization constraints $\sum_{p \in G} x(p|q, J) = 1, \forall q \in G$ and $\sum_{p \in ch(q)} x(p|q, l) = 1, \forall q \in G$.

The model considers a temporal sequence of actions performed by the surfer and it can be used to compute the probability that the surfer is located in page p at time t , $x_p(t)$. The probability distribution on the pages of the Web is updated by taking into account the actions at time $t + 1$ using the following equation

$$\begin{aligned} x_p(t+1) = & \sum_{q \in G} x(p|q, J) \cdot x(J|q) \cdot x_q(t) + \\ & + \sum_{q \in pa(p)} x(p|q, l) \cdot x(l|q) \cdot x_q(t), \end{aligned} \quad (1)$$

where $pa(p)$ is the set of the parents of node p . The relevance of a page p , x_p is computed using the stationary distribution of the Markov Chain of equation (1). This is a simplified version of the model proposed in [Diligenti *et al.*, 2002].

3 Learning the page rank

The random surfer model which is used to compute the page relevance scores depends on the values assigned to the probabilities $x(p|q, J)$, $x(J|q)$, $x(p|q, l)$, and $x(l|q)$. In most of the approaches proposed in the literature, these parameters are predefined or computed from some features describing the page p and/or the page q . For example, in the original PageRank scheme $x(l|q) = d$, $x(J|q) = 1 - d$, $x(p|q, J) = \frac{1}{N}$, and $x(p|q, l) = \frac{1}{ch(q)}$.

However, in the general case, it would be infeasible to estimate the parameters without any assumption to reduce their number. In particular, we assume that the pages are grouped into n clusters according to their content and that the surfer's behavior for any page p is dependent only on the cluster which p belongs to. In our experiments we used classical clustering techniques (e.g. k-means) on the bag-of-words representation of the pages [Kobayashi and Takeda, 2000]. By this hypothesis, the parameters of the model are computed as:

$$\begin{aligned} x(p|q, l) &= \frac{r(c^q \rightarrow c^p|l)}{\sum_{k \in ch(q)} r(c^q \rightarrow c^k|l)} \\ x(p|q, J) &= \frac{x(c^p|c^q, J)}{|c^p|} \\ x(l|p) &= x(l|c^p) \\ x(J|p) &= x(J|c^p) \end{aligned} \quad (2)$$

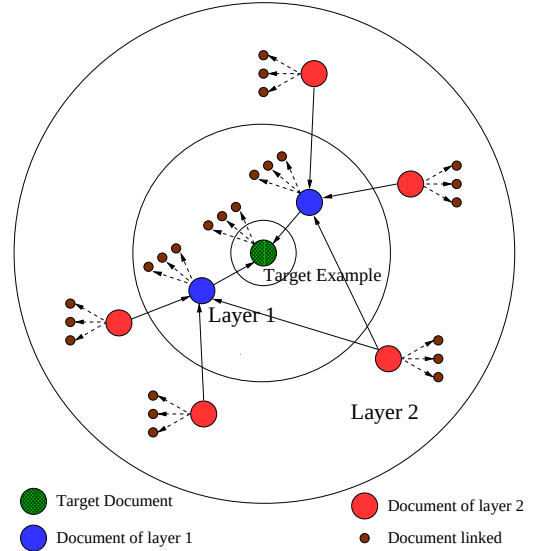


Figure 1: The context graph built by following the links backwards from an example page.

where c^q is the cluster which page q belongs to, $|c^p|$ is the number of pages in cluster c^p , the parameter $r(c^q \rightarrow c^p|l)$ is the tendency of the surfer to follow a hyperlink from a page in cluster c^q to a page in cluster c^p , and the parameter $x(c^p|c^q, J)$ is the tendency of the surfer to jump from a page of cluster c^q to a page of cluster c^p . The values $r(c^q \rightarrow c^p|l)$ are initialized to 1, such that all links are equally likely to be followed. Because of the model assumptions, the stronger is the tendency of the surfer to move to a cluster, the higher are the scores associated to the pages in that cluster.

3.1 The learning set

We assume that a supervisor provides a set pages labeled as *relevant* and *not relevant*. These examples are used to define a *specific* view of the collection of documents contained in a search engine. In particular the supervisor provides:

- a set of pages N^+ which should get a high rank;
- a set of pages N^- which should get a low rank.

In the first case we refer to “positive” or “good” pages, whereas in the latter case to “negative” or “bad” pages.

Each example page is associated to its Context Graph [Diligenti *et al.*, 2000], representing how the page can be reached from the Web. The graph is organized into layers as shown in figure 1. The example page is the only node in layer 0. Then, each node in layer $i > 0$ links only nodes in layer $i - 1$. Thus, a page belongs to layer i if it is at least i links away from the example page. The *depth* K of the context graph is defined as the number of layers in the graph including level 0. For a given example, the context graph can be built by back-crawling the Web adding to layer $i + 1$ the pages which link the pages in layer i , up to a maximum level K . Moreover, all the pages that are linked by at least a page of the graph are also considered.

The learning set consists of the context graphs built from the examples in N^+ and N^- . $\mathcal{CG}$ is the set of all the pages contained in the learning set; L_k^+ (L_k^-) is the union of the pages in the layer k of the context graphs of the positive (negative) examples.

3.2 Learning the transition parameters

We define the set of transition probabilities $\mathcal{T} = \{x(p \in c_j | q \in c_i, l) \mid i, j = 1, \dots, n\}$, which represent the probabilities of the surfer to select a hyperlink from a page in the cluster c_i to a page in the cluster c_j . The rank of the pages in the set N^+ is increased, whereas the rank of the pages in N^- is decreased if we increase the probability of the paths leading to pages in N^+ and we reduce the probability of paths leading to pages in N^- . This effect can be obtained by maximizing the following cost function

$$y(\mathcal{T}) = \sum_{k=0}^{K-1} \alpha^k [x(p \in L_k^+ | q \in L_{k+1}^+, \mathcal{T}, l) + (1 - x(p \in L_k^- | q \in L_{k+1}^-, \mathcal{T}, l))] , \quad (3)$$

where $0 \leq \alpha \leq 1$ is a discount factor which is used to reduce the impact on the cost function of the links too far from the target page.

The probability $x(p \in L_k^+ | q \in L_{k+1}^+, \mathcal{T}, l)$ of moving one level in the context graph towards a good page, can be rewritten as

$$\begin{aligned} x(p \in L_k^+ | q \in L_{k+1}^+, \mathcal{T}, l) &= \quad (4) \\ &= \sum_{i=1}^n x(p \in L_k^+ | q \in L_{k+1}^+, q \in c_i, \mathcal{T}, l) \cdot \\ &\quad \cdot x(q \in c_i | q \in L_{k+1}^+, l) = \\ &= \sum_{i=1}^n \sum_{j=1}^n x(p \in L_k^+ | q \in L_{k+1}^+, q \in c_i, p \in c_j, l) \cdot \\ &\quad \cdot x(p \in c_j | q \in c_i, l) \cdot x(q \in c_i | q \in L_{k+1}^+, l) , \end{aligned}$$

where $x(p \in c_j | q \in c_i, l)$ reflects the surfer model assumption that the links are selected depending only on the clusters c_i and c_j . The probability $x(p \in L_k^+ | q \in L_{k+1}^+, q \in c_i, p \in c_j, l)$ of following a link to a page in the cluster c_j of layer k from a page in the cluster c_i of layer $k+1$ of a context graph can be estimated as

$$\begin{aligned} x(p \in L_k^+ \mid q \in L_{k+1}^+, q \in c_i, p \in c_j, l) &= \quad (5) \\ &= x(p \in L_k^+ \cap c_j | q \in L_{k+1}^+ \cap c_i, l) = \\ &= \frac{n(L_{k+1}^+ \cap c_i \rightarrow L_k^+ \cap c_j)}{n(L_{k+1}^+ \cap c_i \rightarrow c_j)} , \end{aligned}$$

where $n(A \rightarrow B)$ is the number of links from the pages in the set A to the pages in the set B .

On the other hand the term $x(q \in c_i | q \in L_{k+1}^+, l)$ represents the probability that the surfer is located in a page of the cluster c_i , given that the page belongs to the layer $k+1$ of a positive context graph. This probability can be estimated using the scores of the pages in cluster c_i and in L_{k+1}^+ , computed using the current values of the parameters, as

$$x(q \in c_i | q \in L_{k+1}^+, l) = \frac{\sum_{p \in L_{k+1}^+ \cap c_i} x_p}{\sum_{p \in L_{k+1}^+} x_p} . \quad (6)$$

Similar relationships can be derived for the negative set.

The cost function (3) can be optimized by updating the parameters in $\mathcal{T}$ using gradient ascent. The gradient with respect to the transition parameters $x(p \in c_j | q \in c_i, l)$ can be computed using equation (4), yielding

$$\begin{aligned} \nabla_k^{ij} y(\mathcal{T}) &= x(p \in L_k^+ | q \in L_{k+1}^+, q \in c_i, p \in c_j, l) \cdot \\ &\quad \cdot x(q \in c_i | q \in L_{k+1}^+, l) - \quad (7) \\ &= x(p \in L_k^- | q \in L_{k+1}^-, q \in c_i, p \in c_j, l) \cdot \\ &\quad \cdot x(q \in c_i | q \in L_{k+1}^-, l) , \end{aligned}$$

where we neglected that $x(q \in c_i | q \in L_{k+1}^+, l)$ and $x(q \in c_i | q \in L_{k+1}^-, l)$ actually depend on the transition parameters. Since the model parameters $r(c_i \rightarrow c_j | l)$ are essentially a not normalized version of the transition parameters $x(p \in c_j | q \in c_i, l)$, we can update them directly as

$$r(c_i \rightarrow c_j | l)' = r(c_i \rightarrow c_j | l) + \eta \sum_{k=0}^{K-1} \alpha^k \nabla_k^{ij} y(\mathcal{T}) , \quad (8)$$

where η is the learning rate.

Once the model parameters are updated, a new score distribution can be computed, thus updating the estimate of the variables $x(q \in c_i | q \in L_{k+1}^+, l)$ using equation (6). Therefore, the function optimization and the score estimation can be iterated, using an EM style algorithm [Dempster *et al.*, 1977], till a termination criterion is satisfied.

3.3 Learning the jump parameters

The transition parameters for a jump can be optimized to increase the probability of jumping to a “good” page and not to a “bad” page. This effect is obtained by maximizing the cost function

$$y(\mathcal{P}) = x(p \in N^+ | \mathcal{P}, J) + [1 - x(p \in N^- | \mathcal{P}, J)] , \quad (9)$$

where $\mathcal{P} = \{x(p \in c_j | q \in c_i, J) \mid i, j = 1, \dots, n\}$ is the set of the jump parameters.

We start observing that,

$$\begin{aligned} x(p \in N^+ \mid \mathcal{P}, J) &= \quad (10) \\ &= \sum_{i=1}^n x(p \in N^+ | q \in c_i, J) \cdot x(q \in c_i | J) = \\ &= \sum_{i=1}^n \sum_{j=1}^n x(p \in N^+ | q \in c_i, p \in c_j, J) \cdot \\ &\quad \cdot x(p \in c_j | q \in c_i, J) \cdot x(q \in c_i | J) . \end{aligned}$$

By assuming that the jump target is independent on the originating cluster c_i , the term $x(p \in N^+ | q \in c_i, p \in c_j, J)$ can be estimated as

$$\begin{aligned} x(p \in N^+ | q \in c_i, p \in c_j, J) &= x(p \in N^+ | p \in c_j, J) = \\ &= \frac{|N^+ \cap c_j|}{|\mathcal{CG}|} . \quad (11) \end{aligned}$$

Moreover, the value $x(q \in c_i | J)$ is computed at each iteration as

$$x(q \in c_i | J) = x(q \in c_i) = \sum_{p \in c_i} x_p . \quad (12)$$

Similar relationships can be derived for the negative set.

The cost function (9) is maximized by gradient ascent. The gradient can be computed using equation (10) and the equivalent equation for the negative set, yielding

$$\begin{aligned} \nabla^{ij} y(\mathcal{P}) &= \{x(p \in N^+ | q \in c_i, p \in c_j, J) - \\ &\quad - x(p \in N^- | q \in c_i, p \in c_j, J)\} \cdot x(q \in c_i | J), \end{aligned} \quad (13)$$

where we neglected the dependence of $x(q \in c_i | J)$ on the parameters. At each iteration, the parameters are updated using the direction of the gradient as,

$$x(p \in c_j | q \in c_i, J)' = x(p \in c_j | q \in c_i, J) + \eta \nabla^{ij} y(\mathcal{P}), \quad (14)$$

being η the learning rate, and then normalized to meet the probabilistic constraints

$$x(p \in c_j | q \in c_i, J) = \frac{x(p \in c_j | q \in c_i, J)'}{\sum_{f=1}^n x(p \in c_j | q \in c_f, J)'}. \quad (15)$$

The adapted surfer model can re-surf the Web graph yielding a new estimate of equation (12). Thus, the new gradient estimate can be used to update again the parameters, till the stop criterion is satisfied.

3.4 Learning the surfer action bias

The surfer action bias is represented by the value of $x(l | q \in c_i)$ which is the probability of following a link from the current page rather than jumping to another page ($x(J | q \in c_i) = 1 - x(l | q \in c_i)$). In order to favor the action leading to good pages, we maximize the following cost function which represents a weighted average of the probabilities of ending in the layer k of a positive context graph and not of a negative context graph,

$$y(\mathcal{Q}) = \sum_{k=0}^{K-1} \alpha^k [x(p \in L_k^+ | \mathcal{Q}) + (1 - x(p \in L_k^- | \mathcal{Q}))], \quad (16)$$

where $\mathcal{Q} = \{x(l | q \in c_i) \mid i = 1, \dots, n\}$ is the set of the free parameters, and $0 \leq \alpha \leq 1$ is a discount factor to reduce the effect of the outer layers on the cost function.

The probability of moving to a page k steps away from a target is,

$$\begin{aligned} x(p \in L_k^+ | \mathcal{Q}) &= \sum_{i=1}^n x(p \in L_k^+ | q \in c_i) \cdot x(q \in c_i) = \\ &= \sum_{i=1}^n (x(p \in L_k^+ | q \in c_i, l) \cdot \\ &\quad \cdot x(l | q \in c_i) \cdot x(q \in c_i) + \\ &\quad + x(p \in L_k^+ | q \in c_i, J) \cdot \\ &\quad \cdot (1 - x(l | q \in c_i)) \cdot x(q \in c_i)). \end{aligned} \quad (17)$$

The term $x(p \in L_k^+ | q \in c_i, l)$ represents the probability of ending in the layer k of a positive context graph by following a link from a page of cluster i and can be estimated as

$$\begin{aligned} x(p \in L_k^+ | q \in c_i, l) &= \\ &= x(p \in L_k^+ | q \in L_{k+1}^+, q \in c_i, l). \end{aligned} \quad (18)$$

$$\begin{aligned} &\cdot x(q \in L_{k+1}^+ | q \in c_i, l) = \\ &= \frac{n(L_{k+1}^+ \cap c_i \rightarrow L_k^+)}{n(L_{k+1}^+ \cap c_i \rightarrow \mathcal{CG})} \cdot \frac{|L_{k+1}^+ \cap c_i|}{|c_i|}, \end{aligned}$$

where we assumed that all links out of a page are equally likely to be followed. The term $x(p \in L_k^+ | q \in c_i, J)$ is the probability of jumping to the layer k of a positive context graph from a page of cluster i , and can be computed as

$$x(p \in L_k^+ | q \in c_i, J) = \frac{|L_k^+|}{|\mathcal{CG}|}. \quad (19)$$

Finally, $x(q \in c_i)$ can be computed at each iteration as in equation (12). Similar relationships hold for the negative set.

The gradient of the cost function (16) can be computed using equation (17) and the corresponding relationship for the negative set, yielding

$$\begin{aligned} \nabla_k^i y(\mathcal{Q}) &= \{x(p \in L_k^+ | q \in c_i, l) - x(p \in L_k^+ | q \in c_i, J) - \\ &\quad - x(p \in L_k^- | q \in c_i, l) + x(p \in L_k^- | q \in c_i, J)\} \cdot x(q \in c_i), \end{aligned}$$

neglecting the dependence of $x(q \in c_i)$ on $x(l | q \in c_i)$.

The parameters are updated by gradient ascent as

$$x(l | q \in c_i)' = x(l | q \in c_i) + \eta \cdot \sum_{k=0}^{K-1} \alpha^k \cdot \nabla_k^i y(\mathcal{Q}), \quad (20)$$

where the learning rate must be chosen small enough in order to preserve the stochastic model of the surfer.

The adapted surfer model can re-surf the Web graph yielding a new estimate of equation (12). Thus, the new gradient estimate can be used to update again the parameters and the procedure is iterated till the stop criterion is satisfied.

4 Experimental results

In the following experiments, the learning algorithm was applied only to the transition parameters of the random surfer model. The experiments were performed on two datasets, each containing 1.000.000 documents, which were collected by focus crawling the Web on the topics “wine” and “playstation”, respectively. The documents were clustered using a hierarchical version of the k-means algorithm [Fukunaga, 1990]. We considered different numbers of clusters generated by the k-means algorithm. In particular, in two different runs, we clustered the pages from the dataset on topic “playstation” into 16 and 25 sets, whereas the dataset on topic “wine” was clustered into 25 and 100 sets.

Two topic-specific search engines were build creating the inverted lists containing the terms of all the documents. Then, the rank of each page in the dataset was computed using the model described in section 2 by setting its parameters to reproduce the original PageRank ranking. We tested the quality of the ranking function by submitting a set of queries to this focused search engine. The documents matching the query were sorted by the scores assigned by the random surfer. By interacting with the search engine, we found pages which were authorities on the specific topic but were (incorrectly) not inserted in the top positions of the result list. Such pages were selected as examples of pages which should get a higher

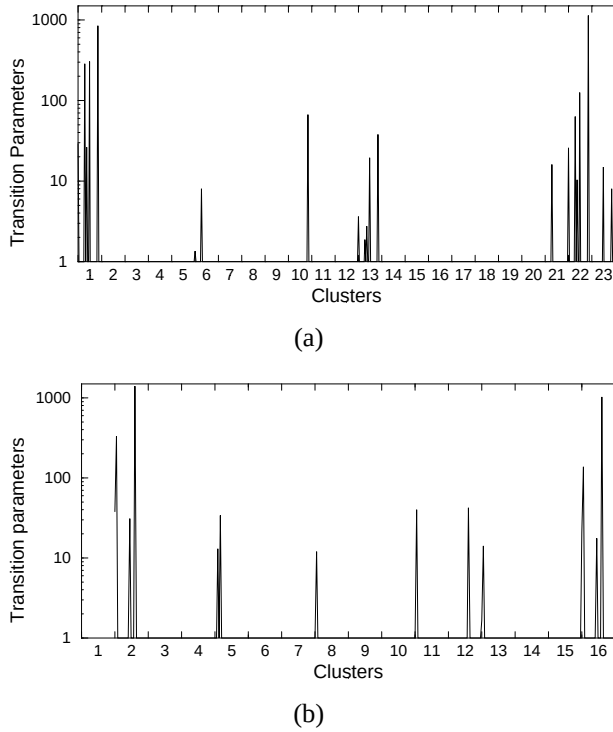


Figure 2: Plots of the values of the transition parameters for each cluster, resulting from the training of the surfer. (a) “wine” dataset. (b) “playstation” dataset.

rank (positive examples). On the other hand, we found pages having a high score which were not authorities on the topic. Such pages were selected as negative examples. The learning algorithm, as described in section 3, was applied to the datasets, using the selected examples. In the experimental setup only a small set of examples (3-15) was typically labelled as positive or negative.

