

Thank You for Purchasing This Ebook!

If you want to share a paper from this ebook with a friend or colleague, or would like to extract single papers, why not go to AAAI's Digital Library instead? Individual papers can be downloaded from www.aaai.org/Library with ease. (Click the title of this book in the title page and you will go directly to this book's digital library!) If you do share this ebook, kindly treat it like a physical book — lend for a quick look, but ask your friend or colleague to buy a copy for careful reading or reference.

We hope you find this volume useful. It was brought to you by the members of the Association for the Advancement of Artificial Intelligence.

To learn more about AAAI, please visit www.aaai.org. We invite you to join us! (<https://aaai.memberclicks.net>)

Terms of Use

This Agreement sets forth the terms and conditions for licensing of this ebook to you (“you” or “Licensee”), and use of this ebook indicates that you have read and understand this Agreement and accept its terms and conditions. If you do not agree with this Agreement, do not use this ebook. This ebook is for your personal, noncommercial use. Content copied or printed must retain all copyright, trademark, and other proprietary notices. Except as expressly set forth in this Agreement, you may not modify, copy, distribute, disclose, retransmit, sell, publish, broadcast, or circulate this ebook in whole or in part without AAAI’s prior written consent. For commercial use of any portion of the contents you must secure written consent from AAAI.

License and Restrictions

Subject to the terms and conditions of this Agreement, you are granted a limited non-exclusive license to use a copy of this publication on one (1) computer used by a single household. You may make one (1) backup copy of the ebook for your own personal use. You may print out one copy of this ebook in hard copy for your own personal use. You agree not to: (1) make additional copies of this ebook except as provided above; (2) print out copies of the papers on this ebook except as provided above; (3) give copies to another person who has not purchased a license for this ebook from the Association for the Advancement of Artificial Intelligence (AAAI); (4) install the publication on computers used by individuals who have not purchased the appropriate licenses for this ebook from AAAI; (5) duplicate the ebook by any other means including electronic transmission. This ebook in its entirety is protected by copyright laws.

Termination

This Agreement may be terminated by AAAI immediately and without notice if you fail to comply with any term or condition of this Agreement. Upon such termination, you must immediately destroy all complete and partial copies of this ebook, including all backup copies. AAAI shall have the right to change or add to the terms of this Agreement at any time, and to change, discontinue or impose conditions on any aspect of this ebook, and your continued use of this ebook will indicate your agreement to any such change.

Disclaimer of Warranties

1. This ebook is provided “as is.” AAAI and its affiliates, licensors, third-party content or service providers, distributors, dealers, or suppliers makes no express or implied warranties with respect to matters such as accuracy, completeness, timeliness, fitness for a particular purpose, non-infringement, and merchantability in regard to this ebook. 2. AAAI assumes no contractual liability for trademark or other infringements on intellectual property rights with respect to infringing material provided by third parties to AAAI. 3. AAAI assumes no liability for errors or omissions of material that appears on this ebook. 4. AAAI shall have no liability for damages arising from any reliance on data, information, or other content published on this ebook. 5. Neither the mention of, citation to, or other reference to any individual, group, organization,

product, viewpoint, opinion, theory or any and all such entities and matters shall constitute an endorsement thereof or infer any association therewith in regard to AAAI. Conversely, the failure to so mention, cite, or otherwise reference shall not be construed, deemed or otherwise understood as implying or inferring anything whatsoever.

Limitation of Liability and Damages

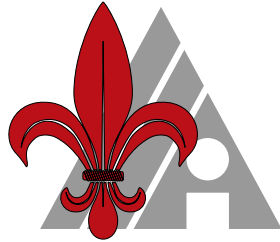
THE ENTIRE LIABILITY OF AAAI FOR ANY REASON SHALL BE LIMITED TO THE AMOUNT PAID BY THE CUSTOMER FOR THE EBOOK LICENSED FROM AAAI OR ITS AUTHORIZED RESELLER. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, AAAI AND ITS REPRESENTATIVES ARE NOT LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON BREACH OF CONTRACT, BREACH OF WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, EVEN IF AAAI OR ITS REPRESENTATIVES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF A REMEDY SET FORTH HEREIN IS FOUND TO HAVE FAILED OF ITS ESSENTIAL PURPOSE. AAAI AND ITS REPRESENTATIVES' TOTAL LIABILITY TO YOU FOR ACTUAL DAMAGES FOR ANY CAUSE WHATSOEVER WILL BE LIMITED TO THE AMOUNT PAID BY YOU FOR THIS EBOOK

General Provisions

This Agreement sets forth AAAI's and its Representatives' entire liability and your exclusive remedy with respect to this ebook. You acknowledge that this Agreement is a complete statement of the agreement between you and AAAI. This Agreement does not limit any rights that AAAI may have under copyright, patent, trademark or other laws. The Representatives of AAAI are not authorized to make modifications to this Agreement other than in writing signed by an officer of AAAI. If any provision of this Agreement is invalid or unenforceable under applicable law, then it is, to that extent, deemed omitted and the remaining provisions will continue in full force and effect. No delay or failure on the part of AAAI in the exercise of any right granted under this Agreement, or available at law or in equity, shall be construed as a waiver of such right. This Agreement shall be governed by California law, except as to copyright and trademark matters, which are covered by federal laws. You agree to submit to the jurisdiction of the federal and state courts of Santa Clara County, California and waive any objection to venue in such courts. This Agreement is deemed entered into at Palo Alto, California, USA.

Copyright

Copyright © 2014 Association for the Advancement of Artificial Intelligence. All rights reserved. This publication, or parts thereof, may not be reproduced in any form without permission.



Statistical Relational AI

Papers Presented at the
Twenty-Eighth AAAI Conference
on Artificial Intelligence

Technical Report WS-14-13



AAAI Press

Association for the Advancement of Artificial Intelligence

Statistical Relational AI

Papers Presented at the
Twenty-Eighth AAAI Conference
on Artificial Intelligence

Technical Report WS-14-13

AAAI Press

Palo Alto, California

Copyright © 2014, AAAI Press

The Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303 USA

AAAI maintains compilation copyright for this technical report and retains the right of first refusal to any publication (including electronic distribution) arising from this AAAI event. Please do not make any inquiries or arrangements for hardcopy or electronic publication of all or part of the papers contained in these working notes without first exploring the options available through AAAI Press and AI Magazine (concurrent submission to AAAI and an another publisher is not acceptable). A signed release of this right by AAAI is required before publication by a third party.

Distribution of this technical report by any means including electronic (including, but not limited to the posting of the papers on any website) without permission is prohibited.

ISBN 978-1-57735-674-5

WS-14-13

Manufactured in the United States of America

Organizers

Organizing Committee

Guy Van den Broeck (UCLA, KU Leuven)
Kristian Kersting (TU Dortmund, Fraunhofer IAIS)
Sriram Natarajan (Indiana University)
David Poole (University of British Columbia)

Program Committee

Hendrik Blockeel (KU Leuven)
Rodrigo Braz (SRI International)
Arthur Choi (UCLA)
Jesse Davis (KU Leuven)
Luc De Raedt (KU Leuven)
Pedro Domingos (University of Washington)
Paolo Frasconi (Università degli Studi di Firenze)
Vibhav Gogate (UT Dallas)
Tuyen Huynh (SRI International)
Manfred Jaeger (Aalborg University)
Roni Khardon (Tufts University)
Tushar Khot (University of Wisconsin-Madison)
Angelika Kimmig (KU Leuven)
Niels Landwehr (University of Potsdam)
Daniel Lowd (University of Oregon)
Brian Milch (Google)
Kee Siong Ng (EMC Greenplum)
Mathias Niepert (University of Washington)
Scott Sanner (NICTA)
Vitor Santos Costa (Universidade do Porto)
Taisuke Sato (Tokyo Institute of Technology)
Jude Shavlik (University of Wisconsin-Madison)
Sameer Singh (University of Massachusetts, Amherst)

*This AAAI workshop was held on
Sunday, July 27, 2014
in conjunction with the Twenty-Eighth
AAAI Conference on Artificial Intelligence.*

Contents

Organizers / iii

Preface / vii

Guy Van den Broeck, Kristian Kersting, Sriraam Natarajan, David Poole

Papers

Lifting Relational MAP-LPs Using Cluster Signatures / 2

Udi Apsel, Kristian Kersting, Martin Mladenov

Efficient Markov Logic Inference for Natural Language Semantics / 9

Islam Beltagy, Raymond J. Mooney

Reasoning in the Description Logic BEL Using Bayesian Networks / 15

Ismail Ilkan Ceylan, Rafael Peñaloza

Parameter Estimation for Relational Kalman Filtering / 22

Jaesik Choi, Eyal Amir, Tianfang Xu, Albert J. Valocchi

Representation, Reasoning, and Learning for a Relational Influence

Diagram Applied to a Real-Time Geological Domain / 29

Matthew C. Dirks, Andrew Csinger, Andrew Bamber, David Poole

Extending PSL with Fuzzy Quantifiers / 35

Golnoosh Farnadi, Stephen H. Bach, Marie-Francine Moens, Lise Getoor, Martine De Cock

Understanding the Complexity of Lifted Inference and Asymmetric Weighted Model Counting / 38

Eric Gribkoff, Guy Van den Broeck, Dan Suciu

Relational Logistic Regression: The Directed Analog of Markov Logic Networks / 41

Seyed Mehran Kazemi, David Buchman, Kristian Kersting, Sriraam Natarajan, David Poole

Classification from One Class of Examples for Relational Domains / 44

Tushar Khot, Sriraam Natarajan, Jude Shavlik

Applying Marginal MAP Search to Probabilistic Conformant Planning: Initial Results / 51

Junkyu Lee, Rina Dechter, Radu Marinescu

Towards Adversarial Reasoning in Statistical Relational Domains / 57

Daniel Lowd, Brenton Lessley, Mino De Raj

Automated Debugging with Tractable Probabilistic Programming / 60

Aniruddh Nath, Pedro Domingos

Learning Tractable Statistical Relational Models / 62

Aniruddh Nath, Pedro Domingos

Tractable Probabilistic Knowledge Bases: Wikipedia and Beyond / 69	<i>Mathias Niepert, Pedro Domingos</i>
A Sparse Parameter Learning Method for Probabilistic Logic Programs / 76	<i>Masaaki Nishino, Akihiro Yamamoto, Masaaki Nagata</i>
A Deeper Empirical Analysis of CBP Algorithm: Grounding Is the Bottleneck / 83	<i>Shrutika Poyrekar, Sriraam Natarajan, Kristian Kersting</i>
Explanation-Based Approximate Weighted Model Counting for Probabilistic Logics / 86	<i>Joris Renkens, Angelika Kimmig, Guy Van den Broeck, Luc De Raedt</i>
A Proposal for Statistical Outlier Detection in Relational Structures / 93	<i>Fatemeh Riahi, Oliver Schulte, Qing Li</i>
WOLFE: Strength Reduction and Approximate Programming for Probabilistic Programming / 100	<i>Sebastian Riedel, Sameer Singh, Vivek Srikumar, Tim Rocktäschel, Larysa Visengeriyeva, Jan Noessner</i>
Hierarchical Reasoning with Probabilistic Programming / 104	<i>Brian E. Ruttenberg, Matthew P. Wilkins, Avi Pfeffer</i>
Scalable Learning for Structure in Markov Logic Networks / 111	<i>Zhengya Sun, Zhuoyu Wei, Jue Wang, Hongwei Hao</i>
Evidence-Based Clustering for Scalable Inference in Markov Logic / 118	<i>Deepak Venugopal, Vibhav Gogate</i>
Solving Distributed Constraint Optimization Problems Using Ranks / 125	<i>Mihaela Verman, Philip Stutz, Abraham Bernstein</i>
Efficient Probabilistic Inference for Dynamic Relational Models / 131	<i>Jonas Vlasselaer, Wannes Meert, Guy Van den Broeck, Luc De Raedt</i>
ProPPR: Efficient First-Order Probabilistic Logic Programming for Structure Discovery, Parameter Learning, and Scalable Inference / 133	<i>William Yang Wang, Kathryn Mazaitis, William W. Cohen</i>

Preface

Much has been achieved in the field of AI, yet much remains to be done if we are to reach the goals we all imagine. One of the key challenges with moving ahead is closing the gap between logical and probabilistic AI. Logical AI has mainly focused on complex representations, and probabilistic AI on uncertainty. Intelligent agents, however, must be able to handle both the complexity and the uncertainty of the real world.

Recent years have seen an explosion of successes in combining probability and (subsets of) first-order logic, programming languages and relational databases in several subfields of AI such as reasoning, learning, knowledge representation, planning, databases, NLP, robotics, vision, and others. Nowadays, we can learn statistical relational models automatically from millions of inter-related objects. We can generate optimal plans and learn to act optimally in uncertain environments involving millions of objects and relations among them. We can even perform lifted probabilistic inference, avoiding explicit state enumeration by manipulating first-order representations directly. So far, however, the researchers combining logic and probability in each of these subfields have been working mostly independently.

This workshop was designed for attempts at synthesis, forming a common core of problems and ideas, cross-pollinating across subareas, and ultimately starting to explore what might be called Statistical Relational AI: the study and design of intelligent agents that act in noisy worlds composed of objects and relations among the objects.

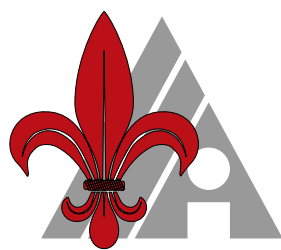
In total, the workshop received 30 submissions. Each submission was reviewed by at least two program committee members. The committee decided to accept 14 full technical papers and 13 short position papers and abstracts. Additionally, we had three invited speakers, Henry Kautz (University of Rochester, USA), Vibhav Gogate (University of Texas, Dallas, USA), and Luc De Raedt (KU Leuven, Belgium). The selected papers cover a wide range of topics in Statistical Relational AI such as parameter and structure learning, lifted inference, probabilistic programming, planning, information extraction, constraint optimization, NLP, among others, and clearly show the promise of statistical relational techniques for AI.

More precisely, Apse et al. present a framework for lifted MAP-LP inference with a new symmetry detection technique. Beltagy and Mooney develop a new inference algorithm for Markov logic that computes probabilities of complete formulae rather than ground atoms. Ceylan and Penaloza study the problem of reasoning in the probabilistic description logic BEL by reduction to Bayesian network inference. The paper by Choi et al. introduces a parameter learning algorithm for relational Kalman filters. Dirks et al. apply relational influence diagrams to a challenging real-world geological planning problem. Farnadi et al. propose to extend probabilistic soft logic with fuzzy quantifiers. Gribkoff et al. analyze the complexity of lifted inference in probabilistic databases and statistical relational learning. Kazemi et al. introduce relational logistic regression, a directed analogue of Markov logic networks. The paper by Khot et al. presents a one-class classification algorithm for relational domains. Lee et al. use marginal MAP inference algorithms for solving conformant planning problems. Lowd et al. argue that existing statistical relational techniques can be readily adapted to do adversarial reasoning. Nath and Domingos investigate tractable learning of relational sum-product networks, as well as automated debugging techniques based on probabilistic programs. Niepert and Domingos present a framework for tractable probabilistic knowledge bases that feature a rich query language. Niepert and Van den Broeck show that exchangeability is a statistical property that renders probabilistic inference tractable. Nishino et al. propose a new parameter learning algorithm for ProbLog that reduces the number of para-

meters. Poyrekar et al. perform an empirical analysis of counting belief propagation, identifying grounding as the bottleneck. Renkens et al. contribute a new bounded approximation method for weighted model counting based on probabilistic logic programming principles. The paper by Riedel et al. introduces a probabilistic programming language for declarative machine learning models. Rocktaschel et al. propose to learn low-dimensional embeddings that simulate the behaviour of first-order logic. Ruttenberg et al. introduce an approach to representing and reasoning about hierarchies in probabilistic programming. The paper by Schulte and Riahi applies Bayesian networks to relational outlier detection. Venugopal and Gogate propose a general method for addressing the evidence problem in lifted inference by reducing the domain size. Verman et al. present an algorithm that models distributed constraint optimization as a graph processing problem. Vlasselaer et al. study efficient inference algorithms for dynamic relational models with local structure. Wang et al. introduce a scalable probabilistic logic that enables efficient learning and inference on graphs. Finally, Zhengya et al. develop a scalable MLN structure learner that combines random walks with subgraph pattern mining.

We believe that this compilation of papers provides useful perspectives on statistical relational AI and hope it may inspire the reader to contribute to this challenging and exciting new research area.

– *Guy Van den Broeck, Kristian Kersting, Sriraam Natarajan, David Poole*



Papers

Lifting Relational MAP-LPs Using Cluster Signatures

Udi Apsel

Computer Science Department
Ben Gurion University of The Negev, Israel
apsel@cs.bgu.ac.il

Kristian Kersting and Martin Mladenov

Computer Science Department
TU Dortmund University, Germany
{kristian.kersting, martin.mladenov}@cs.tu-dortmund.de

Abstract

Inference in large scale graphical models is an important task in many domains, and in particular probabilistic relational models (e.g. Markov logic networks). Such models often exhibit considerable symmetry, and it is a challenge to devise algorithms that exploit this symmetry to speed up inference. Recently, the automorphism group has been proposed to formalize mathematically what “exploiting symmetry” means. However, obtaining symmetry derived from automorphism is GI-hard, and consequently only a small fraction of the symmetry is easily available for effective employment. In this paper, we improve upon efficiency in two ways. First, we introduce the Cluster Signature Graph (CSG), a platform on which greater portions of the symmetries can be revealed and exploited. CSGs classify clusters of variables by projecting relations between cluster members onto a graph, allowing for the efficient pruning of symmetrical clusters even before their generation. Second, we introduce a novel framework based on CSGs for the Sherali-Adams hierarchy of linear program (LP) relaxations, dedicated to exploiting this symmetry for the benefit of tight Maximum A Posteriori (MAP) approximations. Combined with the pruning power of CSG, the framework quickly generates compact formulations for otherwise intractable LPs, as demonstrated by several empirical results.

Introduction

Inference in large scale graphical models is an important task in many domains, and in particular probabilistic relational models (e.g. Markov logic networks). Such models often exhibit considerable symmetry, and it is a challenge to devise algorithms that exploit this symmetry in order to speed up inference, see e.g. (Poole 2003; Singla and Domingos 2008; Kersting, Ahmadi, and Natarajan 2009; Jha et al. 2010; Apsel and Brafman 2011; Van den Broeck et al. 2011; Noessner, Niepert, and Stuckenschmidt 2013; Ahmadi et al. 2013) and (Kersting 2012) for an overview. Recently, (Niepert 2012; Bui, Huynh, and Riedel 2013) have established connections between the automorphism group of graphical models and lifted inference. Specifically, Bui *et al.* have demonstrated how to deduce some of the symmetry available directly from the relational structure, and formulated a compact linear program (LP) for the basic relaxation

of the Maximum A Posteriori (MAP) query. Their framework, however, does not provide one with efficient tools to formulate tighter MAP-LP relaxations, such as those obtained via the Sherali-Adams (SA) hierarchy. In this paper, we make two contributions that jointly allow the computation of much tighter relaxations in relational models.

The first contribution, called Cluster Signature Graph (CSG), is a graphical platform that allows an efficient detection of symmetry between clusters of variables, by projecting the relational structure of each cluster onto a graph. Consequently, isomorphic CSGs denote exchangeable clusters, that is – clusters that are mapped one onto the other in a permutation belonging to the automorphism group.

The second contribution, inspired by the work of (Mladenov, Globerson, and Kersting 2014), is a framework which formulates compact MAP-LPs of the SA hierarchy. At its core, the framework relies on an oracle to provide all non-exchangeable clusters within a given size range. The above requirement is efficiently met by generating non-isomorphic CSGs using automated graph generation tools. Since the generated graphs are small, and the procedure for producing such graphs involves no analysis of the ground graphical model, the combination of our two contributions produces compact MAP-LPs of the SA hierarchy very quickly.

Further, we discuss the integration of evidence on unary and binary atoms into the framework, and finally, validate our new approach with several empirical results, putting special emphasis on the challenging class of transitive relational models. From the empirical results, we highlight several liftings of the SA hierarchy up to level 6, in non-trivial models.

We begin with an elaborated background section, followed by the presentation of our MAP approximation framework. Only then do we proceed to the CSG introduction.

Background

MRF and MAP Estimation

A Markov Random Field (MRF) $\mathcal{M}$ is a probabilistic graphical model, consisting of a set of random variables $x = \{x_1, \dots, x_n\}$, and a set of *factors*. A factor is a pair (θ_f, x_f) , which represents a function $\theta_f : \text{range}(x_f) \rightarrow \mathbb{R}$, mapping from the joint assignment range of variables $x_f \subseteq x$, to the reals. The joint distribution function of MRFs is given by

$$Pr(x = \hat{x}) = \frac{1}{Z} \prod_f \exp(\theta_f(\hat{x}_f)) \quad (1)$$

where $\hat{x}$ is a joint assignment to all x variables, and $\hat{x}_f$ is the respective joint assignment to all x_f variables under $\hat{x}$. Z here denotes a normalization constant called the *partition function*. One common query in probabilistic models is the Maximum A Posteriori (MAP) estimation query, which seeks to obtain a joint assignment that maximizes the joint distribution. The unnormalized maximization task can be expressed in the form of sum of logarithms, as follows.

$$\text{MAP}(\mathcal{M}) = \max_x \sum_f \theta_f(x_f) \quad (2)$$

In this paper, we will be focusing on obtaining an upper bound on the maximal unnormalized log distribution in relational probabilistic models. MAP assignments corresponding to this task may be obtained via rounding schemes, e.g. (Ravikumar, Agarwal, and Wainwright 2010). Optimizing the latter for the benefit of relational environments is an interesting challenge, which we leave to further study.

MAP Relaxation via Linear Programming

Methods for approximating the NP-hard MAP query (Shimony 1994) have become a subject for extensive study in various fields of computer science. Of the many approaches (Yedidia, Freeman, and Weiss 2003; Sontag, Globerson, and Jaakkola 2011), we build our work on the framework of Linear Programming (LP), where the MAP computation problem is translated into an Integer Linear Program (ILP) (Schrijver 1998), and relaxed to an LP by lifting the integrality constraints imposed on the variables. Formally, an LP is a maximization problem over a set of real-valued variables $\mu = \{\mu_i\}_i$, as follows.

$$\max_{\mu} \sum_i c_i \mu_i \quad \text{s.t.} \quad \sum_i a_{ij} \mu_i \leq b_j \quad \forall j \quad (3)$$

where c_i , b_j and a_{ij} denote real-valued coefficients. The polynomial-time complexity of LPs (Khachiyan 1979; Karmarkar 1984) lends itself to fast approximation methods, and can be found suitable for large domains, as in our setting.

For ease of presentation, our formulation, from this point on, will be adapted to triplewise MRFs and Boolean variables. This chosen formulation should suit our running examples throughout the paper. Nevertheless, the method we introduce can be applied to any MRF, of any variable assignment range. Given a triplewise MRF $\mathcal{M}$ over the set of random variables $x = \{x_i, \dots, x_n\}$ and factors $F = \{(\theta_f, x_f)\}_f$, we define a MAP linear program as follows.

For each subset of indices $\mathcal{I}$ taken from $\{1, \dots, n\}$ of size $1 \leq |\mathcal{I}| \leq 3$, let $\mu_{\mathcal{I}}$ denote a vector of variables of size $2^{|\mathcal{I}|}$. A notation $\mu_{ijk}(x_i, x_j, x_k)$ will be used to describe a specific variable in vector $\mu_{\mathcal{I}}$ corresponding to the subset $\mathcal{I} = (i, j, k)$ and entry $(x_i, x_j, x_k) \in \{0, 1\}^3$. Additionally, let $\mathcal{I}_F$ denote the set of all ordered indices for which there exists a factor f with a matching variables scope x_f , and let θ_{ijk} denote the log probability table of a factor whose variables scope is (x_i, x_j, x_k) . The MAP linear program (MAP-LP) is defined as the optimization problem in Figure 1. Note that permutations of $\mathcal{I}$ induce μ vectors that represent the same LP variables. For instance, the LP variable $\mu_{ij}(x_i, x_j)$ is equal to $\mu_{ji}(x_j, x_i)$, for any i, j and assignments to x_i, x_j .

$$\begin{aligned} \max_{\mu} \quad & \sum_{(i) \in \mathcal{I}_F} \langle \theta_i, \mu_i \rangle + \sum_{(i,j) \in \mathcal{I}_F} \langle \theta_{ij}, \mu_{ij} \rangle + \sum_{(i,j,k) \in \mathcal{I}_F} \langle \theta_{ijk}, \mu_{ijk} \rangle \\ \text{s.t.} \quad & \sum_{x_i} \mu_i(x_i) = 1 \quad \forall (i) \in \mathcal{I}_F, \\ & \sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i) \quad \forall (i, j) \in \mathcal{I}_F, \\ & \sum_{x_i} \mu_{ij}(x_i, x_j) = \mu_j(x_j) \\ & \mu_{ijk}(x_i, x_j, x_k) \geq 0 \quad \forall (i, j, k) \in \mathcal{I}_F, \\ & \sum_{x_k} \mu_{ijk}(x_i, x_j, x_k) = \mu_{ij}(x_i, x_j) \\ & \sum_{x_j} \mu_{ijk}(x_i, x_j, x_k) = \mu_{ik}(x_i, x_k) \\ & \sum_{x_i} \mu_{ijk}(x_i, x_j, x_k) = \mu_{jk}(x_j, x_k) \end{aligned}$$

where ,

$$\langle \theta_{ijk}, \mu_{ijk} \rangle = \sum_{(x_i, x_j, x_k)} \theta_{ijk}(x_i, x_j, x_k) \cdot \mu_{ijk}(x_i, x_j, x_k)$$

Figure 1: MAP-LP for triplewise MRFs

For this reason, we sometimes ignore the ordered nature of $\mathcal{I}$ and refer to its respective set of $x_{\mathcal{I}}$ variables as a cluster.

Modern LP solvers are capable of solving MAP-LP instances for some very large models. However, depending on a given instance of a problem, the quality of approximation may prove to be insufficient. Thus, it is common to employ strategies that produce tighter results. One often used strategy is to identify violations of cycle inequalities (Sontag and Jaakkola 2007), and add constraints that remove these solutions from the LP's feasible region. An adaptation of this method to relational probabilistic models was recently introduced in (Bui, Huynh, and Riedel 2013). A different strategy, albeit considered intractable in large or even moderately sized models, is elevating the MAP approximation problem onto high levels of *lift and project hierarchies*. As shown in (Sontag 2010), such approach, if tractable, produces tighter bounds compared with the aforementioned strategy.

The Sherali-Adams Hierarchy of MAP Relaxation

Lift and project hierarchies are frameworks for incrementally tightening the LP's feasible region of a relaxed ILP, starting from loose approximations and ending in exact solutions. At each level of the hierarchy, new variables and constraints are introduced, thereby lifting the dimension of the computational problem. A solution is obtained by projecting the result back onto the smaller dimension of the original problem. Typically, lift and project methods (Sherali and Adams 1990; Lovász and Schrijver 1991; Lasserre 2002) add variables which correspond to clusters of variables taken from the original problem. In this work, we focus on the the Sherali-Adams (SA) hierarchy, which is a natural extension to MAP-LP. The principles we demonstrate, however, apply to other hierarchies as well.

In order to obtain a k -level SA linear program, called MAP-LP $_k$, we define the set of constraints $SA_{\mathcal{I}_k}$.

$$SA_{\mathcal{I}_k} \equiv \left\{ \begin{array}{ll} \mu_{\mathcal{I}_k}(x_{\mathcal{I}_k}) \geq 0 & \forall i \in \mathcal{I}, \forall x_{\mathcal{I}} \in \{0, 1\}^{|\mathcal{I}|} \\ \sum_{x_i} \mu_{\mathcal{I}_k}(x_{\mathcal{I}_k}) = \mu_{\mathcal{I} \setminus \{i\}}(x_{\mathcal{I} \setminus \{i\}}) \end{array} \right.$$

For later references, let the constraint $\mu_{\mathcal{I}_k}(x_{\mathcal{I}_k}) \geq 0$ be called a **non-negativity constraint**, and let the constraint $\sum_{x_i} \mu_{\mathcal{I}_k}(x_{\mathcal{I}_k}) = \mu_{\mathcal{I} \setminus \{i\}}(x_{\mathcal{I} \setminus \{i\}})$ be called a **local constraint**.

MAP-LP_k can be obtained by adding $\mu_{\mathcal{I}}$ vectors and respective $SA_{\mathcal{I}}$ constraints to MAP-LP_{k-1}, for each ordered subset of indices $\mathcal{I}$ of size k taken from $\{1, \dots, n\}$. As a starting point, MAP-LP₃ is defined according to Figure 1, complemented by local constraints on all triples and pairs of variables. MAP-LP₄ can then be obtained from MAP-LP₃, by introducing $\mu_{\mathcal{I}}$ variables corresponding to all clusters of size 4, adding all constraints of the form $\sum_{x_l} \mu_{ijkl}(x_i, x_j, x_k, x_l) = \mu_{ijk}(x_i, x_j, x_k)$.

Exploiting Symmetry in Linear Programs

The SA hierarchy can quickly generate large and intractable LPs, even for levels as low as 3. Fortunately, LPs that are highly symmetrical can have their dimension substantially reduced. This aspect was originally discussed for ILPs and Semi-Definite Programs, see e.g. (Margot 2010; Gatermann and Parrilo 2004), and has recently been adapted to LPs using a quasi-linear time preprocessing step and introduced to the domain of relational probabilistic models (Mladenov, Ahmadi, and Kersting 2012). We will now formalize this notion of symmetry for the benefit of this work.

Let $\mathcal{L}$ denote an LP, and let $\{\mu_i\}_i$ denote $\mathcal{L}$'s variables.

Definition 1. μ_i and μ_j are called **exchangeable** if there exists a permutation on $\mathcal{L}$'s variables that maps μ_i onto μ_j and yields the exact same LP. All variables exchangeable with one another form a set called an **orbit**. The set of all orbits in $\mathcal{L}$ is called $\mathcal{L}$'s **orbit partition**.

Theorem 1. (Bödi, Herr, and Joswig 2013) Let $\bar{\mu}_\rho$ denote the set of variables in orbit ρ . Then, there exists a solution for $\mathcal{L}$ under which $\forall \mu_i, \mu_j \in \bar{\mu}_\rho : \mu_i = \mu_j$, for each orbit ρ in $\mathcal{L}$'s orbit partition.

From theorem 1, it immediately follows that the dimension of the LP can be reduced by defining a variable μ_ρ for each orbit, and replacing each variable μ_i whose orbit is ρ with μ_ρ . However, as the LP becomes substantially bigger and more complex in structure with each level of the SA hierarchy, such direct approach is unlikely to be feasible. In (Ostrowski 2012), the issue is tackled by incrementally generating, row by row, a compact MAP-LP_k, instead of generating the original MAP-LP_k and only then reducing its dimension. The method is able to generate MAP-LP₃ on many problem instances, yet the time required to generate the LP approaches hundreds of seconds in small problem instance, and climbs up to ten of thousands seconds in problem instances of no more than a few hundred of variables and constraints. LPs for relational models, however, can easily reach thousands if not millions of variables and constraints, and may be found unsuitable for this type of approach.

Relational MRF

Probabilistic Relational Models (PRM) are representations of probabilistic models using the first-order predicate logic. Two commonly used PRMs are the *parfactor model* (Poole 2003) and *Markov logic network* (Richardson and Domingos 2006), and each can be considered a relational representation of an MRF. Here, we adapt a slightly modified representation of the parfactor model, as to allow a consistent set of notations for both relational and non-relational cases. We

include a brief representation of the model, and refer to (de Salvo Braz, Amir, and Roth 2005) for a formal introduction.

A *domain* is a set of constants, called *domain objects*, that represent distinctive entities in the modeled world, e.g. $\{Alice, Bob, Carol, \dots\}$. A logical variable (*lvar*) is a variable whose assignment range is associated with some domain. An *atom* is an atomic formula of the form $p(t_1, \dots, t_n)$, where the symbol p is called a *predicate*¹, and each term t_i is either a domain object or an lvar. A *ground atom* is an atom whose t_i terms are all domain objects. Non-ground atoms are collections of ground atoms, all sharing the same assignment range, and describing a certain property of an individual (e.g. smoker) or some relation between individuals (e.g. friendship). A *ground substitution* $\{X_i/o_i\}_i$, is the replacement of each lvar X_i with a domain object o_i .

The parfactor model $\mathcal{M}$ (aka *relational MRF*) is a collection of relational factors, called *parfactors*. A parfactor is a tuple (θ, A, R) , consisting of a function $\theta : \text{range}(A) \rightarrow \mathbb{R}$, an ordered set of atoms A , and a set of constraints R imposed on A 's lvars. Grounding a parfactor is done by applying all ground substitutions that are consistent with R , resulting in a collection of factors. The ground atoms then serve as random variables in the ground MRF. A notation $\theta(A | R)$ is commonly used to denote a parfactor. For example, parfactor $\theta(\text{smokes}(X), \text{smokes}(Y), \text{friends}(X, Y) | X \neq Y)$, whose ground instances in the domain $\{Alice, Bob\}$ are

- $\theta(\text{smokes}(Alice), \text{smokes}(Bob), \text{friends}(Alice, Bob))$
- $\theta(\text{smokes}(Bob), \text{smokes}(Alice), \text{friends}(Bob, Alice))$

We restrict our attention to shattered (de Salvo Braz, Amir, and Roth 2005) models, consisting of inequality constraints of the form $X \neq Y$ only. Additionally, such inequality constraints will be imposed on each pair of lvars X, Y where, in their absence, a ground factor with multiple entries of the same ground atom may be produced. For instance, $\theta(p(X), p(Y))$ may produce a ground factor $\theta(p(o_1), p(o_1))$, and will therefore be split into two parfactors: $\theta(p(X), p(Y) | X \neq Y)$ and some $\theta'(p(X))$.

Lifting the Sherali-Adams Hierarchy

Before introducing our framework, which substantially reduces the size of relational MAP linear programs induced by the SA hierarchy, we make the following observation. If two clusters of MRF variables are symmetrical, then the vectors of MAP-LP variables corresponding to these clusters are symmetrical as well. More formally, let $x_{\mathcal{I}}$ and $x_{\mathcal{J}}$ denote two clusters of variables taken from MRF $\mathcal{M}$.

Definition 2. $x_{\mathcal{I}}$ and $x_{\mathcal{J}}$ are called **exchangeable** if there exists a permutation π on $\mathcal{M}$'s random variables that yields the exact same MRF, and under which each variable $x_i \in x_{\mathcal{I}}$ is mapped onto a distinctive $x_j \in x_{\mathcal{J}}$.

Observation 1. If $x_{\mathcal{I}}$ and $x_{\mathcal{J}}$ are exchangeable, then the LP variables in vectors $\mu_{\mathcal{I}}$ and $\mu_{\mathcal{J}}$ of a respective MAP-LP_k are exchangeable.

Thus, if we obtain indices of only non-exchangeable MRF clusters, then we can form a compact MAP-LP representing

¹Although the term predicate is used, atoms are not restricted to Boolean assignments.

Algorithm 1: GENERATECOMPACTMAPLP_k

Input: Relational MRF $\mathcal{M}^r$, an integer k

Output: MAP-LP_k of reduced dimension

```
1 objective = 0, constraints = {}
2 for  $d = 1$  to  $k$  do
3   for  $\mathcal{I} \in \text{CANONICALCLUSTERS}(\mathcal{M}^r, d)$  do
4     Let  $\mu_{\mathcal{I}}$  be a vector of LP variables of size  $2^d$ 
5     Add non-negativity constraints on  $\mu_{\mathcal{I}}$ 
6      $(\theta_{\mathcal{I}}, x_{\mathcal{I}}) = \text{CANONICALFACTOR}(\mathcal{M}^r, \mathcal{I})$ 
7     Add  $\#\mathcal{I} \cdot \langle \theta_{\mathcal{I}}, \mu_{\mathcal{I}} \rangle$  to the objective
8     for  $i \in \mathcal{I}$  do
9        $\mathcal{I}' = \text{CANONICALCLUSTER}(\mathcal{M}^r, \mathcal{I} \setminus \{i\})$ 
10      Add local constraints between  $\mu_{\mathcal{I}}$  and  $\mu_{\mathcal{I}'}$ 
11 return objective, constraints
```

the k -level of the Sherali-Adams hierarchy. The unique representative clusters will be called *canonical clusters* henceforth, and their generation will be, at this point, associated with an oracle which supports the following queries:

1. $\text{CANONICALCLUSTERS}(\mathcal{M}^r, d)$ – generating the indices of all canonical clusters of size d in the relational MRF.
2. $\text{CANONICALCLUSTER}(\mathcal{M}^r, \mathcal{I})$ – retrieving the indices of a canonical cluster which is exchangeable with $x_{\mathcal{I}}$.
3. $\text{CANONICALFACTOR}(\mathcal{M}^r, \mathcal{I})$ – retrieving a ground factor $(\theta_{\mathcal{I}}, x_{\mathcal{I}})$ associated² with the canonical cluster $x_{\mathcal{I}}$.
4. $\#\mathcal{I}$ – retrieving the number of clusters exchangeable with $x_{\mathcal{I}}$ ³ in the relational MRF.

Algorithm 1 introduces the framework for generating a compact MAP-LP, making use of the oracle’s capabilities. The algorithm generates all LP variables corresponding to canonical clusters of various sizes, injects non-negativity and locality constraints, and updates the linear objective function with the matching log-potential tables. Note that the effect of the log-potential is multiplied by $\#\mathcal{I}$ to compensate for the effective removal of exchangeable variables from the LP. Also note that local constraints cannot be properly formed without invoking the oracle’s CANONICALCLUSTER query, since having $\mathcal{I}$ as the set of indices of a canonical cluster, does not guarantee the same for $\mathcal{I} \setminus \{i\}$. Finally, we define $\mu_{\emptyset}$ as a constant whose value is 1, thereby allowing local constraints to be added for clusters of size 1.

Exchangeable Clusters in the Relational MRF

Algorithm 1 establishes a clear motivation for shifting the attention from symmetry analysis in the LP domain to the domain of the original graphical model. In particular, we’re interested in capturing symmetry of clusters, as induced by

²An all zero factor is returned in case of no such association.

³In the relational setting, clusters corresponding to ground instances of the same relational factor are necessarily exchangeable, if the parfactor consists of inequality constraints on all pairs of lvars. If so, $\#\mathcal{I}$ is simply the number of ground instances of the parfactor. When not assuming this form, it is possible for two (or more) canonical clusters to be associated with the same parfactor. $\#\mathcal{I}$ will then be obtained combinatorically.

the first-order formulation of the relational MRF. We begin with a known result (Bui, Huynh, and Riedel 2013).

Theorem 2. *Let π_o be a renaming permutation on the domain objects of $\mathcal{M}^r$. Let $x_{\mathcal{I}}$ and $x_{\mathcal{J}}$ be clusters of ground atoms, mapped one onto the other under π_o . Then, $x_{\mathcal{I}}$ and $x_{\mathcal{J}}$ are exchangeable.*

We demonstrate the above with an example. Let $\pi_o = \{o_1 \rightarrow o_2, o_2 \rightarrow o_1\}$ be a renaming permutation on domain objects $\{o_1, o_2\}$. Then, the two clusters of ground atoms⁴ $\{r_{(1,2)}, p_2\}$ and $\{r_{(2,1)}, p_1\}$ are necessarily exchangeable. Nevertheless, renaming permutations cannot rule out exchangeability in all cases. For instance, clusters $\{p_1, q_2\}$ and $\{p_1, q_1\}$ are indeed exchangeable in parfactor $\theta(p(X), q(Y))$, although they do not conform to any renaming permutation. In the following subsection we present a generalization to the renaming permutation criterion, allowing us to capture these cases of exchangeability as well.

Cluster Signature Graph (CSG)

Our approach is based on a reduction from a cluster of ground atoms to a graph, such that if two graphs are isomorphic, then their respective clusters are exchangeable. For that purpose, we define the *Cluster Signature Graph* (CSG). Intuitively, the CSG depicts which ground atoms in the respective cluster must “move together” in any permutation which aims to preserve the structure of the relational MRF. For instance, a structure preserving permutation in parfactor $\theta(r(X, Y), p(X))$ would be required to coordinate the images of ground atoms $\{r_{(1,2)}, p_1\}$ in a way that guarantees identical substitutions on lvar X . Consequently, their images could be $\{r_{(3,5)}, p_3\}$, but never $\{r_{(3,5)}, p_4\}$. Before formalizing this restriction, we make a distinction between substitutions in different positions of the ground atom. In $r_{(3,5)}$, a substitution X/o_3 was applied on the lvar in position 1, whereas a substitution Y/o_5 was applied in position 2. With this, we are ready for the following definition.

Definition 3. *Ground atoms x_i and x_j are **substitution coupled** in positions ω_i and ω_j , respectively, if (1) they share a ground factor originating from a substitution on the same lvar in the respective ω positions, or (2) there exists a ground atom x_k that is substitution coupled with both in their respective ω positions.*

Example: In parfactor $\theta(r(X, Y), p(X))$, the ground atoms $r_{(1,2)}$ and $r_{(1,3)}$ are substitution coupled in positions 1 and 1, respectively, since they each share a ground factor with p_1 . However, $r_{(2,1)}$ and $r_{(3,1)}$ are not substitution coupled, although they share the same substitution on lvar Y .

In order to provide a full picture on substitution coupling, the CSG must also integrate inequality constraints found in the relational MRF. One way to achieve the above, is to determine substitution coupling in a slightly modified, constraint-less model, and project the coupling property onto the original model, as follows. For every constraint $X \neq Y$ found in a parfactor consisting of atom $r(X, Y)$, $p(X)$ or $q(Y)$, add a respective atom $r(Y, X)$, $p(Y)$ or $q(X)$ to the parfactor and remove the constraint. The substitution

⁴Let p_i and $r_{(i,j)}$ denote $p(o_i)$ and $r(o_i, o_j)$, respectively.

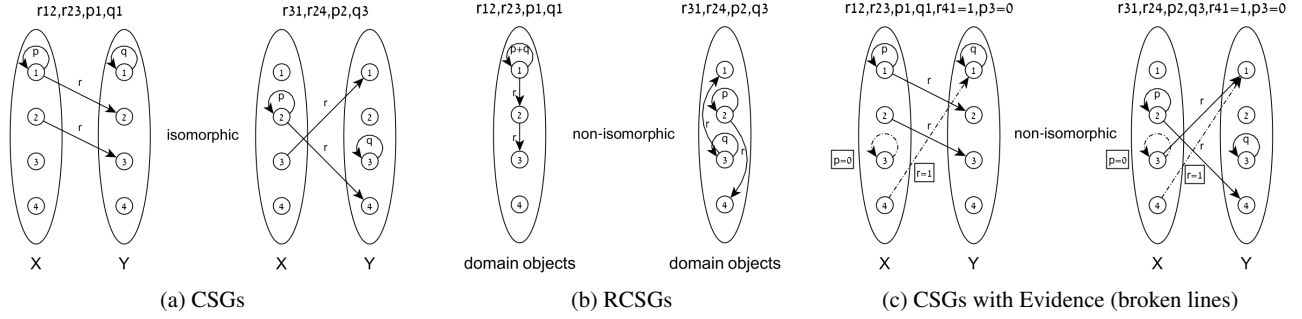


Figure 2: Cluster Signature Graphs in $\theta(r(X, Y), p(X), q(Y))$

coupling of the modified model now requires a "coordination" between lvars X and Y , as they occupy the same position in one of the atoms, and thus fulfills the original requirement. We now proceed with the main definition.

Definition 4. The CSG of cluster $x_{\mathcal{I}}$ in relational MRF $\mathcal{M}^r$, is a directed colored multigraph $G = (V, E, C)$, where V is a set of vertices, E is a set of directed edges and C is a coloring function, mapping each edge to a color. Edges and colors in the CSG are defined as follows.

1. For each $x_i \in x_{\mathcal{I}}$ originating from a unary predicate p , let G contain a node u and a loop carrying the color 'p'.
2. For each $x_i \in x_{\mathcal{I}}$ originating from a binary predicate r , let G contain nodes u and v to represent x_i 's positions, and a directed edge $u \rightarrow v$ carrying the color 'r'.
3. Let all substitution coupled nodes in G be merged into a single node, without removing any of the incident edges.

Theorem 3. If the CSGs of $x_{\mathcal{I}}$ and $x_{\mathcal{J}}$ in $\mathcal{M}^r$ are isomorphic, then $x_{\mathcal{I}}$ and $x_{\mathcal{J}}$ are exchangeable.

Proof outline.

Let $g_{\mathcal{I}}$ and $g_{\mathcal{J}}$ denote two isomorphic CSGs of $x_{\mathcal{I}}$ and $x_{\mathcal{J}}$, respectively. It can be shown that for every ground atom x_i added to CSG $g_{\mathcal{I}}$, there exists a respective ground atom x_j that, once added to CSG $g_{\mathcal{J}}$, induces the same isomorphism as before, but for the additional mapping between the new graph properties. Once both graphs consist of all ground atoms, the mapping between the edges of the graphs serves as a structure preserving permutation in the relational MRF. Hence, $x_{\mathcal{I}}$ and $x_{\mathcal{J}}$ are exchangeable. $\square$

Example: Let $\mathcal{M}^r$ consist of $\theta(r(X, Y), p(X), q(Y))$ and a domain size 4, let $x_{\mathcal{I}} = \{r_{(1,2)}, r_{(2,3)}, p_1, q_1\}$ and let $x_{\mathcal{J}} = \{r_{(3,1)}, r_{(2,4)}, p_2, q_3\}$. The two clusters are exchangeable, since there exists a permutation which maps one to the other while preserving the MRF structure. That permutation is $\pi = \{r_{(1,2)} \leftrightarrow r_{(2,4)}, p_1 \leftrightarrow p_2, q_2 \leftrightarrow q_4, r_{(2,3)} \leftrightarrow r_{(3,1)}, p_2 \leftrightarrow p_3, q_3 \leftrightarrow q_1\}$. See Figure 2a for illustration.

Renaming Signatures As mentioned earlier, renaming permutations constitute a stricter form of exchangeability compared with CSGs. A graphical depiction of this form, called *Renaming Cluster Signature Graph* (RCSG), can be obtained from existing CSGs by projecting their content onto the axis of domain objects (see Figure 2b).

Generating Canonical Clusters

Since isomorphic CSGs denote exchangeable clusters, canonical clusters can be obtained, in principle, by generating valid CSGs and filtering out isomorphic instances. This approach, unfortunately, cannot be easily implemented due to the complex rules that constitute a valid CSG. A more practical approach is to generate non-isomorphic instances of the simpler RCSG, and filter-out further, if one wishes, by converting flat RCSG representations into CSGs.

Generating non-isomorphic instances of graphs up to k edges is clearly harder than determining isomorphism in graphs of order k . However, one must keep in mind that this procedure involves generation of small local graphs, and does not involve any processing of the actual ground (and typically large) graphical model. In fact, this procedure is entirely domain size independent. Fortunately, there are commonly available tools dedicated to this task, producing graph instances very quickly. For instance, we were able to obtain the entire data required to lift MAP-LP₆ in a transitive model, in less than 23 seconds, which is only a fraction of the time the LP engine takes to solve that compact LP (more than 1000 seconds). Much like knowledge compilation, generation of graphs can be carried out once, offline, and serve all models consisting of similar predicate content.

Canonical Clusters in Presence of Evidence

As it turns out, the platform of CSG is almost seamlessly extended for handling evidence. Given cluster $x_{\mathcal{I}}$ and evidence e (a set of ground atoms paired with value assignments), a new CSG is defined on the union of $x_{\mathcal{I}}$ and e , where the substitution coupling property of ground atoms in e is defined as if they contained no evidence. Edges denoting evidence are colored according to the predicate symbol and the associated value assignment, e.g. $r = 1, p = 0$ etc. Exchangeability, as before, is determined via an isomorphism test on the CSGs.

The set of canonical clusters is now obtained by generating all possible compositions between (a) the evidence and (b) all canonical clusters obtained by ignoring the evidence, followed by the filtration of isomorphic instances. In Figure 2c, we see two different compositions of an "evidence free" canonical cluster with constant evidence, resulting in two clusters that are no longer symmetrical. The number of canonical clusters therefore becomes highly dependent on

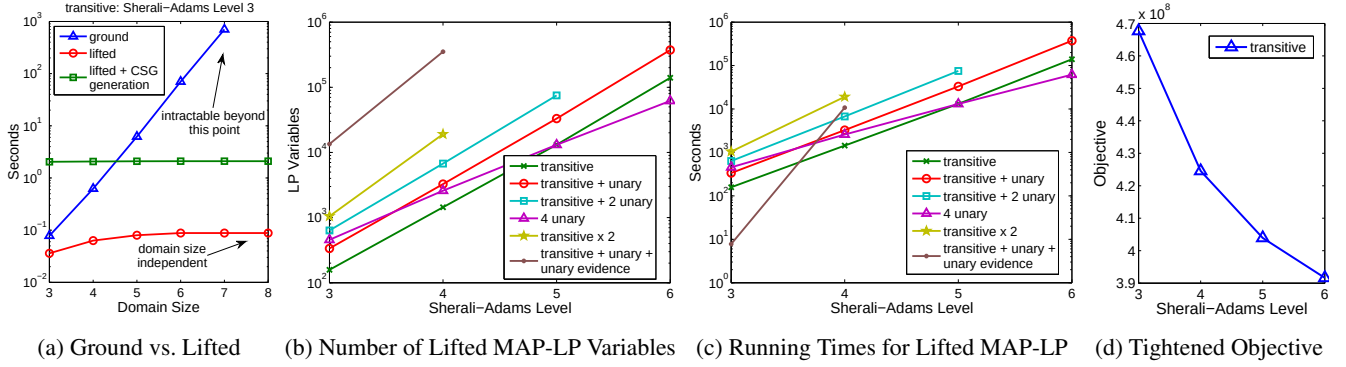


Figure 3: Empirical Results for Lifted MAP-LP

the type of the evidence. Evidence on binary predicates, even in small portions, can very quickly break the symmetry of the model up to intractability.

In contrast, evidence on unary predicates results in LPs which are size independent in both the global domain-size and the amount of evidence. The reason for this very convenient integration, is that the composition between "evidence-free" canonical clusters and the constant evidence partitions the CSG into two disconnected components: (a) one that only carries unary evidence, and (b) one that paints the nodes of the canonical cluster with the unary evidence occupying the shared nodes. Since the evidence is constant, the content of evidence in (b) determines (a) as well. Further, nodes in (a) are graphically indistinguishable, but for the colors of the atoms occupying the nodes. These inherent properties reduce the CSG isomorphism test to a comparison between the (b) components of the CSGs, ignoring (a) altogether.

With the introduction of evidence, a single parfactor in the compact LP may be matched with several canonical clusters, all assuming similar shape but differ in evidence nodes. For instance, let model $\theta(p(X), r(X, Y))$ consist of evidence $p_1 = 0$, and let $\{p_2, r_{(2,5)}\}$ and $\{r_{(1,3)}\}$ denote two different canonical clusters. The first cluster is matched with potential θ . The second is matched with the potential θ conditioned on $p_1 = 0$. The number of clusters exchangeable with each canonical cluster ($\#I$) can be determined combinatorially.

Implementation and Empirical Results

We demonstrate the framework with the challenging task of lifting MAP-LP_k in transitive models. Let parfactor $\theta(r(X, Y), r(X, Z), r(Y, Z) | X \neq Y, X \neq Z, Y \neq Z)$ be called, henceforth, the *transitive model*. We follow Algorithm 1, as follows. (1) Canonical clusters are obtained by generating all non-isomorphic instances of directed graphs with up to k edges, using the *nauty* (McKay and Piperno 2014) software package, where all graphs assume a *canonically labeled* form. (2) A canonical cluster of size 3 consisting of nodes u, v, w and edges $(u, v), (u, w), (v, w)$, is matched with the parfactor, and a linear expression involving its respective μ variables is added to the objective. (3) For each canonical cluster of size d , we obtain subset clus-

ters of size $d - 1$ for which local constraints are added, by removing edges (one each time) from the d size cluster's graph and obtaining a matching canonical labeling (*nauty*).

We conducted experiments on several models, in which table entries were set at random, with explicit discouragement of uniform assignment entries (Apsel and Brafman 2012), as to avoid trivial MAP solutions. The LP engine we used is the GNU Linear Programming Kit (GLPK) simplex solver. Figure 3a depicts a comparison between ground and lifted MAP-LP₃ for the transitive model, where we see that the ground LP could not solve instances beyond domain size 7. The lifted MAP-LP is slightly more compact in smaller domains, since canonical clusters are restricted to include no more nodes than the domain size. Beyond that, lifted MAP-LP is domain size independent. Figures 3b and 3c depict the number of LP variables and the running time of MAP-LPs for the following models: (1) Transitive; (2) Transitive with an additional parfactor and one unary atom; (3) Transitive with three additional parfactors and two unary atoms; (4) Four unary atoms and three parfactors; (5) Two binary atoms with two transitive parfactors; (6) Model no. 2 with unary evidence on 50% of the atoms. Figure 3d depicts the tightening of the objective with each level of the SA hierarchy. To the best of our knowledge, we present the highest lifting of the SA hierarchy in relational models, to date.

Conclusions

We presented a new framework for relational MAP-LP, based on the Sherali-Adams hierarchy, which exploits symmetry between clusters to form a compact linear program. We then presented a novel graphical platform (CSG) which allows for the quick generation of all relevant non symmetrical clusters. In effect, the CSG serves as a bounded-width window onto the automorphism group of the graphical model. Thus, CSGs may be used to apply tighter approximations within various frameworks of lifted inference.

Acknowledgements

We thank the anonymous reviewers for their comments and useful suggestions. We would also like to thank Eden Chlamtac for fruitful discussions, and for the initial insight

that helped to establish the direction pursued in this work. The authors were supported in part by ISF grant 933/13, the Lynn and William Frankel Center for Computer Science, and by the German-Israeli Foundation (GIF) for Scientific Research and Development, 1180-218.6/2011.