4.1 Analysis of the effects of learning

In a first group of experiments we aimed at demonstrating that the surfer model as learnt by the proposed training algorithm, could not be generated by a simpler (but less powerful) schema as the focused versions of PageRank, proposed in the literature.

In figure 2-(a) 2-(b), we show the values of the transition parameters for each cluster, resulting from the training sessions for a surfer on the topic “wine” and “playstation”, respectively. The learnt parameters represent a complex interaction of the resulting surfer with the page clusters. This interaction could not be expressed by a simple schema as, for example, the focused versions of PageRank proposed in the literature [Richardson and Domingos, 2002; Diligenti *et al.*, 2002]. This demonstrates that the model, in order to learn the users’ feedback, needed to exploit hierarchies of topics to model the complex path leading either to positive or negative examples.

In figure 3-(a) and 3-(b), we show for each cluster, the values of the probability of the surfer of being located in a page

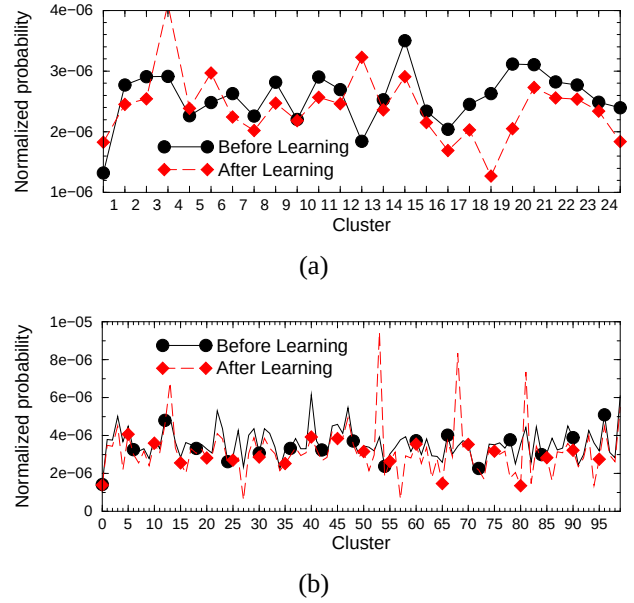


Figure 3: Plots of the probability of the surfer of being located in a page of a given cluster for the topic “wine”. The probabilities are normalized with respect to the number of pages in the cluster. (a) 25 clusters. (b) 100 clusters.

of that cluster. In particular we used the dataset on topic “wine” clustered into 25 and 100 groups, respectively. The probabilities are normalized with respect to the number of pages in the cluster. The learning algorithm is able to increase the likelihood of the random surfer to visit pages belonging to a set of clusters, while decreasing its probability of visiting pages belonging to other clusters.

In figure 4, we report the rank of pages after the learning session with respect of their rank before learning. In particular, on the x axis the pages are sorted according to their page rank before the learning session. The closer a page to the origin, the more “relevant” the page is. On the y axis it is reported the corresponding page rank after the learning session. The plot shows the generalization capability of the learning algorithm, which is able to change the ranks of a significant percentage of pages, even if a small set of example was provided.

4.2 Qualitative results

After training the surfer and computing the scores for all pages in the dataset, we compared the rank before and after the training process.

Figure 5 reports the pages which obtained the largest variation in their rank. The pages that obtained a negative rank variation were either topic-generic authoritative pages (e.g. www.netscape.com, www.yahoo.com) or pages relevant for different topics than “wine” (e.g. “www.forgottensoldier.org”). Only the page “www.yahoo.com” was explicitly provided to the system as a negative example. On the other hand, the pages that obtained a positive rank variation were effectively authoritative pages on the topic “wine” (e.g. “www.wine-searcher.com” or “webwinery.com”). Among the most

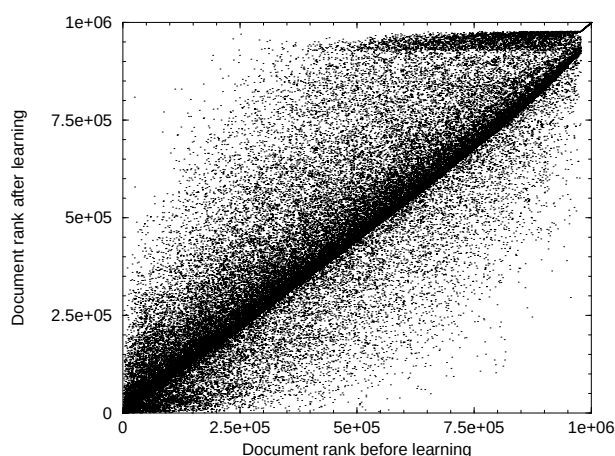


Figure 4: Plot of the page ranks before (x axis) versus after learning (y axis). The closer a page is to the origin, the most “relevant” it is.

ascending documents, only the page “www.winespectator.com” was explicitly provided to the system as a positive example, whereas all other pages were pushed up in the rank for the capability of the system to generalize from a small set of training data.

5 Conclusions

In this paper we introduced a novel algorithm to learn the ranking function over a collection of Web pages. The proposed framework allows us to tune a previously computed rank distribution, specifying which pages should get a higher or lower score. Like in the original PageRank, we assume that the score of a page is proportional to the probability that a Web surfer is visiting the page. The learning algorithm adapts the parameters of the surfer in order to increase the probability that the surfer is visiting a “good” page, while decreasing the probability that the surfer is visiting a “bad” page. The parameters are estimated from the Context Graphs, which compactly represent the topological context in which a Web page is inserted. The learning algorithm is fast since only close ancestors of example pages are considered in the Context Graphs. Thus, it can be applied to large scale repositories with little overhead (i.e. the ranking function must be computed more many times). Further investigations are needed to compare the accuracy of the learnt ranking functions with respect to the focused versions of the PageRank and to exploit the complete learning algorithm which considers all the model parameters.

References

- [Dempster *et al.*, 1977] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. B*, 39:185–197, 1977.
- [Diligenti *et al.*, 2000] M. Diligenti, F. Coetzee, S. Lawrence, L. Giles, and M. Gori. Focus crawling

-
- Most descending pages on topic “wine”
- <http://www.macromedia.com/go/getflashplayer>
 - <http://www.netscape.com>
 - <http://www.macromedia.com/downloads>
 - <http://www.yahoo.com>
 - <http://www.adobe.com/products/acrobat/readstep.html>
 - <http://help.yahoo.com>
 - <http://www.nl.placestostay.com/index.html>
 - <http://home.netscape.com>
 - <http://www.forgottensoldier.com>
 - <http://www.inntravels.com>
-
- Most ascending pages on topic “wine”
- <http://www.winecountry.com>
 - <http://webwinery.com>
 - <http://www.ny-wine.com/wineinfo.asp>
 - <http://www.wine-searcher.com>
 - <http://www.insidewine.com>
 - <http://webwinery.com>
 - <http://www.viniesapordipuglia.com/en/index.html>
 - http://www.vinitaly.com/home_en.asp
 - <http://www.wineinstitute.org>
 - <http://www.winespectator.com>

Figure 5: URLs of the pages that yielded either the largest negative or positive rank changes among the pages that were initially in the top 1000 scoring documents.

by context graphs. In *Proceedings of the International Conference on Very Large DataBases, 11-15 September 2000, Il Cairo, Egypt*, pages 527–534, 2000.

- [Diligenti *et al.*, 2002] M. Diligenti, M. Gori, and M. Maggini. Web Page Scoring Systems for vertical and horizontal search engines. In *Proceedings of the 11-th World Wide Web conference (WWW11)*, 2002.
- [Fukunaga, 1990] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, CA, US, 1990.
- [Haveliwala, 2002] T. H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the eleventh international conference on World Wide Web*, pages 517–526. ACM Press, 2002.
- [Kobayashi and Takeda, 2000] Mei Kobayashi and Koichi Takeda. Information retrieval on the web. *ACM Computing Surveys*, 32(2):144–173, 2000.
- [Lempel and Moran, 2000] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In *Proceedings of the 9th World Wide Web Conference*, 2000.
- [Page *et al.*, 1998] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Computer Science Department, Stanford University, 1998.
- [Richardson and Domingos, 2002] Mathew Richardson and Pedro Domingos. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

Does a New Simple Gaussian Weighting Approach Perform Well in Text Categorization?

Giorgio Maria Di Nunzio

Dip. di Ingegneria dell'Informazione
Università degli Studi di Padova
Via Gradenigo 6/b, 35131 Padova - Italia
dinunzio@dei.unipd.it

Alessandro Micarelli

Dip. di Informatica e Automazione
Università degli Studi "Roma Tre"
Via della Vasca Navale 79, 00146 Roma - Italia
micarel@dia.uniroma3.it

Abstract

A new approach to the Text Categorization problem is here presented. It is called Gaussian Weighting and it is a supervised learning algorithm that, during the training phase, estimates two very simple and easily computable statistics which are: the Presence P , how much a term t is present in a category c ; the Expressiveness E , how much t is present outside c in the rest of the domain. Once the system has learned this information, a Gaussian function is shaped for each term of a category, in order to assign the term a weight that estimates the level of its importance for that particular category. We tested our learning method on the task of single-label classification using the Reuters-21578 benchmark. The outcome of the result was quite impressive: in different experimental setups, we reached a micro-averaged F1-measure of 0.89, with a peak of 0.899. Moreover, a macro-averaged Recall and Precision was calculated: the former reported a 0.72, the latter a 0.79. These results reach most of the state-of-the-art techniques of machine learning applied to Text Categorization, demonstrating that this new weighting scheme does perform well on this particular task.

1 Introduction

Consider the problem of automated classification of text documents. This problem is of great importance as the accessible textual information increases and the volume of online texts available through the Internet expands rapidly. One solution is to categorize documents according to their topics beforehand or in real time.

A number of different Machine Learning methods have been applied to Text Categorization (TC), including probabilistic classifiers, decision trees, regression methods, neural networks and support vector machines (see [Sebastiani, 2002]). Most of these methodologies share a common factor of high level complexity, that, often, makes a classifier design difficult to understand.

The question addressed in this paper is if it is possible to find a simple learning algorithm that uses naive, human comprehensible parameters, in such a way as to act like a non-

expert human being in front of the problem of classifying new unknown documents.

Starting from this idea of simplicity, we decided to design a training algorithm that corresponds to the action of finding two information parameters: given a set of categories C of documents, for each term t of a category $c \in C$ we calculate its Presence P , that is, the percent of documents of the category c in which the term t appears at least once; and its Expressiveness E , that is, how much the term t is absent in the other categories of the domain. Then, we model a Gaussian Function with the two parameters above, whose value in the abscissae equal to 1 is used as the weight of the term t . We called this learning approach Gaussian Weighting. It is important to stress the fact that we don't use any stemming algorithm or Feature Selection function (see [Yang and Pedersen, 1997]). An eventual reduction of the number of features per category is done using only two thresholds named $ThresP$ and $ThresE$, relative to the parameters P and E .

To make the evaluation of Gaussian Weighting comparable to most of the published results on TC, we chose the Reuters-21578¹ corpus as benchmark for the single-label classification task. Three different tests have been performed with a complete automated learning approach: the first run has been done without the use of a Gaussian Function, but simply giving a weight proportional to P and E ; the second and the third test runs have been done with the Gaussian Weighting approach proposed here.

The outcome of the results was quite surprising: throughout the different experimental setups, we repeatedly reached a micro-averaged F1-measure around the 0.89, with a peak of 0.899. Then, since the micro-average is known to be highly influenced by frequent categories (see [Yang and Liu, 1999]), we decided to compute the macro-averaged Recall and Precision. Again, the results reached a satisfactory macro-Recall around 0.72 and a macro-Precision of 0.79. These results reach most of the state-of-the-art techniques of machine learning applied to Text Categorization (see [Dumais et al., 1998]), demonstrating that this new weighting scheme does perform well on this particular task. Moreover, the low computational cost of the algorithm we propose, makes this approach particularly preferable when the computing power is low.

¹www.daviddlewis.com/resources/testcollection/reuters21578

2 The Algorithm

In this section we present our supervised algorithm. First, we will analytically define the two parameters: Presence P and Expressiveness E . Then, we will explain how the training algorithm works and how the system classifies new documents at run-time.

2.1 Presence P and Expressiveness E

A central key of this naive approach is the computation of two easy human understandable parameters that we call: Presence P , and Expressiveness E . The former captures how much a term t appears at least once in the documents belonging to a category c , the latter estimates how much the same term t does not appear in the documents of the other categories. Given a set of categories $C = \{c_1, c_2, \dots, c_i, \dots, c_n\}$, each category having a number of documents m , (e.g. $c_1 = \{d_{1,1}, d_{1,2}, \dots, d_{1,j}, \dots, d_{1,m}\}$), we give the following definitions:

- The number of documents D_i of the i -th category is:

$$D_i = |c_i| = \sum_{j=1}^m d_{i,j} \quad (1)$$

- the number of documents of category i in which the term t is present:

$$D_{i|t} = |c_{i|t}| = \sum_{j=1}^m (d_{i,j}|t) \quad (2)$$

Note that we use the symbol ‘|’ to denote the presence of a term inside a category or a document. This is not to be confused with the same symbol when used for the conditional probability $P(x|y)$.

Then, the Presence P of a term t in the i -th category is, using (1) and (2) :

$$P_{i|t} = \frac{D_{i|t}}{D_i} \quad (3)$$

while, the Expressiveness E of term t in the i -th category is calculated using 1, 2):

$$E_{i|t} = 1 - \frac{\sum_{p=1}^n P_{p|t}}{|C|-1} \quad p \neq i \quad (4)$$

It is important to stress that the same term in different categories has different values of expressiveness. This fact is better explained in Table 1. A term t is present in all the three categories with presence $P_{i|t}$ shown in the first row. The second row is the expressiveness $E_{i|t}$ of the same term for each category. Note how t of category c_1 has a greater expressiveness given the relatively smaller presence of the term in the rest of the domain.

	c_1	c_2	c_3
$P_{i t}$	0.9	0.4	0.2
$E_{i t}$	0.7	0.45	0.35

Table 1: A numerical example of Presence and Expressiveness.

Locality of Presence and Expressiveness

The approach uses a weighting method for terms per each class, which is in a sense a local type of weighting/modeling scheme of classes. This idea is similar to the Local LSI representation [Hull, 1994] where, in order to characterize the term space, the singular value decomposition is applied to a matrix consisting only of the known relevant documents (in our case the documents belonging to a class).

There are substantial differences with the *Tf-Idf* weighting scheme. *Tf* counts the number of occurrences of a term t in a document, while P counts the documents in which t appears at least once. *Idf* computes the number of documents in which t appears over the whole collection, meanwhile E calculates, according to which class we are computing the weights, the partial averaged Presence of t in the domain.

2.2 Learning the Gaussian Weights (GW)

In this paragraph we explain how we shape a Gaussian function using the values P and E . Starting from the definition of a generic Gaussian function:

$$G(x) = \exp\left(-\frac{(x - \mu)^2}{2 \cdot \sigma^2}\right) \quad (5)$$

we estimate the parameters $\hat{\mu}$ and $\hat{\sigma}^2$ as follows:

$$\hat{\mu} = P_{t|i} \quad (6)$$

$$\hat{\sigma}^2 = P_{t|i} \cdot E_{t|i} \quad (7)$$

Then, the Gaussian Weighting GW for a term t of a category i is defined, substituting (6) and (7) in (5), as:

$$GW(x)|_{x=1} = \exp\left(-\frac{(x - \hat{\mu})^2}{2 \cdot \hat{\sigma}^2}\right) \quad (8)$$

The GW calculated for $x = 1$, returns a real value belonging to the (0,1] interval. In particular, the maximum weight can be reached only when the Presence of a term is equal to 1, that is to say, when it appears in each document of the category of interest. Figure 1 shows an example of a GW with $P = 0.5$ and $E = 0.5$ (continuous), and another one with $P = 0.5$ and $E = 0.9$ (dashed); when two terms have the same Presence, the one with a higher Expressiveness has a higher weight too.

Computing the parameter P

The algorithm to compute P is the following:

1. For each category $c_i \in C$
2. $D_i = 0$
3. For each document $d_{i,j} \in c_i$
4. $D_i = D_i + 1$
5. For each unique term t in $d_{i,j}$
6. if t is in $D_{i|t}$
7. then $D_{i|t} = D_{i|t} + 1$
8. else Add $D_{i|t} = 1$
9. end For
10. end For
11. For each term in $D_{i|t}$

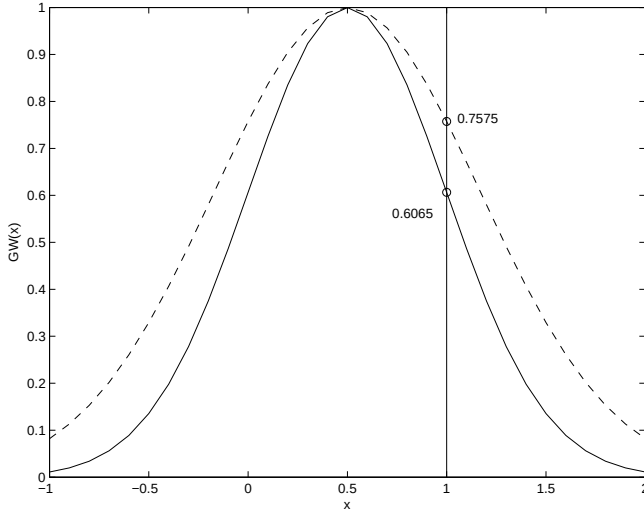


Figure 1: Example of a GW. The continuous line is a GW with $P = 0.5$ and $E = 0.5$. The dashed line is a GW with $P = 0.5$ and $E = 0.9$. For the same P , the higher expressiveness a term possesses, the bigger the output GW gives.

12. $P_{i|t} = \frac{D_{i|t}}{D_i}$
13. end For
14. end For

If we assume we can use a HashMap HM to store the results of $D_{i|t}$, such that any update or search in the HM has a constant cost k , in the worst case the algorithm to compute the parameter P has a computational cost of $O(C \times D \times T \times k)$.