References

- Ahmadi, B.; Kersting, K.; Mladenov, M.; and Natarajan, S. 2013. Exploiting symmetries for scaling loopy belief propagation and relational training. *Machine Learning Journal* 92:91–132.
- Apse, U., and Brafman, R. I. 2011. Extended lifted inference with joint formulas. In Cozman, F. G., and Pfeffer, A., eds., *UAI*, 11–18. AUAI Press.
- Apse, U., and Brafman, R. I. 2012. Exploiting uniform assignments in first-order MPE. In de Freitas, N., and Murphy, K. P., eds., *UAI*, 74–83. AUAI Press.
- Bödi, R.; Herr, K.; and Joswig, M. 2013. Algorithms for highly symmetric linear and integer programs. *Math. Program.* 137(1-2):65–90.
- Bui, H.; Huynh, T.; and Riedel, S. 2013. Automorphism groups of graphical models and lifted variational inference. In *Proceedings of the Twenty-Ninth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-13)*, 132–141. Corvallis, Oregon: AUAI Press.
- de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted first-order probabilistic inference. In Kaelbling, L. P., and Saffioti, A., eds., *IJCAI*, 1319–1325. Professional Book Center.
- Gatermann, K., and Parrilo, P. A. 2004. Symmetry groups, semidefinite programs, and sums of squares. *Journal of Pure and Applied Algebra* 192(1):95–128.
- Jha, A.; Gogate, V.; Meliou, A.; and Suciu, D. 2010. Lifted inference from the other side: The tractable features. In *Proc. of the 23rd Annual Conference on Advances in Neural Information Processing Systems (NIPS-2010)*, 973–981.
- Karmarkar, N. 1984. A new polynomial-time algorithm for linear programming. *Combinatorica* 4(4):373–396.
- Kersting, K.; Ahmadi, B.; and Natarajan, S. 2009. Counting Belief Propagation. In *Proc. of the 25th Conf. on Uncertainty in Artificial Intelligence (UAI-09)*.
- Kersting, K. 2012. Lifted probabilistic inference. In *Proceedings of ECAI-2012*. IOS Press.
- Khachiyan, L. G. 1979. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR* 244:1093–1096.
- Lasserre, J. B. 2002. An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. *SIAM Journal on Optimization* 12(3):756–769.
- Lovász, L., and Schrijver, A. 1991. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization* 1(2):166–190.
- Margot, F. 2010. Symmetry in integer linear programming. In *50 Years of Integer Programming 1958-2008*. Springer. 647–686.
- McKay, B. D., and Piperno, A. 2014. Practical graph isomorphism, {II}. *Journal of Symbolic Computation* 60(0):94 – 112.
- Mladenov, M.; Ahmadi, B.; and Kersting, K. 2012. Lifted linear programming. In *15th Int. Conf. on Artificial Intelligence and Statistics (AISTATS 2012)*, 788–797. Volume 22 of JMLR: W&CP 22.
- Mladenov, M.; Globerson, A.; and Kersting, K. 2014. Efficient lifting of MAP LP relaxations using k -locality. In *17th Int. Conf. on Artificial Intelligence and Statistics (AISTATS 2014)*.
- Niepert, M. 2012. Markov chains on orbits of permutation groups. In *Proc. of the 28th Conf. on Uncertainty in Artificial Intelligence (UAI)*.
- Noessner, J.; Niepert, M.; and Stuckenschmidt, H. 2013. Rockit: Exploiting parallelism and symmetry for MAP inference in statistical relational models. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI)*.
- Ostrowski, J. 2012. Using symmetry to optimize over the Sherali-Adams relaxation. In *ISCO*, 59–70.
- Poole, D. 2003. First-order probabilistic inference. In Gottlob, G., and Walsh, T., eds., *IJCAI*, 985–991. Morgan Kaufmann.
- Ravikumar, P.; Agarwal, A.; and Wainwright, M. J. 2010. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *The Journal of Machine Learning Research* 11:1043–1080.
- Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.
- Schrijver, A. 1998. *Theory of Linear and Integer Programming*. Wiley Series in Discrete Mathematics & Optimization. John Wiley & Sons.
- Sherali, H. D., and Adams, W. P. 1990. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math.* 3(3):411–430.
- Shimony, S. E. 1994. Finding MAPs for belief networks is NP-hard. *Artif. Intell.* 68(2):399–410.
- Singla, P., and Domingos, P. 2008. Lifted First-Order Belief Propagation. In *Proc. of the 23rd AAAI Conf. on Artificial Intelligence (AAAI-08)*, 1094–1099.
- Sontag, D., and Jaakkola, T. 2007. New outer bounds on the marginal polytope. In *NIPS*.
- Sontag, D.; Globerson, A.; and Jaakkola, T. 2011. Introduction to dual decomposition for inference. *Optimization for Machine Learning* 1.
- Sontag, D. A. 2010. *Approximate inference in graphical models using LP relaxations*. Ph.D. Dissertation, Massachusetts Institute of Technology.
- Van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and Raedt, L. D. 2011. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, 2178–2185.
- Yedidia, J. S.; Freeman, W. T.; and Weiss, Y. 2003. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium* 8:236–239.

Efficient Markov Logic Inference for Natural Language Semantics

Islam Beltagy

Department of Computer Science
The University of Texas at Austin
Austin, Texas 78712
beltagy@cs.utexas.edu

Raymond J. Mooney

Department of Computer Science
The University of Texas at Austin
Austin, Texas 78712
mooney@cs.utexas.edu

Abstract

Using Markov logic to integrate logical and distributional information in natural-language semantics results in complex inference problems involving long, complicated formulae. Current inference methods for Markov logic are ineffective on such problems. To address this problem, we propose a new inference algorithm based on SampleSearch that computes probabilities of complete formulae rather than ground atoms. We also introduce a modified closed-world assumption that significantly reduces the size of the ground network, thereby making inference feasible. Our approach is evaluated on the recognizing textual entailment task, and experiments demonstrate its dramatic impact on the efficiency of inference.

1 Introduction

A novel approach to natural-language semantics that uses Markov Logic Networks (MLNs) (Richardson and Domingos 2006) to integrate logical and distributional approaches to linguistic meaning was recently proposed by Beltagy et al. (2013). However, using this approach to solve problems in recognizing textual entailment (RTE) (i.e. determining whether one natural-language sentence reasonably implies or contradicts another) (Dagan et al. 2013) results in complex inference problems that are intractable for existing MLN inference methods.

Given a set of weighted formulae in first-order logic, MLNs use them as a template to construct a Markov network that defines a probability distribution over possible worlds. MLN inference calculates the probability of query Q given a set of evidence E and a set of weighted clauses R in first-order logic. MLN inference is computationally difficult, and various sampling algorithms have been proposed for approximate inference (Domingos and Lowd 2009).

The inference task addressed by Beltagy et al. (2013) has two distinct features that tend to violate the implicit assumptions behind the currently available MLN inference algorithms and their implementations. First, the query is a complex logical formula, while available MLN tools calculate marginal probabilities of ground atoms only. In this paper, we introduce an inference algorithm based on Sample-

Search (Gogate and Dechter 2011) that directly calculates the probability of complete formulae.

The second problem with the resulting inference tasks is the large number of predicates and long formulae. This results in an extremely large number of ground clauses that existing inference algorithms fail to handle efficiently. Even recent advancements in lifted inference (Singla and Domingos 2008; Gogate and Domingos 2011) are not able to reduce the size of the ground network to a feasible size. Lifted inference relies on finding structural regularities (symmetries) in the MLN and avoiding grounding repeated structures. However, for these regularities to be frequent enough, the implicit assumption is that formulae are typically short and the set of evidence is large. These assumptions do not fit the inference problems we are addressing, which involve limited number of ground facts but complex logical formulae, which makes lifting ineffective.

To address this problem, we propose a modified closed-world assumption that helps reduce the size of the ground network and make inference tractable. Any ground atom that is *unreachable* from the evidence, is considered to be False. A ground atom is said to be *reachable* from the evidence if there is a way to propagate the evidence through the formulae and reach this ground atom. Note that this notion of reachability is different from reachability in the graph theoretic sense. The intuition behind our modified closed-world assumption is related to the idea of imposing a low prior on all predicates, which means that all ground atoms are effectively assumed to be False by default unless they can be inferred from the available evidence and background knowledge. However, by explicitly providing evidence that unreachable atoms are False, the size of the ground network is dramatically reduced.

The rest of the paper is organized as follows, section 2 is the background section, section 3 discusses an inference algorithm that supports complex query formulae, section 4 explains the modified closed-world assumption, section 5 is the experimental evaluation, and sections 6 and 7 discuss future work and conclusions, respectively.

2 Background

2.1 Logical Semantics

Logic-based representations of natural language meaning map a natural sentence into a logical form (Montague 1970). They handle many complex semantic phenomena such as relational propositions, logical operators, and quantifiers; however, they cannot handle “graded” aspects of meaning in language because they are binary by nature.

Boxer (Bos 2008) is an example of a wide-coverage logical-semantic tool that maps a natural sentence to first-order logic. For example, consider the two sentences “A man is driving a car”, and “A guy is driving a vehicle”. They become:

$$T : \exists x, y, z. \text{man}(x) \wedge \text{agent}(y, x) \wedge \text{drive}(y) \wedge \text{patient}(y, z) \wedge \text{car}(z)$$

$$H : \exists x, y, z. \text{guy}(x) \wedge \text{agent}(y, x) \wedge \text{drive}(y) \wedge \text{patient}(y, z) \wedge \text{vehicle}(z)$$

2.2 Distributional Semantics

Distributional models use statistics on large corpora to predict semantic similarity of words and phrases (Turney and Pantel 2010; Mitchell and Lapata 2010), based on the observation that semantically similar words occur in similar contexts. Distributional models capture the graded nature of meaning, but do not adequately capture logical structure (Grefenstette 2013).

Distributional semantic knowledge can be encoded as weighted inference rules. For the example above, relevant rules could look like:

$$r_1 : \forall x. \text{man}(x) \Rightarrow \text{guy}(x) \mid w_1$$

$$r_2 : \forall x. \text{car}(x) \Rightarrow \text{vehicle}(x) \mid w_2$$

where w_1, w_2 are weights calculated from the distributional similarity between the rule’s antecedent and consequent.

2.3 Recognizing Textual Entailment

Recognizing Textual Entailment (RTE) (Dagan et al. 2013) is the task of determining whether one natural language text, the *premise* T , Entails, Contradicts, or not related (Neutral) to another, the *hypothesis* H .

Beltagy et al. (2013) proposed a framework to perform the RTE task using MLNs. Given T and H in first-order logic, and given the knowledge base KB generated from distributional semantics, MLN inference is then used to compute $P(H|T, KB)$, which is then used as a measure of the degree to which T entails H .

2.4 Markov Logic Network

Markov Logic Networks (MLNs) (Richardson and Domingos 2006) are a framework for probabilistic logic that employ weighted formulae in first-order logic to compactly encode complex undirected probabilistic graphical models (i.e., Markov networks). MLNs define a probability distribution over possible worlds. Probability of a given world x is denoted by:

$$P(X = x) = \frac{1}{Z} \exp \left(\sum_i w_i n_i(x) \right) \quad (1)$$

where Z is the partition function, i ranges over all formulae F_i is the MLN, w_i is the weight of F_i and n_i is the number of true groundings of F_i . MLN’s marginal inference calculates the probability $P(Q|E, R)$, where Q is a query, E is the evidence set, and R is the set of weighted formulae.

Alchemy (Kok et al. 2005) is the most widely used MLN implementation. It is a software package that contains implementations of a variety of MLN inference and learning algorithms.

2.5 MC-SAT

MC-SAT (Poon and Domingos 2006) is a popular and generally effective MLN inference algorithm. It combines ideas from MCMC and satisfiability. MC-SAT generates samples from the probability distribution specified by the weighted rules, and use these samples to estimate marginal probabilities of query ground atoms. For each sample, MC-SAT randomly selects a set of weighted clauses to sample from, where the probability of a clause to be added to the set exponentially increases with the rule’s weight. Hard clauses are always included in this set. Once a set of clauses is selected, clauses are sampled using SampleSAT (Wei, Erenrich, and Selman 2004). SampleSAT uses WalkSAT (Selman et al. 1993), an efficient satisfiability solver, to efficiently generate nearly uniform samples.

2.6 Lifted Inference

The grounding process of a first-order formula generates a large number of ground atoms and ground clauses. Lifting techniques exploit the logical structure of the problem and the similarities in the ground network to reduce computation and the size of the ground network.

Lifted First-Order Belief Propagation proposed in (Singla and Domingos 2008) is a lifted version of the well known inference algorithm, Belief Propagation. It starts by building a *lifted network* where similar ground atoms of a first-order atom are grouped in one *supernode*, and similar ground clauses of a first-order clause are grouped in one *superfeature*. Ground atoms are considered similar (and merged in the same supernode) if they send and receive the same messages at each step of belief propagation. Similarly, ground clauses are considered similar (and merged in the same superfeature) if the ground clauses send and receive the same messages at each step of belief propagation. The minimum size of the lifted network is the size of the MLN, and the maximum size is that of a full ground Markov network. After building the lifted network, belief propagation proceeds normally with different messages that take lifting into account.

Another lifting technique is proposed in (Gogate and Domingos 2011). It starts by showing that inference in the propositional case (full ground network) is equivalent to “weighted model counting”, then suggests an algorithm for weighted model counting that involves two basic steps, a decomposition and a splitting step. Then, it shows that inference in the first-order case is equivalent to “lifted weighted model counting”, and shows how to lift the basic decomposition and splitting steps.

2.7 SampleSearch

SampleSearch (Gogate and Dechter 2011) is an importance sampling algorithm for graphical models that have a mix of probabilistic and deterministic constraints. Importance sampling in general is problematic in the presence of determinism, because many of the generated samples violate the deterministic constraints, and they get rejected. Instead, SampleSearch combines the sampling process with a backtracking search that uses a SAT solver to modify the generated sample if it violates the deterministic constraints.

SampleSearch works on top of a base-level sampler. The base sampler generates samples, and then SampleSearch makes sure the final samples satisfy all hard constraints. We use an implementation of SampleSearch that uses a generalized belief propagation algorithm called Iterative Join-Graph Propagation (IJGP) (Dechter, Kask, and Mateescu 2002). This version is available online (Gogate 2014).

3 Inference with Complex Query Formulae

Current implementations of MLNs like Alchemy (Kok et al. 2005) do not allow queries to be complex formulae, they can only calculate probabilities of ground atoms. This section discusses an inference algorithm for arbitrary query formulae.

3.1 Standard Work-Around

Although current MLN implementations can only calculate probabilities of ground atoms, they can be used to calculate the probability of a complex formula through a simple work-around. The complex query formula Q is added to the MLN using the hard formula:

$$Q \leftrightarrow \text{result}(D) \mid \infty \quad (2)$$

where $\text{result}(D)$ is a new ground atom that is not used anywhere else in the MLN. Then, inference is run to calculate the probability of $\text{result}(D)$, which is equal to the probability of the formula Q .

However, this approach can be very inefficient for some queries. For example, consider the query Q ,

$$Q : \exists x, y, z. \text{man}(x) \wedge \text{agent}(y, x) \wedge \text{drive}(y) \wedge \text{patient}(y, z) \wedge \text{car}(z) \quad (3)$$

This form of existentially quantified formula with a list of conjunctively joined atoms, is very common in the inference problems we are addressing, so it is important to have efficient inference for such queries. However, using this Q in equation 2 results in a very inefficient MLN. The direction $Q \leftarrow \text{result}(D)$ of the double-implication in equation 2 is very inefficient because the existentially quantified formula is replaced with a large disjunction over all possible combinations of constants for variables x, y and z (Gogate and Domingos 2011). Generating this disjunction, converting it to clausal form, and running inference on the resulting ground network becomes increasingly intractable as the number of variables and constants grows.

3.2 New Inference Method

Instead, we propose an inference algorithm to directly calculate the probability of complex query formulae. The probability of a formula is the sum of the probabilities of the possible worlds that satisfy it. Gogate and Domingo (2011) show that to calculate the probability of a formula Q given a probabilistic knowledge base K , it is enough to compute the partition function Z of K with and without Q added as a hard formula:

$$P(Q \mid K) = \frac{Z(K \cup \{(Q, \infty)\})}{Z(K)} \quad (4)$$

Therefore, all we need is an appropriate algorithm to estimate the partition function Z of a Markov network. Then, we construct two ground networks, one with the query and one without, and estimate their Z s using that estimator. The ratio between the two Z s is the probability of Q .

We tried to estimate Z using a harmonic-mean estimator on the samples generated by MC-SAT, but we found that the estimates are highly inaccurate as shown in (Venugopal and Gogate 2013). So, the partition function estimator we use is SampleSearch (Gogate and Dechter 2011). SampleSearch has been shown to be an effective sampling algorithm when there is a mix of probabilistic and deterministic (hard) constraints, a fundamental property of the inference problems we address.

For the example Q in equation 3, in order to avoid generating a large disjunction because of the existentially quantified variables, we replace Q with its negation $\neg Q$, so the existential quantifiers are replaced with universals, which are easier to ground and perform inference upon. Finally, we compute the probability of the query $P(Q) = 1 - P(\neg Q)$. Note that replacing Q with $\neg Q$ cannot make inference with the technique discussed in section 3.1 faster, because with $\neg Q$, the direction $\neg Q \rightarrow \text{result}(D)$ suffers from the same problem of the existential quantifiers instead of the other direction $\neg Q \leftarrow \text{result}(D)$.

4 Modified Closed-World Assumption

This section explains the problem with the explosive size of the ground network, and why lifting techniques are not enough to solve it. Next it discusses the relationship between the traditional low prior on predicates, and our modified closed-world assumption. Finally, it defines our modified closed-world assumption and describes how it is implemented.

4.1 Problem Description

In the inference problems we address, typically formulae are long, especially the query formula. First-order formulae result in an exponential number of ground clauses, where the number of ground clauses of a formula is $O(c^v)$, where c is number of constants in the domain, and v is number of variables in the formula. For any moderately long formula, the number of resulting ground clauses is infeasible to process in any reasonable time using available inference algorithms. Even recent lifting techniques (Singla and Domingos 2008; Gogate and Domingos 2011) that try to group similar ground

clauses to reduce the total number of nodes in the ground network, are not applicable here. Lifting techniques implicitly assume that c is large compared to v , and number of ground clauses is large because c is large. In our case, c and v are typically in the same range, and v is large, and this makes lifting algorithms fail to find similarities to lift.

4.2 Low prior

In the inference problems we address, as in most MLN applications, all atoms are initialized with a low prior. This low prior means that, by default, all groundings of an atom have very low probability, unless they can be inferred from the evidence and knowledge base. However, we found that a large fraction of the ground atoms cannot be inferred, and their probabilities remain very low. This suggests that these ground atoms can be identified and removed in advance with very little impact on the approximate nature of the inference. As the number of such ground atoms is large, this has the potential to dramatically decrease the size of the ground network. Our modified closed-world assumption was created to address this issue.

4.3 Definition

Closed-world, open-world and our modified closed-world assumptions are different ways of specifying what ground atoms are initialized to True, False or Unknown. True and False ground atoms are used to construct the appropriate network but are not part of the final ground Markov network. Only Unknown ground atoms participate in probabilistic inference. All ground atoms specified as evidence are known (True or False). The difference between the three assumptions is in the non-evidence ground atoms. With a closed-world assumption, non-evidence ground atoms are all False. In case of the open-world assumption, non-evidence ground atoms are all Unknown and they are all part of the inference task. In case of our modified closed-world atom, non-evidence ground atoms are False by default, unless they are *reachable* from any of the evidence, or from a ground atom in an input formula.

4.4 Reachability

A ground atom is said to be *reachable* from the evidence if there is a way to propagate the evidence through the formulae and reach this ground atom. The same applies for ground atoms specified in an input formula. For example, consider the evidence set E , and clauses r_1, r_2 :

$$\begin{aligned} E &: \{ g(C_1), h(C_2) \} \\ r_1 &: \forall x, y. g(x) \vee h(y) \vee i(x, y) \\ r_2 &: \forall x, y. j(x) \vee k(y) \vee i(x, y) \end{aligned}$$

From r_1 , variables x, y can be assigned the constants C_1, C_2 respectively because of the evidence $g(C_1), h(C_2)$. Then, this evidence gets propagated to $i(C_1, C_2)$, so the ground atom $i(C_1, C_2)$ is Unknown. From r_2 , the variables x, y can be assigned the constants C_1, C_2 respectively because of the Unknown ground atom $i(C_1, C_2)$, and this gets propagated to $j(C_1), k(C_2)$, so ground atoms $j(C_1), k(C_2)$ are also Unknown. All other ground atoms, except the evidence $g(C_1)$

Algorithm 1 Grounding with modified closed-world assumption

Input $R: \{K \cup Q\}$ set of first-order clauses, where K is the set of clauses from the input MLN, and Q is the set of clauses from the query.

Input E : set of evidence (list of ground atoms)

Output : a set of ground clauses with the modified closed-world assumption applied

```

1: Add all  $E$  to the reachable ground atoms
2: Add all ground atoms in  $R$  to reachable
3: repeat
4:   for all  $r \in R$  do
5:      $p$  = propagate reachable ground atoms between
       predicates sharing the same variable
6:     add propagated ground atoms ( $p$ ) to reachable
7:     if  $p$  not empty then
8:        $changed = true$ 
9:     end if
10:  end for
11: until not  $changed$ 
12: Generate False evidence for ground atoms
     $\notin$  reachable and add them to  $E$ 
13:  $GC$  = Use MLN's grounding process to ground clauses
     $R$ 
14: for all  $gc \in GC$  do
15:    $gc$  =  $gc$  after substituting values of known ground
       atoms in  $E$ 
16:   if  $gc = True$  then
17:     drop  $gc$ 
18:   else if  $gc = False$  then
19:     if  $gc$  is a grounding of one of  $Q$ 's clauses then
20:       Terminate inference with  $Z = 0$ 
21:     else
22:       if  $gc$  is hard clause then
23:         Error inconsistent MLN
24:       else
25:         drop  $gc$ 
26:       end if
27:     end if
28:   else
29:     keep  $gc$  in  $GC$ 
30:   end if
31: end for
32: return  $GC$ 

```

and $h(C_2)$, are False because they are not reachable from any evidence.

Note that the definition of reachability here (mcw-reachable) is different from the definition of reachability in graph theory (graph-reachable). Nodes can be graph-reachable but not mcw-reachable. For the example above, consider the full ground network of E and r_1 , which contains 8 nodes, and 4 cliques. It is a connected graph, and all nodes are graph-reachable from each others. However, as explained in the example, $i(C_1, C_2)$ is the only mcw-reachable node.

4.5 Algorithm and Implementation

Algorithm 1 describes the details of the grounding process with the modified closed-world assumption applied. Lines 1 and 2 initialize the reachable set with the evidence and any ground atom in R . Lines 3-11 repeatedly propagate evidence until there is no change in the reachable set. Line 12 generates False evidence for all unreachable ground atoms. Line 13 generates all ground clauses, then lines from 14-31 substitute values of the known ground atoms in the ground clauses. Alchemy drops all True and False ground clauses, but this does not work when the goal of the inference algorithm is to calculate Z . Lines from 16-30 describe the change. True ground clauses are dropped, but not False ground clauses. If a False ground clause is a grounding of one of Q 's clauses, then $Z = 0$ and there is no need to perform inference since there is no way to satisfy Q given E and R . If there is False hard clause, then this MLN is inconsistent. Otherwise, the False ground clause can be dropped. The resulting list of ground clauses GC are then passed to the inference algorithm to estimate Z .

5 Evaluation

This section evaluates the two techniques proposed in this paper using the RTE task, demonstrating the effectiveness of both components.

5.1 Task

Given the two natural-language sentences T and H represented in first-order logic, and given the background knowledge KB generated as in (Beltagy et al. 2013), two inferences are run, $P(H|T, KB)$ and $P(H|\neg T, KB)$.

The dataset used in our evaluation is "Sentences Involving Compositional Knowledge" (SICK) (Marelli et al. 2014). SICK is a new RTE dataset collected for the SemEval 2014 competition. Only the "training set" is available at this point, which consists of 5,000 pairs of sentences.

5.2 Systems Compared

- **mln**: This system uses MC-SAT (Richardson and Domingos 2006) for inference without any modifications. It uses the work-around explained in section 3.1 to calculate the probability of a complex query formula, and uses an open-world assumption.
- **mln+qf**: This system uses our SampleSearch inference to directly calculate the probability of a query formula (qf), while making an open-world assumption.
- **mln+mcw**: This system uses MC-SAT with the work-around for computing the probability of a complex query formula, but uses our modified closed-world (mcw) assumption.
- **mln+qf+mcw**: This is our proposed technique, inference that supports a query formula (qf) and makes a modified closed-world (mcw) assumption.

For all systems, we run the two inferences $P(H|T, KB)$ and $P(H|\neg T, KB)$, then train a classifier that maps their outputs to one of the classes: Entails, Contradicts or Neutral.

	Accuracy	CPU Time	Timeouts
mln	56.94%	2min 27s	9,578
mln+qf	68.74%	1min 51s	2,964
mln+mcw	65.80%	10s	252
mln+qf+mcw	71.80%	7s	212

Table 1: Systems' performance, accuracy, CPU Time for completed runs only, and number of Timeouts out of 10,000 runs

We use a 30 minute timeout for each MLN inference problem in order to make the experiments tractable. If the system times out, it outputs -1 indicating an error, and the final classifier learns to assign it to one of the three RTE classes. Usually, because the Neutral class is the largest, timeouts are classified as Neutral.

5.3 Metrics

- **Accuracy**: Percentage of correct classifications (Entail, Contradict, or Neutral) using 10-fold cross validation.
- **CPU Time (completed runs)**: Average CPU time per run for the completed runs only, i.e. timed out runs are not included.
- **Timeouts**: Number of inferences that timeout after 30 minutes. Total number of runs is 10,000.

5.4 Results and Discussion

Table 1 summarizes the results of the experiments. First, for the four different systems, the CPU time (average time per run for completed runs only) is very short compared to the length of the timeout (30 minutes). This shows the exponential nature of the inference algorithms, either the problem is small enough to finish in few minutes, or if it is slightly larger, it fails to finish in reasonable time.

Comparing the systems, the results clearly show that the base system, (**mln**), is not effective for the type of inference problems that we are addressing, almost all of the runs timed out. System **mln+qf** shows the impact of being able to calculate the probability of a complex query directly. It significantly improves the accuracy, and it lowers the number of timeouts; however, the number of timeouts is still large. System **mln+mcw** shows the impact of the modified closed-world assumption, demonstrating that makes inference significantly faster, since the number of unreachable ground atoms in our application is large compared to the total number of ground atoms. However, the accuracy of **mln+mcw** is lower than that of **mln+qf**, since calculating the probability of a query directly is more accurate than the standard work-around. Finally, **mln+qf+mcw** is both more accurate and faster than the other systems, clearly demonstrating the effectiveness of our overall proposed approach.

6 Future Work

An obvious extension to this work is a better algorithm for computing the probability of an arbitrary formula. Instead of making two separate runs of SampleSearch to estimate

two different Z s, it would be helpful to exploit the similarities between the two Markov networks (one with Q and one without Q) to reduce the amount of repeated computation. Also, it should be possible to optimize the calculations, or simplify them, knowing that we are really only interested in the ratio between the two Z s and not their individual values.

7 Conclusion

This paper has addressed the problem of making MLN inference feasible for textual inference that combines logical and distributional semantics, where queries are long, complex logical formulae (not just single ground atoms). We proposed a simple MLN inference algorithm based on Sample-Search to compute probabilities of formulae, and proposed a modified closed-world assumption that can dramatically reduce the size of the ground network. Experiments on a recent SemEval task in textual entailment demonstrated the effectiveness of these techniques.

Acknowledgments

We would like to thank Deepak Venugopal and Vibhav Gogate for insightful discussions.

This research was supported by the DARPA DEFT program under AFRL grant FA8750-13-2-0026. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the view of DARPA, DoD or the US government. Some experiments were run on the Mastodon Cluster supported by NSF Grant EIA-0303609.

References

- Beltagy, I.; Chau, C.; Boleda, G.; Garrette, D.; Erk, K.; and Mooney, R. 2013. Montague meets Markov: Deep semantics with probabilistic logical form. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM-13)*.
- Bos, J. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of Semantics in Text Processing (STEP-08)*.
- Dagan, I.; Roth, D.; Sammons, M.; and Zanzotto, F. M. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*.
- Dechter, R.; Kask, K.; and Mateescu, R. 2002. Iterative join-graph propagation. In *Proceedings of the Eighteenth conference on Uncertainty in Artificial Intelligence (UAI-02)*.
- Domingos, P., and Lowd, D. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.
- Gogate, V., and Dechter, R. 2011. Samplesearch: Importance sampling in presence of determinism. *Artificial Intelligence* 175(2):694–729.
- Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *27th Conference on Uncertainty in Artificial Intelligence (UAI-11)*.
- Gogate, V. 2014. IJGP-sampling and samplesearch. <http://www.hlt.utdallas.edu/~vgogate/ijgp-samplesearch.html>.
- Grefenstette, E. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Proceedings of Second Joint Conference on Lexical and Computational Semantics (*SEM-13)*.
- Kok, S.; Singla, P.; Richardson, M.; and Domingos, P. 2005. The Alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington. <http://alchemy.cs.washington.edu/>.
- Marelli, M.; Menini, S.; Baroni, M.; Bentivogli, L.; Bernardi, R.; and Zamparelli, R. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *LREC-2014*.
- Mitchell, J., and Lapata, M. 2010. Composition in distributional models of semantics. *Journal of Cognitive Science*.
- Montague, R. 1970. Universal grammar. *Theoria* 36:373–398.
- Poon, H., and Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies.
- Richardson, M., and Domingos, P. 2006. Markov logic networks. 62:107–136.
- Selman, B.; Kautz, H.; Cohen, B.; et al. 1993. Local search strategies for satisfiability testing. *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge*.
- Singla, P., and Domingos, P. 2008. Lifted first-order belief propagation. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI-08)*.
- Turney, P., and Pantel, P. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research* 37(1):141–188.
- Venugopal, D., and Gogate, V. 2013. Giss: Combining samplesearch and importance sampling for inference in mixed probabilistic and deterministic graphical models. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI-13)*.
- Wei, W.; Erenrich, J.; and Selman, B. 2004. Towards efficient sampling: Exploiting random walk strategies. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI-04)*.