Computing the parameter E

The algorithm to compute E is the following:

1. For each category $c_i \in C$
2. sum = 0
3. For each term t in $P_{i|t}$
4. For each category $c_j \in C$ where $i \neq j$
5. if t in $P_{j|t}$
6. then sum = sum + $P_{j|t}$
7. end For
8. $E_{i|t} = 1 - \frac{\text{sum}}{|C|-1}$
9. end For
10. end For

Again, if we assume that $P_{j|t}$ is a HashMap with a constant cost k of access and search, the computational cost of this second algorithm is $O(C^2 \times T \times k)$, which is quadratic cost with respect to the number of categories C .

2.3 Categorizing a new document

Once the system has been trained and the parameters P and E have been found for each term of the domain, it is possible to feed the system with an unknown document and, according to the value of a couple of optional thresholds $ThresP$ and $ThresE$, classify it with the following algorithm:

1. For each $c_i \in C$
2. $outputc_i = 0$, $Nterm = 0$
3. For each term in $P_{i|t}$
4. if t is in test
And $P_{i|t} > ThresP$ And $E_{i|t} > ThresE$
5. then $outputc_i = outputc_i + GW(1, t, c)$
 $Nterm = Nterm + 1$
6. end For
7. $outputc_i = \frac{outputc_i}{n}$
8. end For
9. Assign the document to the c_i with the highest $outputc_i$

For each category, the algorithm calculates, the mean of the activated Gaussian functions of the terms whose $P_{i|t}$ and $E_{i|t}$ are greater than two fixed thresholds, respectively $ThresP$ and $ThresE$. The computational cost of this algorithm is $O(C \times T \times k)$.

3 Experimental Setup

3.1 Test Collection

To make the evaluation of GW comparable to most of the published results on TC, we chose the Reuters-21578 corpus as benchmark. During the last few years this corpus has been used as a standard benchmark on which many TC methods have been evaluated, although the results are sometimes difficult to compare as slightly different version have been used. For this paper we used the ModApte split of Reuters-21578 in which 75% of the stories (9603) are used as training documents to build the classifier and the remaining 25% (3299) to test the accuracy of our single-label classifier. Now, the Reuters-21578 is known to be quite unbalanced on the distribution of stories per category. Therefore, of the 135 potential topics categories only the 10 most frequent are here used; these 10 categories account for almost the 75% of the training instances, while the remainder is distributed among the other 115. Table 2 shows the number of training and test samples for each category.

Category	# Training	# Test
Earn	2877	1087
Acquisitions	1650	719
Money-fx	538	179
Grain	433	149
Crude	389	189
Trade	369	118
Interest	347	131
Ship	197	89
Wheat	212	71
Corn	182	56

Table 2: Number of Training and Test documents for the ten most frequent categories of Reuters-21578 ModApte split

3.2 Parameters Setting

We performed three different test runs on our system:

1. The ‘what if?’ test run: what if we didn’t use a Gaussian function? we tried to run our system with the most simple function we could use: $GW = P \cdot E$
2. The GW test run: uses the GW function defined in equation (8)
3. The simple GW test run: we simplified the Gaussian function with a variance equal to $\sigma^2 = P_{t,l}$

All the tests maintained $ThresE$ equal to 0.6, that is to say, a term in a category should have an Expressivity value greater than 0.6. This value permitted to achieve the highest performance during the test phase. $ThresP$ has been varied in the range between 0.0 (all the terms of the categories were used) and 0.5 (only the terms present in the 50% of the training documents were used).

The number of terms per category according to the value P is reported in Table 3.

	0,0	0,1	0,2	0,3	0,4	0,5
acq	11115	65	15	7	6	3
corn	3257	82	20	12	7	3
crude	6126	99	24	12	4	1
earn	10455	46	19	12	9	6
grain	5448	64	17	10	5	3
interest	3863	92	20	9	6	4
money-fx	5545	116	30	13	4	2
ship	4084	80	17	3	0	0
trade	5795	149	37	17	6	3
wheat	3543	75	17	10	4	4
feat/cat	592.3	87.8	21.6	10.5	5.1	2.9

Table 3: The table shows the number of terms per category according to the threshold of Presence P . The last row reports the average of features per category.

A stoplist of 331 terms has been used to remove the most frequent words of the English Language, but no stemming or Feature Selection have been performed.

In order to evaluate the accuracy of the classifier we have computed the standard IR measures, such as Recall and Precision, and the following averaging measures: micro-Recall (μRe), micro-Precision (μPr) and micro-averaged F1 measure (micro-F1):

$$\mu Re = \frac{\sum_{category} \#correctly\ classified}{\sum_{category} \#belonging\ to\ category}$$

$$\mu Pr = \frac{\sum_{category} \#correctly\ classified}{\sum_{category} \#classified\ under\ category}$$

$$microF1 = \frac{2 \cdot \mu Re \cdot \mu Pr}{\mu Re + \mu Pr}$$

In particular we have reported the values of the F1-measure in order to directly compare our results with the ones in the literature. But, since the micro-averaged measure is known to be highly influenced by the most frequent category (see

[Yang and Pedersen, 1997]), we have decided to calculate the Macro-Recall (MRe) and the Macro-Precision (MPr), as well:

$$MRe = \frac{\sum_{category} Re}{|category|}$$

$$MPr = \frac{\sum_{category} Pr}{|category|}$$

4 Results

4.1 The ‘What if?’ test run

In this test, the weight of each term t of a category i is computed as:

$$GW = P_{t|i} \cdot E_{t|i} \quad (9)$$

The ‘What if?’ test run results are shown in Figure 2.

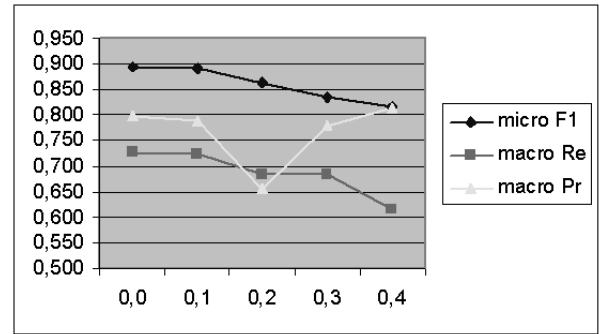


Figure 2: The ‘What if?’ test run. The x-axis represents the values of Presence P .

	0.0	0.10	0.20	0.30	0.40	0.50
micro F1	0.894	0.891	0.862	0.836	0.816	*
macro Re	0.727	0.725	0.682	0.684	0.615	*
macro Pr	0.797	0.789	0.656	0.778	0.814	*

Table 4: The ‘What if?’ test run.

In this particular test we did not expect so remarkable a performance. Considering the weight of a term proportional to the parameters P and E seems to be another possible solution to investigate. The performance of the system seems not to be influenced significantly when the threshold $ThresP$ is set either to 0.0 or 0.1. It is worth stressing that this test performed better when all the terms (except those belonging to the stoplist) were used.

4.2 The GW test run

The results of this run are shown in Figure 3, while numerical values are reported in Table 5.

This test has been performed using equation (8) to calculate the weight of each term of the category. The best results have been obtained with $ThresP = 0.1$. Macro-Recall and macro-Precision are quite high, and they indicate that the system works well for every category. The performance of the system starts to decrease sensibly when $ThresP > 0.2$

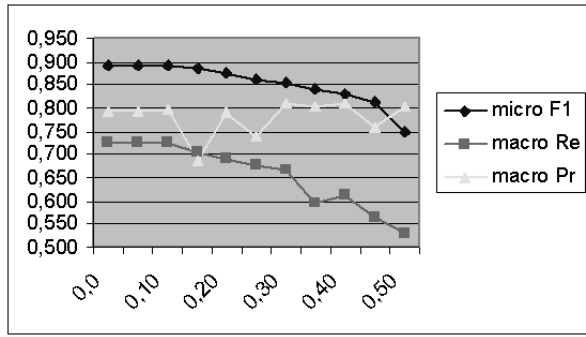


Figure 3: The GW test run. In this test the function 8 is used to weight each term.

	0.0	0.10	0.20	0.30	0.40	0.50
micro F1	0.892	0.893	0.876	0.856	0.832	0.748
macro Re	0.726	0.725	0.692	0.666	0.613	0.532
macro Pr	0.794	0.797	0.789	0.810	0.809	0.804

Table 5: The GW test run. Each column reports the values of micro F1, macro Recall and macro Precision for each *ThresP*.

4.3 The Simple GW test run

The results of this second run are shown in Figure 4. Numerical values are reported in Table 6.

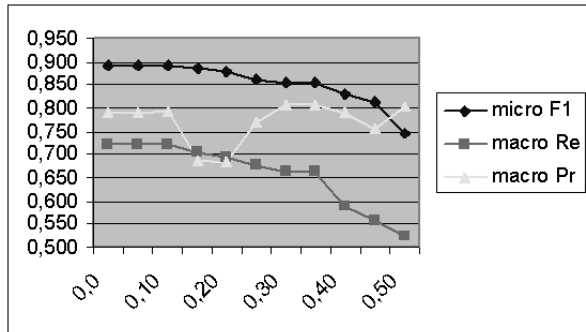


Figure 4: The simple GW test run. In this test the variance of the GW is equal to $\sigma^2 = P_{t,l}$.

This test has been performed using a simplified version of equation (8) to calculate the weight of each term of the category. The variance of the Gaussian function is set to $\sigma^2 = P_{t,l}$. The best results have been obtained with *ThresP* = 0.1, there does not seem to be any particular difference with the GW test run, except in cases when *ThresP* equals 0.2, where the macro Precision is sensibly lower.

4.4 Discussion

The results reported above are satisfactory. In many situations the system reached an accuracy comparable to the ones in the state-of-the-art systems. Some aspects need to be discussed here: the performance of the system in all the test runs decays when *ThresP* exceeds the 0.2, that is to say, looking back to Table (3), when the number of features per

	0.0	0.10	0.20	0.30	0.40	0.50
micro F1	0.891	0.893	0.877	0.855	0.830	0.747
macro Re	0.722	0.720	0.693	0.662	0.588	0.524
macro Pr	0.791	0.793	0.685	0.806	0.791	0.802

Table 6: The simple GW test run.

category decreases from almost 90 to almost 20. This sharp decay influences the macro-averaged performance of the system. In fact, while the macro-Recall retains the same values, the macro-Precision is cut-off by almost 10%. This fact indicates that the system is incorrectly assigning the stories to all the categories of the domain rather than to the biggest ones only. As the *ThresP* gets bigger, the performance of the system shows the classic behavior in which the Recall tends to 0 and the Precision to 1.

For the ‘what if?’ test run we didn’t expect such good results. The simple proportional weight assignment performed as well as the other approaches, nevertheless the performance decays slightly more rapidly. This test was very important for us to confirm the idea of using a simplified learning approach to solve the problem of TC.

The second and the third test runs show almost the same values. The one-per-thousand differences are not to be considered relevant. At the moment, we cannot draw any conclusion about which of the three learning approaches will perform better in general. We plan to investigate this matter testing on the complete Reuters-21578 and on other test collections.

Comparative Results

Dumais et al. tested a number of inductive learning algorithms on the same Reuters-21578 ModApte split we used ([Dumais et al., 1998]). These results are briefly summarized in Table 7 for the 10 most frequent categories. The

Method	Performance
FindSim	0.64
Naive Bayes	0.81
BayesNets	0.85
Tree	0.88
Linear SVM	0.92

Table 7: Microaveraged F1-measure of the 10 most frequent categories of Reuters-21578, reported by Dumais

FindSim method is a variant of Rocchio’s method for relevance feedback([Rocchio, 1971]). The Naive Bayes classifier is constructed by using the training data to estimate the probability of each category given the document feature values of a new instance. A discussion of the independence assumption of naive bayes classifier can be found in ([Lewis, 1998]). BayesNets is a bayesian network, that uses 2-dependence Bayesian (see [Sahami, 1996] for another example of Bayes nets for classification). Tree is a Decision Tree approach described in ([Chickering et al., 1997]). Meanwhile, Linear SVM is a linear hyperplane that separates a set of positive examples from a set of negative ones ([Joachims, 1998]).

Both SVMs and Decision Trees produce very high overall classification accuracy. As the results of Table 7 are directly comparable with ours, our Gaussian Weighting approach is placed just behind the SVMs, which are known to be the best machine learning method in the field of text classification, with a micro-averaged F1-measure of 0.893 of the first two test runs. This result is important for two reasons: it demonstrates that this new learning method does perform well on this particular task, and encourage us to further investigate other fields of application, in which GW's simplicity could perform as well as other state-of-the-art machine learning methods.

5 Future Work

Two are the most compelling issues that we are going to complete in the future work are two:

- Incorporate a wrapper phase [Kohavi and John, 1997] to run systematically varying experiments with (i) using Expressiveness or not, (ii) varying values of ThresP and ThresE, (iii) varying values of GW variance, and (iv) using a Gaussian function or not;
- Test the full Reuters benchmark data rather than restricting ourselves to only the 10 most frequent categories, and on other collections like 20 Newsgroups² or Med-Line³.

6 Conclusions

In this paper we addressed the problem of finding a simple learning algorithm that uses naive, human comprehensible parameters. A very accurate text classifier can be learned automatically by using only two information parameters: presence P and expressivity E . The algorithm proposed here has, in the worst case, a quadratic computational cost, with respect to the number of categories of documents. We tested GW on the Reuters-21578 benchmark and calculated the micro-averaged F1-measure to directly compare our results with the ones of the literature, and the macro-Recall and macro-Precision. The results achieved are satisfactory and can be compared to most state-of-the-art machine learning methods applied to TC.

Acknowledgments

We are grateful to the anonymous reviewers for their insightful and helpful comments. The authors would also like to thank Prof. Maristella Agosti for her valuable comments and suggestions to improve the paper.

References

[Chickering *et al.*, 1997] D. Chickering, D. Heckerman, and C. Meek. A bayesian approach for learning bayesian networks with local structure. In *Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 307–315. Springer Verlag, Heidelberg, DE, 1997.

[Dumais *et al.*, 1998] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, pages 148–155. ACM Press, 1998.

[Hull, 1994] D.A. Hull. *Information Retrieval Using Statistical Classification*. PhD thesis, University of Stanford, 1994.

[Joachims, 1998] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[Kohavi and John, 1997] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[Lewis, 1998] D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[Rocchio, 1971] J.J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. G.Salton editor, 1971.

[Sahami, 1996] M. Sahami. Learning limited dependence Bayesian classifiers. In *Second International Conference on Knowledge Discovery in Databases*, 1996.

[Sebastiani, 2002] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, Vol. 34, No. 1, pp.1-47, 2002.

[Yang and Liu, 1999] Y. Yang and X. Liu. A re-examination of text categorization methods. In *22nd Annual International SIGIR*, pages 42–49, Berkley, 1999.

[Yang and Pedersen, 1997] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

²www.ai.mit.edu/~jrennie/20Newsgroups/

³www.nlm.nih.gov

Learning to Classify Texts Using Positive and Unlabeled Data

Xiaoli Li

School of Computing
National University of Singapore/
Singapore-MIT Alliance
Singapore 117543
lixl@comp.nus.edu.sg

Bing Liu

Department of Computer Science
University of Illinois at Chicago
851 South Morgan Street
Chicago, IL 60607-7053
liub@cs.uic.edu

Abstract

In traditional text classification, a classifier is built using labeled training documents of every class. This paper studies a different problem. Given a set P of documents of a particular class (called *positive class*) and a set U of unlabeled documents that contains documents from class P and also other types of documents (called *negative class documents*), we want to build a classifier to classify the documents in U into documents from P and documents not from P . The key feature of this problem is that there is no labeled negative document, which makes traditional text classification techniques inapplicable. In this paper, we propose an effective technique to solve the problem. It combines the Rocchio method and the SVM technique for classifier building. Experimental results show that the new method outperforms existing methods significantly.

1 Introduction

Text classification is an important problem and has been studied extensively in information retrieval and machine learning. To build a text classifier, the user first collects a set of training examples, which are labeled with pre-defined classes (labeling is often done manually). A classification algorithm is then applied to the training data to build a classifier. This approach to building classifiers is called *supervised learning/classification* because the training examples/documents all have pre-labeled classes.

This paper studies a special form of *semi-supervised text classification*. This problem can be regarded as a two-class (*positive* and *negative*) classification problem, where there are only labeled *positive* training data, but no labeled *negative* training data. Due to the lack of negative training data, the classifier building is thus *semi-supervised*. Since traditional classification techniques require both labeled positive and negative examples to build a classifier, they are thus not suitable for this problem. Although it is possible to manually label some negative examples, it is labor-intensive and very time consuming. In this research, we want to build a classifier using only a set of positive examples and a set of unlabeled examples. Collecting unlabeled examples or documents is normally easy and inexpensive in many text or Web page

domains, especially those involving online sources [Nigam *et al.*, 1998; Liu *et al.*, 2002].

In [Liu *et al.*, 2002; Yu *et al.*, 2002], two techniques are proposed to solve the problem. One is based on the EM algorithm [Dempster *et al.*, 1977] (called S-EM) and the other is based on Support Vector Machine (SVM) [Vapnik, 1995] (called PEBL). However, both techniques have some major shortcomings. S-EM is not accurate because of its weak classifier. PEBL is not robust because it performs well in certain situations and fails badly in others. We will discuss these two techniques in detail in Section 2 and compare their results with the proposed technique in Section 4.

As discussed in our earlier work [Liu *et al.*, 2002], positive class based learning occurs in many applications. With the growing volume of text documents on the Web, Internet news feeds, and digital libraries, one often wants to find those documents that are related to one's interest. For instance, one may want to build a repository of machine learning (ML) papers. One can start with an initial set of ML papers (e.g., an ICML Proceedings). One can then find those ML papers from related online journals or conference series, e.g., AI journal, AAAI, IJCAI, etc.

The ability to build classifiers without negative training data is particularly useful if one needs to find positive documents from many text collections or sources. Given a new collection, the algorithm can be run to find those positive documents. Following the above example, given a collection of AAAI papers (unlabeled set), one can run the algorithm to identify those ML papers. Given a set of SIGIR papers, one can run the algorithm again to find those ML papers. In general, one cannot use the classifier built using the AAAI collection to classify the SIGIR collection because they are from different domains. In traditional classification, labeling of negative documents is needed for each collection. A user would obviously prefer techniques that can provide accurate classification without manual labeling any negative documents.

This paper proposes a more effective and robust technique to solve the problem. It is based on the Rocchio method [Rocchio, 1971] and SVM. The idea is to first use Rocchio to extract some reliable negative documents from the unlabeled set and then apply SVM iteratively to build and to select a classifier. Experimental results show that the new method outperforms existing methods significantly.

2 Related Work

Many studies on text classification have been conducted in the past. Existing techniques including Rocchio, naive Bayes (NB), SVM, and many others [e.g., Rocchio, 1971; Lewis & Gale, 1994; Vapnik, 1995; Joachims, 1999; Nigam *et al.*, 1998; Yang & Liu, 1999]. These existing techniques, however, all require labeled training data of all classes. They are not designed for the positive class based classification.

In [Denis, 1998], a theoretical study of PAC learning from positive and unlabeled examples under the statistical query model [Kearns, 1998] is reported. [Letouzey *et al.*, 2000] presents an algorithm for learning using a modified decision tree algorithm based on the statistical query model. It is not for text classification, but for normal data. In [Muggleton, 2001], Muggleton studied the problem in a Bayesian framework where the distribution of functions and examples are assumed known. The result is similar to that in [Liu *et al.*, 2002] in the noiseless case. [Liu *et al.*, 2002] extends the result to the noisy case. The purpose of this paper is to propose a robust practical method to improve existing techniques.

In [Liu *et al.*, 2002], Liu *et al.* proposed a method (called S-EM) to solve the problem in the text domain. It is based on naïve Bayesian classification (NB) and the EM algorithm [Dempster *et al.*, 1977]. The main idea of the method is to first use a spy technique to identify some *reliable negative documents* from the unlabeled set. It then runs EM to build the final classifier. Since NB is not a strong classifier for texts, we will show that the proposed technique is much more accurate than S-EM.

In [Yu *et al.*, 2002], Yu *et al.* proposed a SVM based technique (called PEBL) to classify Web pages given positive and unlabeled pages. The core idea is the same as that in [Liu *et al.*, 2002], i.e., (1) identifying a set of reliable negative documents from the unlabeled set (called *strong negative documents* in PEBL), and (2) building a classifier using SVM. The difference between the two techniques is that for both steps the two techniques use different methods. In [Yu *et al.*, 2002], strong negative documents are those documents that do not contain any features of the positive data. After a set of strong negative documents is identified, SVM is applied iteratively to build a classifier. PEBL is sensitive to the number of positive examples. When the positive data is small, the results are often very poor. The proposed method differs from PEBL in that we perform negative data extraction from the unlabeled set using the Rocchio method and clustering. Although our second step also runs SVM iteratively to build a classifier, there is a key difference. Our technique selects a good classifier from a set of classifiers built by SVM, while PEBL does not. The proposed method performs well consistently under a variety of conditions.

In [Scholkopf *et al.*, 1999], one-class SVM is proposed. This technique uses only positive data to build a SVM classifier. However, our experiment results show that its classification performance is poor.

Another line of related work is the use of a small labeled set of every class and a large unlabeled set for learning [Ni-

gam *et al.*, 1998; Blum & Mitchell, 1998; Goldman & Zhou, 2000; Basu *et al.*, 2002; Muslea *et al.*, 2002; Bockhorst & Craven, 2002]. It has been shown that the unlabeled data helps classification. This is different from our work, as we do not have any labeled negative training data.

3 The Proposed Technique

This section presents the proposed approach, which also consists of two steps: (1) extracting some reliable negative documents from the unlabeled set, (2) applying SVM iteratively to build a classifier. Below, we present these two steps in turn. In Section 4, we show that our technique is much more robust than existing techniques.

3.1 Finding reliable negative documents

This sub-section gives two methods for finding reliable negative documents: (1) Rocchio and (2) Rocchio with clustering. The second method often results in better classifiers, especially when the positive set is small. The key requirement for this step is that the identified negative documents from the unlabeled set must be reliable or pure, i.e., with no or very few positive documents, because SVM (used in our second step) is very sensitive to noise.

3.1.1 Method 1: Rocchio

This method treats the entire unlabeled set U as negative documents and then uses the positive set P and U as the training data to build a Rocchio classifier. The classifier is then used to classify U . Those documents that are classified as negative are considered (reliable) negative data, denoted by RN . The algorithm is shown in Figure 1.

1. Assign the unlabeled set U the negative class, and the positive set P the positive class;
2. Let $\vec{c}^+ = \alpha \frac{1}{|P|} \sum_{d \in P} \frac{\vec{d}}{\|\vec{d}\|} - \beta \frac{1}{|U|} \sum_{d \in U} \frac{\vec{d}}{\|\vec{d}\|}$;
3. Let $\vec{c}^- = \alpha \frac{1}{|U|} \sum_{d \in U} \frac{\vec{d}}{\|\vec{d}\|} - \beta \frac{1}{|P|} \sum_{d \in P} \frac{\vec{d}}{\|\vec{d}\|}$;
4. **for** each document d' in U **do**
5. **if** $\text{sim}(\vec{c}^+, \vec{d}') \leq \text{sim}(\vec{c}^-, \vec{d}')$ **then**
6. $RN = RN \cup \{d'\}$;

Figure 1: Rocchio extraction using U as negative

In Rocchio classification, each document d is represented as a vector [Salton & McGill, 1983], $\vec{d} = (q_1, q_2, \dots, q_n)$. Each element q_i in $\vec{d}$ represents a word w_i and is calculated as the combination of *term frequency* (tf) and *inverse document frequency* (idf), i.e., $q_i = tf_i * idf_i$. tf_i is the number of times that word w_i occurs in d , while

$$idf_i = \log(|D| / df(w_i)).$$

Here $|D|$ is the total number of documents and $df(w_i)$ is the number of documents where word w_i occurs at least once.

Building a classifier is achieved by constructing positive and negative prototype vectors $\vec{c}^+$ and $\vec{c}^-$ (lines 2 and 3 in Figure 1). α and β parameters adjust the relative impact of positive and negative training examples. $\alpha = 16$ and $\beta = 4$ are

recommended in [Buckley *et al.*, 1994].

In classification, for each test document $\vec{d}'$, it simply uses the cosine measure [Salton & McGill, 1983] to compute the similarity (*sim*) of $\vec{d}'$ with each prototype vector. The class whose prototype vector is more similar to $\vec{d}'$ is assigned to the test document (lines 4-6 in Figure 1). Those documents classified as negative form the negative set RN .

In general, it is also possible to use other classification methods (e.g., naïve Bayesian, SVM) to extract negative data from the unlabeled set. However, not all classification techniques perform well in the absence of a “pure” negative training set. We experimented with SVM, naïve Bayesian and Rocchio, and observed that Rocchio produces the best results. It is able to extract a large number of true negative documents from U with very high precision.

The reason that Rocchio works well is as follow: In our positive class based learning, the unlabeled set U typically has the following characteristics:

1. The proportion of positive documents in U is very small. Thus, they do not affect $\vec{c}^-$ very much.
2. The negative documents in U are of diverse topics. Thus, in the vector space, they cover a very large region.

Since the positive set P is usually homogenous (focusing on one topic), they cover a much smaller region in the vector space. Assume that there is a true decision surface S that separates positive and negative data. Then, the positive prototype vector $\vec{c}^+$ will be closer to S than the negative prototype vector $\vec{c}^-$ due to the vector summation in Rocchio (line 2 and 3 of Figure 1). Because of this reason, when we apply the similarity computation (in line 5 of Figure 1) to classify documents, many negative documents will be classified as positive because they are closer to $\vec{c}^+$. This explains why Rocchio can extract negative documents with high precision, and positive documents with low precision (but very high recall). Our experiments show that this is indeed the case.

After some reliable negative documents are found from U by Rocchio, we can run SVM using P and RN . Experimental results show that this simple approach works very well. Note that we could actually use $\vec{c}^+$ and $\vec{c}^-$ in Figure 1 as the final classifier without using SVM. However, it does not perform well because it tends to classify too many negative documents as positive documents (low precision for positive). SVM using RN and P as training data can correct the situation and produce a more accurate final classifier.

3.1.2 Method 2: Rocchio with Clustering

Rocchio is a linear classifier based on cosine similarity. When the decision boundary is non-linear or does not conform to the separating plane resulted from cosine similarity, Rocchio may still extract some positive documents and put them in RN . This will damage the SVM classifier later. Below, we propose an enhancement to the Rocchio approach in order to purify RN further, i.e., to discard some likely positive documents from RN .

Figure 2 shows a possible scenario where some positive documents in U may still be in RN . $\vec{c}^+$ and $\vec{c}^-$ represent the positive and negative prototype vectors respectively. H is the decision hyperplane produced by Rocchio. It has the same

distance (similarity) to $\vec{c}^+$ and $\vec{c}^-$. If a document is located on the right-hand-side of H , it is classified as negative otherwise it is classified as positive. However, the positive and negative classes in the data cannot be separated by H well. In Figure 2, the positive documents within the regions 1 and 2 will be misclassified as negative documents.

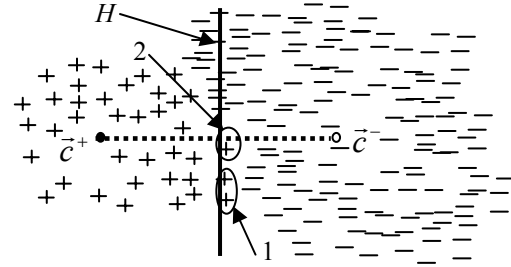


Figure 2: Rocchio classifier is unable to remove some positive examples in U

In order to further “purify” RN , we need to deal with the above problem. We propose the following approach.

This approach uses clustering to partition RN into many similarity groups (or clusters). It then uses Rocchio again to build a classifier using each cluster and the positive set P . The classifier is then applied to identify likely positive documents in the cluster and delete them. The idea is to identify and remove some positive documents in RN in a localized manner, which gives better accuracy. Since both the cluster and the positive set are homogeneous, they allow Rocchio to build better prototype vectors.

The clustering technique that we use is k -means, which is an efficient technique. The k -means method needs the input cluster number k . From our experiments in Section 4, we observe that the choice of k does not affect classification results much as long as it is not too small. Too few clusters may not be effective because the documents in a cluster can still be diverse. Then, Rocchio still cannot build an accurate classifier. Too many clusters will not cause much problem.

This new method can further purify RN without removing too many negative documents from RN . The detailed algorithm is given in Figure 3.

Note that in line 1 we still perform the initial Rocchio extraction as discussed in Section 3.1.1. Thus, the input data for clustering is RN , not U . The reason is that the purity of RN is higher, which allows us to produce better clusters without the influence of too many noisy (positive) documents in U . Lines 2-3 perform the standard k -means clustering of RN , which produces k clusters, $N_1, N_2, \dots, N_k$. Based on these clusters, we construct a positive prototype vector $\vec{p}_j$ and a negative prototype vector $\vec{n}_j$ for the positive set P and each cluster N_j ($j \in \{1, 2, \dots, k\}$) (lines 5 and 6). Line 8 starts the extraction. For each document $d_i \in RN$, we try to find the nearest positive prototype vector $\vec{p}_j$ to d_i (line 9). If the similarity $\text{sim}(\vec{p}_j, d_i)$ is less than the similarity between d_i and any negative prototype vector $\vec{n}_j$, we put it in our final negative set RN' (line 10). The reason for using the procedures in lines 9 and 10 is that we want to be conservative in removing likely positive documents from RN , i.e., not to remove too many negative documents.

1. Perform the algorithm in Figure 1 and generate the initial negative set RN ;
2. Choose k initial cluster centers $\{O_1, O_2, \dots, O_k\}$ randomly from RN ;
3. Perform k -means clustering to produce k clusters, i.e. $N_1, N_2, \dots, N_k$;
4. **for** $j = 1$ to k
5.
$$\bar{n}_j = \alpha \frac{1}{|N_j|} \sum_{d \in N_j} \frac{\bar{d}}{\|\bar{d}\|} - \beta \frac{1}{|P|} \sum_{d \in P} \frac{\bar{d}}{\|\bar{d}\|};$$
6.
$$\bar{p}_j = \alpha \frac{1}{|P|} \sum_{d \in P} \frac{\bar{d}}{\|\bar{d}\|} - \beta \frac{1}{|N_j|} \sum_{d \in N_j} \frac{\bar{d}}{\|\bar{d}\|};$$
7. $RN' = \{\};$
8. **for** each document $d_i \in RN$ **do**
9. Find the nearest positive prototype vector $\bar{p}_v$ to d_i , where $v = \arg \max_j \text{sim}(\bar{p}_j, d_i)$;
10. **if** there exist a $\bar{n}_j$ ($j = 1, 2, \dots, k$), s.t. $\text{sim}(\bar{p}_v, d_i) \leq \text{sim}(\bar{n}_j, d_i)$ **then**
11. $RN' = RN' \cup \{d_i\};$

Figure 3: Rocchio extraction with clustering.

After RN' is determined, we can build the final classifier using SVM, which takes P and RN' as the positive and negative training data respectively.

3.2 Step 2: Classifier building

This step builds the final classifier by running SVM iteratively with the document sets P and RN or RN' depending on the method used in step 1. Let Q be the set of the remaining unlabeled documents, $Q = U - RN$ (or $U - RN'$). The algorithm for this second step is given in Figure 4.

1. Every document in P is assigned the class label +1;
2. Every document in RN (RN') is assigned the label -1;
3. Use P and RN (or RN') to train a SVM classifier S_i , with $i = 1$ initially and $i = i+1$ with each iteration (line 3-5);
4. Classify Q using S_i . Let the set of documents in Q that are classified as negative be W ;
5. **if** $W = \{\}$ **then** *stop*;
else $Q = Q - W$;
 $RN = RN \cup W$ (or $RN' = RN' \cup W$);
goto (3);
6. Use the last SVM classifier S_{last} to classify P ;
7. **if** more than 5% positive are classified as negative **then** use S_1 as the final classifier;
else use S_{last} as the final classifier;

Figure 4: Constructing the final classifier.

The reason that we run SVM iteratively (lines 3-5) is that the reliable negative set RN from step 1 may not be sufficiently large to build the best classifier. SVM classifiers (S_i in line 3) can be used to iteratively extract more negative documents from Q . The iteration stops when there is no negative document that can be extracted from Q (line 5).

There is, however, a danger in running SVM iteratively. Since SVM is very sensitive to noise, if some iteration of SVM goes wrong and extracts many positive documents

from Q and put them in the negative set RN , then the last SVM classifier will be extremely poor. This is the problem with PEBL, which also runs SVM iteratively. In our algorithm, we decide whether to use the first SVM classifier or the last one (line 7). Basically, we use the SVM classifier at convergence (called S_{last} , line 6) to classify the positive set P . If too many ($> 5\%$) positive documents in P are classified as negative, it indicates that SVM has gone wrong. We then use the first SVM classifier (S_1). Otherwise, we use S_{last} as the final classifier. We choose 5% as the threshold because we want to be very conservative. Since SVM is very sensitive to noise, when noise is large, the resulting classifier is often very poor. Since our first classifier is always quite strong, even without catching the last classifier which may be better, it is acceptable.

Note that this strategy is not applicable to PEBL because PEBL's step 1 extracts too few negative documents from U . Its first SVM classifier is thus very inaccurate. Step 1 of our proposed technique is able to extract a large number of negative documents from U . Hence, our first SVM classifier is always quite good, although it may not be the best.

Note also that neither the first nor the last SVM classifier may be the best. This is the case for both PEBL and our algorithm. In many cases, a classifier somewhere in the middle is the best. However, it is hard to catch it. We leave this issue to our future research.

4 Empirical Evaluation

This section evaluates the proposed techniques using the Reuters-21578 text collection¹, which is commonly used in evaluating text classification methods. We will also compare the proposed techniques with S-EM and PEBL.

The Reuters-21578 collection contains 21578 text documents. We used only the most populous 10 out of the 135 topic categories. Table 1 gives the number of documents in each of the ten topic categories. In our experiments, each category is used as the positive class, and the rest of the categories as the negative class. This gives us 10 datasets.

Table 1: Categories and numbers of documents in Reuters

acq	corn	crude	earn	grain	interest	money	ship	trade	wheat
2369	238	578	3964	582	478	717	286	486	283

Our task is to identify positive documents from the unlabeled set. The construction of each dataset for our experiments is done as follows: Firstly, we randomly select $a\%$ of the documents from the positive class and the negative class, and put them into P and N classes respectively. The remaining $(1 - a\%)$ documents from both classes form the unlabeled set U . Our training set for each dataset consists of P and U . N is not used as we do not want to change the class distribution of the unlabeled set U . We also used U as the test set in our experiments because our objective is to recover those positive documents in U . We do not need separate test sets. In our experiments, we use 7 a values to create different settings, i.e., 5%, 15%, 25%, 35%, 45%, 55% and 65%.

¹ <http://www.research.att.com/~lewis/reuters21578.html>

4.1 Evaluation measures

Two popular measures used for evaluating text classification are the F value and breakeven point. F value is defined as, $F = 2pr/(p+r)$, where p is the precision and r is the recall. F value measures the performance of a system on a particular class (in our case, the positive class). Breakeven point is the value at which recall and precision are equal. However, breakeven point is not suitable for our task as it only evaluates the sorting order of class probabilities of documents. It does not give indication of classification performance. F value, on the other hand, reflects an average effect of both precision and recall. This is suitable for our purpose, as we want to identify positive documents. It is undesirable to have either too small a precision or too small a recall.

In our experimental results, we also report *accuracies*. It, however, should be noted that accuracy does not fully reflect the performance of our system, as our datasets has a large proportion of negative documents. We believe this reflects realistic situations. In such cases, the accuracy can be high, but few positive documents may be identified.

4.2 Experimental results

We now present the experimental results. We use Roc-SVM and Roc-Clu-SVM to denote the classification techniques that employ Rocchio and Rocchio with clustering to extract reliable negative set respectively (both methods use SVM for classifier building). We observe from our experiments that using Rocchio and Rocchio with clustering alone for classification do not provide good classification results. SVM improves the results significantly.

For comparison, we include the classification results of NB, S-EM and PEBL. Here, NB treats all the documents in the unlabeled set as negative. SVM for the noisy situation (U as negative) performs poorly because SVM does not tolerate noise well. Thus, its results are not listed (in many cases, its F values are close to 0). In both Roc-SVM and Roc-Clu-SVM, we used the linear SVM as [Yang & Liu, 1999] reports that linear SVM gives slightly better results than non-linear models on the Reuters dataset. For our experiments, we implemented PEBL as it is not publicly available. For SVM, we used the SVM^{light} system [Joachims, 1999]. PEBL also used SVM^{light}. S-EM is our earlier system. It is publicly available at <http://www.cs.uic.edu/~liub/S-EM/S-EM-download.html>.

Table 2 shows the classification results of various techniques in terms of F value and accuracy (Acc) for $a = 15\%$ (the positive set is small). The final row of the table gives the average result of each column. We used 10 clusters (i.e., $k = 10$) for k -means clustering in Roc-Clu-SVM (later we will see that the number of clusters does not matter much).

We observe that Roc-Clu-SVM produces better results than Roc-SVM. Both Roc-SVM and Roc-Clu-SVM outperform NB, S-EM and PEBL. PEBL is extremely poor in this case. In fact, PEBL performs poorly when the number of positive documents is small. When the number of positive documents is large, it usually performs well (see Table 3 with $a = 45\%$). Both Roc-SVM and Roc-Clu-SVM perform well consistently. We summarize the average F value results of all a settings in Figure 5. Due to space limitations, we are unable

to show the accuracy results (as noted earlier, accuracy does not fully reflect the performance of our system).