Reasoning in the Description Logic BEL Using Bayesian Networks

İsmail İlkan Ceylan*

Theoretical Computer Science
TU Dresden, Germany
ceylan@tcs.inf.tu-dresden.de

Rafael Peñaloza†

Theoretical Computer Science, TU Dresden, Germany
Center for Advancing Electronics Dresden
penaloza@tcs.inf.tu-dresden.de

Abstract

We study the problem of reasoning in the probabilistic Description Logic $\mathcal{BEL}$. Using a novel structure, we show that probabilistic reasoning in this logic can be reduced in polynomial time to standard inferences over a Bayesian network. This reduction provides tight complexity bounds for probabilistic reasoning in $\mathcal{BEL}$.

1 Introduction

Description Logics (DLs) (Baader et al. 2007) are a family of knowledge representation formalisms tailored towards the representation of terminological knowledge in a formal manner. In their classical form, DLs are unable to handle the inherent uncertainty of many application domains. To overcome this issue, several probabilistic extensions of DLs have been proposed. The choice of a specific probabilistic DL over others depends on the intended application; these logics differ in their logical expressivity, their semantics, and their independence assumptions.

Recently, the DL $\mathcal{BEL}$ (Ceylan and Peñaloza 2014) was introduced as a means of describing certain knowledge that depends on an uncertain context, expressed by a Bayesian network (BN). An interesting property of this logic is that reasoning can be decoupled between the logical part and the BN inferences. However, despite the logical component of this logic being decidable in polynomial time, the best known algorithm for probabilistic reasoning in $\mathcal{BEL}$ runs in exponential time.

In this paper we use a novel structure, called the *proof structure*, to reduce probabilistic reasoning for a $\mathcal{BEL}$ knowledge base to probabilistic inferences in a BN. In a nutshell, a proof structure describes the class of contexts that entail the wanted consequence. A BN can be constructed to compute the probability of these contexts, which yields the probability of the entailment. Since this reduction can be done in polynomial time, it provides tight upper bounds for the complexity of reasoning in $\mathcal{BEL}$.

*Supported by DFG in the Research Training Group “RoSI” (GRK 1907).

†Partially supported by DFG within the Cluster of Excellence ‘cfAED’.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

2 Proof Structures in $\mathcal{EL}$

$\mathcal{EL}$ is a light-weight DL that allows for polynomial-time reasoning. It is based on *concepts* and *roles*, corresponding to unary and binary predicates from first-order logic, respectively. Formally, let N_C and N_R be disjoint sets of *concept names* and *role names*, respectively. $\mathcal{EL}$ concepts are defined through the syntactic rule $C ::= A \mid \top \mid C \sqcap C \mid \exists r.C$, where $A \in N_C$ and $r \in N_R$.

The *semantics* of $\mathcal{EL}$ is given in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty domain and $\cdot^{\mathcal{I}}$ is an *interpretation function* that maps every concept name A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every role name r to a set of binary relations $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is extended to $\mathcal{EL}$ concepts by defining $\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and $(\exists r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$.

The knowledge of a domain is represented through a set of axioms restricting the interpretation of the concepts.

Definition 1 (TBox). A *general concept inclusion (GCI)* is an expression of the form $C \sqsubseteq D$, where C, D are concepts. A *TBox* $\mathcal{T}$ is a finite set of GCIs. The *signature* of $\mathcal{T}$ ($\text{sig}(\mathcal{T})$) is the set of concept and role names appearing in $\mathcal{T}$. An interpretation $\mathcal{I}$ satisfies the GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; $\mathcal{I}$ is a *model* of the TBox $\mathcal{T}$ iff it satisfies all the GCIs in $\mathcal{T}$.

The main reasoning service in $\mathcal{EL}$ is subsumption checking, i.e., deciding the sub-concept relations between given concepts based on their semantic definitions. A concept C is *subsumed by* D w.r.t. the TBox $\mathcal{T}$ ($\mathcal{T} \models C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. It has been shown that subsumption can be decided in $\mathcal{EL}$ in polynomial time by a completion algorithm (Baader, Brandt, and Lutz 2005). This algorithm requires the TBox to be in *normal form*; i.e., where all GCIs are one of the forms $A \sqsubseteq B \mid A \sqcap B \sqsubseteq C \mid A \sqsubseteq \exists r.B \mid \exists r.B \sqsubseteq A$. It is well known that every TBox can be transformed into an equivalent one in normal form of linear size (Brandt 2004; Baader, Brandt, and Lutz 2005); for the rest of this paper, we assume that $\mathcal{T}$ is a TBox in normal form and GCI denotes a normalized subsumption relation.

In this paper, we are interested in deriving the GCIs in normal form that follow from $\mathcal{T}$; i.e. the *normalised logical closure* of $\mathcal{T}$. We introduce the *deduction rules* shown in Table 1 to produce the normalised logical closure of a TBox.

Table 1: Deduction rules for $\mathcal{EL}$.

$\mapsto$	Premises (S)	Result (α)
1	$\langle A \sqsubseteq B \rangle, \langle B \sqsubseteq C \rangle$	$\langle A \sqsubseteq C \rangle$
2	$\langle A \sqsubseteq \exists r.B \rangle, \langle B \sqsubseteq C \rangle$	$\langle A \sqsubseteq \exists r.C \rangle$
3	$\langle A \sqsubseteq \exists r.B \rangle, \langle C \sqsubseteq A \rangle$	$\langle C \sqsubseteq \exists r.B \rangle$
4	$\langle \exists r.A \sqsubseteq B \rangle, \langle B \sqsubseteq C \rangle$	$\langle \exists r.A \sqsubseteq C \rangle$
5	$\langle \exists r.A \sqsubseteq B \rangle, \langle C \sqsubseteq A \rangle$	$\langle \exists r.C \sqsubseteq B \rangle$
6	$\langle \exists r.A \sqsubseteq B \rangle, \langle B \sqsubseteq \exists r.C \rangle$	$\langle A \sqsubseteq C \rangle$
7	$\langle A \sqsubseteq \exists r.B \rangle, \langle \exists r.B \sqsubseteq C \rangle$	$\langle A \sqsubseteq C \rangle$
8	$\langle A \sqcap B \sqsubseteq C \rangle, \langle C \sqsubseteq X \rangle$	$\langle A \sqcap B \sqsubseteq X \rangle$
9	$\langle A \sqcap B \sqsubseteq C \rangle, \langle X \sqsubseteq A \rangle$	$\langle X \sqcap B \sqsubseteq C \rangle$
10	$\langle A \sqcap B \sqsubseteq C \rangle, \langle X \sqsubseteq B \rangle$	$\langle A \sqcap X \sqsubseteq C \rangle$
11	$\langle X \sqcap X \sqsubseteq C \rangle$	$\langle X \sqsubseteq C \rangle$

Each rule maps a set of premises to a GCI that is implicitly encoded in the premises. It is easy to see that the sets of premises cover all pairwise combinations of GCIs in normal form and that the deduction rules produce the normalised logical closure of a TBox. Moreover, the given deduction rules introduce axioms only in normal form, and do not create any new concept or role name. Hence, if $n = |\text{sig}(\mathcal{T})|$, the logical closure of $\mathcal{T}$ is computed after n^3 rule applications, at most.

Later on we will associate a probability to the GCIs in the TBox $\mathcal{T}$, and will be interested in computing the probability of a consequence. It will then be useful to be able not only to deduce the GCI, but also *all* the sub-TBoxes of $\mathcal{T}$ from which this GCI follows. Therefore, we store the traces of the deduction rules using a directed hypergraph.

Definition 2. A *directed hypergraph* is a tuple $H = (V, E)$ where V is a non-empty set of *vertices* and E is a set of *directed hyper-edges* of the form $e = (S, v)$ where $S \subseteq V$ and $v \in V$. A *path* from S to v in H is a sequence of hyper-edges $(S_1, v_1), (S_2, v_2), \dots, (S_n, v_n)$ such that $v_n = v$ and $S_i \subseteq S \cup \{v_j \mid 0 < j < i\}$ for every $i, 1 \leq i \leq n$. In this case, the path has *length* n .

Given a TBox $\mathcal{T}$ in normal form, we build the hypergraph $H_{\mathcal{T}} = (V_{\mathcal{T}}, E_{\mathcal{T}})$, where $V_{\mathcal{T}}$ is the set of all GCIs that follow from $\mathcal{T}$ and $E_{\mathcal{T}} = \{(S, \alpha) \mid S \mapsto \alpha, S \subseteq V_{\mathcal{T}}\}$ where $\mapsto$ is the deduction relation defined in Table 1. We call this hypergraph the *proof structure* of $\mathcal{T}$. The following lemma follows from the correctness of the deduction rules.

Lemma 3. Let $\mathcal{T}$ be a TBox in normal form, its proof structure $H_{\mathcal{T}} = (V_{\mathcal{T}}, E_{\mathcal{T}})$, $\mathcal{O} \subseteq \mathcal{T}$, and $C \sqsubseteq D \in V_{\mathcal{T}}$. There is a path from $\mathcal{O}$ to $C \sqsubseteq D$ in $H_{\mathcal{T}}$ iff $\mathcal{O} \models C \sqsubseteq D$.

Intuitively, $H_{\mathcal{T}}$ is a compact representation of all the possible ways in which a GCI can be derived from the GCIs present in $\mathcal{T}$. Traversing this hypergraph backwards, from a GCI α being entailed by $\mathcal{T}$, it is possible to construct all proofs for α ; hence the name “proof structure.” As mentioned before, $|V_{\mathcal{T}}| \leq |\text{sig}(\mathcal{T})|^3$; thus, it is enough to consider paths of length at most $|\text{sig}(\mathcal{T})|^3$.

Clearly, the proof structure $H_{\mathcal{T}}$ can be cyclic. To simplify the process of finding the causes of a GCI being entailed, we construct an *unfolded* version of this hypergraph by making different copies of each node in each level in or-

Algorithm 1 Construction of the pruned proof structure

Input: TBox $\mathcal{T}$

Output: $H = (W, F)$

```

1:  $V_0 \leftarrow \mathcal{T}, E_0 \leftarrow \emptyset, i \leftarrow 0$ 
2: do
3:    $i \leftarrow i + 1$ 
4:    $V_i := V_{i-1} \cup \{\alpha \mid S \mapsto \alpha, S \subseteq V_{i-1}\}$ 
5:    $E_i = \{(S, \alpha) \mid S \mapsto \alpha, S \subseteq V_{i-1}\}$ 
6: while  $V_i \neq V_{i-1}$  or  $E_i \neq E_{i-1}$ 
7:  $W := \{(\alpha, k) \mid \alpha \in V_k, 0 \leq k \leq i\}$ 
8:  $E := \{(S, (\alpha, k)) \mid (S, \alpha) \in E_k, 0 \leq k \leq i\} \cup$ 
9:    $\{(\{(\alpha, k-1)\}, (\alpha, k)) \mid \alpha \in V_{k-1}, 0 \leq k \leq i\}$ 
10: return  $(W, E)$ 

```

der to avoid cycles. In this case, nodes are pairs of GCIs and labels, where the latter indicates to which level the nodes belong in the hypergraph. We write $S^i = \{(\alpha, i) \mid \alpha \in S\}$ to denote the i -labeled set of GCIs in S . Let $n := |\text{sig}(\mathcal{T})|^3$, we start with the set $W_0 := \{(\alpha, 0) \mid \alpha \in \mathcal{T}\}$ and define the levels $0 \leq i < n$ inductively by

$$W_{i+1} := \{(\alpha, i+1) \mid S^i \subseteq W_i, S \mapsto \alpha\} \cup \{(\alpha, i+1) \mid (\alpha, i) \in W_i\}.$$

For each $i, 0 \leq i \leq n$, W_i contains all the consequences that can be derived by at most i applications of the deduction rules from Table 1. The *unfolded proof structure* of $\mathcal{T}$ is the hypergraph $H_{\mathcal{T}}^u = (W_{\mathcal{T}}, F_{\mathcal{T}})$, where $W_{\mathcal{T}} := \bigcup_{i=0}^n W_i$ and $F_{\mathcal{T}} := \bigcup_{i=1}^n F_i$, with

$$F_{i+1} := \{(S^i, (\alpha, i+1)) \mid S^i \subseteq W_i, S \mapsto \alpha\} \cup \{(\{(\alpha, i)\}, (\alpha, i+1)) \mid (\alpha, i) \in W_i\}$$

The following is a simple consequence of our constructions and Lemma 3.

Theorem 4. Let $\mathcal{T}$ be a TBox, and $H_{\mathcal{T}} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ and $H_{\mathcal{T}}^u = (W_{\mathcal{T}}, F_{\mathcal{T}})$ the proof structure and unfolded proof structure of $\mathcal{T}$, respectively. Then,

- for all $C \sqsubseteq D \in V_{\mathcal{T}}$ and all $\mathcal{O} \subseteq \mathcal{T}$, $\mathcal{O} \models C \sqsubseteq D$ iff there is a path from $\{(\alpha, 0) \mid \alpha \in \mathcal{O}\}$ to $(C \sqsubseteq D, n)$ in $H_{\mathcal{T}}^u$, and
- $(S, \alpha) \in E_{\mathcal{T}}$ iff $(S^{n-1}, (\alpha, n)) \in F_{\mathcal{T}}$.

The unfolded proof structure of a TBox $\mathcal{T}$ is thus guaranteed to contain the information of all possible causes for a GCI to follow from $\mathcal{T}$. Moreover, this hypergraph is acyclic, and has polynomially many nodes on the size of $\mathcal{T}$. Yet, this hypergraph may contain many redundant nodes. Indeed, it can be the case that all the simple paths in $H_{\mathcal{T}}$ starting from a subset of $\mathcal{T}$ are of length $k < n$. In that case, $W_i = W_{i+1}$ and $F_i = F_{i+1}$ hold for all $i \geq k$, modulo the second component. It thus suffices to consider the sub-hypergraph of $H_{\mathcal{T}}^u$ that contains only the nodes $\bigcup_{i=0}^k W_i$. Algorithm 1 describes a method for computing this pruned hypergraph. In the worst case, this algorithm will produce the whole unfolded proof structure of $\mathcal{T}$, but will stop the unfolding procedure earlier if possible. The **do-while** loop is executed at most $|\text{sig}(\mathcal{T})|^3$ times, and each of these loops requires at most $|\text{sig}(\mathcal{T})|^3$ steps; hence we obtain the following.

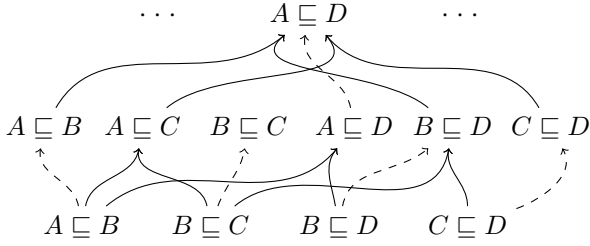


Figure 1: The first levels of an unfolded proof structure and the paths to $\langle A \sqsubseteq D \rangle$

Lemma 5. *Algorithm 1 terminates in time polynomial on the size of $\mathcal{T}$.*

We briefly illustrate the execution of Algorithm 1 on a simple TBox.

Example 6. Consider the following $\mathcal{EL}$ TBox $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C, B \sqsubseteq D, C \sqsubseteq D\}$. The first levels of the unfolded proof structure of $\mathcal{T}$ are shown in Figure 1.¹ The first level V_0 of this hypergraph contains a representative for each GCI in $\mathcal{T}$. To construct the second level, we first copy all the GCIs in V_0 to V_1 , and add a hyperedge joining the equivalent ones (represented by a dashed line in Figure 1). Afterwards, we apply all possible deduction rules to the elements of V_0 , and add a hyperedge from the premises at level V_0 to the conclusion at level V_1 (continuous lines). The same procedure is repeated at each subsequent level. Notice that the set of GCIs at each level is monotonically increasing. Additionally, for each GCI, the in-degree of each representative monotonically increases throughout the levels.

In the next section, we briefly recall $\mathcal{BEL}$, a probabilistic extension of $\mathcal{EL}$ based on Bayesian networks (Ceylan and Peñaloza 2014), and use the construction of the (unfolded) proof structure to reduce reasoning in this logic, to standard Bayesian network inferences.

3 The Bayesian Description Logic $\mathcal{BEL}$

The probabilistic Description Logic $\mathcal{BEL}$ extends $\mathcal{EL}$ by associating every GCI in a TBox with a probability. To handle the joint probability distribution of the GCIs, these probabilities are encoded in a Bayesian network (Darwiche 2009). Formally, a *Bayesian network* (BN) is a pair $\mathcal{B} = (G, \Phi)$, where $G = (V, E)$ is a finite directed acyclic graph (DAG) whose nodes represent Boolean random variables,² and Φ contains, for every node $x \in V$, a conditional probability distribution $P_{\mathcal{B}}(x \mid \pi(x))$ of x given its parents $\pi(x)$. If V is the set of nodes in G , we say that $\mathcal{B}$ is a BN over V .

BNs encode a series of conditional independence assumptions between the random variables; more precisely, every

¹For the illustrations we drop the second component of the nodes, but visually make the level information explicit.

²In their general form, BNs allow for arbitrary discrete random variables. We restrict w.l.o.g. to Boolean variables for ease of presentation.

variable $x \in V$ is conditionally independent of its non-descendants given its parents. Thus, every BN $\mathcal{B}$ defines a unique joint probability distribution (JPD) over V given by

$$P_{\mathcal{B}}(V) = \prod_{x \in V} P_{\mathcal{B}}(x \mid \pi(x)).$$

As with classical DLs, the main building blocks in $\mathcal{BEL}$ are concepts, which are syntactically built as $\mathcal{EL}$ concepts. The domain knowledge is encoded by a generalization of TBoxes, where GCIs are annotated with a context, defined by a set of literals belonging to a BN.

Definition 7 (KB). Let V be a finite set of Boolean variables. A V -literal is an expression of the form x or $\neg x$, where $x \in V$; a V -context is a consistent set of V -literals.

A V -restricted general concept inclusion (V -GCI) is of the form $\langle C \sqsubseteq D : \kappa \rangle$ where C and D are $\mathcal{BEL}$ concepts and κ is a V -context. A V -TBox is a finite set of V -GCIs. A $\mathcal{BEL}$ knowledge base (KB) over V is a pair $\mathcal{K} = (\mathcal{B}, \mathcal{T})$ where $\mathcal{B}$ is a BN over V and $\mathcal{T}$ is a V -TBox.

The semantics of $\mathcal{BEL}$ extends the semantics of $\mathcal{EL}$ by additionally evaluating the random variables from the BN. Given a finite set of Boolean variables V , a V -interpretation is a tuple $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \mathcal{V}^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set called the *domain*, $\mathcal{V}^{\mathcal{I}} : V \rightarrow \{0, 1\}$ is a *valuation* of the variables in V , and $\cdot^{\mathcal{I}}$ is an *interpretation function* that maps every concept name A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.³

The interpretation function $\cdot^{\mathcal{I}}$ is extended to arbitrary $\mathcal{BEL}$ concepts as in $\mathcal{EL}$ and the valuation $\mathcal{V}^{\mathcal{I}}$ is extended to contexts by defining, for every $x \in V$, $\mathcal{V}^{\mathcal{I}}(\neg x) = 1 - \mathcal{V}^{\mathcal{I}}(x)$, and for every context κ , $\mathcal{V}^{\mathcal{I}}(\kappa) = \min_{\ell \in \kappa} \mathcal{V}^{\mathcal{I}}(\ell)$, where $\mathcal{V}^{\mathcal{I}}(\emptyset) := 1$. Intuitively, a context κ can be thought as a conjunction of literals, which is evaluated to 1 iff each literal in the context is evaluated to 1.

The V -interpretation $\mathcal{I}$ is a *model* of the V -GCI $\langle C \sqsubseteq D : \kappa \rangle$, denoted as $\mathcal{I} \models \langle C \sqsubseteq D : \kappa \rangle$, iff (i) $\mathcal{V}^{\mathcal{I}}(\kappa) = 0$, or (ii) $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. It is a *model* of the V -TBox $\mathcal{T}$ iff it is a model of all the V -GCIs in $\mathcal{T}$. The idea is that the restriction $C \sqsubseteq D$ is only required to hold whenever the context κ is satisfied. Thus, any interpretation that violates the context trivially satisfies the V -GCI.

Example 8. Let $V_0 = \{x, y, z\}$, and consider the V_0 -TBox

$$\mathcal{T}_0 := \{ \langle A \sqsubseteq C : \{x, y\} \rangle, \langle A \sqsubseteq B : \{\neg x\} \rangle, \langle B \sqsubseteq C : \{\neg x\} \rangle \}.$$

The interpretation $\mathcal{I}_0 = (\{d\}, \cdot^{\mathcal{I}_0}, \mathcal{V}_0)$ given by $\mathcal{V}_0(\{x, \neg y, z\}) = 1$, $A^{\mathcal{I}_0} = \{d\}$, and $B^{\mathcal{I}_0} = C^{\mathcal{I}_0} = \emptyset$ is a model of $\mathcal{T}_0$, but is not a model of the V -GCI $\langle A \sqsubseteq B : \{x\} \rangle$, since $\mathcal{V}_0(\{x\}) = 1$ but $A^{\mathcal{I}_0} \not\subseteq B^{\mathcal{I}_0}$.

A V -TBox $\mathcal{T}$ is in *normal form* if for each V -GCI $\langle \alpha : \kappa \rangle \in \mathcal{T}$, α is an $\mathcal{EL}$ GCI in normal form. A $\mathcal{BEL}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{B})$ is in *normal form* if $\mathcal{T}$ is in normal form. As for $\mathcal{EL}$, every $\mathcal{BEL}$ KB can be transformed into an equivalent one in normal form in polynomial time (Ceylan 2013). Thus, we consider only $\mathcal{BEL}$ KBs in normal form in the following.

³When there is no danger of ambiguity, we will usually drop the prefix V and speak simply of e.g. a *TBox*, a *KB* or an *interpretation*.

The DL $\mathcal{EL}$ is a special case of $\mathcal{BEL}$ in which all V -GCIs are of the form $\langle C \sqsubseteq D : \emptyset \rangle$. Notice that every valuation satisfies the empty context $\emptyset$; thus, a V -interpretation $\mathcal{I}$ satisfies the V -GCI $\langle C \sqsubseteq D : \emptyset \rangle$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We say that $\mathcal{T}$ entails $\langle C \sqsubseteq D : \emptyset \rangle$, denoted by $\mathcal{T} \models C \sqsubseteq D$, if every model of $\mathcal{T}$ is also a model of $\langle C \sqsubseteq D : \emptyset \rangle$. For a valuation $\mathcal{W}$ of the variables in V , we can define a TBox containing all axioms that must be satisfied in any V -interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \mathcal{V}^{\mathcal{I}})$ with $\mathcal{V}^{\mathcal{I}} = \mathcal{W}$.

Definition 9 (restriction). Let $\mathcal{K} = (\mathcal{B}, \mathcal{T})$ be a KB. The restriction of $\mathcal{T}$ to a valuation $\mathcal{W}$ of the variables in V is

$$\mathcal{T}_{\mathcal{W}} := \{ \langle C \sqsubseteq D : \emptyset \rangle \mid \langle C \sqsubseteq D : \kappa \rangle \in \mathcal{T}, \mathcal{W}(\kappa) = 1 \}.$$

To handle the probabilistic knowledge provided by the BN, we extend the semantics of $\mathcal{BEL}$ through multiple-world interpretations. Intuitively, a V -interpretation describes a possible world; by assigning a probabilistic distribution over these interpretations, we describe the required probabilities, which should be consistent with the BN provided in the knowledge base.

Definition 10 (probabilistic model). A probabilistic interpretation is a pair $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$, where $\mathcal{J}$ is a set of V -interpretations and $P_{\mathcal{J}}$ is a probability distribution over $\mathcal{J}$ such that $P_{\mathcal{J}}(\mathcal{I}) > 0$ only for finitely many interpretations $\mathcal{I} \in \mathcal{J}$. This probabilistic interpretation is a model of the TBox $\mathcal{T}$ if every $\mathcal{I} \in \mathcal{J}$ is a model of $\mathcal{T}$. $\mathcal{P}$ is consistent with the BN $\mathcal{B}$ if for every possible valuation $\mathcal{W}$ of the variables in V it holds that

$$\sum_{\mathcal{I} \in \mathcal{J}, \mathcal{V}^{\mathcal{I}} = \mathcal{W}} P_{\mathcal{J}}(\mathcal{I}) = P_{\mathcal{B}}(\mathcal{W}).$$

It is a model of the KB $(\mathcal{B}, \mathcal{T})$ iff it is a (probabilistic) model of $\mathcal{T}$ and consistent with $\mathcal{B}$.

One simple consequence of this semantics is that probabilistic models preserve the probability distribution of $\mathcal{B}$ for contexts; the probability of a context κ is the sum of the probabilities of all valuations that extend κ .

Just as in classical DLs, we want to extract the information that is implicitly encoded in a $\mathcal{BEL}$ KB. In particular, we are interested in solving different reasoning tasks for this logic. One of the fundamental reasoning problems in $\mathcal{EL}$ is subsumption: is a concept C always interpreted as a sub-concept of D ? In the case of $\mathcal{BEL}$, we are also interested in finding the probability with which such a subsumption relation holds. For the rest of this section, we formally define this reasoning task, and provide a method for solving it, by reducing it to a decision problem in Bayesian networks.

3.1 Probabilistic Subsumption

Subsumption is one of the most basic decision problems in $\mathcal{EL}$. In $\mathcal{BEL}$, we generalize this problem to consider also the contexts and probabilities provided by the BN.

Definition 11 (p -subsumption). Let C, D be two $\mathcal{BEL}$ concepts, κ a context, and $\mathcal{K}$ a $\mathcal{BEL}$ KB. For a probabilistic interpretation $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$, we define $P(\langle C \sqsubseteq_{\mathcal{P}} D : \kappa \rangle) := \sum_{\mathcal{I} \in \mathcal{J}, \mathcal{I} \models \langle C \sqsubseteq D : \kappa \rangle} P_{\mathcal{J}}(\mathcal{I})$. The probability of $\langle C \sqsubseteq D : \kappa \rangle$ w.r.t. $\mathcal{K}$ is defined as

$$P(\langle C \sqsubseteq_{\mathcal{K}} D : \kappa \rangle) := \inf_{\mathcal{P} \models \mathcal{K}} P(\langle C \sqsubseteq_{\mathcal{P}} D : \kappa \rangle).$$

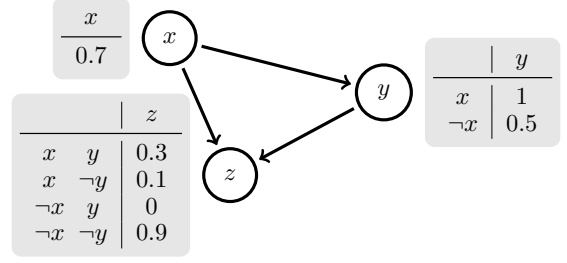


Figure 2: A simple BN

We say that C is p -subsumed by D in κ , for $p \in (0, 1]$ if $P(\langle C \sqsubseteq_{\mathcal{K}} D : \kappa \rangle) \geq p$.

The following proposition was shown in (Ceylan and Peñaloza 2014).