Table 2: Experiment results for $a = 15\%$.

	NB		S-EM		PEBL		Roc-SVM		Roc-Clu-SVM	
	F	Acc	F	Acc	F	Acc	F	Acc	F	Acc
acq	0.744	0.920	0.876	0.954	0.001	0.817	0.846	0.948	0.867	0.952
corn	0.452	0.983	0.452	0.983	0.000	0.982	0.804	0.993	0.822	0.995
crude	0.731	0.979	0.820	0.984	0.000	0.955	0.782	0.983	0.801	0.985
earn	0.910	0.949	0.947	0.968	0.000	0.693	0.858	0.924	0.891	0.959
grain	0.728	0.977	0.807	0.982	0.020	0.955	0.845	0.987	0.869	0.986
interest	0.609	0.972	0.648	0.970	0.000	0.963	0.704	0.976	0.724	0.983
money	0.754	0.974	0.793	0.975	0.000	0.945	0.768	0.973	0.785	0.973
ship	0.701	0.989	0.742	0.990	0.008	0.978	0.578	0.986	0.596	0.989
trade	0.627	0.977	0.698	0.979	0.000	0.962	0.759	0.983	0.778	0.985
wheat	0.579	0.982	0.611	0.979	0.000	0.978	0.834	0.994	0.854	0.997
Avg	0.684	0.970	0.739	0.976	0.003	0.923	0.778	0.975	0.799	0.980

Table 3: Experiment results for $a = 45\%$.

	NB		S-EM		PEBL		Roc-SVM		Roc-Clu-SVM	
	F	Acc	F	Acc	F	Acc	F	Acc	F	Acc
acq	0.802	0.934	0.891	0.959	0.891	0.963	0.905	0.964	0.909	0.965
corn	0.502	0.972	0.517	0.970	0.663	0.991	0.635	0.990	0.645	0.992
crude	0.833	0.985	0.850	0.986	0.798	0.984	0.811	0.985	0.811	0.985
earn	0.924	0.956	0.950	0.970	0.956	0.974	0.886	0.937	0.923	0.956
grain	0.768	0.975	0.772	0.975	0.900	0.992	0.903	0.992	0.903	0.992
interest	0.617	0.959	0.614	0.956	0.770	0.983	0.614	0.957	0.616	0.957
money	0.751	0.969	0.760	0.968	0.714	0.973	0.764	0.969	0.764	0.970
ship	0.791	0.991	0.806	0.991	0.672	0.989	0.829	0.993	0.843	0.994
trade	0.678	0.973	0.693	0.972	0.728	0.982	0.728	0.982	0.728	0.982
wheat	0.595	0.973	0.595	0.972	0.783	0.992	0.779	0.992	0.792	0.994
Avg	0.726	0.969	0.745	0.972	0.788	0.982	0.785	0.976	0.793	0.979

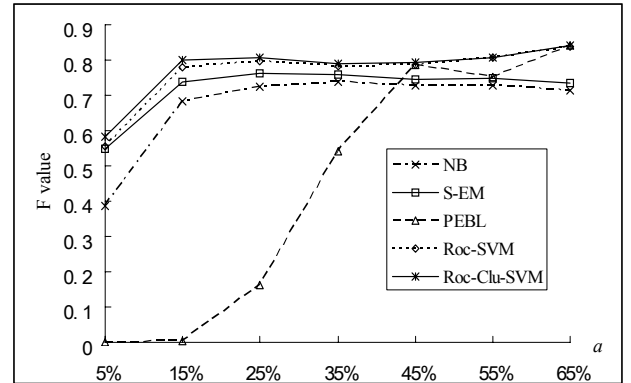


Figure 5 Average results for all a settings

From Figure 5, we can draw the following conclusions:

1. S-EM's results are quite consistent under different settings. However, its results are worse than Roc-SVM and Roc-Clu-SVM. The reason is that the negative documents extracted from U by its spy technique are not that reliable, and S-EM uses a weaker classifier, NB.
2. PEBL's results are extremely poor when the number of positive documents is small. We believe that this is because its strategy of extracting the initial set of strong negative documents could easily go wrong without sufficient positive data. Even when the number of positive documents is large, it may also go wrong. For example, for $a = 55\%$, one F value (for the dataset, trade) is only 0.073. This shows that PEBL is not robust.
3. Both Roc-SVM and Roc-Clu-SVM are robust with different numbers of positive documents. This is important

because in practice one does not know how many positive documents are sufficient. Using a smaller set of positive documents also reduce the manual labeling effort.

From Figure 5, we also observe that Roc-Clu-SVM is slightly better than Roc-SVM in both F value and accuracy, especially when a is small because pure RN is more important in such cases. When a is large, their results are similar. In practice, Roc-SVM may be sufficient due to its simplicity, efficiency and good results.

We also ran one-Class SVM using the LIBSVM package (<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>). It gives very poor results (mostly with F values less than 0.4). Due to space limitations, we do not list their results here.

Comparison of different number of clusters: As discussed above, the number of clusters used in Roc-Clu-SVM makes little difference to the final classifier. We now provide experiment results in Figure 6 to support this claim by using different numbers of clusters (here, $a = 25\%$ and for other a values, the results are similar). The cluster number k varies from 2 to 30. We observe that the choice of k has little influence on the results as long as it is not too small.

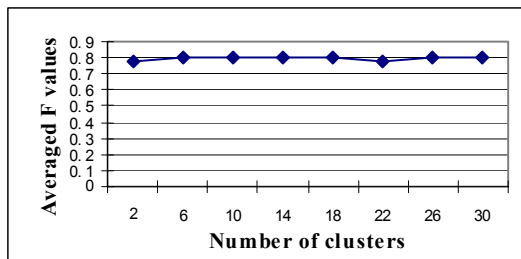


Figure 6: Averaged F values of different clusters ($a=25\%$)

Execution times: Our technique consists of two steps: extracting negative documents from U and iteratively running SVM. As SVM is a standard technique, we will not discuss it here. If we only use Rocchio for the first step, it is very efficient because Rocchio is a linear algorithm, $O(|U \cup P|)$. If we use Rocchio with clustering, more time is needed because of k -means clustering. The time complexity of k -means is $O(k * |RN| * I)$, where k is the number of clusters, $|RN|$ is the size of the reliable negative set, and I is the number of iterations. Since k and I are normally small, k -means is generally regarded as a linear algorithm $O(|RN|)$. Thus, the time complexity of the extraction step of Roc-Clu-SVM is $O(|U \cup P|)$ since $|RN| < |U \cup P|$. In our experiments, every dataset takes less than 7 seconds for step one for Roc-Clu-SVM (on Pentium 800MHz PC with 256MB memory).

5 Conclusions

This paper studied the problem of text classification with only partial information, i.e., with only one class of labeled documents and a set of unlabeled documents. An effective technique is proposed to solve the problem. Our algorithm first utilizes the Rocchio classifier and/or clustering to extract a set of reliable negative documents from the unlabeled set, and then builds a SVM classifier iteratively. Experimental results show that the proposed technique is more robust than both S-EM and PEBL.

Acknowledgement: This research is supported by a research grant from A-STAR and NUS to the second author.

References

- [Basu *et al.*, 2002] S. Basu, A. Banerjee, & R. Mooney. Semi-supervised clustering by seeding. *ICML-2002*, 2002.
- [Blum & Mitchell, 1998] A. Blum & T. Mitchell. Combining labeled & unlabeled data with co-training. *COLT-98*.
- [Bockhorst & Craven, 2002] J. Bockhorst & M. Craven. Exploiting relations among concepts to acquire weakly labeled training data. *ICML-2002*, 2002.
- [Buckley *et al.*, 1994] C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. *SIGIR-94*, 1994.
- [Dempster *et al.*, 1977] A. Dempster, N. Laird & D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1977.
- [Denis, 1998] F. Denis. PAC learning from positive statistical queries. *ALT-98*, pages 112-126. 1998.
- [Goldman & Zhou, 2000] S. Goldman & Y. Zhou. Enhancing supervised learning with unlabeled data. *ICML-2000*.
- [Joachims, 1999] T. Joachims. Making large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.
- [Kearns, 1998] M. Kearns. Efficient noise-tolerant learning from statistical queries. *J. of the ACM*, 45:983-1006, 1998.
- [Letouzey *et al.*, 2000] F. Letouzey, F. Denis, R. Gilleron, & E. Grappa. Learning from positive and unlabeled examples. *ALT-00*, 2000.
- [Lewis & Gale, 1994] D. Lewis & W. Gale. A sequential algorithm for training text classifiers. *SIGIR-94*, 1994.
- [Liu *et al.*, 2002] B. Liu, W. Lee, P. Yu, & X. Li. Partially supervised classification of text documents. *ICML-2002*.
- [Muslea *et al.*, 2002] I. Muslea, S. Minton & C. Knoblock. Active + semi-supervised learning = robust multi-view learning. *ICML-2002*, 2002.
- [Muggleton, 2001] S. Muggleton. Learning from the positive data. *Machine Learning*, 2001, to appear.
- [Nigam *et al.*, 1998] K. Nigam, A. McCallum, S. Thrun, & T. Mitchell. Text classification from labeled and unlabeled documents using EM, *machine learning*, 2000.
- [Rocchio, 1971] J. Rocchio. Relevant feedback in information retrieval. In G. Salton (ed.). *The smart retrieval system: experiments in automatic document processing*, Englewood Cliffs, NJ, 1971.
- [Salton & McGill, 1983] G. Salton & M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
- [Scholkopf *et al.*, 1999] B. Scholkopf, J. Platt, J. Shawe, A. Smola & R. Williamson. Estimating the support of a high-dimensional distribution. *Technical Report MSR-TR-99-87*, Microsoft Research, 1999.
- [Vapnik, 1995] V. Vapnik. *The nature of statistical learning theory*, 1995.
- [Yang & Liu, 1999] Y. Yang & X. Liu. A re-examination of text categorization methods. *SIGIR-99*, 1999.
- [Yu *et al.*, 2002] H. Yu, J. Han, & K. Chang. PEBL: Positive example based learning for Web page classification using SVM. *KDD-02*, 2002.

LEARNING

TREE LEARNING

Inductive Learning in Less Than One Sequential Data Scan

Wei Fan, Haixun Wang, and Philip S. Yu

IBM T.J.Watson Research
Hawthorne, NY 10532

{weifan, haixun, psyu}@us.ibm.com

Shaw-Hwa Lo

Statistics Department, Columbia University
New York, NY 10027

slo@stat.columbia.edu

Abstract

Most recent research of scalable inductive learning on very large dataset, decision tree construction in particular, focuses on eliminating memory constraints and reducing the number of sequential data scans. However, state-of-the-art decision tree construction algorithms still require multiple scans over the data set and use sophisticated control mechanisms and data structures. We first discuss a general inductive learning framework that scans the dataset exactly once. Then, we propose an extension based on Hoeffding's inequality that scans the dataset less than once. Our frameworks are applicable to a wide range of inductive learners.

1 Introduction

Most recent research on scalable inductive learning over very large dataset focuses on eliminating memory-constraints and reducing the number of sequential data scans (or the total number of times that the training file is accessed from secondary storage), particularly for decision tree construction. State-of-the-art decision tree algorithms (SPRINT [Shafer *et al.*, 1996], RainForest [Gehrke *et al.*, 1998], and later BOAT [Gehrke *et al.*, 1999] among others) still scan the data multiple times, and employ rather sophisticated mechanisms in implementation. Most recent work [Hulten and Domingos, 2002] applies Hoeffding inequality to decision tree learning on streaming data in which a node is reconstructed iff it is statistically necessary. Besides decision tree, there hasn't been much research on reducing the number of data scans for other inductive learners. The focus of this paper is to propose a *general approach* for a wide range of inductive learning algorithms to scan the dataset *less than once* from the secondary storage. Our approach is applicable not only to decision trees but also to other learners, e.g., rule and naive Bayes learners.

Ensemble of classifiers has been studied as a general approach for scalable learning. Previously proposed meta-learning [Chan, 1996] reduces the number of data scans to 2. However, empirical studies have shown that the accuracy of the multiple model is sometimes lower than respective single model. Bagging [Breiman, 1996] and boosting [Freund and Schapire, 1997] are not scalable since both methods scan the dataset multiple times. Our proposed method scans the

dataset less than once and has been empirically shown to have higher accuracy than a single classifier.

Based on averaging ensemble, we propose a statistically-based multiple model inductive learning algorithm that scans the dataset less than once. Previous research [Fan *et al.*, 2002b] on averaging ensemble has shown that it is more efficient and accurate than both bagging and meta-learning. In this paper, we apply Hoeffding inequality to estimate the probability that the partial and complete models are equal in accuracy. When the probability is higher than a threshold, the algorithm stops model construction and returns the current model, resulting in less than one sequential scan of the dataset. Our objective is completely different from [Hulten and Domingos, 2002] on determining whether to change the shape of a decision tree. Unlike previous research [Hulten and Domingos, 2002; Gehrke *et al.*, 1999], our algorithm is not limited to decision tree, but is applicable to a wide range of inductive learners. When it is applied to decision tree learning, the accuracy is higher than a single decision tree. Another advantage is that the ensemble reduces the asymptotic complexity of the algorithm besides scanning less data. One important distinction is that we are interested in reducing sequential scan from secondary storage. Once the data can be held entirely in memory, the base learning algorithm is allowed to scan the data *in memory* multiple times.

2 One Scan

We first describe a strawman algorithm that scans the data set exactly once, then propose the extension that scans the data set less than once. The strawman algorithm is based on probabilistic modeling.

2.1 Probabilistic Modeling

Suppose $p(\ell_i|x)$ is the probability that x is an instance of class ℓ_i . In addition, we have a benefit matrix $b[\ell_{i'}, \ell_i]$ that records the benefit received by predicting an example of class $\ell_{i'}$ to be an instance of class ℓ_i . For traditional accuracy-based problems, $\forall i, b[\ell_i, \ell_i] = 1$ and $\forall i' \neq i, b[\ell_{i'}, \ell_i] = 0$. For cost-sensitive application such as credit card fraud detection, assume that the overhead to investigate a fraud is \$90 and $y(x)$ is the transaction amount, then $b[\text{fraud}, \text{fraud}] = y(x) - \90 and $b[\neg \text{fraud}, \text{fraud}] = -\90 . Using benefit matrix and probability, the expected benefit received by pre-

dicting x to be an instance of class ℓ_i is

$$\text{Expected Benefit: } e(\ell_i|x) = \sum_{\ell_{i'}} b[\ell_{i'}, \ell_i] \cdot p(\ell_{i'}|x) \quad (1)$$

Based on optimal decision policy, the best decision is the label with the highest expected benefit:

$$\ell_{\max} = \operatorname{argmax}_{\ell_i} e(\ell_i|x) \quad (2)$$

Assuming that $\ell(x)$ is the true label of x , the accuracy of the decision tree on a test data set ST is

$$\text{Accuracy: } A = \sum_{x \in \text{ST}} b[\ell(x), \ell_{\max}] \quad (3)$$

For traditional accuracy-based problems, A is always normalized by dividing $|\text{ST}|$; for cost-sensitive problems, A is usually represented in some measure of benefits such as dollar amount. For cost-sensitive problems, we sometimes use “total benefits” to mean accuracy.

2.2 The Strawman Algorithm

The strawman algorithm is based on averaging ensemble [Fan *et al.*, 2002b]. Assume that a data set S is partitioned into K disjoint subsets S_j with equal size. A base level model $\mathcal{C}_j$ is trained from each S_j . Given an example x , each classifier outputs individual expected benefit based on probability $p_j(\ell_{i'}|x)$

$$e_j(\ell_i|x) = \sum_{\ell_{i'}} b[\ell_{i'}, \ell_i] \cdot p_j(\ell_{i'}|x) \quad (4)$$

The averaged expected benefit from all base classifiers is therefore

$$E_K(\ell_i|x) = \frac{\sum_1^K e_j(\ell_i|x)}{K} \quad (5)$$

We then predict the class label with the highest expected return as in Eq[2].

$$\text{Optimal Decision: } L_K = \operatorname{argmax}_{\ell_i} E_K(\ell_i|x) \quad (6)$$

The obvious advantage is that the strawman algorithm scans the dataset exactly once as compared to two scans by meta-learning and multiple scans by bagging and boosting. In previous research [Fan *et al.*, 2002b], the accuracy by the strawman algorithm is also significantly higher than both meta-learning and bagging. [Fan *et al.*, 2002b] explains the statistical reason why the averaging ensemble is also more likely to have higher accuracy than a single classifier trained from the same dataset.

3 Less Than One Scan

The less-than-one-scan algorithm returns the current ensemble with k ($< K$) number of classifiers when the accuracy of current ensemble is the same as the complete ensemble with high confidence. For a random variable y in the range of $R = a - b$ with observed mean of $\bar{Y}$ after n observations, without any assumption about the distribution of y , Hoeffding's inequality states that with probability $\geq p$, the error of $\bar{Y}$ to the true mean is at most

$$\epsilon_n = R \left(\frac{1}{2n} \ln \left(\frac{1}{1-p} \right) \right)^{\frac{1}{2}} \quad (7)$$

Train(S, S_V, K, p)

Data : training set S , validation set S_V , partition number K , confidence p

Result : multiple model with size $k \leq K$

begin

partition S into K disjoint subsets of equal size $\{S_1, \dots, S_K\}$;

train $\mathcal{C}_1$ from S_1 ;

test $\mathcal{C}_1$ on S_V ;

$k \leftarrow 1$;

while $k \leq K$ **do**

train $\mathcal{C}_k$ from S_k ;

test $\mathcal{C}_k$ on S_V ;

$\forall \ell_i$, compute Hoeffding error $\epsilon_k(\ell_i)$ (Eq[8]);

confidence_satisfied $\leftarrow$ true;

for $x \in S_V$ **do**

$\forall \ell_i$, compute $E(\ell_i|x)$;

$E(\ell_a|x)$ is the highest and $E(\ell_b|x)$ is the second highest;

if $E(\ell_a|x) - E(\ell_b|x) \leq \epsilon_k(\ell_a) + \epsilon_k(\ell_b)$ **then**

confidence_satisfied $\leftarrow$ false;

break;

end

end if confidence_satisfied **then**

return $\{\mathcal{C}_1, \dots, \mathcal{C}_k\}$

else

$k \leftarrow k + 1$

end

end return $\{\mathcal{C}_1, \dots, \mathcal{C}_K\}$;

end

Algorithm 1: Less than one data scan

For finite population of size N , the adjusted error is

$$\epsilon_n = R \left(\frac{1-f}{2n} \ln \left(\frac{1}{1-p} \right) \right)^{\frac{1}{2}} \quad \text{where } f = \frac{n}{N} \quad (8)$$

The range R of expected benefit for class label ℓ_i can be found from the index to the data, or predefined. When k base models are constructed, the Hoeffding error ϵ_k can be computed by using Eq[8]. For data example x , assume that $E(\ell_a|x)$ is the highest expected benefit and $E(\ell_b|x)$ is the second highest, $\epsilon_k(\ell_a)$ and $\epsilon_k(\ell_b)$ are the Hoeffding errors. If $E(\ell_a|x) - \epsilon_k(\ell_a) > E(\ell_b|x) + \epsilon_k(\ell_b)$ or $E(\ell_a|x) - E(\ell_b|x) > \epsilon_k(\ell_a) + \epsilon_k(\ell_b)$, with confidence $\geq p$, the prediction on x by the complete multiple model and the current multiple model is the same. Otherwise, more base models will be trained. The algorithm is summarized in Algorithm 1.