Proposition 12. Let $\mathcal{K} = (\mathcal{B}, \mathcal{T})$ be a KB. Then

$$P(\langle C \sqsubseteq_{\mathcal{K}} D : \kappa \rangle) = 1 - P_{\mathcal{B}}(\kappa) + \sum_{\substack{\mathcal{T}_{\mathcal{W}} \models \langle C \sqsubseteq D \\ \mathcal{W}(\kappa) = 1}} P_{\mathcal{B}}(\mathcal{W}).$$

Example 13. Consider the KB $\mathcal{K}_0 = (\mathcal{B}_0, \mathcal{T}_0)$, where $\mathcal{B}_0$ is the BN from Figure 2 and $\mathcal{T}_0$ the TBox from Example 8. It follows that $P(\langle A \sqsubseteq_{\mathcal{K}_0} C : \{x, y\} \rangle) = 1$ from the first GCI in $\mathcal{T}$ and $P(\langle A \sqsubseteq_{\mathcal{K}_0} C : \{\neg x\} \rangle) = 1$ from the others since any model of $\mathcal{K}_0$ needs to satisfy the GCIs asserted in $\mathcal{T}$ by definition. Notice that $A \sqsubseteq C$ does not hold in context $\{x, \neg y\}$, but $P(\langle A \sqsubseteq_{\mathcal{K}_0} C : \{x, \neg y\} \rangle) = 1$. Since this describes all contexts, we conclude $P(\langle A \sqsubseteq_{\mathcal{K}_0} C : \emptyset \rangle) = 1$.

3.2 Deciding p -subsumption

We now show that deciding p -subsumption can be reduced to exact inference in Bayesian networks. This latter problem is known to be PP-complete (Roth 1996). Let $\mathcal{K} = (\mathcal{T}, \mathcal{B})$ be an arbitrary but fixed $\mathcal{BEL}$ KB. From the V -TBox $\mathcal{T}$, we construct the classical $\mathcal{EL}$ TBox $\mathcal{T}' := \{ \alpha \mid \langle \alpha : \kappa \rangle \in \mathcal{T} \}$; that is, $\mathcal{T}'$ contains the same axioms as $\mathcal{T}$, but ignores the contextual information encoded in their labels. Let now $H_{\mathcal{T}'}^u$ be the (pruned) unraveled proof structure for this TBox $\mathcal{T}'$. By construction, $H_{\mathcal{T}'}^u$ is a directed acyclic hypergraph. Our goal is to transform this hypergraph into a DAG. Using this DAG, we will construct a BN, from which all the p -subsumption relations can be read through standard BN inferences. We explain this construction in two steps.

From Hypergraph to DAG Hypergraphs generalize graphs by allowing several vertices to be connected by a single edge. Intuitively, the hyperedges in a hypergraph encode a formula in disjunctive normal form. Indeed, an edge (S, v) expresses that if all the elements in S can be reached, then v is also reachable; this can be seen as an implication: $\bigwedge_{w \in S} w \Rightarrow v$. Suppose that there exist several edges sharing the same head $(S_1, v), (S_2, v), \dots, (S_k, v)$ in the hypergraph. This situation can be described through the implication $\bigvee_{i=1}^k (\bigwedge_{w \in S_i} w) \Rightarrow v$. We can thus rewrite any directed acyclic hypergraph into a DAG by introducing auxiliary conjunctive and disjunctive nodes (see the upper part of Fig-

Algorithm 2 Construction of a DAG from a hypergraph**Input:** $H = (V, E)$ directed acyclic hypergraph**Output:** $G = (V', E')$ directed acyclic graph

```

1:  $V' \leftarrow V, i, j \leftarrow 0$ 
2: for each  $v \in V$  do
3:    $\mathcal{S} \leftarrow \{S \mid (S, v) \in E\}, j \leftarrow i$ 
4:   for each  $S \in \mathcal{S}$  do
5:      $V' \leftarrow V' \cup \{\wedge_i\}, E' \leftarrow E' \cup \{(u, \wedge_i) \mid u \in S\}$ 
6:     if  $i > j$  then
7:        $V' \leftarrow V' \cup \{\vee_i\}, E' \leftarrow E' \cup \{(\wedge_i, \vee_i)\}$ 
8:      $i \leftarrow i + 1$ 
9:   if  $i = j + 1$  then
10:     $E' \leftarrow E' \cup \{(\wedge_j, v)\}$ 
11:   else
12:     $E' \leftarrow E' \cup \{(\vee_k, \vee_{k+1}) \mid j < k < i - 1\} \cup$ 
13:     $\{(\vee_{i-1}, v), (\wedge_j, \vee_{j+1})\}$ 
14: return  $G = (V', E')$ 

```

ure 3); the proper semantics of these nodes will be guaranteed by the conditional probability distribution defined later. Since the space needed for describing the conditional probability tables in a BN is exponential on the number of parents that a node has, we ensure that these auxiliary nodes, as well as the elements in $W_{\mathcal{T}}$ have at most two parent nodes.

Algorithm 2 describes the construction of such a DAG from a directed hypergraph. Essentially, the algorithm adds a new node $\wedge_i$ for each hyperedge (S, v) in the input hypergraph H , and connects it with all the nodes in S . Additionally, if there are k hyperedges that lead to a single node v , it creates $k-1$ nodes $\vee_i$. These are used to represent the binary disjunctions among all the hyperedges leading to v . Clearly, the algorithm runs in polynomial time on the size of H , and if H is acyclic, then the resulting graph G is acyclic too. Moreover, all the nodes $v \in V$ that existed in the input hypergraph will have at most one parent node after the translation; every $\vee_i$ node has exactly two parents, and the number of parents of a node $\wedge_i$ is given by the set S from the hyperedge $(S, v) \in E$ that generated it. In particular, if the input hypergraph is the unraveled proof structure for a TBox $\mathcal{T}$, then the size of the generated graph G is polynomial on the size of $\mathcal{T}$, and each node has at most two parent nodes.

From DAG to Bayesian Network The next step is to build a BN that preserves the probabilistic entailments of a $\mathcal{BEL}$ KB. Let $\mathcal{K} = (\mathcal{T}, \mathcal{B})$ be such a KB, with $\mathcal{B} = (G, \Phi)$, and let $G_{\mathcal{T}}$ be the DAG obtained from the unraveled proof structure of $\mathcal{T}$ using Algorithm 2. Recall that the nodes of $G_{\mathcal{T}}$ are either (i) pairs of the form (α, i) , where α is a GCI in normal form built from the signature of $\mathcal{T}$, or (ii) an auxiliary disjunction ($\vee_i$) or conjunction ($\wedge_i$) node introduced by Algorithm 2. Moreover, $(\alpha, 0)$ is a node of $G_{\mathcal{T}}$ iff there is a context κ with $\langle \alpha : \kappa \rangle \in \mathcal{T}$. We assume w.l.o.g. that for node $(\alpha, 0)$ there is *exactly one* such context. If there were more than one, then we could extend the BN $\mathcal{B}$ with an additional variable which describes the disjunctions of these contexts, similarly to the construction of Algorithm 2. Similarly, we assume that $|\kappa| \leq 2$, to ensure that 0-level nodes

have at most two parent nodes. This restriction can be easily removed by introducing conjunction nodes as before. For a context κ , let $\text{var}(\kappa)$ denote the set of all variables appearing in κ . We construct a new BN $\mathcal{B}_{\mathcal{K}}$ as follows.

Let $G = (V, E)$ and $G_{\mathcal{T}} = (V_{\mathcal{T}}, E_{\mathcal{T}})$. The DAG $G_{\mathcal{K}}$ is given by $G_{\mathcal{K}} = (V_{\mathcal{K}}, E_{\mathcal{K}})$, where $V_{\mathcal{K}} := V \cup V_{\mathcal{T}}$ and

$$E_{\mathcal{K}} := E \cup E_{\mathcal{T}} \cup \{(x, (\alpha, 0)) \mid \langle \alpha : \kappa \rangle \in \mathcal{T}, x \in \text{var}(\kappa)\}.$$

Clearly, $G_{\mathcal{K}}$ is a DAG. We now need only to define the conditional probability tables for the nodes in $V_{\mathcal{T}}$ given their parents in $G_{\mathcal{K}}$; notice that the structure of the graph G remains unchanged for the construction of $G_{\mathcal{K}}$. For every node $(\alpha, 0) \in V_{\mathcal{T}}$, there is a κ such that $\langle \alpha : \kappa \rangle \in \mathcal{T}$; the parents of $(\alpha, 0)$ in $G_{\mathcal{K}}$ are then $\text{var}(\kappa) \subseteq V$. The conditional probability of $(\alpha, 0)$ given its parents is given by: $P_{\mathcal{B}}((\alpha, 0) = \text{true} \mid \mathcal{V}(\text{var}(\kappa))) = \mathcal{V}(\kappa)$; that is, the probability of $(\alpha, 0)$ being true given a valuation of its parents is 1 if the valuation makes the context κ true; otherwise, it is 0. Each auxiliary node has at most two parents, and the conditional probability of a conjunction node $\wedge_i$ being true is 1 iff all parents are true; and the conditional probability of a disjunction node $\vee_i$ being true is 1 iff at least one parent is true; Finally, every (α, i) with $i > 0$ has exactly one parent node v ; (α, i) is true with probability 1 iff v is true.

Example 14. Consider the $\mathcal{BEL}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{B}_0)$, where

$$\mathcal{T} = \{ \langle A \sqsubseteq B : \{x\} \rangle, \langle B \sqsubseteq C : \{\neg x, y\} \rangle, \\ \langle C \sqsubseteq D : \{z\} \rangle, \langle B \sqsubseteq D : \{y\} \rangle \}.$$

The BN obtained from this KB is depicted in Figure 3. The upper part of the figure represents the DAG obtained from the unraveled proof structure of $\mathcal{T}$, while the lower part shows the original BN $\mathcal{B}_0$. The gray arrows depict the connection between these two DAGs, which is given by the labels in the V -GCIs in $\mathcal{T}$. The gray boxes denote the conditional probability of the different nodes given their parents.

Suppose that we are interested in $P(\langle A \sqsubseteq_{\mathcal{K}} D : \emptyset \rangle)$. From the unraveled proof structure, we can see that $A \sqsubseteq D$ can be deduced either using the GCIs $A \sqsubseteq B, B \sqsubseteq C, C \sqsubseteq D$, or through the two GCIs $A \sqsubseteq B, B \sqsubseteq D$. The probability of any of these combinations of GCIs to appear is given by $\mathcal{B}_0$ and the contextual connection to the axioms at the lower level of the proof structure. Thus, to deduce $P(\langle A \sqsubseteq_{\mathcal{K}} D : \emptyset \rangle)$ we need only to compute the probability of the node $(A \sqsubseteq D, n)$, where n is the last level.

From the properties of proof structures and Theorem 4 we have that

$$\begin{aligned} P_{\mathcal{B}_{\mathcal{K}}}((\alpha, n) \mid \kappa) &= \sum_{\mathcal{V}(\kappa)=1} P_{\mathcal{B}_{\mathcal{K}}}((\alpha, n) \mid \mathcal{V}(\kappa)) P_{\mathcal{B}_{\mathcal{K}}}(\mathcal{V}(\kappa)) \\ &= \sum_{\substack{\mathcal{T}_{\mathcal{W}} \models \alpha \\ \mathcal{W}(\kappa)=1}} P_{\mathcal{B}_{\mathcal{K}}}(\mathcal{W}), \end{aligned}$$

which yields the following result.

Theorem 15. Let $\mathcal{K} = (\mathcal{T}, \mathcal{B})$ be a $\mathcal{BEL}$ KB, where $\mathcal{B}$ is over V , and $n = |\text{sig}(\mathcal{T})|^3$. For a V -GCI $\langle \alpha : \kappa \rangle$, it holds that $P(\langle \alpha : \kappa \rangle) = 1 - P_{\mathcal{B}}(\kappa) + P_{\mathcal{B}_{\mathcal{K}}}((\alpha, n) \mid \kappa)$.

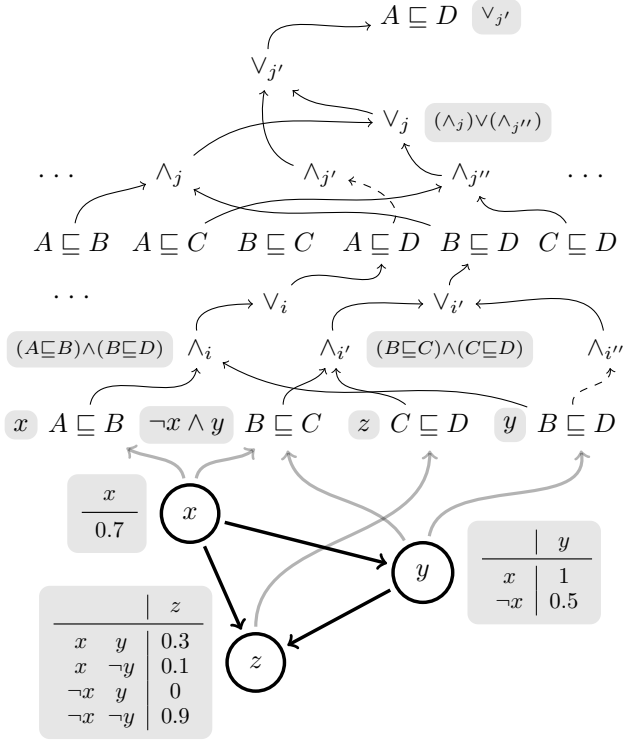


Figure 3: A portion of the constructed BN

This theorem states that we can reduce the problem of p -subsumption w.r.t. the $\mathcal{BEL}$ KB $\mathcal{K}$ to a probabilistic inference in the BN $\mathcal{B}_{\mathcal{K}}$. Notice that the size of $\mathcal{B}_{\mathcal{K}}$ is polynomial on the size of $\mathcal{K}$. This means that p -subsumption is at most as hard as deciding the probability of query variables, given an evidence, which is known to be in PP (Roth 1996). Since p -subsumption is already PP-hard (Ceylan and Peñaloza 2014), we obtain the following result.

Corollary 16. *Deciding p -subsumption w.r.t. a $\mathcal{BEL}$ KB $\mathcal{K}$ is PP-complete on the size of $\mathcal{K}$.*

4 Related Work

An early attempt for combining BNs and DLs was P-CLASSIC (Koller, Levy, and Pfeffer 1997), which extends CLASSIC through probability distributions over the interpretation domain. In the same line, in PR-OWL (da Costa, Laskey, and Laskey 2008) the probabilistic component is interpreted by providing individuals with a probability distribution. As many others in the literature (see (Lukasiewicz and Straccia 2008) for a thorough survey on probabilistic DLs) these approaches differ from our multiple-world semantics, in which we consider a probability distribution over a set of classical DL interpretations.

DISPONTE (Riguzzi et al. 2012) is one representative of the approaches that consider a multiple-world semantics. The main difference with our approach is that in DISPONTE, the authors assume that all probabilities are independent, while we provide a joint probability distribution through the BN. Another minor difference is that $\mathcal{BEL}$ al-

lows for classical consequences whereas DISPONTE does not. Closest to our approach is perhaps the Bayesian extension of DL-Lite called BDL-Lite (d’Amato, Fanizzi, and Lukasiewicz 2008). Abstracting from the different logical component, BDL-Lite looks almost identical to ours. There is, however, a subtle but important difference. In our approach, an interpretation $\mathcal{I}$ satisfies a V -GCI $\langle C \sqsubseteq D : \kappa \rangle$ if $\mathcal{V}^{\mathcal{I}}(\kappa) = 1$ implies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. In (d’Amato, Fanizzi, and Lukasiewicz 2008), the authors employ a closed-world assumption over the contexts, where this implication is substituted for an equivalence; i.e., $\mathcal{V}^{\mathcal{I}}(\kappa) = 0$ also implies $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$. The use of such semantics can easily produce inconsistent KBs, which is impossible in $\mathcal{BEL}$.

Other probabilistic extensions of $\mathcal{EL}$ are (Lutz and Schröder 2010) and (Niepert, Noessner, and Stuckenschmidt 2011). The former introduces probabilities as a concept constructor, while in the latter the probabilities of axioms, which are always assumed to be independent, are implicitly encoded through a weighting function, which is interpreted with a log-linear model. Thus, both formalisms differ greatly from our approach.

5 Conclusions

We have described the probabilistic DL $\mathcal{BEL}$, which extends the light-weight DL $\mathcal{EL}$ with uncertain contexts. We have shown that it is possible to construct, from a given $\mathcal{BEL}$ KB $\mathcal{K}$, a BN $\mathcal{B}_{\mathcal{K}}$ that encodes all the probabilistic and logical knowledge of $\mathcal{K}$ w.r.t. to the signature of the KB. Moreover, the size of $\mathcal{B}_{\mathcal{K}}$ is polynomial on the size of $\mathcal{K}$. We obtain that probabilistic reasoning over $\mathcal{K}$ is at most as hard as deciding inferences in $\mathcal{B}_{\mathcal{K}}$, which yields a tight complexity bound for deciding p -subsumption in this logic.

While the construction is polynomial on the input KB, the obtained DAG might not preserve all the desired properties of the original BN. For instance, it is known that the efficiency of the BN inference engines depends on the treewidth of the underlying DAG (Pan, McMichael, and Lendjel 1998); however, the proof structure used by our construction may increase the treewidth of the graph. One direction of future research will be to try to optimize the reduction by bounding the treewidth and reducing the amount of nodes added to the graph.

Clearly, once we have constructed the associated BN $\mathcal{B}_{\mathcal{K}}$ from a given $\mathcal{BEL}$ KB $\mathcal{K}$, this can be used for additional inferences, beyond deciding subsumption. We think that reasoning tasks such as *contextual subsumption* and finding the *most likely context*, defined in (Ceylan and Peñaloza 2014) can be solved analogously. Studying this and other reasoning problems is also a task of future work.

Finally, our construction does not depend on the chosen DL $\mathcal{EL}$, but rather on the fact that a simple polynomial-time consequence-based method can be used to reason with it. It should thus be a simple task to generalize the approach to other consequence-based methods, e.g. (Simancik, Kazakov, and Horrocks 2011). It would also be interesting to generalize the probabilistic component to consider other kinds of probabilistic graphical models (Koller and Friedman 2009).

References

- Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2nd edition.
- Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proc. IJCAI-05*. Morgan-Kaufmann.
- Brandt, S. 2004. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In *Proc. ECAI-2004*, 298–302. IOS Press.
- Ceylan, İ. İ., and Peñaloza, R. 2014. The Bayesian Description Logic $\mathcal{BEL}$. In *Proc. IJCAR 2014*. To appear.
- Ceylan, İ. İ. 2013. Context-sensitive bayesian description logics. Master’s thesis, Dresden University of Technology, Germany.
- da Costa, P. C. G.; Laskey, K. B.; and Laskey, K. J. 2008. Prowl: A bayesian ontology language for the semantic web. In *Uncertainty Reasoning for the Semantic Web I, URSW 2005-2007*, volume 5327 of *LNCS*, 88–107. Springer.
- d’Amato, C.; Fanizzi, N.; and Lukasiewicz, T. 2008. Tractable reasoning with bayesian description logics. In *Proc. Second International Conference on Scalable Uncertainty Management (SUM 2008)*, volume 5291 of *LNCS*, 146–159. Springer.
- Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.
- Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press.
- Koller, D.; Levy, A. Y.; and Pfeffer, A. 1997. P-classic: A tractable probabilistic description logic. In *Proc. 14th National Conference on Artificial Intelligence (AAAI-97)*, 390–397. AAAI Press.
- Lukasiewicz, T., and Straccia, U. 2008. Managing uncertainty and vagueness in description logics for the semantic web. *J. of Web Semantics* 6(4):291–308.
- Lutz, C., and Schröder, L. 2010. Probabilistic description logics for subjective uncertainty. In Lin, F.; Sattler, U.; and Truszczyński, M., eds., *KR*. AAAI Press.
- Niepert, M.; Noessner, J.; and Stuckenschmidt, H. 2011. Log-linear description logics. In Walsh, T., ed., *IJCAI*, 2153–2158. IJCAI/AAAI.
- Pan, H.; McMichael, D.; and Lendjel, M. 1998. Inference algorithms in bayesian networks and the probanet system. *Digital Signal Processing* 8(4):231 – 243.
- Riguzzi, F.; Bellodi, E.; Lamma, E.; and Zese, R. 2012. Epistemic and statistical probabilistic ontologies. In *Proc. 8th Int. Workshop on Uncertainty Reasoning for the Semantic Web (URSW-12)*, volume 900, 3–14. CEUR-WS.
- Roth, D. 1996. On the hardness of approximate reasoning. *Artif. Intel.* 82(1-2):273–302.
- Simancik, F.; Kazakov, Y.; and Horrocks, I. 2011. Consequence-based reasoning beyond horn ontologies. In *Proc. IJCAI-11*, 1093–1098. IJCAI/AAAI.

Parameter Estimation for Relational Kalman Filtering

Jaesik Choi

School of Electrical and Computer Engineering
Ulsan National Institute of Science and Technology
Ulsan, Korea

Eyal Amir

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL, USA

Tianfang Xu and Albert J. Valocchi

Department of Civil and Environmental Engineering
University of Illinois at Urbana-Champaign
Urbana, IL, USA

Abstract

The Kalman Filter (KF) is pervasively used to control a vast array of consumer, health and defense products. By grouping sets of symmetric state variables, the Relational Kalman Filter (RKF) enables to scale the exact KF for large-scale dynamic systems. In this paper, we provide a parameter learning algorithm for RKF, and a regrouping algorithm that prevents the degeneration of the relational structure for efficient filtering. The proposed algorithms significantly expand the applicability of the RKFs by solving the following questions: (1) how to learn parameters for RKF in partial observations; and (2) how to regroup the degenerated state variables by noisy real-world observations. We show that our new algorithms improve the efficiency of filtering the large-scale dynamic system.

1 Introduction

Many real-world systems can be modeled by continuous variables and relationships (or dependences) among them. The Kalman Filter (KF) (Kalman 1960) accurately estimates the state of variables in a linear dynamic system with Gaussian noise given a sequence of control-inputs and observations. The KF has been applied in a broad range of domains such as robotics, finance (Bahmani-Oskooee and Brown 2004) and environmental science (F.P. and Bierkens 2001; Clark et al. 2008). Given a sequence of observations and linear dependences with Gaussian noise between variables, the KF calculates the conditional probability density of the state variables at each time step.

Unfortunately, the KF computations are cubic in the number of state variables, which limits the use of existing exact methods to domains with a large number of state variables. This has led to the combination of approximation and sampling in the Ensemble Kalman Filter (Evensen 1994), and recently to the Relational Kalman Filters (RKFs) over grouped state variables (Choi, Guzman-Rivera, and Amir 2011; Ahmadi, Kersting, and Sanner 2011). The RKFs

leverage the ability of relational languages to specify models with size of representation independent of the size of populations involved (Friedman et al. 1999; Poole 2003; Richardson and Domingos 2006).

Lifted inference algorithms for relational continuous models (Wang and Domingos 2008; Choi, Hill, and Amir 2010; Ahmadi, Kersting, and Sanner 2011) degenerate (or split) relational structures upon individual observations. Lifted RKF (Choi, Guzman-Rivera, and Amir 2011) maintains relational structure when the same number of observations are made. Otherwise, it also degenerates (possibly rapidly) the relational structure, thus lifted RKF may not be useful with sparse observations.

The main contributions of this paper are (1) to regroup the degenerated state variables from noisy real-world observations with a tight error bounds; and (2) to learn parameters for RKFs.

We propose a new learning algorithm for RKFs. We show that the relational learning expedites filtering, and achieves accurate filtering in theory and practice. The key intuition is that the Maximum Likelihood Estimate (MLE) of RKF parameters is the empirical mean and variance over state variables included in each group. For partial observations, the parameters can be calculated similarly. We show that under reasonable conditions variances of degenerated state variables on partial observations converge exponentially. Thus, our approximate regrouping algorithm has bounded errors compared to the exact KF. We could show that the RKF with regrouping is more robust against degeneracy than the Lifted RKF in practice with partial observations.

2 Relational Linear Dynamic Systems

In this section, we define relational linear dynamic systems. Dependencies among variables are represented using **relational atoms**, or just **atoms**.¹ The relational atoms are useful when the joint probability of variables involves common types of functions. When representing the joint probability

¹For comprehensive definitions, see (Poole 2003; de Salvo Braz, Amir, and Roth 2005)

distribution, there are products of the parameterized functions (or potentials).

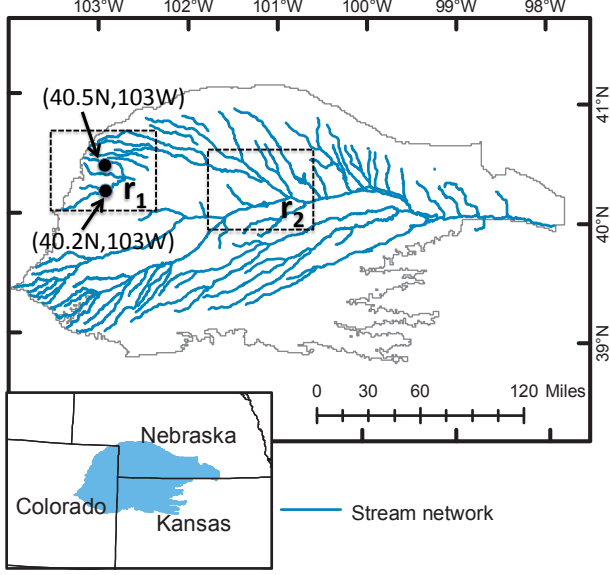


Figure 1: Republican River Basin covering portions of eastern Colorado, northwest Kansas and southwest Nebraska. This figure shows two clustered water wells; $region_1(r_1)$ and $region_2(r_2)$. Water wells in each region have the same (linear Gaussian) relationships with wells in other regions.

Relational atoms represent the set of state variables corresponding to all ground substitutions of its parameter variables. For example, let $X^{r_1}(\text{Latitude}, \text{Longitude})$ be an atom for the (water level of) wells in $region_1$, $\theta = (40.2N, 103W)$. When we substitute *Latitude* and *Longitude* with θ , the atom becomes a state variable $X^{r_1}(40.2N, 103W)$ which represents the level (or prediction) of well head at (*Latitude*=40.2N, *Longitude*=103W). Formally, applying a substitution θ to an atom $\mathbf{X}(L)$ yields a new atom $\mathbf{X}(L\theta)$ where $L\theta$ is obtained by renaming the parameter variables in L according to θ . If θ is a ground substitution, $\mathbf{X}(L\theta)$ is a ground state variable like $X^{r_1}(40.2N, 103W)$.² $|X(L)|$ or just $|X|$ denotes the the number of distinct state variables generated from X by all substitutions.

A **pairwise Gaussian parfactor** $((\mathbf{X}, \mathbf{X}'), \phi)$ is composed of a pair of two atoms $(\mathbf{X}, \mathbf{X}')$ and a linear Gaussian potential ϕ between two atoms in the following form.

$$\phi(\mathbf{X}, \mathbf{X}') \propto \exp \left[-\frac{(\mathbf{X} - \mathbf{X}' - \mu)^2}{\sigma^2} \right]$$

For example, a pairwise Gaussian parfactor $\phi_{r_1, r_2}(X_{r_1}, X_{r_2})$ represents the linear Gaussian relationship between two ground variables chosen from $region_1$ and $region_2$ respectively.

²Here, we assume that the ground state variables are univariate, e.g., domain of x is $\mathbb{R}$. However, it is straightforward to extend this model to the one with multivariate ground variables.

A **pairwise Gaussian factor**, or just a **factor**, $f = ((\mathbf{x}, \mathbf{x}'), \phi)$ is a pair where ϕ is a potential function on $(\mathbf{x}, \mathbf{x}')$ from $\mathbb{R}^2$ to $\mathbb{R}^+$ where $(\mathbf{x}, \mathbf{x}')$ is a pair of ground random variables derived by ground substitutions from $(\mathbf{X}(L\theta), \mathbf{X}'(L'\theta'))$. A factor f defines a weighting function on a **valuation** $(x, x') = (v, v')$: $w_f(x, x') = \phi(v, v')$. The weighting function for a *parfactor* g is the product of the weighting functions over all of its ground substitutions (factors), $w_g(v) = \prod_{f \in g} w_f(v)$. Hence, a set of parfactors G defines a probability density,

$$w_G(v) = \frac{1}{Z} \prod_{g \in G} \prod_{f \in g} w_f(v),$$

where Z is the normalizing constant.³ In this way, we can represent the joint probability of all random variables (e.g. all wells in $region_1$ and $region_2$).