Scan of validation set S_V

If an example x satisfies the confidence p when k classifiers are computed, there is no utility to check its satisfaction when more classifiers are computed. This is because that an ensemble with more classifiers is likely to be a more accurate model. In practice, we can only read and keep one example x from the validation set in memory at one time. We only read a new

instance from the validation set if the current set of classifiers satisfy the confidence test. In addition, we keep only the predictions on one example at any given time. This guarantees that the algorithm scans the validation dataset once with nearly no memory requirement.

Training Efficiency

The extra overhead of the Hoeffding-based less than one scan algorithm is the cost for the base classifiers to predict on the validation set and calculate the statistics. All these can be done in main memory. As discussed above, we can predict on one example from the validation set at any given time. Assume that we have k classifiers at the end and n is the size of the validation set, the total number of predictions is approximately $\frac{n \times k}{2}$ on average. The calculation of both mean and standard deviation can be done incrementally. We only need to keep $\sum X_i$ and $\sum X_i^2$ for just one example at anytime and calculate as follows:

$$\bar{X} = \frac{\sum X_i}{k} \quad (9)$$

$$\sigma^2(X) = \frac{\sum X_i^2 - k \cdot \bar{X}^2}{k - 1} \quad (10)$$

The average number of arithmetic operation is approximately $3 \times \frac{n \times k}{2}$.

The problem that the proposed algorithm solves is one in which the training set is very large and the I/O cost of data scan is the major overhead. When I/O cost is the bottleneck, the extra cost of prediction and statistical analysis is minimum.

4 Experiment

In empirical evaluation, we first compare the accuracy of the complete multiple model (one scan as well as less than one scan) and the accuracy of the single model trained from the same data set. We then evaluate the amount of data scan and accuracy of the less than one scan algorithm as compared to the one scan models. Additionally, we generate a dataset with biased distribution and study the results of the less than one scan algorithm.

4.1 Datasets

The first one is the well-known donation data set that first appeared in KDDCUP'98 competition. Suppose that the cost of requesting a charitable donation from an individual x is \$0.68, and the best estimate of the amount that x will donate is $Y(x)$. Its benefit matrix is:

	predict <i>donate</i>	predict $\neg$ <i>donate</i>
actual <i>donate</i>	$Y(x) - \$0.68$	0
actual $\neg$ <i>donate</i>	$-\$0.68$	0

As a cost-sensitive problem, the total benefit is the total amount of received charity minus the cost of mailing. The data has already been divided into a training set and a test set. The training set consists of 95412 records for which it is known whether or not the person made a donation and how much the donation was. The test set contains 96367 records for which similar donation information was not published until after the KDD'98 competition. We used the standard training/test set splits to compare with previous results. The feature subsets were based on the KDD'98 winning submission.

To estimate the donation amount, we employed the multiple linear regression method.

The second data set is a credit card fraud detection problem. Assuming that there is an overhead \$90 to dispute and investigate a fraud and $y(x)$ is the transaction amount, the following is the benefit matrix:

	predict <i>fraud</i>	predict $\neg$ <i>fraud</i>
actual <i>fraud</i>	$y(x) - \$90$	0
actual $\neg$ <i>fraud</i>	$-\$90$	0

As a cost-sensitive problem, the total benefit is the sum of recovered frauds minus investigation costs. The data set was sampled from a one year period and contained a total of 5M transaction records. We use data of the last month as test data (40038 examples) and data of previous months as training data (406009 examples).

The third data set is the adult data set from UCI repository. For cost-sensitive studies, we artificially associate a benefit of \$2 to class label **F** and a benefit of \$1 to class label **N**, as summarized below:

	predict F	predict N
actual F	\$2	0
actual N	0	\$1

We use the natural split of training and test sets, so the results can be easily replicated. The training set contains 32561 entries and the test set contains 16281 records.

4.2 Experimental Setup

We have selected three learning algorithms, decision tree learner C4.5, rule builder RIPPER, and naive Bayes learner. We have chosen a wide range of partitions, $K \in \{8, 16, 32, 64, 128, 256\}$. The validation dataset *SV* is the complete training set. All reported accuracy results were run on the test dataset.

4.3 Experimental Results

In Tables 1 and 2, we compare the results of the single classifier (which is trained from the complete dataset as a whole), one scan algorithm, and the less than one scan algorithm. We use the original "natural order" of the dataset. Later on in Section 4.4, we use a biased distribution. Each data set under study is treated both as a traditional and cost-sensitive problem. The less than one scan algorithm is run with confidence $p = 99.7\%$.

Accuracy Comparison

The baseline traditional accuracy and total benefits of the single model are shown in the two columns under "single" in Tables 1 and 2. These results are the baseline that the one scan and less than one scan algorithms should achieve. For the one scan and less than one scan algorithm, each reported result is the average of different multiple models with K ranging from 2 to 256. In Tables 1 and 2, the results are shown in two columns under accuracy and benefit. As we compare the respective results in Tables 1 and 2, the multiple model either significantly beat the accuracy of the single model or have very similar results. The most significant increase in both accuracy and total benefits is for the credit card data set. The total benefits have been increased by approximately \$7,000 $\sim$ \$10,000; the accuracy has been increased by approximately 1% $\sim$ 3%. For the KDDCUP'98 donation data set, the total

C4.5				
	Single	OneScan	LessThanOne	
			accuracy	datascan
Donation	94.94%	94.94%	94.94%	0.61
Credit Card	87.77%	90.37%	90.41%	0.62
Adult	84.38%	85.6%	85.0%	0.76

RIPPER				
	Single	OneScan	LessThanOne	
			accuracy	datascan
Donation	94.94%	94.94%	94.94%	0.45
Credit Card	90.14%	91.46%	91.42%	0.56
Adult	84.84%	86.1%	86.0%	0.59

NB				
	Single	OneScan	LessThanOne	
			accuracy	datascan
Donation	94.94	94.94%	94.94%	0.51
Credit Card	85.46%	88.64%	88.71%	0.57
Adult	82.86%	84.94%	84.6%	0.61

Table 1: Comparison of single model, one scan ensemble, and less than one scan ensemble for accuracy-based problems

benefit has been increased by \$1400 for C4.5 and \$250 for NB.

We next study the trends of accuracy when the number of partitions K increases. In Figure 1, we plot the accuracy and total benefits for the credit card data sets, and the total benefits for the donation data set with increasing number of partitions K . C4.5 was the base learner for this study. As we can see clearly that for the credit card data set, the multiple model consistently and significantly improve both the accuracy and total benefits over the single model by at least 1% in accuracy and \$40000 in total benefits for all choices of K . For the donation data set, the multiple model boosts the total benefits by at least \$1400. Nonetheless, when K increases, both the accuracy and total benefits show a slow decreasing trend. It would be expected that when K is extremely large, the results will eventually fall below the baseline.

Another important observation is that the accuracy and total benefit of the less than one scan algorithm are very close to the one scan algorithm. Their results are nearly identical.

Data Scan

In both Tables 1 and 2, we show the amount of data scanned for the less than one scan algorithm. It ranges from 40% (0.4) to about 70% (0.7). The adult dataset has the most amount of data scanned since the training set is the smallest and it requires more data partitions to compute an accurate model. C4.5 scans more data than both RIPPER and NB. This is because we generate the completely unpruned tree for C4.5, and there are wide variations among different models.

In Table 3, we compare the differences in accuracy and amount of training data when the validation set is either read completely by every classifier (under “Batch”) or sequentially only by newly computed base classifiers (under “Seq”) (as discussed in Section 3). Our empirical studies have found that “Batch” mode usually scans approximately 1% to 2% more training data, and the models computed by both methods are nearly identical in accuracy. The extra training data from the

C4.5				
	Single	OneScan	LessThanOne	
			benefit	datascan
Donation	\$13292.7	\$14702.9	\$14828	0.71
Credit Card	\$733980	\$804964	\$804914	0.65
Adult	\$16443	\$16435	\$16205	0.77

RIPPER				
	Single	OneScan	LessThanOne	
			benefit	datascan
Donation	\$0	\$0	\$0	0.47
Credit Card	\$712541	\$815612	\$815310	0.57
Adult	\$19725	\$19875	\$19615	0.62

NB				
	Single	OneScan	LessThanOne	
			benefit	datascan
Donation	\$13928	\$14282	\$14278	0.55
Credit Card	\$704285	\$798943	\$799104	0.59
Adult	\$16269	\$19169	\$16102	0.63

Table 2: Comparison of single model, one scan ensemble, and less than one scan ensemble for cost-sensitive problems

“batch” method is due to the fact that some examples satisfied by previously learned classifiers have high probability, but may not necessarily be satisfied by more base classifiers. However, our empirical studies have shown that the difference in how the validation set is handled doesn’t significantly influence the final model accuracy.

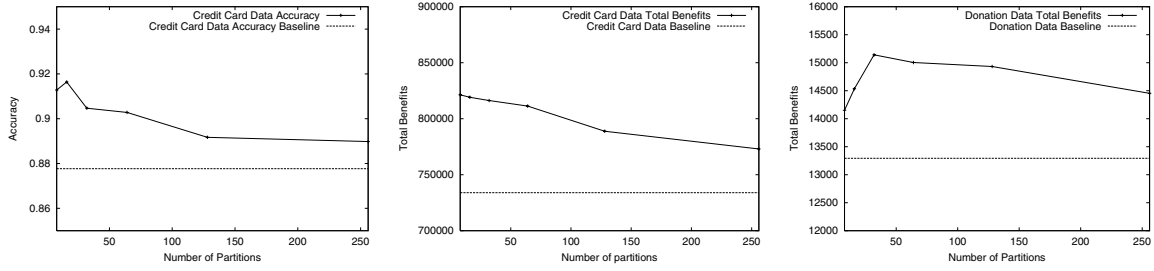
4.4 Biased Distribution

When a data is biased in its distribution, the less than one scan algorithm need to scan more data than uniform distribution to produce an accurate model. With the same amount of datascan, it may not have the same accuracy as uniform distribution. We have created a “trap” using the credit card dataset. We sorted the training data with increasing transaction amount. The detailed results are shown in Table 4(a) and (b). The accuracy (and total benefits) in Table 4(a) are nearly identical to the results of “natural distribution” as reported in Tables 1 and 2. However, the amount of datascan by the less than one scan algorithm is over 0.9 as compared to approximately 0.6 for natural distribution. As shown in Table 4(b), when the datascan is less than 0.9 (the confidence is not satisfied and less one scan will continue to compute more model), the total benefits are much lower. When distribution is biased, the variations in base classifiers’ prediction are wider. It requires more data to compute an accurate model and the less than one scan algorithm is performing in the correct way.

4.5 Training Efficiency

We recorded both the training time of the batch mode single model, and the training time of both the one scan algorithm and less than one scan algorithm *plus* the time to classify the validation set multiple times and statistical estimation. We then computed serial improvement, which is the ratio that the one scan and less than one scan algorithm are faster than training the single model. In Figure 2, we plot results for the credit card dataset using C4.5. Our

Figure 1: Plots of accuracy and total benefits for credit card data sets, and plot of total benefits for donation data set with respect to K when using C4.5 as the base learner



C4.5				
	Accuracy		Data Scan	
	Batch	Seq	Batch	Seq
Donation	94.94%	94.94%	0.64	0.61
Credit Card	90.39%	90.41%	0.62	0.62
Adult	85.1%	85.0%	0.78	0.76

RIPPER				
	Accuracy		Data Scan	
	Batch	Seq	Batch	Seq
Donation	94.94%	94.94%	0.48	0.45
Credit Card	91.44%	91.42%	0.56	0.55
Adult	85.9%	86.0%	0.62	0.59

NB				
	Accuracy		Data Scan	
	Batch	Seq	Batch	Seq
Donation	94.96%	94.94%	0.54	0.51
Credit Card	88.62%	88.71%	0.59	0.57
Adult	84.84%	84.6%	0.62	0.61

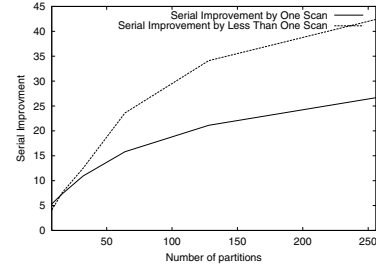
Table 3: Comparison of accuracy and amount of datascan using the batch (or all in memory) and sequential method for accuracy-based problems

training data can fit into the main memory of the machine. Any single classifier algorithm that reduces the number of data scan [Shafer *et al.*, 1996; Gehrke *et al.*, 1998; 1999; Hulten and Domingos, 2002] will not have training time less than this result. As shown in Figure 2, both one scan and less than one scan algorithm are significantly faster than the single classifier, and the less than one scan algorithm is faster than the one scan algorithm.

5 Related Work

To scale up decision tree learning, SPRINT [Shafer *et al.*, 1996] generates multiple sorted attribute files. Decision tree is constructed by scanning and splitting attribute lists, which eliminates the need for large main memory. Since each attribute list is sorted, for a data file with f attributes and N examples, the total cost to produce the sorted lists is $f \cdot O(N \cdot \log(N))$. External sort is used to avoid the need for main memory. Each attribute list has three columns—a unique record number, the attribute value and the class label; the total size of attribute lists is approximately three times the size of the original data set. When a split takes places at a node,

Figure 2: Serial improvement for credit card dataset using one scan and less than one scan



	Accuracy Based		Cost-sensitive	
	Accuracy	DataScan	Beneht	Datascan
C4.5	89.7%	0.95	\$794933	0.96
RIPPER	90%	0.93	\$769344	0.97
NB	86.9%	0.89	\$774854	0.93

(a). Performance for different classifier for biased distribution

DataScan	0.6	0.7	0.8	0.96
Total Benefits	\$561382	\$614575	\$728315	\$794933

(b). Performance of C4.5 with different amount of data scanned under the biased distribution

Table 4: Performance of less than one scan under biased distribution

SPRINT reads the attribute lists at this node and breaks them into r sets of attribute lists for r child nodes, for a total of f file read and $r \cdot f$ file writes. Later, RainForest [Gehrke *et al.*, 1998] improves the performance of SPRINT by projecting attribute list on each unique attribute value into an AVC-set. The complete AVC-sets of all attributes are called AVC-group. When the AVC-group can be held in main memory, the selection of predictor attribute can be done efficiently. To construct the initial AVC-group, it incurs the same cost as SPRINT to construct initial attribute lists plus one more scan over the sorted lists to project into AVC-sets. Whenever a split happens, RainForest has to access the data file again and reconstruct the AVC-group for child nodes. The exact number of read and write is based on variations of RainForest that is chosen. In the best scenario where the AVC-group of every node in the tree fit in memory, the RF-read version still has to scan the whole data once at each level of the tree and write it into r files. When this condition is not met, RainForest solves the problem by multiple read and write. More recently, BOAT [Gehrke *et al.*, 1999] constructs a “coarse”

tree from a data sample that can fit into main memory. The splitting criterion in each node of the tree is tested against multiple decision trees trained from bootstrap samples of the sampled data. It refines the tree later by scanning the complete data set, resulting in a total of two complete data read.

Meta-learning [Chan, 1996] builds a tree of classifiers and combine class label outputs from base classifiers. It is based on heuristics and the total number of datascan is two. The improvements by our methods are in many folds. We combine probabilities instead of class labels. The combining technique is straightforward and can estimate the final accuracy prior to full construction. The total amount of sequential scan is less than once. There is a strong statistical reason to support why the multiple model method works [Fan *et al.*, 2002b]. Besides meta-learning, both bagging [Breiman, 1996] and boosting [Freund and Schapire, 1997] have been shown to have higher accuracy than a single model. However, bagging and boosting scan the dataset many times, and are not scalable for large dataset. Previous research [Fan *et al.*, 2002b] has shown the our combining method is more accurate than bagging.

One earlier work to use Hoeffding's inequality is online aggregation [Hellerstein *et al.*, 1997] to estimate the accuracy of aggregate queries. When the confidence about the accuracy of the partial query is sufficient, the user can terminate the query to avoid a complete scan. One recent work using Hoeffding inequality is on building decision tree from streaming data [Hulten and Domingos, 2002]. They keep the number of examples that satisfy or fail the predicate of a node in the tree. When it is confident that the current node is not accurate for the new data, it will be reconstructed. One limitation of [Hulten and Domingos, 2002] is that it is only applicable to decision tree learning. Our ensemble method using Hoeffding's inequality is applicable to a wide range of inductive learners. Empirically, we have applied it to decision tree, rule learner as well as naive Bayes learner. Beside its generality, the proposed one scan and less than one scan algorithms have been shown to have potentially higher accuracy than the single model (existing approaches are single model method [Hulten and Domingos, 2002; Gehrke *et al.*, 1999; Shafer *et al.*, 1996; Gehrke *et al.*, 1998]). Another advantage is the asymptotic complexity using the less than one scan algorithm is approximately $k \cdot O(\frac{n}{K})$ while computing a single model is still $O(n)$. Our empirical studies have shown that both the one scan and less than one scan algorithms are significantly faster than learning a single model. In previous work [Fan *et al.*, 2002a], we have used Central limit theorem (CLT) to implement a concept of progressive modeling where we estimate the range of accuracy of the final model. The learning stops when the estimated accuracy of the final falls within a tolerable range with confidence p . CLT requires the data distribution to be uniform. Hoeffding inequality has a different implication, in which we estimate the probability that the partial and final models are exactly the same.

6 Conclusion

In this paper, we propose two scalable inductive learning algorithms. The strawman multiple model algorithm scans the data set exactly once. We then propose a less than one scan

extension based on Hoeffding's inequality. It returns a partial multiple model when its accuracy is the same as the complete multiple model with confidence $\geq p$. Since Hoeffding inequality makes no assumption about the data distribution, the advantage of this method is that the data items can be retrieved sequentially. We have also discussed how to sequentially read the validation set exactly once using minimal memory. We have evaluated these methods on several data sets as both traditional accuracy-based and cost-sensitive problems using decision tree, rule and naive Bayes learners. We have found that the accuracy of all our methods are the same or far higher than the single model. The amount of data scan by the less than one scan algorithms range from 0.45 to 0.7 for the original natural distribution of data. For a significantly biased dataset, the amount of datascan by the less than one scan algorithm is over 0.9. It needs extra data to resolve the bias in data distribution in order to compute an accurate model. In addition, our empirical studies have shown that both methods are significantly faster than computing a single model even when the training data can be held in main memory, and the less than one scan algorithm is faster than the one scan algorithm.