Relational Transition Models (RTMs) characterize the dependence of relational atoms between time steps. $X_t^i(a)$ and $X_{t+1}^j(a')$ are relational atoms at time step t and $t+1$ respectively when a and a' are ground substitutions, e.g., $\theta = (40.2N, 98W)$. $U_t^i(a)$ is the control-input information. A RTM takes the following form,

$$X_{t+1}^j(a') = B_X^{ij} X_t^i(a) + B_U^{ij} U_t^i(a) + G_{RTM}^{ij}, \quad (1)$$

where $G_{RTM}^{ij} \sim \mathcal{N}(0, \sigma_{RTM}^{ij})$ and $\mathcal{N}(m, \sigma^2)$ is the normal distribution with mean m and variance σ^2 . B_X^{ij} and B_U^{ij} are the linear transition coefficients.

In the linear Gaussian representation, the transition models take the following form,

$$\begin{aligned} &\phi_{RTM}(X_{t+1}^j(a') | X_t^i(a), U_t^i(a)) \\ &\propto \exp \left[-\frac{(X_{t+1}^j(a') - B_X^{ij} X_t^i(a) - B_U^{ij} U_t^i(a))^2}{2\sigma_{RTM}^{ij}{}^2} \right]. \end{aligned} \quad (2)$$

The most common transition is the one from the state $X_t^i(a)$ to the state itself $X_{t+1}^i(a)$ at the next time step,

$$X_{t+1}^i(a) = B_X^i X_t^i(a) + B_U^i U_t^i(a) + G_{RTM}^i. \quad (3)$$

Relational Observation Models (ROMs) represent the relationships between the hidden (state) variables, $X_t^i(a)$, and the observations made directly on the related variable, $O_t^i(a)$ (*direct observations*),

$$O_t^i(a) = H_t^i X_t^i(a) + G_{ROM}^i, G_{ROM}^i \sim \mathcal{N}(0, \sigma_{ROM}^i{}^2) \quad (4)$$

H_t^i is the linear coefficient from $X_t^i(a)$ to $O_t^i(a)$.

ROMs also represent the relationships between the hidden variables $X_t^i(a)$ and the observations made indirectly on a variable in the atom $O_t^i(a')$ where $a \neq a'$ (*relational observations*),

$$O_t^i(a') = H_t^i X_t^i(a) + G_{ROM}^i, G_{ROM}^i \sim \mathcal{N}(0, \sigma_{ROM}^i{}^2) \quad (5)$$

³The condition of being a probability density is that at least a random variable has a prior distribution, see (Choi, Hill, and Amir 2010).

in most cases, it is reasonable to set the variance of direct observation $\text{var}(G_{ROM}^i)$ is smaller than the variance of relational one $\text{var}(G_{ROM}^{ij})$ s.t. $\sigma_{ROM}^i \ll \sigma_{ROM}^{ij}$. For the well example, $X_t^{r1}(40.2N, 103W)$, an observation made at the exact location $O_t^{r1}(40.2N, 103W)$ will have the smaller variance (more certain) than an observation made at a nearby location $O_t^{r1}(40.5N, 103W)$.

In the linear Gaussian representation, ROMs take the following form,

$$\phi_{ROM}(O_t^i(a)|X_t^i(a)) \propto \exp \left[-\frac{(O_t^i(a) - H_t^i X_t^i(a))^2}{2\sigma_{ROM}^i} \right].$$

Relational Pairwise Models (RPMs) represent linear dependences between pairs of relational atoms,

$$X_t^i(a) = R_t^{ij} X_t^j(a') + G_{RPM}^{ij}, G_{RPM}^{ij} \sim \mathcal{N}(0, \sigma_{RPM}^{ij}) \quad (6)$$

R_t^{ij} is the coefficient.

Note that RTMs and ROMs represent the nature of dynamic systems (e.g. the state at the next time step depends on the current time step). The product of RPMs is an efficient way to represent the relational structure over atoms, groups of state variables.

Relational Kalman Filter (RKF) is a filtering procedure with a relational linear dynamic system which is composed of RTMs, ROMs and RPMs. That is, the joint probability of state variables are represented by the product of pairwise Gaussian parafactors. **Lifted RKF** computes the posterior of the state variables given a prior (current) belief and full or partial observations. The input to the problem is: (1) relational parafactors (RTMs, RPMs and ROMs); (2) current belief over atoms (X_0^i); (3) a sequence of control-inputs ($U_1^i, \dots, U_T^i$); and (4) a sequence of observations ($O_1^i, \dots, O_T^i$). The output is the multivariate Gaussian distribution over the atoms (X_T^i) at each time step T . The filtering problem is solved by algorithms represented (Choi, Guzman-Rivera, and Amir 2011; Ahmadi, Kersting, and Sanner 2011). Here, we turn our focus to the parameter learning problem.

3 Learning Relational Kalman Filter

The two important parameters of the RKF are the transition models and observation models. In this section, we present a learning algorithm that derives the maximum likelihood estimates of RTMs and ROMs. For simplicity, we will present a solution with fully observed model, first. A solution for partial observations can be derived with a slight modification.

3.1 Algorithm LearningRKF

Algorithm *LearningRKF* estimates the parameter of RKF given a sequence of observations such as measurements of water wells for several years. The overall procedure is similar to the one with the parameter learning for the ground KF. Here, the main difference is that the coefficients and covariances of RTMs and ROMs are the block matrices. A subroutine, *BlockAverage*, averages out the diagonal and non-diagonal entries of an input matrix, and then outputs a block

matrix where each block includes the empirical means, variances and covariances in each block. In the following sections, we will show that the block matrix computed by *BlockAverage* is the MLE estimate.

Algorithm 1 LearningRKF

input: a sequence of obs ($O_1, \dots, O_T$)
 $(\mathbf{B}, \Sigma_T, \mathbf{C}, \Sigma_O) \leftarrow (I, I, I, I)$
currentLL $\leftarrow \infty$
repeat
 prevLL $\leftarrow$ currentLL
 $(\mathbf{B}', \Sigma_T', \mathbf{C}', \Sigma_O') \leftarrow \text{LearnGroundTM}(O_t, \mathbf{B}, \Sigma_T, \mathbf{C}, \Sigma_O)$
 $(\mathbf{B}, \Sigma_T, \mathbf{C}, \Sigma_O) \leftarrow \text{BlockAverage}(\mathbf{B}', \Sigma_T', \mathbf{C}', \Sigma_O')$
 currentLL $\leftarrow \sum_t \log P(O_t|X_t, \mathbf{B}, \Sigma_T, \mathbf{C}, \Sigma_O)$
until $|\text{prevLL} - \text{currentLL}| < \epsilon$
output: estimated parameters $(\mathbf{B}, \Sigma_T, \mathbf{C}, \Sigma_O)$

3.2 Learning Transition Models

Here, we derive the parameter of the RTMs: linear coefficient B and Gaussian noise G_{RTM} .

Learning Transition Noise is to calculate the mean and the covariance matrix in the following block forms,

$$\mu_T = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}, \Sigma_T = \begin{bmatrix} \Sigma^{1,1} & \Sigma^{1,2} & \dots & \Sigma^{1,n} \\ \Sigma^{2,1} & \Sigma^{2,2} & \dots & \Sigma^{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma^{n,1} & \Sigma^{n,2} & \dots & \Sigma^{n,n} \end{bmatrix} \quad (7)$$

Where μ_i is a vector of size $|X^i| (=n_i)$; $\Sigma^{i,j}$ is a matrix of size n_i by n_j .

Given a prior, a linear coefficient $\mathbf{B}$ and a sequence of full observations, we derive the estimate X_t at time step t assuming the Gaussian noise in the transition model. The MLE estimation of μ and Σ for the RTM can be derived:

$$(\mu_{Tmax}, \Sigma_{Tmax}) = \arg \max_{\mu_T, \Sigma_T} \sum_{t=2, \dots, T} \log f_{\mathcal{N}}(\vec{X}_t; \mu_T, \Sigma_T)$$

where $\vec{X}_t = X_t - \mathbf{B}X_{t-1}$.

Proposition 1. *Given a RKF with a single atom, the maximum likelihood estimates of the Gaussian transition noise are the empirical mean, variance and covariance as follows,*

$$\mu_{MLE} = \begin{bmatrix} m \\ m \\ \vdots \\ m \end{bmatrix}, \Sigma_{MLE} = \begin{bmatrix} \sigma^2 & \sigma' & \dots & \sigma' \\ \sigma' & \sigma^2 & \dots & \sigma' \\ \vdots & \vdots & \ddots & \vdots \\ \sigma' & \sigma' & \dots & \sigma^2 \end{bmatrix}$$

such that,

$$m = \frac{1}{n\bar{T}} \sum_{t=2}^T \sum_{a \in A} \vec{X}_t(a), \sigma^2 = \frac{1}{n\bar{T}} \sum_{t=2}^T \sum_{a \in A} (\vec{X}_t(a) - m)^2$$

$$\sigma' = \frac{1}{n(n-1)\bar{T}} \sum_{t=2}^T \sum_{\substack{a, a' \in A \\ a \neq a'}} (\vec{X}_t(a) - m) (\vec{X}_t(a') - m).$$

where $n = |\vec{X}_t(A)|$ and $\bar{T} = T-1$.

Proof. The MLEs of the parameters (μ_T, Σ_T) are derived by the partial derivatives of the log likelihood,

$$\begin{aligned}\frac{\partial}{\partial \mu_T} \sum_{t=2}^T \log f_{\mathcal{N}}(\vec{X}_t; \mu_T, \Sigma_T) &= 0, \\ \frac{\partial}{\partial \Sigma_T} \sum_{t=2}^T \log f_{\mathcal{N}}(\vec{X}_t; \mu_T, \Sigma_T) &= 0.\end{aligned}$$

All ground variables generated from the atom $\vec{X}_t$ have the same mean, variance and covariances as shown in Equation (8). Now, we can specify the following linear constraints;

$$m = \frac{1}{\bar{T}} \sum_t \vec{X}_t(a_1) = \dots = \frac{1}{\bar{T}} \sum_t \vec{X}_t(a_n).$$

That is, $m = \frac{1}{n\bar{T}} \sum_t \sum_a \vec{X}_t(a)$.

The covariance matrix of the RTM is also calculated from the empirical covariance matrix. The diagonal entries σ^2 are derived for the variances;

$$\sigma^2 = \frac{1}{\bar{T}} \sum_t (\vec{X}_t(a_1) - m)^2 = \dots = \frac{1}{\bar{T}} \sum_t (\vec{X}_t(a_{n_i}) - m)^2$$

Thus, $\sigma^2 = \frac{1}{n\bar{T}} \sum_t \sum_a (\vec{X}_t(a) - m)^2$. Non-diagonal entries (covariances) are derived similarly with $n(n-1)$ empirical covariances.⁴ $\square$

This result is consistent with the result in non-relational KF because the MLE estimates of the ground KF (μ_T and Σ_T) are known to be the empirical mean and the empirical covariance matrix (Roweis and Ghahramani 1999).

In the general case, when we have multiple atoms, the mean vector and the covariance matrix are block forms as shown in Equation 7. That is, the mean and covariance values are same in each subblock. In case of, two atoms X^i and X^j , the means and covariances are as follows:

$$\mu_T = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \Sigma_T = \begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,1} & \Sigma_{2,2} \end{bmatrix}$$

The MLE parameters of the RTM are derived similarly with empirical means and covariances of subblocks.

Proposition 2. *Given a RKF with multiple atoms, the maximum likelihood estimates of the Gaussian transition noise are the empirical means, variances and covariances,*

$$\mu_i = [m_i, \dots, m_i]^T \text{ s.t. } m_i = \frac{1}{n_i \bar{T}} \sum_{t=2}^T \sum_a \vec{X}_t^i(a),$$

$$\Sigma^{i,i} = \begin{bmatrix} \sigma^2 & \sigma' & \dots & \sigma' \\ \sigma' & \sigma^2 & \dots & \sigma' \\ \vdots & \vdots & \ddots & \vdots \\ \sigma' & \sigma' & \dots & \sigma^2 \end{bmatrix} \Sigma^{i,j} = \begin{bmatrix} \sigma'' & \dots & \sigma'' \\ \vdots & \ddots & \vdots \\ \sigma'' & \dots & \sigma'' \end{bmatrix}$$

⁴One gets the same result when differentiating m , σ^2 and σ' directly from the log-likelihood.

$$\sigma^2 = \frac{1}{n_i \bar{T}} \sum_{t=2}^T \sum_a (\vec{X}_t^i(a) - m_i) (\vec{X}_t^i(a) - m_i)$$

$$\sigma' = \frac{1}{n_i(n_i-1)\bar{T}} \sum_{t=2}^T \sum_{\substack{a, a' \in A \\ (a \neq a')}} (\vec{X}_t^i(a) - m_i) (\vec{X}_t^i(a') - m_i),$$

$$\sigma'' = \frac{1}{n_i n_j \bar{T}} \sum_{t=2}^T \sum_{a \in A, b \in B} (\vec{X}_t^i(a) - m_i) (\vec{X}_t^j(b) - m_j)$$

where $n_i = |\vec{X}_t^i|$ and $\bar{T} = T - 1$.

Proof. The principles used in the proof of Proposition 1 are applied because the $\Sigma^{i,i}$ and $\Sigma^{i,j}$ are block matrices. $\square$

Learning Linear Coefficient is to estimate the linear coefficient $\mathbf{B}$ between X_{t-1} and X_t . In this case, given other parameters, Gaussian noise of RTMs and ROMs, the MLE of $\mathbf{B}$ is derived as follows (Roweis and Ghahramani 1999),

$$\hat{\mathbf{B}} = \left(\sum_{t=2, \dots, T} X_t X_{t-1}^T \right) \left(\sum_{t=1, \dots, T-1} X_t X_t^T \right)^{-1}$$

Here, we call the $\hat{\mathbf{B}}$ linear coefficient of the ground TM, and $\hat{\mathbf{B}}$ is reshaped as a block matrix by averaging the coefficient in each subblock. When $\hat{\mathbf{B}}^{i,j}$ denotes the subblock for the linear transition from X_{t-1}^j to X_t^i . The MLE of the block coefficient is represented as follows,

$$\mathbf{B}^{i,i} = \begin{bmatrix} b & b' & \dots & b' \\ b' & b & \dots & b' \\ \vdots & \vdots & \ddots & \vdots \\ b' & b' & \dots & b \end{bmatrix} \mathbf{B}^{i,j} = \begin{bmatrix} b'' & \dots & b'' \\ \vdots & \ddots & \vdots \\ b'' & \dots & b'' \end{bmatrix}$$

such that,

$$\begin{aligned}b &= \frac{1}{n_i} \sum_{k=1}^{n_i} \hat{\mathbf{B}}_{k,k}^{i,i}, b' = \frac{1}{n_i(n_i-1)} \sum_{\substack{(k,l)=(1,1) \\ k \neq l}}^{(n_i, n_i)} \hat{\mathbf{B}}_{k,l}^{i,i}, \\ b'' &= \frac{1}{n_i n_j} \sum_{(k,l)=(1,1)}^{(n_i, n_j)} \hat{\mathbf{B}}_{k,l}^{i,j}\end{aligned}$$

BlockAverage in Algorithm *LearningRKF* denotes this averaging-out procedure. The block coefficient matrix $\mathbf{B}$ is also the MLE of RTM.

3.3 Learning Observation Models

Given RTMs and a sequence of full observations, we derive the estimate X^t at time step t assuming that there is no observation noise.

Learning Observation Noise is to estimate the mean vector and covariance matrix for the ROM. The MLEs problem is formulated as follows,

$$(\mu_{\text{MLE}}, \Sigma_{\text{MLE}}) = \arg \max_{\mu_O, \Sigma_O} \sum_{t=1}^T \log \mathcal{N}(\vec{O}_t; \mu_O, \Sigma_O).$$

where $\vec{O}_t^i = O_t - \mathbf{C} \cdot X_t^i$. The derivation is similar to RTMs. One can substitute $\vec{O}_t^i$ for $\vec{X}_t^i$ in Proposition 2.

Learning Linear Coefficient C is to calculate the linear coefficient between X_t and O_t .

$$\hat{\mathbf{C}} = \left(\sum_{t=1, \dots, T} O_t X_t^T \right) \left(\sum_{t=1, \dots, T} X_t X_t^T \right)^{-1}$$

Here $\mathbf{C}$ is also calculated from $\hat{\mathbf{C}}$ by averaging out each sub-block as in learning $\mathbf{B}$.

4 LRKF with Regroupings

With the estimated parameters, the RKF predicts the state variables in the relational linear dynamic models. This section presents a new lifted Kalman filtering algorithm, which approximately regroupes degenerated relational structures. Existing lifted Kalman filtering algorithms (Choi, Guzman-Rivera, and Amir 2011; Ahmadi, Kersting, and Sanner 2011) suffer degenerations of relational structures when sparse observations are made.⁵ Algorithm *LRKF-Regroup* also degenerates the domains of relational atoms by calling *DegenAtom* when state variables are observed in different time, thus different *Obs* (e.g., $Obs_{(i,a)} \neq Obs_{(i,a')}$). Here, $Obs_{(i,a)}$ stores the most recently observed time for a ground substitution a in the i -th atom.

To overcome such degeneracy, *LRKF-Regroup* introduces a new subroutine, called *MergeAtom*, which merges covariance structures when random variables are not directly observed for a certain time steps, say k .

Algorithm 2 LRKF-Regroup (Prediction w/ testing data)

Input: params $(\mathbf{B}, \Sigma_T, \mathbf{C}, \Sigma_O)$, obs $(O_1, \dots, O_T)$
repeat
 $\mu_0 \leftarrow 0, \Sigma_0 \leftarrow 0$
 $(Obs_{(1,1)}, \dots, Obs_{(n,n_i)}) \leftarrow (0, \dots, 0)$
for $t \leftarrow 1$ **to** T **do**
 $(\mu'_t, \Sigma'_t) \leftarrow \text{Predict-RTM}(\mu_{t-1}, \Sigma_{t-1}, \mathbf{B}, \Sigma_T)$
for all (i, a) s.t. $O_t^i(a)$ is observed **do**
 $Obs_{(i,a)} \leftarrow t$
end for
 $(\mathbf{B}, \Sigma_T, \mathbf{C}, \Sigma_O) \leftarrow \text{DegenAtom}(Obs, \mathbf{B}, \Sigma_T, \mathbf{C}, \Sigma_O)$
 $(\mu_t, \Sigma_t) \leftarrow \text{Update-ROM}(\mu'_t, \Sigma'_t, \mathbf{C}, \Sigma_O)$
 $(\mathbf{B}, \Sigma_T, \mathbf{C}, \Sigma_O) \leftarrow \text{MergeAtom}(Obs, t, \mathbf{B}, \Sigma_T, \mathbf{C}, \Sigma_O)$
end for
until t is T
Output: state estimations $((\mu_0, \Sigma_0), \dots, (\mu_T, \Sigma_T))$

The *MergeAtom* operation iterates atoms and find all state variables which are not observed for a certain time steps k . The selected variables are stored in *mlist*. Then, the *BlockMerge* respectively averages diagonal entries and non-diagonal entries in *mlist*, and sets the averaged values to state variables in *mlist*. In this way, it rebuilds the compact relational structure again.

⁵Note that, the lifted algorithm in (Choi, Guzman-Rivera, and Amir 2011) only valid when the same number of observations are made at the same time steps.

Algorithm 3 MergeAtom

input: recent obs time *Obs*, time t , params $(\mathbf{B}, \Sigma_T, \mathbf{C}, \Sigma_O)$
 $mlist \leftarrow \emptyset$
for $i = 1$ **to** n **do**
for each a s.t. $Obs_{(i,a)} + k \leq t$ **do**
 $mlist^i \leftarrow mlist^i \cup \{a\}$
end for
end for
 $(\mathbf{B}', \Sigma'_T, \mathbf{C}', \Sigma'_O) = \text{BlockAverage}(mlist, \mathbf{B}, \Sigma_T, \mathbf{C}, \Sigma_O)$
output: merged relational structures $(\mathbf{B}', \Sigma'_T, \mathbf{C}', \Sigma'_O)$

Lemma 3. When (1) at least one relational observation is made on O_i at each time step; the variance of any state variable $X^i(a)$ in LRKF-Regroup is bounded by σ_{ROM}^i ² and converges to $\sqrt{\sigma_{ROM}^i \sigma_{RTM}^i + \frac{\sigma_{ROM}^i{}^4}{4} - \frac{\sigma_{RTM}^i{}^2}{2}}$.

Proof. Let the variance of the i -th atom be σ_{RTM}^i and the variances of direct and relational observations respectively be σ_{ROM}^i ² and σ_{ROM}^i ² where $\sigma_{ROM}^i < \sigma_{ROM}^i$ as Equation (4) and (5).

Let $\sigma_t^i(a)$ ² be the variance of a state variable $X^i(a)$ at time t . This variable will be updated by at least one relational observation by the LRKF-Regroup. Then, the new variance is $\frac{1}{\sigma_{t+1}^i(a)^2} = \frac{1}{\sigma_t^i(a)^2} + \frac{1}{\sigma_{ROM}^i{}^2}$. That is, $\sigma_{t+1}^i(a)^2 \leq \min(\sigma_t^i(a)^2, \sigma_{ROM}^i{}^2)$. Use the following equation for transition and update in each filtering step,

$$\frac{1}{\sigma_{t+1}^i(a)^2} \leq \frac{1}{\sigma_t^i(a)^2 + \sigma_{RTM}^i{}^2} + \frac{1}{\sigma_{ROM}^i{}^2}$$

For the convergence, let $\sigma_{t+1}^i(a) = \sigma_t^i(a) = \sigma_*^i(a)$. The variance is $\sigma_*^i(a)^2 = \sqrt{\sigma_{ROM}^i \sigma_{RTM}^i + \frac{\sigma_{ROM}^i{}^4}{4} - \frac{\sigma_{RTM}^i{}^2}{2}}$ $\square$

Theorem 4. When (1) at least one relational observation is made on O_i at each time step; and (2) no direct observation is made on two state variables $X^i(a)$ and $X^i(a')$ at least for k time steps, the difference of the variances of two state variables $\sigma_t^i(a)$ and $\sigma_t^i(a')$ is bounded by $c^k \sigma_{RTM}^i{}^2 (1 + \sigma_*^i(a)^2 / \sigma_{ROM}^i{}^2)$ where c is $\sigma_*^i(a)^2 / (\sigma_*^i(a)^2 + \sigma_{ROM}^i{}^2)$ and $c \leq 1$.

Proof. We follow the result of the Lemma 3 and use $\sigma_*^i(a)$. The variance of each time step follows the recursive form,

$$\frac{1}{\sigma_{t+1}^i(a)^2} = \frac{1}{\sigma_t^i(a)^2 + \sigma_{RTM}^i{}^2} + \frac{1}{\sigma_{ROM}^i{}^2}. \quad (8)$$

An exact (non-recursive) formula for $\sigma_{t+1}^i(a)$ is non trivial. Thus, we introduce another simpler, converging sequence,

$$\bar{\sigma}_{t+1}^i(a)^2 = c(\bar{\sigma}_t^i(a)^2 + \sigma_{RTM}^i{}^2).$$

Since $\sigma_t^i(a)^2 - \bar{\sigma}_t^i(a)^2$ is positive and convex when $c < 1$,

$$0 \leq \bar{\sigma}_t^i(a) \leq \sigma_t^i(a) \leq \sigma_*^i(a), \quad \sigma_*^i(a)^2 - \bar{\sigma}_t^i(a)^2 \geq \sigma_*^i(a)^2 - \sigma_t^i(a)^2$$

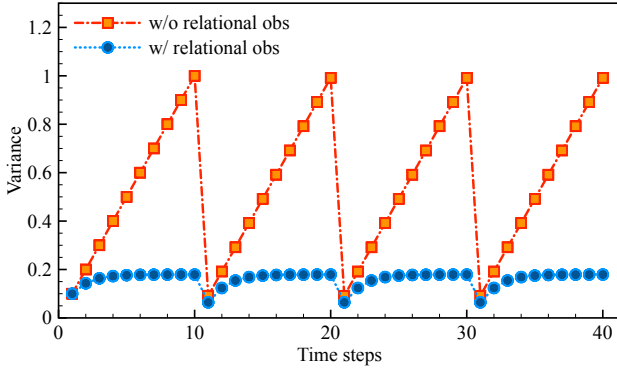


Figure 2: Variances of state variables with one observation per 10 time steps without a relational obs. (square shaped marks) and with relational obs. (circle shaped marks). This figure shows the variances (y axis) in time steps (x axis). Setting that $\sigma_{RTM}^2=0.1$, $\sigma_{ROM}^2=0.1$, $\sigma_{ROM}^{\prime 2}=0.5$, and let observations made at different time steps. This simulation shows the intuition of Lemma 3.

The convergence of the simpler form is slower than the original one in Equation (8). However, it provides an exact formulation and converge exponentially,

$$\bar{\sigma}_t^i(a)^2 = c^k \bar{\sigma}_{t-k}^i(a)^2 + \sigma_{RTM}^2 \frac{2(1-c^k)}{1-c}$$

WLOG, we set $X^i(a')$ has no direct observation longer than $X^i(a)$. The variance of $X^i(a')$ has the same formulation with a substitution of $k+\alpha$ for k . Thus, variance of $X^i(a')$ is σ^i

$$\bar{\sigma}_t^i(a')^2 = c^{k+\alpha} \cdot \bar{\sigma}_{t-k-\alpha}^i(a')^2 + \sigma_{RTM}^2 \frac{2(1-c^{k+\alpha})}{1-c}$$

Note that, $\bar{\sigma}_{t+\alpha}^i(a) \leq \bar{\sigma}_{t+\alpha}^i(a') \leq \bar{\sigma}_{t+\infty}^i(a')$.

$$\begin{aligned} |\sigma_{t+\alpha}^i(a')^2 - \sigma_t^i(a)^2| &\leq |\sigma_{t+\infty}^i(a')^2 - \sigma_t^i(a)^2| \\ &= c^k (\sigma_{RTM}^2 / (1-c) - \sigma_{t+k}^i(a)^2) \\ &= c^k (\sigma_{RTM}^2 (1 + \sigma_*^i(a)^2 / \sigma_{ROM}^2) - \sigma_{t+k}^i(a)^2) \\ &\leq c^k \sigma_{RTM}^2 (1 + \sigma_*^i(a)^2 / \sigma_{ROM}^2). \end{aligned}$$

□

5 Experimental Results

To compare the improve the efficiency, we implement the *Lifted RKF* (Choi, Guzman-Rivera, and Amir 2011) and our *LRKF-Regroup*. The algorithm is compared in two datasets with sparse observations: one synthetic dataset and one real-world ground water dataset. Note that, the lifted RKF will not degenerate model on full, dense, observations. In both experiment, we set k to be 4. That is, two state variables will be merged if they have the same observation numbers and types when at least a relational observation is made. With relational observations, the variances of degenerated state variables are reasonably small even after 4 time steps.

In the synthetic dataset, we assume that an atom with 300 ground substitutions, $|X^i|=30$. Then we make a sparse observations with a rate of 90%. That is, 90% of state variables will be observed in each time step. Then, we report the numbers of shattered (or degenerated) groups and average filtering time in the Lifted RKF and in our *LRKF-Regroup*. The result is shown in Figure 3.

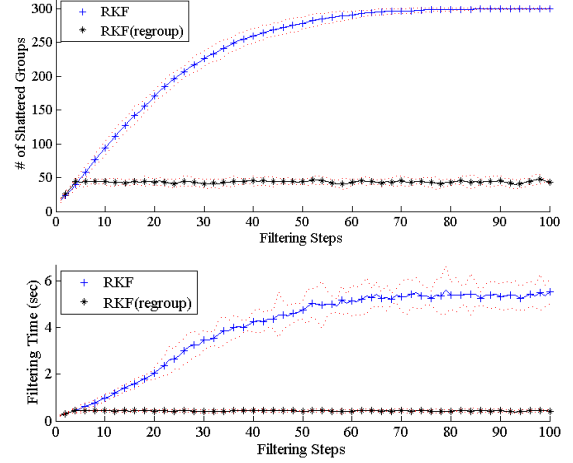


Figure 3: Comparisons of Lifted RKF and LRKF-Regroup in a simulation dataset.