References

- [Breiman, 1996] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [Chan, 1996] P Chan. *An Extensible Meta-learning Approach for Scalable and Accurate Inductive Learning*. PhD thesis, Columbia University, Oct 1996.
- [Fan *et al.*, 2002a] Wei Fan, Haixun Wang, Philip S Yu, Shawhwa Lo, and Salvatore Stolfo. Progressive modeling. In *Proceedings of Second IEEE International Conference on Data Mining (ICDM2002)*, 2002.
- [Fan *et al.*, 2002b] Wei Fan, Haixun Wang, Philip S Yu, and Salvatore Stolfo. A framework for scalable cost-sensitive learning based on combining probabilities and benefits. In *Second SIAM International Conference on Data Mining (SDM2002)*, April 2002.
- [Freund and Schapire, 1997] Y Freund and R Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Computer and System Sciences*, 55(1):119–139, 1997.
- [Gehrke *et al.*, 1998] Johannes Gehrke, Venkatesh Ganti, and Raghu Ramakrishnan. RainForest: a framework for fast decision construction of large datasets. In *Proceeding of 24th International Conference on Very Large Databases (VLDB'1998)*, 1998.
- [Gehrke *et al.*, 1999] Johannes Gehrke, Venkatesh Ganti, Raghu Ramakrishnan, and Wei-Yin Loh. BOAT-optimistic decision tree construction. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD 1999)*, 1999.
- [Hellerstein *et al.*, 1997] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online aggregation. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'97)*, 1997.
- [Hulten and Domingos, 2002] Geolff Hulten and Pedro Domingos. Learning from infinite data in finite time. In *Advances in neural information processing systems*. MIT Press, 2002.
- [Shafer *et al.*, 1996] J Shafer, Ramesh Agrawl, and M Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proceedings of Twenty-second International Conference on Very Large Databases (VLDB-96)*, pages 544–555, San Francisco, California, 1996. Morgan Kaufmann.

Skewing: An Efficient Alternative to Lookahead for Decision Tree Induction

David Page^{*†}
page@biostat.wisc.edu

^{*}Department of Biostatistics & Medical Informatics
University of Wisconsin
Madison, Wisconsin 53706

Soumya Ray^{†*}
sray@cs.wisc.edu

[†]Department of Computer Sciences
University of Wisconsin
Madison, Wisconsin 53706

Abstract

This paper presents a novel, promising approach that allows greedy decision tree induction algorithms to handle problematic functions such as parity functions. *Lookahead* is the standard approach to addressing difficult functions for greedy decision tree learners. Nevertheless, this approach is limited to very small problematic functions or subfunctions (2 or 3 variables), because the time complexity grows more than exponentially with the depth of lookahead. In contrast, the approach presented in this paper carries only a constant run-time penalty. Experiments indicate that the approach is effective with only modest amounts of data for problematic functions or subfunctions of up to six or seven variables, where the examples themselves may contain numerous other (irrelevant) variables as well.

1 Introduction

Algorithms for the top-down induction of decision trees (TDIDT) are among the most widely used algorithms for machine learning, data mining and statistical classification. TDIDT implementations such as ID3 [Quinlan, 1983], C4.5 [Quinlan, 1997], C5.0 (www.rulequest.com) and CART [Breiman *et al.*, 1984] are easy to use and (often) produce human-comprehensible models. Nevertheless, TDIDT algorithms are well-known to be myopic because of their greedy strategy for choosing split variables, or internal node labels. This myopia is at its worst when the data are labeled according to parity functions such as exclusive-or (2-bit odd parity, denoted by $\oplus$). These and related other problematic functions naturally arise in real-world data. For example, in fruitfly experiments, flies that are *either* male and have an active *Sxl* gene *or* that are female and have an inactive *Sxl* gene survive, while other flies die; hence survival is an exclusive-or function of gender and *Sxl* activity (Table 1).

Of course, parity and parity-like functions are also problematic for other machine learning or statistical algorithms that employ (explicitly or implicitly) a linear assumption in order to gain efficiency. Such models include perceptrons, logistic regression, linear support vector machines, Fischer's linear discriminant and naive Bayes. They also include any of

Gender Female	Sxl active	Survival
0	0	0
0	1	1
1	0	1
1	1	0

Table 1: Truth table for *Drosophila* (fruitfly) survival based on gender and *Sxl* gene activity.

a variety of data analysis approaches that employ an information gain or Kullback-Leibler divergence filter to do variable selection or to control computation time, for example as in the sparse candidate algorithm for learning Bayesian networks [Friedman *et al.*, 1999]. The inability of such approaches to learn functions like $\oplus$ is frequently noted, for example, in the context of analyzing gene expression microarray data [Friedman *et al.*, 1999; Szallasi, 2001].

In TDIDT algorithms, the myopia of the search can be reduced at the cost of increased computation time. The standard approach is through depth- k lookahead [Norton, 1989], where the default for TDIDT algorithms is depth-1 lookahead. However, the time to perform a split grows exponentially with k , and problematic functions remain no matter how large a k is chosen.

The purpose of this paper is to introduce an alternative approach to problematic functions such as parity, which does not incur the high computational cost of lookahead. The approach relies on the observation that these functions are not actually problematic if the distribution over the data is significantly different from uniform. In such cases other functions may become problematic for TDIDT algorithms, but functions such as exclusive-or become relatively easy. Hence we first observe that TDIDT algorithm performance on a data set can be improved if the algorithm has access to a second, significantly different distribution over the data. Of course such a second distribution often is unavailable. Therefore, we next show how the second distribution often can be simulated by *skewing* the data from the first distribution. We note that the present paper focuses entirely on classification, ignoring regression trees, or trees that predict continuous outputs at their leaves. Furthermore, we limit our entire discussion to TDIDT algorithms that perform only binary splits, although the data may use variables that can take more than two values.

x_1	x_2	x_3	f
0	0	0	1
1	0	1	0
0	1	0	0
0	1	1	1

Table 2: For subfunctions at each of $x_2 = 0, x_2 = 1, x_3 = 0$ and $x_3 = 1$, the fraction of assignments that are positive is $\frac{1}{2}$, as for f itself. Hence x_2 and x_3 have zero gain, while x_1 has nonzero gain according to both entropy and Gini.

2 Review of Lookahead for Hard Functions

We assume the reader is familiar with the approach of TDIDT algorithms. The assumed familiarity includes a knowledge of the commonly-used functions to measure node purity, such as entropy [Quinlan, 1997] or Gini index [Breiman *et al.*, 1984]. It also includes familiarity with the greedy heuristic of choosing to split on the variable that maximizes the improvement, or *gain*, in the node purity score. We now review the notions of problematic functions and lookahead. For ease of discussion, this section limits itself to two-class problems, where the classes are *positive* and *negative*.

For the various node purity functions employed by different TDIDT algorithms, splitting on a variable x_i can yield a non-zero gain only if the class distribution changes for at least one of the values (or ranges) that x_i can take. If the distribution of classes is the same for every value of x_i (or range) then x_i will have zero gain according to any node purity measure in common usage. As an example, in Table 2 the variable x_1 has non-zero gain according to either Gini or entropy, whereas the variables x_2 and x_3 have zero gain.

Consider a data set drawn from a uniform distribution over binary-valued variables $x_1, x_2 \dots x_{100}$, labeled according to the target function $x_{99} \oplus x_{100}$. Even if we are fortunate enough to have a *complete data set*—one occurrence of each truth assignment over $x_1 \dots x_{100}$ —it is clear that for every variable x_i , the class distribution is exactly the same whether x_i is 0 or 1. So, regardless of how large a uniform sample we choose to draw, a variable will have non-zero gain only because of chance. Thus, the probability that one of the *correct* variables (x_{99} or x_{100}) will have a higher gain than every one of the *incorrect* variables is extremely low. Hence the learning task is virtually impossible for a TDIDT algorithm.

With depth-2 lookahead the preceding task becomes trivial. A depth-2 lookahead from a given node chooses not only the next split variable, but also the split variables at the next level. A TDIDT algorithm augmented in this way will consider among the possible depth-2 trees the two shown in Figure 1, each of which will have the maximum possible gain. For any reasonably large data set, with high probability all other depth-2 trees will have gain only marginally different from zero. Hence we see that with depth-2 lookahead, $\oplus$ becomes easy. Because depth-2 lookahead is repeated at every step in tree construction, many other functions that have 2-variable $\oplus$ as a subfunction become easy.

Of course, depth-2 lookahead comes with a price. Where n is the number of variables and m is the number of examples, the time to choose the split goes from $O(mn)$ to $O(mn^3)$, be-

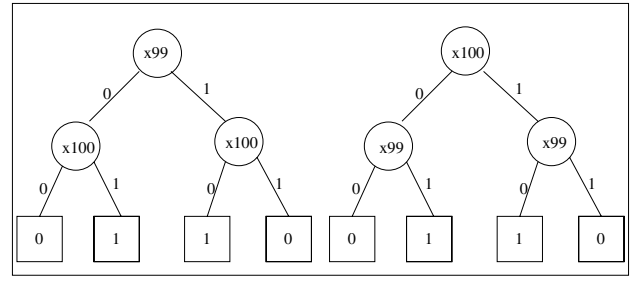


Figure 1: Two trees representing $x_{99} \oplus x_{100}$

x_1	x_2	x_3	f_1	f_2	f_3	f_4	f_5	f_6
0	0	0	0	0	0	0	1	0
0	0	1	1	1	1	0	0	0
0	1	0	1	0	1	1	0	0
0	1	1	0	0	0	0	0	1
1	0	0	0	0	1	0	0	1
1	0	1	1	0	0	1	0	0
1	1	0	1	1	0	0	0	0
1	1	1	0	0	1	0	1	0

Table 3: Six of the 12 functions over three variables that are problematic even using depth-2 lookahead. The other six problematic functions are the inverses of these.

cause labels have to be selected for three nodes from among n variables. Furthermore, there are many functions that require a higher lookahead. For example, suppose we have examples constructed from variables $x_1 \dots x_{100}$ and the target is one of the functions in Table 3 involving x_1, x_2 , and x_3 . Even with depth-2 lookahead, TDIDT is highly likely to choose incorrect variables. These problems can be solved with depth-3 lookahead, but the time to choose a split becomes $O(mn^7)$, and other problematic targets remain even then.¹

3 Motivation for Skewing

Consider the first target function discussed in the previous section, $x_{99} \oplus x_{100}$, but now suppose the data are distributed differently from uniform. For example, we might introduce dependencies not present in the uniform distribution: for every odd number i , $1 \leq i \leq 99$, if x_i is 0 then x_{i+1} has probability 0.99 of being 1. Or we might suppose all variables are independent as in the uniform distribution, but every variable has probability only $\frac{1}{4}$ of taking the value 0. In either case, with a large enough sample we expect that the class distribution among examples with $x_{99} = 0$ will differ significantly from the class distribution among examples with $x_{99} = 1$.

To examine the preceding claim more closely, consider the second distribution, where each variable has probability $\frac{1}{4}$ of being 0. If we draw a sample of 400 examples, we expect roughly 100 of these to have $x_{99} = 0$ and roughly 300 to

¹We can reduce this super-exponential growth of lookahead to “mere” exponential growth and still address parity-like functions if we require the tree to be “leveled”—all nodes at a level are labeled by the same variable. But even this non-standard lookahead procedure imposes a high computational burden.

have $x_{99} = 1$. Of the 100 with $x_{99} = 0$, we expect roughly $\frac{3}{4}$ of these, or 75, to have $x_{100} = 1$ and hence to belong to the positive class. Of the 300 with $x_{99} = 1$, we expect only $\frac{1}{4}$ of these, or 75 again, to belong to the positive class (to have $x_{100} = 0$). Hence the fraction of positive examples is quite different for the two values of x_{99} : $\frac{3}{4}$ of the examples with $x_{99} = 0$ are positive, while $\frac{1}{4}$ of the examples with $x_{99} = 1$ are positive. As a result x_{99} (and x_{100}) will have non-zero gain; for instance, information gain is roughly 0.19 (out of maximum 1.0 possible) and Gini gain is roughly 0.06 (out of maximum 0.25 possible). On the other hand, every variable other than x_{99} or x_{100} is likely to have nearly zero gain. Hence unless a highly unlikely sample is drawn, a TDIDT algorithm will choose to split on either x_{99} or x_{100} , at which point the remainder of the learning task is trivial.

Notice that in the preceding discussion, moving to our second distribution changed the marginal distribution for *every* variable, not just for those in the target. It would have revealed the correct variables if the target function had been $x_2 \oplus x_{69}$ or even one of the problematic functions of three variables in Table 3. Notice also that the important aspect of the second distribution was that it changed the frequency distributions for the variables; the specific change for any variable could have gone the other way—to probability $\frac{3}{4}$ of taking value 0—and the second distribution still would have given non-zero gain to exactly the variables in the target.

From the preceding discussion we conclude that if we have access to two distributions that are “different enough,” then choosing good variables to split on becomes relatively easy. However, in real-world problems we rarely have access to two different distributions over the data, or the ability to request data according to a second distribution that we choose. Instead, the next section discusses how in practice we can simulate a second distribution different from the first. We call this procedure *skewing*. The simulation approach tends to magnify idiosyncrasies in the data set, for example, introducing some dependencies that were not present in the original distribution. Nevertheless, our experiments indicate that, if the data set is large enough, the magnification of idiosyncrasies is not a major problem.

4 Skewing Algorithm

The desired effect of the skewing procedure is that the skewed data set should exhibit significantly different frequencies from the original data set. Because we cannot draw new examples, we change the frequency distributions for variables by attaching various weights to the existing examples. The procedure initializes the weight of every example to 1. We next present the details of the re-weighting procedure for binary-valued variables only. Nominal variables can be converted to binary variables. We discuss extensions to continuous variables in Section 7.

We may assume that every variable takes the value 0 in at least one example and takes the value 1 in at least one example—otherwise, the variable carries no information and can be removed. For each variable x_i , $1 \leq i \leq n$, we randomly, uniformly (independently for each variable) select a “favored setting” v_i of either 0 or 1. We then increase the

weight of each example in which x_i takes the value v_i , by multiplying the weight by a constant; for the sake of illustration, suppose we double the weight.

At the end of this process, each example has a weight between 1 and 2^n . It is likely that each variable has a significantly different weighted frequency distribution than previously, as desired. But this is not guaranteed. For example, suppose the original data set consists of 100 truth assignments over variables x_1 and x_2 . Suppose further that in half of these examples $x_1 = 0$ and $x_2 = 1$, and in the other half $x_1 = 1$ and $x_2 = 0$. If the favored setting for each variable happens to be 1, then all examples get assigned weight 2, so the new frequency distribution for each variable is the same as the original frequency distribution. In addition to this potential difficulty, a second difficulty is that this process can magnify idiosyncrasies in the original data. For instance, suppose we have a data set over $x_1, \dots, x_n$, and (for simplicity) the favored setting for each variable is 1. If we happen to have one example with many variables set to 1, it will get an inordinately high weight compared with other examples, potentially giving some insignificant variable a high gain. Can we mitigate these potential problems with the skewing procedure?

The difficulties in the preceding paragraph occur with some data sets combined with some choices of favored settings. Other selections of favored settings for the same data set may leave other variables’ frequencies unchanged, but it is relatively unlikely they will leave the same variables’ frequencies unchanged. Furthermore, while other selections of favored settings may magnify other idiosyncrasies in the data, it is unlikely they will magnify the same idiosyncrasies. Therefore, instead of using skewing to create only a second distribution, we use it to create some number T of additional distributions. The T different distributions come about from randomly (without replacement) selecting T different combinations of favored settings for the n variables according to a uniform distribution. To ensure that tree construction is not thrown off course by any single bad distribution (either original or skewed), the tree construction process is modified as described in the following paragraph.

Suppose we have $T + 1$ weightings of the data (the original data set plus T reweighted versions of this data set), and we are considering a split. We score each of the n variables against each of the $T + 1$ weightings of the data. A variable that is not part of the target function should have nearly zero gain on every weighting, although as already noted, it may occur that on some weightings some of these variables can achieve high gain. But only variables that appear in the target should have significantly non-zero gain on *most* of the weightings (though not necessarily on all). Therefore, we set a *gain threshold*, and the variable that exceeds the gain threshold for the greatest number of weightings is selected as the split variable. Our expectation is that the selected variable is highly likely to be *correct* in the sense that it actually is a part of the target function. Yet the time for choosing the split remains $O(mn)$, in contrast to lookahead. We have increased the run-time only by a small constant.

Pseudocode for the algorithm is shown in Algorithm 1. Rather than actually doubling or tripling weights, the algorithm takes a parameter $\frac{1}{2} < s < 1$. The weight of an ex-

Algorithm 1 Skewing Algorithm

Input: A matrix D of m data points over n boolean variables, gain fraction G , number of trials T , skew $\frac{1}{2} < s < 1$

Output: A variable x_i to split on, or -1 if no variable with sufficient gain could be found

```
1:  $N \leftarrow$  Entropy of class variable in  $D$ 
2:  $v \leftarrow$  Variable with max gain in  $D$ 
3:  $g \leftarrow$  Gain of  $v$  in  $D$ 
4: if  $g < G \times N$  then
5:    $v \leftarrow -1$ 
6: for  $i = 1$  to  $n$  do
7:    $F(i) \leftarrow 0$ 
   {begin skewing loop}
8:   for  $t = 1$  to  $T$  do
9:     for  $i = 1$  to  $n$  do
10:       $V(i) \leftarrow$  Randomly chosen favored value for  $x_i$ 
11:      for  $e = 1$  to  $m$  do
12:         $W(e) = 1$ 
13:        for  $i = 1$  to  $n$  do
14:          if  $t > 1$  then
15:            if  $D(e, i) = V(i)$  then
16:               $W(e) \leftarrow W(e) \times s$ 
17:            else
18:               $W(e) \leftarrow W(e) \times (1 - s)$ 
19:       $N \leftarrow$  Entropy of class variable in  $D$  under  $W$ 
20:      for  $i = 1$  to  $n$  do
21:         $E \leftarrow$  Gain of  $x_i$  under distribution  $W$ 
22:        if  $E \geq G \times N$  then
23:           $F(i) \leftarrow F(i) + 1$ 
   {end skewing loop}
24:  $j \leftarrow \arg \max F(i)$ 
25: if  $F(j) > 0$  then
26:   return  $x_j$ 
27: else
28:   return  $v$ 
```

ample is multiplied by s if x_i takes preferred value v_i in the example, and is multiplied by $1 - s$ otherwise. Hence for illustration, if s is $\frac{2}{3}$, the weight of every example in which x_i takes value v_i is effectively doubled relative to examples in which x_i does not take value v_i .

Our hypothesis is that in practical experiments the new algorithm will run somewhat slower than an ordinary TDIDT algorithm, only by a constant factor. It will rarely produce trees with lower accuracy than those of an ordinary TDIDT algorithm. It will often produce trees with slightly to moderately higher accuracy—when the target contains one or more problematic subfunctions. And it will sometimes produce trees with much higher accuracy—when the target is itself a problematic function. When the target is a problematic function over many variables, even after skewing the gain of any individual variable in the target is likely to be small. Therefore, we also hypothesize that unless the data set is large, the benefits of the skewing approach will not apply to problematic target functions of five or more variables. Note that while a large number of variables in the *target* reduces the potential gain, the number of variables in the *examples* does not. The

following section describes the experiments designed to test the preceding hypotheses.

5 Experiments

In this section, we discuss experiments with synthetic and real data, designed to test the hypotheses in the preceding paragraph. In addition, the question arises of whether problematic functions or subfunctions occur with high enough frequency to justify the additional work of skewing. The experiments in this section also address that question. We begin with a discussion of experiments using synthetic data, where target functions as well as examples are drawn randomly and uniformly, with replacement. In these experiments we compare simple ID3 against ID3 with skewing. We selected ID3 to eliminate issues to do with more sophisticated pruning. The parameters input to the skewing algorithm (algorithm 1) were $T = 30$, $s = \frac{3}{4}$ and $G = 0.05$. These parameters were chosen before the experiments and were held constant across all experiments. Improved results could perhaps be obtained by tuning s and G .

In the first set of experiments with synthetic data, examples are generated according to a uniform distribution over 30 binary variables. Target functions are drawn by randomly generating DNF formulae over subsets of 3 to 6 of the 30 variables. The number of terms in each target is drawn randomly, uniformly from between 1 and 25, and each term is drawn by choosing for each variable whether it will appear negated, unnegated, or not at all (all with equal probabilities). All targets are ensured to be satisfiable. Examples over the 30 variables that satisfy the target are labeled *positive*, and all other examples are labeled *negative*. Figures 2-5 show learning curves for different target sizes. Each point on each curve is the average over several runs, each with a different target and with a different sample of the specified sample size.

In general, these figures fit our expectations. Both algorithms perform well but skewing provides slightly yet consistently better results (we note that the differences are not statistically significant). Probably the difference is skewed ID3 is less likely than ordinary ID3 to include irrelevant variables, particularly when faced with problematic functions. One surprise is that the figures indicate that an ordinary TDIDT algorithm outperforms skewing on average when the sample size is small relative to target size. As sample size grows, a crossover point is reached after which skewing consistently outperforms the ordinary TDIDT algorithm. Furthermore, the sample size required for effective skewing grows with the number of variables in the target, although these results alone do not make clear the order of this growth. It may well be exponential, because target complexity can grow exponentially with the number of variables in the target. Further experiments are needed to explore the order of this growth. This observation implies a limitation of skewing—that skewing may be undesirable for learning tasks with small samples or target concepts that potentially employ many variables.

The next set of experiments focuses on the problematic functions alone. The methodology is the same as before, with the following exception. Targets are drawn randomly from functions that can be described entirely by variable co-

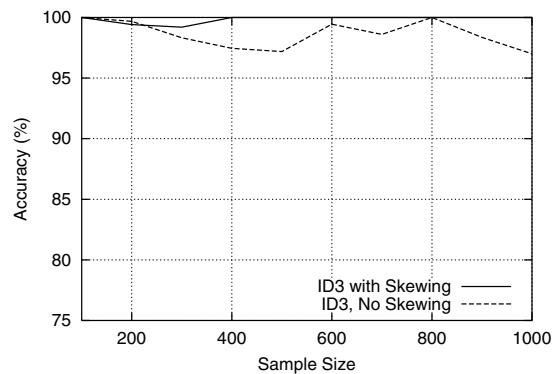


Figure 2: Three-Variable Targets

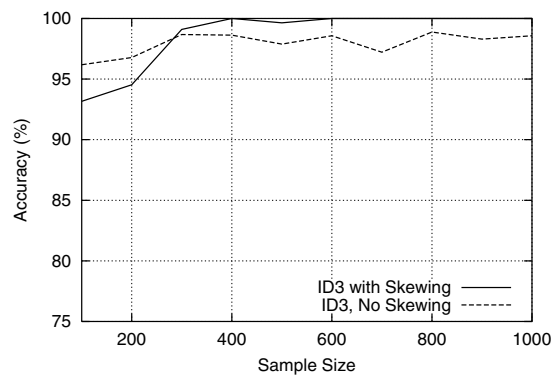


Figure 3: Four-Variable Targets

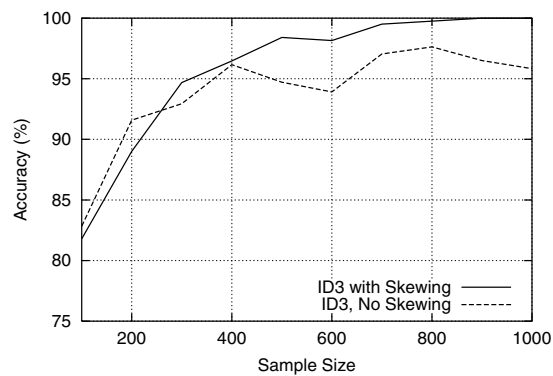


Figure 4: Five-Variable Targets

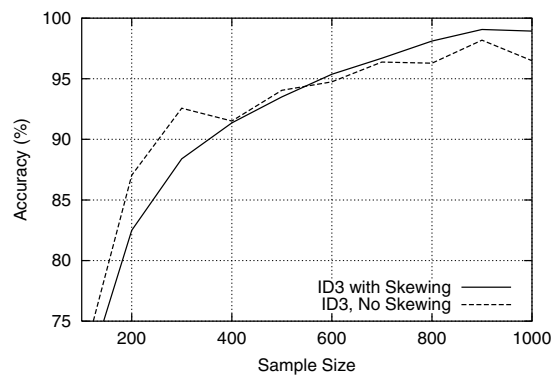


Figure 5: Six-Variable Targets

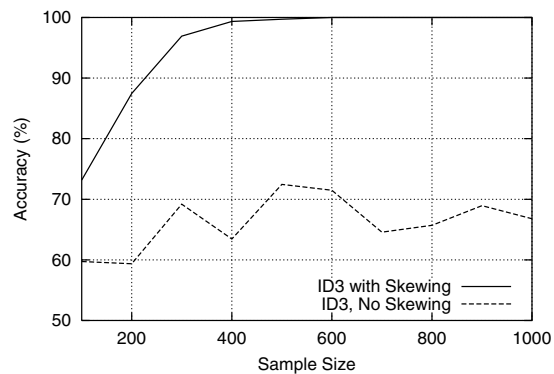


Figure 6: Three-Variable Hard Targets

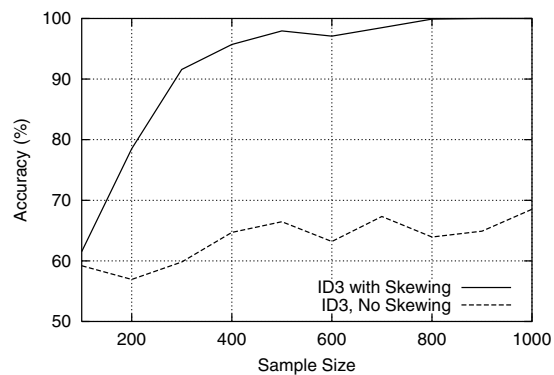


Figure 7: Four-Variable Hard Targets

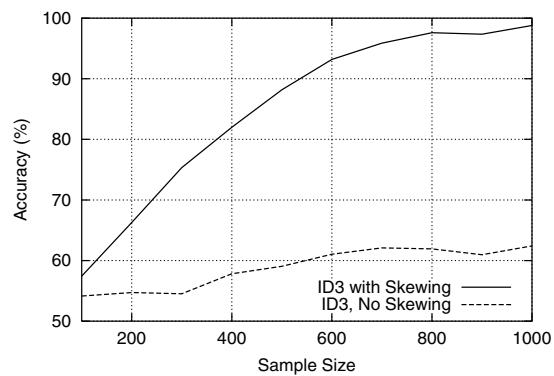


Figure 8: Five-Variable Hard Targets

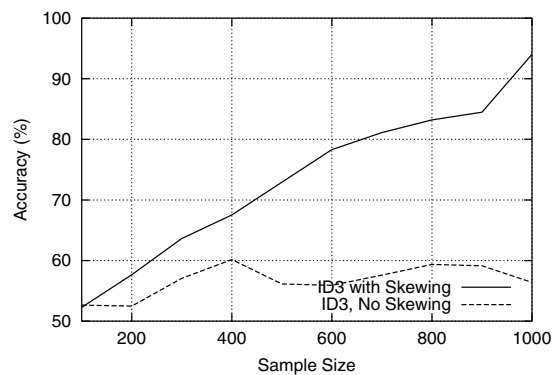


Figure 9: Six-Variable Hard Targets

Data Set	Standard ID3	ID3 with Skewing
Heart	71.9	74.5
Voting	94.0	94.2
Voting-2	87.4	88.6
Contra	60.4	61.5
Monks-1	92.6	100.0
Monks-2	86.5	89.3
Monks-3	89.8	91.7

Table 4: Accuracies of ID3 and ID3 with skewing on 4 UCI data sets. **Heart** is Cleveland Heart Disease, **Voting** is Congressional Voting, **Contra** is Contraceptive Method Choice, and **Monks-*x*** are the Monk’s problems. **Voting-2** is the same as **Voting**, with one feature (physician-fee-freeze) removed to make the problem more difficult.

references together with the standard logical connectives *and*, *or*, and *not*. Many such functions exist, and for all such functions, even given a complete data set, no variable has gain. Examples of such functions are exclusive-or and exclusive-nor, and all those in Table 3. Figures 6-9 show the results for these experiments.

We observe that if the target is a problematic function, skewing outperforms standard ID3 by a wide margin. The difference in accuracy was statistically significant — the 95% confidence intervals around each sample point in these graphs do not overlap once the sample sizes become moderately large. We repeated the experiment with 6-variable hard targets with 5000 examples to verify that skewing did indeed achieve 100% accuracy (in our noise-free setting) in this case. We also verified this behavior for 7 and 8 variable targets.

In addition to the preceding experiments, we also compared ID3 against ID3 with skewing on several data sets in the UCI machine learning repository [Blake and Merz, 1998] using 5-fold cross validation. For these data sets, we discretized continuous variables by binning. Further, nominal variables were binarized using the standard 1-of-*N* representation. The results of these experiments appear in Table 5. In this case, we do not know whether the target concepts involve problematic subfunctions except for the concepts in the Monk’s problems, where the first two involve problematic subfunctions. The only task for which there is a significant difference in accuracy is the first Monk’s problem, which has an exclusive-or subfunction. We believe the reason ID3 with skewing does not dramatically outperform ID3 on the second Monk’s problem is that the number of training instances is small (169) relative to the target function. We verified this by constructing a larger data set of 2000 examples using the concept that generated this data. In this case, skewing achieves an accuracy of 95%, while standard ID3 achieves 80%.

In the experiments reported in Figures 2-5, the run-time for ID3 with skewing was on average a constant times the run-time for ordinary ID3, regardless of sample size or target size. This constant was (roughly) equal to our value for *T* in Algorithm 1, 30. In the experiments involving hard targets, we observed that as sample size increased, ID3 with skewing became more efficient relative to ID3. This can be explained by the fact that though it takes more time to choose a split, ID3 with skewing chooses many fewer splits when the target

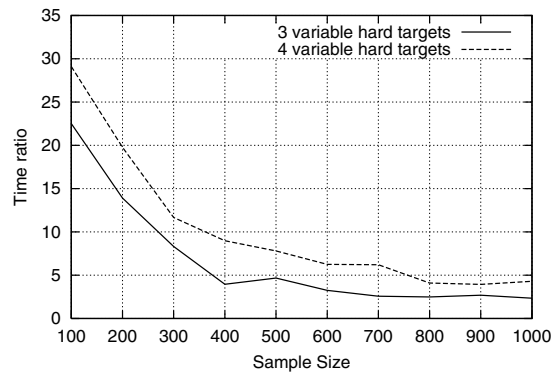


Figure 10: Time complexity of ID3 with skewing relative to standard ID3 for hard targets. The *y* axis represents the ratio of the average time taken by ID3 with skewing to induce a tree against the same quantity for standard ID3 for hard targets. Observe that, though the ratio is close to our value for *T* for small samples, it drops rapidly as the sample size increases, and the final value is much smaller.

is a hard function. In this case, the constant-factor overhead for skewing is significantly smaller than *T*. This behavior is shown in Figure 10. Thus in all cases, provided the sample is sufficiently large, skewing provides benefits similar to lookahead, but with only a constant increase in run-time. This is the primary result of the paper.

6 Related Work

A natural question to ask of the preceding results is whether they could as easily be obtained by other techniques that effectively re-weight the data. We know of two such related techniques: boosting [Freund and Schapire, 1997] and bagging [Breiman, 1996] (sampling with replacement may be thought of as providing a re-weighting of the data). While these techniques were not developed to enable tree induction algorithms to address problematic functions, it is possible that their re-weighting schemes might nevertheless be successful in this task. Therefore, we ran ID3 with each of these re-weighting techniques on hard targets, using the same methodology as in the preceding section. We also ran ID3 with a procedure that randomly reweighted the data as in the data perturbation approaches of Elidan et al[2002]. We observed that ID3 with each of the three re-weighting schemes performed on average no better than ordinary ID3. Hence we conclude that the benefit of skewing comes from the type of re-weighting being performed, not merely from the general notion of re-weighting.

7 Conclusions and Future Work

We have shown that the advantages of lookahead for decision trees can be obtained with only a constant increase in run-time, rather than a super-exponential increase, by the process of *skewing*. Nevertheless, the approach has limitations that need to be addressed in future work. It also has potential applications beyond decision trees, to be investigated in future work. The remainder of this section briefly outlines these directions.

Decision tree induction algorithms often are applied to data sets that involve some continuous variables in addition to nominal variables only. Therefore, one direction for further work is to extend the skewing algorithm to handle continuous variables. Here we briefly outline a natural such extension that we plan to test in future work. For continuous variables, the favored value is either *less* or *greater* than the split point, rather than 0 or 1. This makes skewing trickier because we do not know ahead of time what split point will be chosen. Hence for each continuous value v that a variable x takes in the data set, we compute the probability that v will be less than or greater than the split point. To do this computation, we might assume the split point is drawn from a uniform distribution over the values that x takes in the data set. We can then reweight an example with the expected weight over all possible split points. For example, suppose we have 100 examples, and the favored value for a variable x is *less*. An example that takes the tenth lowest value for x has probability 0.9 of being lower than the split point. Hence the weight of this example will get multiplied by $(0.9)(2) + (0.1)(1) = 1.9$.

Second, while we have demonstrated that skewing works for parity and other problematic functions of up to seven variables, for more variables it appears that large numbers of examples (at least several thousand) will be required. If the number of examples is too small relative to the size of the problematic portion of the target, then skewing can cause predictive accuracy to degrade somewhat, though the reason for this is not entirely clear. Hence if a data set is small it may be desirable to try both skewed and normal versions of the tree learner. In spite of this limitation, skewing makes it possible to gain the effect of five- or six- step lookahead where only 2- or perhaps 3-step lookahead was computationally feasible previously.

The third direction for future work is to address high-dimensional data sets. The skewing approach may have trouble with such data sets for the following reason. If each data point has thousands of features, then one data point is likely to get a much higher weight than all others (to have many more variables with the preferred values), merely by chance, leading to a model that overfits this data point. One solution may be to lessen the degree of skewing. This and other approaches should be tested on high-dimensional data sets.

We would also like to apply the skewing approach to other types of learning algorithms that have trouble with similar hard functions. To focus on a specific algorithm for illustration, the introduction noted that the sparse candidate algorithm for learning Bayesian networks [Friedman *et al.*, 1999] was susceptible to functions like exclusive-or. This susceptibility is because of its use of information gain or more generally Kullback-Leibler divergence to narrow the candidate parents for any given node. For example, suppose the variables x_1 and x_2 together are highly predictive of the value of x_3 and hence would be excellent parents of x_3 , as shown in the Bayesian network fragment in Figure 11. If x_1 and x_2 are independent of one another and take the value 0 roughly half of the time, then among tens or hundreds of other variables neither x_1 nor x_2 is likely to be considered as a candidate parent for x_3 . Nevertheless, if several skewed versions of the data are used to select candidate parents (a variable is a potential

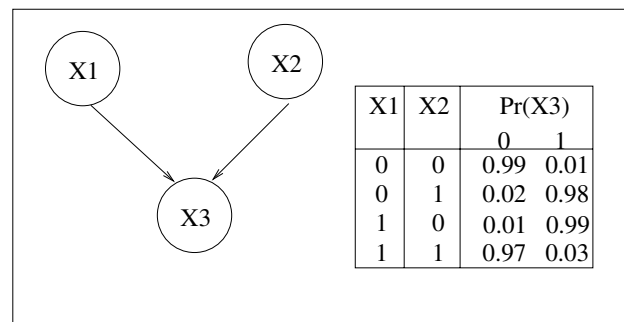


Figure 11: Variable x_3 is an approximation of $x_1 \oplus x_2$

parent if it scores well according to most of the skews), then x_1 and x_2 are likely to be selected. Modifying the sparse candidate algorithm is an intriguing direction for further work. More generally, we believe the *skewing* approach presented in this paper may be applicable to a wide variety of learning algorithms, in addition to decision trees.

Acknowledgements

This work was supported in part by NSF grant 9987841, NIH grant 1R01LM07050-01, and grants from the University of Wisconsin Graduate School and Medical School.

References

- [Blake and Merz, 1998] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
- [Breiman *et al.*, 1984] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks/Cole, Monterey, 1984.
- [Breiman, 1996] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [Elidan *et al.*, 2002] G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. Data perturbation for escaping local maxima in learning. In *AAAI-02/IAAI-02*, pages 132–139, 2002.
- [Freund and Schapire, 1997] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.
- [Friedman *et al.*, 1999] Nir Friedman, Iftach Nachman, and Dana Peér. Learning bayesian network structure from massive datasets: The “sparse candidate” algorithm. In *UAI-99*, pages 206–215, 1999.
- [Norton, 1989] S. Norton. Generating better decision trees. In *IJCAI-89*, pages 800–805, 1989.
- [Quinlan, 1983] J.R. Quinlan. Learning efficient classification procedures and their application to chess end games. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning I*, chapter 15, pages 463–482. Morgan Kaufmann Publishers, 1983.
- [Quinlan, 1997] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Kaufmann, 1997.
- [Szallasi, 2001] Z. Szallasi. Genetic network analysis: From the bench to computers and back. In *Tutorial Notes, Second International Conference on Systems Biology*, 2001. www.icsb2001.org/SzallasiTutorial.pdf.