To verify the efficiency of Lifted-Regroup, we use a regional groundwater flow MODFLOW model, the Republican River Compact Association (RRCA) Model (McKusick 2003). The extracted dataset is a small subset of the whole ground water model, monthly measured head (water level) at over 3,000 wells. We choose a closely related 200 wells and randomly select 100 measurement periods (100 months). Then, we report the degeneracies in *Lifted-RKF* and *LRKF-Regroup* and average filtering time affected by the degenerations in Figure 4.

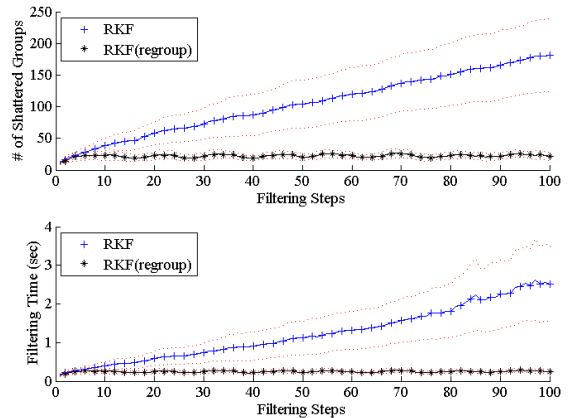


Figure 4: Comparisons of Lifted RKF and LRKF-Regroup in the groundwater model.

6 Conclusion

This paper provides new answers and insights on (1) how to learn parameters for RKF; and (2) how to regroup the state variables from noisy real-world data. We propose a new algorithm that regroups the state variables when individual observations are made to RKF in different time steps. We use the RKF in a simulated and a real-world dataset, and demonstrate that the RKF improves the efficiency of filtering the large-scale dynamic system.

7 Acknowledgments

This work is supported by the National Science Foundation Hydrologic Science Program under Grant No. 0943627. This work was supported by the year of 2014 Research Fund of UNIST (Ulsan National Institute of Science and Technology).

References

- Ahmadi, B.; Kersting, K.; and Sanner, S. 2011. Multi-evidence lifted message passing, with application to pagerank and the kalman filter. In *Proceedings of the International Joint Conference on Artificial Intelligence*.
- Bahmani-Oskooee, M., and Brown, F. 2004. Kalman filter approach to estimate the demand for international reserves. *Applied Economics* 36(15):1655–1668.
- Choi, J.; Guzman-Rivera, A.; and Amir, E. 2011. Lifted relational kalman filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence*.
- Choi, J.; Hill, D. J.; and Amir, E. 2010. Lifted inference for relational continuous models. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 126–134.
- Clark, M. P.; Rupp, D. E.; Woods, R. A.; Zheng, X.; Ibbitt, R. P.; Slater, A. G.; Schmidt, J.; and Uddstrom, M. J. 2008. Hydrological data assimilation with the ensemble kalman filter: Use of streamflow observations to update states in a distributed hydrological model. *Advances in Water Resources* 31(10):1309 – 1324.
- de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted first-order probabilistic inference. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1319–1325.
- Evensen, G. 1994. Sequential data assimilation with a non-linear quasi-geostrophic model using monte carlo methods to forecast error statistics. *Journal of Geophysical Research* 99:10143–10162.
- F.P., M., and Bierkens. 2001. Spatio-temporal modelling of the soil water balance using a stochastic model and soil profile descriptions. *Geoderma* 103(1–2):27–50.
- Friedman, N.; Getoor, L.; Koller, D.; and Pfeffer, A. 1999. Learning probabilistic relational models. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1300–1309.
- Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering* 82(Series D):35–45.
- McKusick, V. 2003. Final report for the special master with certificate of adoption of rca groundwater model. *State of Kansas v. State of Nebraska and State of Colorado, in the Supreme Court of the United States* 3605(1145):4440.
- Poole, D. 2003. First-order probabilistic inference. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 985–991.
- Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.
- Roweis, S., and Ghahramani, Z. 1999. A unifying review of linear gaussian models. *Neural Comput.* 11(2):305–345.
- Wang, J., and Domingos, P. 2008. Hybrid markov logic networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1106–1111.

Representation, Reasoning, and Learning for a Relational Influence Diagram Applied to a Real-Time Geological Domain

Matthew C. Dirks*, Andrew Csinger†, Andrew Bamber†, David Poole*

*University of British Columbia (UBC), †Minesense Technologies Ltd.
mcdirks@cs.ubc.ca

Abstract

Mining companies typically process all the material extracted from a mine site using processes which are extremely consumptive of energy and chemicals. Sorting the good material from the bad would effectively reduce required resources by leaving behind the bad material and only transporting and processing the good material. We use a relational influence diagram with an explicit utility model applied to the scenario in which an unknown number of rocks in unknown positions with unknown mineral compositions pass over 7 sensors toward 7 diverters on a high-throughput rock-sorting machine developed by MineSense™ Technologies Ltd. After receiving noisy sensor data, the system has 400 ms to decide whether to activate diverters which will divert the rocks into either a keep or discard bin. We learn the model offline and do online inference. Our result improves over the current state-of-the-art.

This paper considers the problem of sorting rock ore based on mineralogy, separating the valuable, high-grade rocks from the low-grade rocks as they pass over an array of electromagnetic sensors. By sorting more effectively, we reduce costs and help preserve the environment, because the amount of material sent to further downstream mining processes is reduced. MineSense™ is a Vancouver-based company developing conductivity-based sensing and sorting systems to sort ore more effectively and have developed a rock sorting platform called SortOre™ pictured in 1.

We have developed an anytime algorithm (Zilberstein 1996) for sorting called Rock Predictor Sorting Algorithm (RPSA) which uses the SortOre™ rock sorting platform on which we have performed training and evaluation.

In SortOre™, rocks are dumped on a conveyor belt, moving downward on the y-axis as in the schematic in Figure 2. As rocks travel, they pass over a **sensor array** consisting of 7 electromagnetic coils which measure magnetic and electromagnetic flux and respond to materials which have conductive or magnetic properties. Each sensor takes readings of the magnitude of the magnetic field disrupted by the rock. The computer processes the sensor data while the rocks travel to the end of the **conveyor belt** where they fall



Figure 1: MineSense™ SortOre™

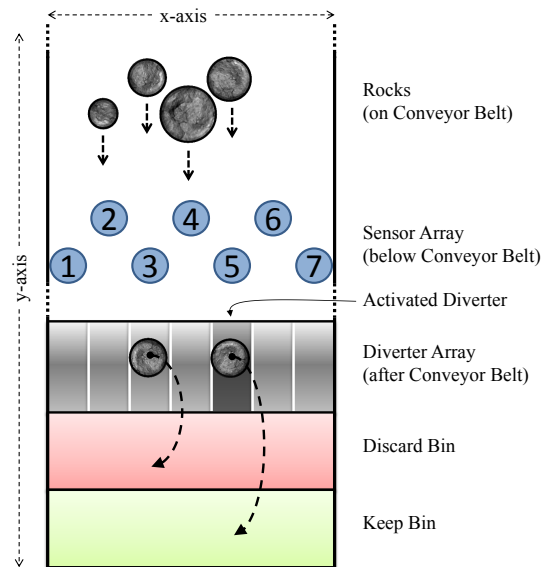


Figure 2: A schematic of the sorter.

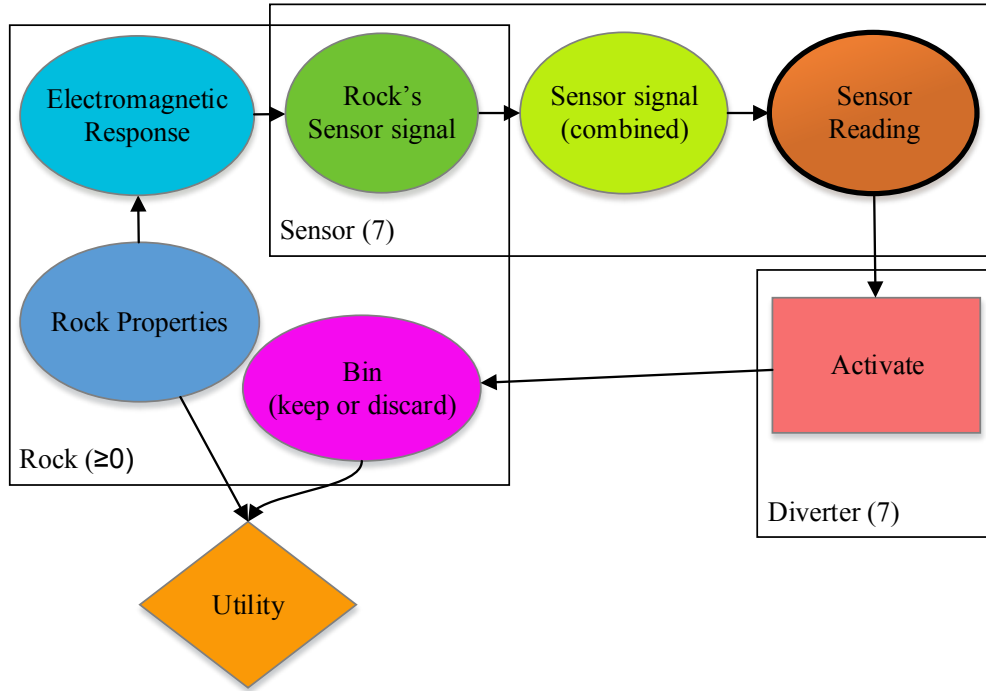


Figure 3: Relational Influence Diagram (High-Level)

verters **activated**, displacing rocks into the **keep bin** or else leaving them to fall into the **discard bin**.

Related Work

Surprisingly, little research has been done in the use of electromagnetic coils in sensors to detect and locate minerals. Similar research is that of detecting and locating unexploded ordnance beneath the ground, however at a long range and with the goal of identifying single objects at a time (Das et al. 1990). Likewise, inductive sensing has also been used in coin recognition, but again only considers the case of single objects (Passeraub et al. 1997).

The Model

The system can be described using a relational influence diagram (Shachter 1988) with plate notation (Buntine 1994). We chose to use a relational influence diagram to explain the model and to reveal explicitly what is and isn't being modelled, though the solution we develop does not necessarily require it. Furthermore, we hope to use this diagram to discover new research areas by spotting where the diagram is lacking and to describe changes in the future in terms of adding or removing diagram elements (variables, arrows, plates, etc) making future work more clear. We have two diagrams: the first one (Figure 3), shows the system at a higher-level and we believe is conceptually easier to understand, and the second (Figure 4) describes the same system

broken down into simpler components with their relations explained more clearly.

First, refer to the high-level influence diagram (Figure 3). There is an unknown number of rocks (zero or more), which may or may not be overlapping, each with **rock properties** describing the size, minerality, and position of the rock. To simplify the problem we model rocks as circles with some amount of mineral content spread equally throughout and currently we do not attempt to model mineral type.

A rock's properties influence the **electromagnetic response** it gives. The electromagnetic response variable models how a rock would respond to a sensor regardless of distance to the sensor.

The rocks do not necessarily pass directly over a sensor, but may pass between sensors or over multiple sensors. A **rock's sensor signal** is the signal of the rock over a particular sensor. The **sensor signal** combines the rock sensor signals from the rocks passing (partially or wholly) over it, and we explain it more later.

Finally, we model a **sensor reading** as a sample from the sensor signal at discrete times (milliseconds). Sensor reading is the only observed variable, and is taken at 1 millisecond intervals.

Given only the sensor signals of all 7 sensors, a decision must be made to **activate** or not activate each diverter for each millisecond in time. The diverters then guide each rock into either the **keep bin** or the **discard bin**. This action is stochastic. Please note, there are two ways for a good rock

to incorrectly end up in the discard bin (i.e. a false positive). First, imagine a good rock at some location, the model may incorrectly predict that the rock is bad at that location or simply that there are no rocks at this location and then will choose not to activate the diverter nearest this rock resulting in a false positive. Second, the model may correctly predict a good rock at its location and activate the diverter at the correct time but the rock may tumble unexpectedly and fall into the discard bin resulting in a false positive.

It is with regard to the second which we model in the diverter's activate variable. Currently we have trained a simple Bernoulli model for when an activated diverter fails to divert the rock into the keep bin due to the rock tumbling or hitting another rock. Other unexpected results may also occur such as a rock tumbling over toward a diverter it would not otherwise have been hit by, colliding with another rock, or splitting into 2 smaller rocks; However, we did not model these because we deemed the accuracy to be good enough to compare one algorithm relative to another (see Evaluation Section for details).

The **utility** takes into account the number of false positives (bad rocks in the keep bin, FP) and false negatives (good rocks in the discard bin, FN) as well as the cost of false positives ($C(FP)$) and the cost of false negatives ($C(FN)$). To make an appropriate decision, we take into account utility and associated misclassification costs.

We use normalized expected cost ($Norm(E[Cost])$) (Drummond and Holte 2006) as the cost (or loss) function for utility, where smaller values are better.

$Norm(E[Cost])$:

$$\frac{FN \cdot C(FN) + FP \cdot C(FP)}{P(+)\cdot C(FN) + P(-)\cdot C(FP)} \quad (1)$$

Where FP and FN are the counts of bad rocks in the keep bin and good rocks in the discard bin. For example, FN is the number of rocks which ended up in the discard bin but whose true mineral content actually represents a good rock. $P(+)$ and $P(-)$ are the distributions of a good (positive) rock occurring and of a bad (negative) rock occurring respectively.

The costs, $C(FP)$ and $C(FN)$, will depend on various factors outside of the model (e.g. environmental costs, the cost of transportation, and the price of minerals). The ratio between the two is what matters.

As in Cost Curves (Drummond and Holte 2006), misclassification costs (utility) and class distributions are expressed in a single term called **Probability Cost** (PC):

$$PC(+) = \frac{P(+)\cdot C(FN)}{P(+)\cdot C(FN) + P(-)\cdot C(FP)} \quad (2)$$

We discretized the domain of $PC(+)$ into a set of 9 and learned the model for each. Only misclassification costs varies, the class distributions remained constant in our evaluation.

Model Details

The specifics of the relations in the system are described next, please refer to the detailed relational influence diagram in Figure 4. This detailed diagram is just as accurate as

the previous, but helps to explain how each variable is connected. Rocks are denoted with an i subscript and sensors are denoted with a j subscript (1 thru 7).

As shown in the detailed diagram, we model rock properties as 4 components: mineral content (c_i), size as a radius (r_i), and position (x_i , y_i). Electromagnetic response is modelled as an un-normalized symmetric 2-D Gaussian function which is broken down into two parameters: width and height. The width (s_i) parameter is a linear function of the rock's size:

$$s_i(r_i) = (ar_i + b)^2 \quad (3)$$

where slope (a) and bias (b) are parameters which are learned. The height (h_i) is a linear function of the rock's mineral content:

$$h_i(c_i) = c_i m \quad (4)$$

where m is a parameter which is learned. The 2-D mean of the 2-D Gaussian is the same as the rock's position so is not modelled separately.

We model a rock sensor signal as an un-normalized 1-D Gaussian function. A 1-D Gaussian typically has 3 parameters: width, height, and mean. We exploit a property of 2-D Gaussians to relate electromagnetic response to rock sensor signal: a cut of a 2-D Gaussian will be a 1-D Gaussian with the same width, and if the cut is parallel to the y-axis, then the mean of the 1-D Gaussian will be equal to the y-position of the 2-D Gaussian's mean. Since the rocks move vertically on the y-axis, all cuts will be parallel to the y-axis.

Therefore, the 1-D Gaussian width is equal to the electromagnetic response's width and the mean is equal to the y-position of the electromagnetic response which is equal to the y-position of the rock. Hence, we reuse the x_i node from rock properties and the width (s_i) node from electromagnetic response. The Gaussian property is used during inference and is the reason we chose to use Gaussians.

The height of the 1-D Gaussian is modelled as the variable height (h_{ij}), shown below:

$$h_{ij}(h_i, x_j, x_i, s_i) = h_i e^{-1/2 \frac{(x_j - x_i)^2}{s_i}} \quad (5)$$

Where h_{ij} is the height of the rock sensor signal for rock i on sensor j , x_j is the x-position of the j^{th} sensor, x_i is the x-position of rock i , and s_i is the width of the electromagnetic response for rock i .

Lastly, we model sensor signal as a sum of all the rock sensor signals. Since rock sensor signal is a 1-D Gaussian, summing creates a sum of Gaussians.

$$f_j(t) = \sum_{i=1}^n h_{ij} e^{-1/2 \frac{(t - y_i)^2}{s_i}} \quad (6)$$

Where t is continuous time and y_i is rock i 's y-position.

Runtime Inference

The problem is resource bounded for a variety of reasons including: the speed of the rocks, the length of the conveyor belt, and communication overhead. In short, we have 400ms to make a decision about which diverters to activate, and we can only observe sensor readings of each sensor. To meet the

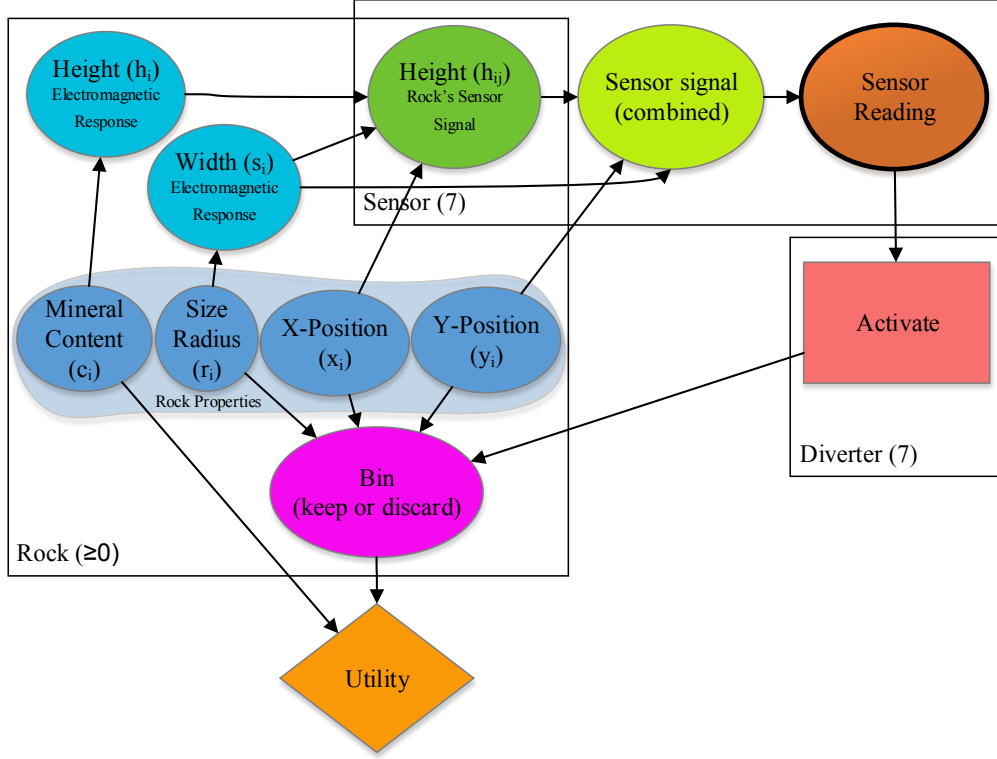


Figure 4: Relational Influence Diagram (Detailed)

400ms deadline, we collect a group (called a **data window**) of 200 sensor readings at a time and run inference on each group independently.

At runtime our algorithm hypothesizes the positions, sizes and mineral contents of the rocks on the conveyor belt within the data window and determines for each diverter whether to activate it.

We have experimented with a number of inference algorithms to determine the positions and compositions of the rocks, including MCMC (Robert and Casella 2009), gradient-based methods (Snyman 2005), and various methods within the SciPy library (Jones et al. 2001). We found that what works the best is a mix between random and coarse exhaustive search.

For each sensor, and within the current data window, we exhaustively search over all values of rock size and rock mineral content.

However, rock size is limited to only 2 possible values and mineral content is limited to 11 values including 0. By including 0 we allow for an unknown number of rocks to be predicted, because a rock with 0 mineral content is effectively removing the rock (we treat non-valuable rocks and non-existent rocks the same). The space of a rock’s position (x, y) is continuous; We randomly choose values for x and y , typically performing around 100 samples (we sample as many as we have time for within the time limit).

For each sample in the search, we generate the sensor signal due to the rocks and compare it to the observed sensor readings. Sampling repeats until time runs out, at which point the best hypothesis is chosen and decisions to activate are made.

Offline Learning

For offline learning we recorded sensor data and manually labelled the rocks and their properties. Generating data to use offline is expensive and time-consuming.

Using the recorded sensor data offline, we automatically configure our model’s parameters using a state-of-the-art algorithm configuration method called SMAC (Hutter, Hoos, and Leyton-Brown 2011). SMAC searches over the space of parameters optimizing a given cost function — in our case we optimize for utility. We trained on 3/4 of our labelled data and held the remaining 1/4 aside as the test set.

For each $PC(+)$ (i.e. misclassification cost) we ran SMAC for 4 hours on 8 2.67-GHz Xeon x5550 cores on WestGrid’s Hermes cluster.

Evaluation

The previously used sorting algorithm, which we call VBSA, looks at the sensor readings of each sensor, and when the magnitude of the reading passes some threshold it activates the diverter corresponding to the sensor. This sorting

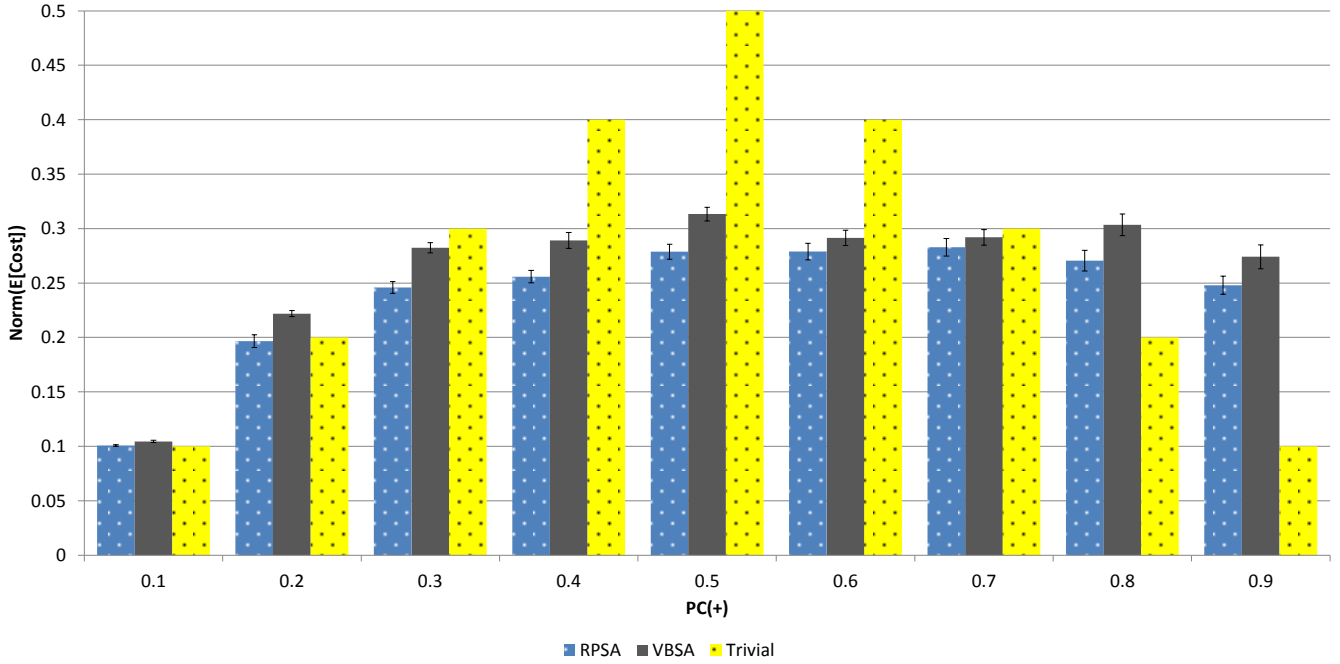


Figure 5: Comparison of two sorting algorithms using the learned parameters for each algorithm for each $PC(+)$ (representing misclassification costs), where lower $Norm(E[Cost])$ is better.

algorithm made no attempt to infer rocks or their properties, but does work surprisingly well given its simplicity.

We ran both the previously used algorithm, VBSA, and our new algorithm, RPSA, on the test set. The sorting decisions outputted from VBSA is also fed through the same diverter and utility model as ours - for the purpose of comparing two algorithms, the diverter model does not have to be perfect for a relative comparison.

Results can be seen in Figure 5. The “trivial” algorithm shows the cost of rejecting (for $PC(+)$ < 0.5) or accepting (for $PC(+)$ > 0.5) all the rocks all the time. At $PC(+)$ of 0.5 one can accept or reject all or randomly sort in order to achieve the trivial cost. We show improvement using our relational influence diagram based sorting algorithm versus VBSA across the 9 values of $PC(+)$, while still being able to make decisions online within the real-time constraint.

Our algorithm, RPSA, improves on VBSA by 9% on average across the 9 values of $PC(+)$ and is statistically significant for where $PC(+)$ is 0.5 and lower. Error bars represent standard error of the mean.

Future Work

Future challenges are to allow us to have a more detailed and accurate model. More specifically, a better model of rock properties; Including a model of various mineral types (one type per rock) as well as heterogeneous rocks (multiple mineral types per rock), and rocks of odd shapes and sizes (removing the assumption that rocks are circles). We would also like to consider alternate models for the sensor signal, other than Gaussians.

Furthermore, we would like to be able to draw an influence diagram and automatically perform inference and learning on the model, enabling us to quickly try new changes to the model and evaluate the results.

Conclusion

Sorting rocks using electromagnetic sensors is a challenging real-world planning problem where there is an unknown number of objects and we have to act in real-time. We created a new sorting algorithm, RPSA, by modeling the problem using a relational influence diagram and automatically configured its parameters offline. RPSA does online inference and beats the previously used algorithm, VBSA.

References

- Buntine, W. L. 1994. Operations for learning with graphical models. *Journal of Artificial Intelligence Research* 2:159–225.
- Das, Y.; McFee, J. E.; Toews, J.; and Stuart, G. C. 1990. Analysis of an electromagnetic induction detector for real-time location of buried objects.
- Drummond, C., and Holte, R. C. 2006. Cost curves: An improved method for visualizing classifier performance. *Machine Learning* 65(1):95–130.
- Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2011. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization, LION’05*, 507–523. Berlin, Heidelberg: Springer-Verlag.

- Jones, E.; Oliphant, T.; Peterson, P.; et al. 2001–. SciPy: Open source scientific tools for Python.
- Passeraub, P. A.; Besse, P.-A.; de Raad, C.; Dezuari, O.; Quinet, F.; and Popovic, R. S. 1997. Coin recognition using an inductive proximity sensor microsystem. In *Proceedings of International Conference on Solid State Sensors and Actuators*, volume 1, 389–392.
- Robert, C., and Casella, G. 2009. A short history of Markov chain monte carlo: Subjective recollections from incomplete data.
- Shachter, R. D. 1988. Probabilistic inference and influence diagrams. *Operations Research* 36(4):589–604.
- Snyman, J. A. 2005. *Practical mathematical optimization : an introduction to basic optimization theory and classical and new gradient-based algorithms*. Applied optimization. New York: Springer.
- Zilberstein, S. 1996. Using anytime algorithms in intelligent systems. *AI Magazine* 17(3):73–83.

Extending PSL with Fuzzy Quantifiers

Golnoosh Farnadi^{1,2}, Stephen H. Bach³, Marie-Francine Moens²,
Lise Getoor⁴, Martine De Cock^{1,5}

¹Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Belgium,

²Department of Computer Science, Katholieke Universiteit Leuven, Belgium,

³Statistical Relational Learning Group, University of Maryland, USA,

⁴University of California, Santa Cruz, USA,

⁵Center for Data Science, University of Washington Tacoma, USA

Introduction

Many interesting tasks in artificial intelligence require the ability to work with *imperfect relational* data. Examples include social network applications in which partial information about users and about their connections is available. For instance, we may have *attribute* information about users; we may know the age, gender, personality, likes and dislikes etc. of a user, but this knowledge is usually incomplete in the sense that we do not have all attribute values for all users. The knowledge can also be uncertain, for instance when the attribute values are not given directly by the user but instead they are inferred from user generated content.

The *relational* aspect of social network data stems from the connections between the users, e.g., Facebook users are connected with their friends; Twitter users can follow one another; in Amazon, users can bookmark other interesting users; in Epinions users can trust other users or include them in their block list (i.e. distrust) etc. Again, the available knowledge about the connections is typically incomplete and uncertain. An interesting task is to perform most probable explanation (MPE) inference over a given social network, i.e. based on (1) given attribute values about users and their connections and (2) some background knowledge about the domain, expressed as probabilistic rules in first-order logic, infer missing attribute values such that the given and the inferred values combined adhere to the background knowledge as well as possible. MPE inference is an important task in *statistical relational learning* (SRL).

Probabilistic Soft Logic (PSL)

A particularly interesting framework for SRL is *probabilistic soft logic* (PSL). Unlike in other forms of SRL, in which predicates are Boolean, in PSL they can take soft truth values in the interval $[0, 1]$. Conjunction and disjunction in PSL rules are respectively modelled by the Łukasiewicz t-norm and t-conorm because their particular piecewise linear form allows to cast MPE inference for a PSL program as a convex optimization problem. Negation is the standard negation

from fuzzy logic, i.e. for x and y in $[0, 1]$ (the $\sim$ indicates the relaxation over Boolean values):

$$x \tilde{\wedge} y = \max(0, x + y - 1), \quad x \tilde{\vee} y = \min(x + y, 1) \\ \tilde{\neg} x = 1 - x$$

In a PSL program, relationships and attributes are modelled by predicates, and first-order rules model constraints on these predicates. MPE inference is concerned with finding the most probable assignment of truth values. The probability of truth value assignments is defined by a weighting function, namely, distance to satisfaction (d_r). Generally, the distance to satisfaction of an unsatisfied rule is the difference between the truth values of the body and the head. A rule r is satisfied when the truth value of its head is at least as high as the truth value of its body. Therefore, the rule's distance to satisfaction under an interpretation I is defined in Equation (1). An interpretation I is a mapping that associates a truth value to each element, i.e., $I(x) \in [0, 1]$.

$$d_r(I) = \max\{0, I(r_{body}) - I(r_{head})\} \quad (1)$$

A PSL program, i.e. a set of PSL rules, induces a distribution over interpretations I . The probability density function is (Brocheler, Mihalkova, and Getoor 2012):

$$f(I) = \frac{1}{Z} \exp\left[-\sum_{r \in R} \lambda_r (d_r(I))^p\right] \quad (2)$$

where λ_r is the weight of the rule r , Z is a normalization constant (see Equation (3)), and $p \in \{1, 2\}$ provides a choice of two different loss functions. $p = 1$ favors the satisfaction of one rule, and $p = 2$ favors the satisfaction of all rules to some degree. These probabilistic models are instances of hinge-loss Markov random fields (Bach et al. 2013).

$$Z = \int_I \exp\left[-\sum_{r \in R} \lambda_r (d_r(I))^p\right] \quad (3)$$

For example $Trusts(A, B) \rightarrow Knows(A, B)$ models that “if A trusts B then A knows B ” where A and B are variables referring to arbitrary objects (Huang et al. 2012). An example of a grounded version of this PSL rule is $Trusts(Alice, Bob) \rightarrow Knows(Alice, Bob)$. If

$Trusts(Alice, Bob)$ and $Knows(Alice, Bob)$ are true to respectively degree 0.7 and 0.5, then the rule is satisfied to degree $1 - (0.7 + 0.5) = 0.8$. The truth degree of some predicates is given in advance, and, roughly speaking, the goal of MPE inference is to find a truth assignment for the other predicates such that the combined degree of satisfaction of all grounded rules is as high as possible.

Variables in PSL rules are implicitly universally quantified. A PSL rule $Trusts(A, X) \wedge Trusts(X, B) \rightarrow Trusts(A, B)$, models that “ A trusts B ” is true to the degree to which there is a trusted third party X . In standard PSL there is no direct way to express that A can trust B if *most* friends of A trust B . The notion *most* in this statement represents a fuzzy quantifier. Atoms whose truth values are functions of other atoms’ truth values are defined using *aggregates*. PSL supports linear aggregates, e.g., the average truth value of a set of atoms. However, non-linear aggregates such as the fuzzy quantifiers as we propose in the next section require additional capabilities.

Fuzzy quantifiers

The idea of fuzzy quantifiers (Delgado et al. 2013) was first introduced by Zadeh (Zadeh 1983). Quantifiers represent notions such as “most” and “a few”. The necessity of introducing a new notation to define quantifiers based on a linguistic perspective has also been studied in (Barwise and Cooper 1981; Keenan and Westerstaahl 2011) in the context of generalization quantifiers.

Intuitively, quantifiers relate to the concept of cardinality of sets. Recall that a fuzzy set A in a universe X is a $X \rightarrow [0, 1]$ mapping. The cardinality of A is defined as the sum of the individual membership degrees, i.e.

$$|A| = \sum_{x \in X} A(x) \quad (4)$$

A fuzzy quantifier Q is a $[0, 1] \rightarrow [0, 1]$ mapping. Q is called coherent if Q is non-decreasing and if it satisfies the boundary conditions $Q(0) = 0$ and $Q(1) = 1$. Figures 1 and 2 depict two coherent fuzzy quantifiers.

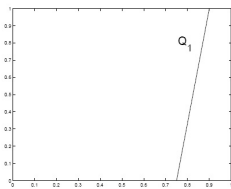


Figure 1: Fuzzy quantifier Q_1

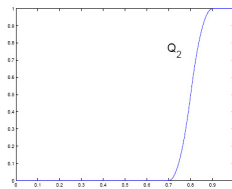


Figure 2: Fuzzy quantifier Q_2

Zadeh (Zadeh 1983) suggested to calculate the truth value of “ Q A ’s are B ’s”, with A and B fuzzy sets in X , as:

$$Q\left(\frac{|A \cap B|}{|A|}\right) \quad (5)$$

Other approaches for fuzzy quantifiers and aggregation operators have been proposed, most notably Yager’s OWA-operators (Yager 1988).

Extending the semantics of PSL to allow the use of these quantifiers is non-trivial. The main challenge is finding a way in which MPE inference over PSL programs can be cast as a convex optimization problem.

Extension approach

A PSL model consists of a collection of weighted rules. Given the data, rules are grounded out with substitution of logical terms, and form hinge-loss potentials for Equation (2). Hinge-loss MRFs have log concave density functions and finding a MPE is a convex optimization problem, which is solvable in polynomial time. Although, the numerator of the fuzzy quantifier (i.e., Equation (5)) is grounded out using the Łukasiewicz t-norm, grounding the function to some threshold is not linear and cannot be represented with linear constraints.

However, one could initialize the aggregate values by grounding the quantifier operator. Each fuzzy aggregate can be defined as a new predicate, e.g., most of the trusted friends can be defined by $MostFriendsTrust(A, B)$ which calculates the cardinality of the trusted friends of A who trust B over all trusted friends of A . By using this new predicate, we are able to define rules similar to the standard PSL rules using conjunction and disjunction, e.g., the following PSL rule indicates that if most friends of A trust B , A trusts B : $MostFriendsTrust(A, B) \rightarrow Trusts(A, B)$

The value of each grounded predicate will be updated to satisfy all grounded rules during the MPE inference or learning phase which maximizes the likelihood of each variable conditioned on all other variables. Thus, to update the value of the fuzzy aggregates, we propose to ground the aggregate function iteratively during the inference and weight learning phase until the aggregate value has converged.

Conclusion

PSL is a probabilistic modeling framework which uses first-order logic and soft truth values in the interval $[0, 1]$ for reasoning in relational domains. PSL uses the Łukasiewicz t-norm and t-conorm from fuzzy logic to model respectively conjunction and disjunction. In the current version of PSL, rules representing notions such as “most” and “a few” have not been addressed. We propose an extension to PSL with fuzzy quantifiers to overcome this limitation. We believe fuzzy quantifiers expand the ability of PSL to model social network applications which work with *imperfect relational data*, e.g., trust propagation, link prediction, node labelling.

Acknowledgements

This project is part of the SBO Program of the IWT-SBO-Nr. 110067 and NSF CCF0937094.

References

- Bach, S. H.; Huang, B.; London, B.; and Getoor, L. 2013. Hinge-loss Markov random fields: Convex inference for structured prediction. In *Uncertainty in Artificial Intelligence (UAI)*.
- Barwise, J., and Cooper, R. 1981. Generalized quantifiers and natural language. *Linguistics and philosophy* 4(2):159–219.
- Brocheler, M.; Mihalkova, L.; and Getoor, L. 2012. Probabilistic similarity logic. *arXiv preprint arXiv:1203.3469*.

- Delgado, M.; Ruiz, M. D.; Sánchez, D.; and Vila, M. A. 2013. Fuzzy quantification: a state of the art. *Fuzzy Sets and Systems* 242:1–30.
- Huang, B.; Kimmig, A.; Getoor, L.; and Golbeck, J. 2012. Probabilistic soft logic for trust analysis in social networks. In *International Workshop on Statistical Relational AI*, 1–8.
- Keenan, D., and Westerstahl, D. 2011. Generalized quantifiers in linguistics and logic. *van Benthem and A. ter Meulen (eds.), Handbook of Logic and Language* 837–893.
- Yager, R. R. 1988. On ordered weighted averaging aggregation operators in multicriteria decision making. *Systems, Man and Cybernetics, IEEE Transactions on* 18(1):183–190.
- Zadeh, L. A. 1983. A computational approach to fuzzy quantifiers in natural languages. *Computers & Mathematics with Applications* 9(1):149–184.

Understanding the Complexity of Lifted Inference and Asymmetric Weighted Model Counting *

Eric Gribkoff

University of Washington
eagribko@cs.uw.edu

Guy Van den Broeck

UCLA, KU Leuven
guyvdb@cs.ucla.edu

Dan Suciu

University of Washington
suciu@cs.uw.edu

Abstract

We highlight our work on lifted inference for the asymmetric Weighted First-Order Model Counting problem (WFOMC), which counts the assignments that satisfy a given sentence in first-order logic. This work is at the intersection of probabilistic databases and statistical relational learning. First, we discuss how adding negation can lower the query complexity, and describe the essential element (resolution) necessary to extend a previous algorithm for positive queries to handle queries with negation. Second, we describe our novel dichotomy result for a non-trivial fragment of first-order CNF sentences with negation. Finally, we discuss directions for future work.

Overview

Our work is concerned with weighted first-order model counting (WFOMC), where we sum the weights of assignments that satisfy a sentence (or query) in finite-domain first-order logic. This reasoning task underlies efficient algorithms for probabilistic reasoning in AI, with statistical relational representations such as Markov logic (Van den Broeck et al. 2011; Gogate and Domingos 2011) and probabilistic programs (Van den Broeck, Meert, and Darwiche 2014). Moreover, WFOMC uncovers a deep connection between AI and database research, where query evaluation in *probabilistic databases* (PDBs) (Suciu et al. 2011) essentially considers the same task. A PDB defines a probability, or weight, for every possible world, and each database query is a sentence encoding a set of worlds, whose combined probability we want to compute. We refer to our setting as *asymmetric* WFOMC, because it allows the weights of each atom to be distinct. This subsumes the symmetric setting considered in AI, where probabilities are set per relation.

The WFOMC task captures query answering in probabilistic database. Take for example the database:

Prof(Arne) : 0.9	Prof(Charlie) : 0.1
Student(Bob) : 0.5	Student(Charlie) : 0.8
Advises(Arne, Bob) : 0.7	Advises(Bob, Charlie) : 0.1

*Extended Abstract of (Gribkoff, Van den Broeck, and Suciu 2014)

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

and the monotone DNF query:

$$Q = \exists x, \exists y, \text{Prof}(x) \wedge \text{Advises}(x, y) \wedge \text{Student}(y).$$

If we set $\Delta = Q$ and weight function w to map each literal to its probability in the database, then our query answer is:

$$\Pr(Q) = \text{WFOMC}(\Delta, w) = 0.9 \cdot 0.7 \cdot 0.5 = 0.315.$$

We refer to the general case above as *asymmetric* WFOMC, because it allows $w(\text{Prof}(\text{Anne}))$ to be different from $w(\text{Prof}(\text{Charlie}))$. We use *symmetric* WFOMC to refer to the special case where w simplifies into two weight functions $w^*, \bar{w}^*$ that map *predicates* to weights, instead of literals, that is:

$$w(\ell) = \begin{cases} w^*(P) & \text{when } \ell \text{ is of the form } P(c) \\ \bar{w}^*(P) & \text{when } \ell \text{ is of the form } \neg P(c) \end{cases}$$

Symmetric WFOMC no longer directly captures PDBs. Yet it can still encode many SRL models, including parfactor graphs (Poole 2003), Markov logic networks (MLNs) (Richardson and Domingos 2006) and probabilistic logic programs (De Raedt et al. 2008). We refer to (Van den Broeck, Meert, and Darwiche 2014) for the details.

We introduce a new WFOMC algorithm **Lift^R** that extends the algorithm of (Dalvi and Suciu 2012). This latter algorithm only supports queries without negation, but comes with strong theoretical guarantees in the form of a dichotomy theorem. **Lift^R** applies to general CNF queries, with arbitrary negation, by introducing a resolution operation to discover new prime implicates. As we show below, this step is crucial to proving completeness of the algorithm for a subclass of CNF sentences with negation. Our algorithm performs *lifted inference*, meaning that it exploits the relational structure of the query. Moreover, it performs domain-lifted inference, because it runs in time polynomial in the database size (Van den Broeck 2013). **Lift^R** is outlined in figure 1; see (Gribkoff, Van den Broeck, and Suciu 2014) for a full description

Negation Can Lower the Complexity

The presence of negations can lower a query's complexity, and **Lift^R** exploits this. To see this, consider the query

$$Q = \forall x \forall y (\text{Tweets}(x) \vee \neg \text{Follows}(x, y)) \\ \wedge \forall x \forall y (\text{Follows}(x, y) \vee \neg \text{Leader}(y)).$$

Algorithm Lift^R

Input: Ranked and shattered query Q Probabilistic DB with domain D Output: $\Pr(Q)$

- 1 Step 0: If Q is a single ground literal ℓ , return its probability $\Pr(\ell)$ in PDB
 - 2 Step 1: Write Q as a union-CNF: $Q = Q_1 \vee Q_2 \vee \dots \vee Q_m$
 - 3 Step 2: If $m > 1$ and Q can be partitioned into two sets $Q = Q' \vee Q''$ with disjoint relation symbols, return $1 - (1 - \Pr(Q_1)) \cdot (1 - \Pr(Q_2))$
 - 4 /* **Decomposable Disjunction** */
 - 5 Step 3: If Q cannot be partitioned, return $\sum_{s \subseteq [m]} \Pr(\bigwedge_{i \in s} Q_i)$
 - 6 /* **Inclusion/Exclusion** – perform cancellations before recursion */
 - 7 Step 4: Write Q in CNF: $Q = C_1 \wedge C_2 \wedge \dots \wedge C_k$
 - 8 Step 5: If $k > 1$, and Q can be partitioned into two sets $Q = Q' \wedge Q''$ with disjoint relation symbols, return $\Pr(Q_1) \cdot \Pr(Q_2)$
 - 9 /* **Decomposable Conjunction** */
 - 10 Step 6: If Q has a separator variable, return $\prod_{a \in D} \Pr(C_1[a/x_1] \wedge \dots \wedge C_k[a/x_k])$
 - 11 /* **Decomposable Universal Quantifier** */
 - 12 Otherwise **FAIL**
-

Figure 1: **Algorithm for Computing $\Pr(Q)$**

The query says that if x follows anyone then x tweets, and that everybody follows the leader¹.

Our goal is to compute the probability $\Pr(Q)$, knowing the probabilities of all (ground) atoms in the domain. We note that the two clauses are dependent (since both refer to the relation `Follows`), hence we cannot simply multiply their probabilities; in fact, we will see that if we remove all negations, then the resulting query is #P-hard; the algorithm described by (Dalvi and Suciu 2012) would immediately get stuck on this query. Instead, Lift^R takes advantage of the negation, by first computing the prime implicate

$$\forall x \text{ Tweets}(x) \vee \forall y \neg \text{Leader}(y),$$

which is a disconnected clause (the two literals use disjoint logical variables, x and y respectively). After applying distributivity we obtain:

$$\begin{aligned} Q &= (Q \wedge (\forall x \text{ Tweets}(x))) \vee (Q \wedge (\forall y \neg \text{Leader}(y))) \\ &= Q_1 \vee Q_2 \end{aligned}$$

and Lift^R applies the inclusion-exclusion formula:

$$\Pr(Q) = \Pr(Q_1) + \Pr(Q_2) - \Pr(Q_1 \wedge Q_2)$$

¹To see this, rewrite the query as $(\text{Follows}(x, y) \Rightarrow \text{Tweets}(x)) \wedge (\text{Leader}(y) \Rightarrow \text{Follows}(x, y))$.

After simplifying the three queries, they become:

$$Q_1 = \forall x \forall y (\text{Follows}(x, y) \vee \neg \text{Leader}(y)) \wedge \forall x (\text{Tweets}(x))$$

$$Q_2 = \forall x \forall y (\text{Tweets}(x) \vee \neg \text{Follows}(x, y)) \wedge \forall y (\neg \text{Leader}(y))$$

$$Q_1 \wedge Q_2 = \forall x (\text{Tweets}(x)) \wedge \forall y (\neg \text{Leader}(y))$$

The probability of Q_1 can now be obtained by multiplying the probabilities of its two clauses; same for the other two queries. As a consequence, our algorithm computes the probability $\Pr(Q)$ in polynomial time in the size of the domain and the probabilistic database.

If we remove all negations from Q and rename the predicates we get the following query:

$$h_1 = \forall x \forall y (R(x) \vee S(x, y)) \wedge (S(x, y) \vee T(y))$$

It was proven in (Dalvi and Suciu 2012) that computing the probability of the dual of h_1 is #P-hard in the size of the PDB. Thus, the query Q with negation is *easy*, while h_1 is hard, and Lift^R takes advantage of this by applying resolution to find the disconnected prime implicate $\forall x \forall y \text{ Tweets}(x) \vee \neg \text{Leader}(y)$.

A Dichotomy for Type-1 Queries

We prove a novel dichotomy for a subclass of CNF queries. First we review an earlier result, to put ours in perspective. In (Dalvi and Suciu 2012), an algorithm and dichotomy result for positive queries is shown. This result can be adapted to show that Lift^R restricted to monotone (i.e., negation-free) queries admits a dichotomy, and the following theorem continues to hold.

Theorem 1 *If algorithm Lift^R FAILS on a Monotone CNF query Q , then computing $\Pr(Q)$ is #P-hard.*

However, the inclusion of negations in our query language increases significantly the difficulty of analyzing query complexities. Our major technical result extends Theorem 1 to a class of CNF queries with negation.

Define a *Type-1* query to be a CNF formula where each clause has at most two variables denoted x, y , and each atom is of one of the following three kinds:

- Unary symbols $R_1(x), R_2(x), R_3(x), \dots$
- Binary symbols $S_1(x, y), S_2(x, y), \dots$
- Unary symbols $T_1(y), T_2(y), \dots$

or the negation of these symbols.

Our main result is:

Theorem 2 *For every Type-1 query Q , if algorithm Lift^R FAILS then computing $\Pr(Q)$ is #P-hard.*

The proof of this theorem required significant extension of the techniques used by (Dalvi and Suciu 2012) to prove Theorem 1. Details of our proof appear in (Gribkoff, Van den Broeck, and Suciu 2014).

Future Directions

Theorem 2 represents the first step towards proving the following conjecture: **Lift^R** provides a dichotomy for probabilistic queries on arbitrary universally-quantified CNF sentences where negation can be applied anywhere. Future work aims to prove this conjecture by expanding the existing dichotomy beyond Type-1 queries. We are also exploring new types of queries, and different assumptions about the probabilistic database, in particular symmetric probabilities, and how they affect our algorithm and dichotomy.

Acknowledgments

This work was partially supported by ONR grant #N00014-12-1-0423, NSF grants IIS-1115188 and IIS-1118122, and the Research Foundation-Flanders (FWO-Vlaanderen).

References

- Dalvi, N., and Suciu, D. 2012. The dichotomy of probabilistic inference for unions of conjunctive queries. *Journal of the ACM (JACM)* 59(6):30.
- De Raedt, L.; Frasconi, P.; Kersting, K.; and Muggleton, S., eds. 2008. *Probabilistic inductive logic programming: theory and applications*. Berlin, Heidelberg: Springer-Verlag.
- Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *Proceedings of UAI*, 256–265.
- Gribkoff, E.; Van den Broeck, G.; and Suciu, D. 2014. Understanding the Complexity of Lifted Inference and Asymmetric Weighted Model Counting. *ArXiv e-prints*. arXiv:1405.3250 [cs.AI].
- Poole, D. 2003. First-order probabilistic inference. In *IJCAI*, volume 3, 985–991. Citeseer.
- Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine learning* 62(1-2):107–136.
- Suciu, D.; Olteanu, D.; Ré, C.; and Koch, C. 2011. Probabilistic databases. *Synthesis Lectures on Data Management* 3(2):1–180.
- Van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, 2178–2185.
- Van den Broeck, G.; Meert, W.; and Darwiche, A. 2014. Skolemization for weighted first-order model counting. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*.
- Van den Broeck, G. 2013. *Lifted Inference and Learning in Statistical Relational Models*. Ph.D. Dissertation, Ph. D. Dissertation, KU Leuven.

Relational Logistic Regression: The Directed Analog of Markov Logic Networks

Seyed Mehran Kazemi, David Buchman,
Kristian Kersting, Sriraam Natarajan, and David Poole

cs.ubc.ca/~smkazemi/ cs.ubc.ca/~davidbuc/
www-ai.cs.uni-dortmund.de/PERSONAL/kersting.html
homes.soic.indiana.edu/natarasr/ cs.ubc.ca/~poole/

Abstract

Relational logistic regression (RLR) was presented at the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR-2014). RLR is the directed analogue of Markov logic networks. Whereas Markov logic networks define distributions in terms of weighted formulae, RLR defines conditional probabilities in terms of weighted formulae. They agree for the supervised learning case when all variables except a query leaf variable are observed. However, they are quite different in representing distributions. The KR-2014 paper defined the RLR formalism, defined canonical forms for RLR in terms of positive conjunctive (or disjunctive) formulae, indicated the class of conditional probability distributions that can and cannot be represented by RLR, and defined many other aggregators in terms of RLR. In this paper, we summarize these results and compare RLR to Markov logic networks.

Introduction

Relational probabilistic models (Getoor and Taskar 2007; de Raedt et al. 2008) or template-based models (Koller and Friedman 2009) represent the probabilistic dependencies among relations of individuals. In these models, individuals about which we have the same information are exchangeable (i.e. the individuals are treated identically when we have no evidence to distinguish them), and the probabilities are about relations among individuals and can be specified independently of actual individuals. Relational models can be directed or undirected.

Markov logic networks (MLNs) (Richardson and Domingos 2006; Domingos et al. 2008) are undirected models that define distributions in terms of weighted formulae. MLNs can also be used to define conditional probabilities when all the neighbours of a variable are observed. However, when weighted formulae are used to define conditional probabilities in a directed model, which we call relational logistic regression (RLR) (Kazemi et al. 2014), this results in quite different models.

Markov Logic Networks and Relational Logistic Regression

Markov logic networks (MLNs) (Richardson and Domingos 2006; Domingos et al. 2008) and relational logistic regression (RLR) (Kazemi et al. 2014) are defined in terms of weighted formulae. In MLNs the formulae are used to define joint probability distributions, whereas in RLR the formulae are used to define conditional probabilities. MLNs and RLR are identical when all but a query leaf variable are observed, but otherwise differ.

A **weighted formula** is a triple $\langle L, F, w \rangle$ where L is a set of logical variables, F is a formula where all of the free logical variables in F are in L , and w is a real-valued weight.

An MLN is a set of weighted formulae¹ where the probability of any world (complete assignment to all ground random variables) is proportional to the exponential of the sum of the weights of the instances of the formulae that are true in the world.

RLR is a form of aggregation, defining a conditional probability in terms of a set of weighted formula for a given relation (defining the parents of the relation). The conditional probability of the relation given its parents is proportional² to the exponential of the sum of the weights of the instances of the formulae that are true for that parent assignment and for the assignment to the relation.

Example 1 Suppose we have weighted formulae:

$$\begin{aligned} &\langle \{\}, q, \alpha_0 \rangle \\ &\langle \{x\}, q \wedge \neg r(x), \alpha_1 \rangle \\ &\langle \{x\}, q \wedge r(x), \alpha_2 \rangle \\ &\langle \{x\}, r(x), \alpha_3 \rangle \end{aligned}$$

Treating this as an MLN, if the truth value of $r(x)$ for every individual x is observed:

$$P(q \mid \text{obs}) = \text{sigmoid}(\alpha_0 + n_F \alpha_1 + n_T \alpha_2) \quad (1)$$

¹MLNs typically do not explicitly include the set of logical variables as part of the weighted formulae, but use the free variables in F . If one wanted to add an extra logical variable, x , one could conjoin $\text{true}(x)$ to F where true is a property that is true of all individuals.

²In MLNs there is a single normalizing constant, guaranteeing the probabilities of the worlds sum to 1. In RLR, normalization is done separately for each possible assignment to the parents.

where obs has $r(x)$ is true for n_T individuals, and false for n_F individuals out of a population of n individuals (so $n = n_T + n_F$), and $sigmoid(x) = 1/(1 + e^{-x})$.

Note that in the MLN, α_3 is not required for representing the conditional probability (because it cancels out), but can be used to affect $P(r(A_i))$, where A_i is an individual of x .

In Kazemi et al. (2014), the sigmoid, as in Equation (1), is used as the definition of RLR. (Kazemi et al. (2014) assumed all formulae were conjoined with $q \wedge$, and omitted $q \wedge$ from the formulae.) RLR only defines the conditional probability of q given each combination of assignments to the $r(x)$ (using Equation (1)); when not all $r(x)$ are observed, separate models of the probability of $r(x)$ are needed.

In summary: RLR uses the weighted formulae to define the conditional probabilities, and MLNs use the weighted formulae to define the joint probability distribution.

Example 2 Suppose people want to go to a party and the party is fun for them if they know at least one social person in the party. In this case, we have a parametrized random variable $funFor(x)$ which is a child of two parametrized random variables, $knows(x, y)$ and $social(y)$. The following weighted formulae can be used to model the dependence of the relation $funFor(x)$ on its parents:

$$\begin{aligned} &\langle \{x\}, funFor(x), -5 \rangle \\ &\langle \{x, y\}, funFor(x) \wedge knows(x, y) \wedge social(y), 10 \rangle \end{aligned}$$

RLR sums over the above weighted formulae and takes the *sigmoid*, resulting in:

$$P(funFor(x) = True | \Pi) = sigmoid(sum), \quad sum = -5 + 10n_T$$

where Π is the assignment of values to the parents, and n_T represents the number of individuals y for which $knows(x, y) \wedge social(y)$ is *True* for the given x . When $n_T = 0$, $sum < 0$ and the probability is close to 0; when $n_T > 0$, $sum > 0$ and the probability is close to 1.

Example 3 This example is similar to Example 1, but uses only positive conjunctions³, and also involves multiple logical variables of the same population.

$$\begin{aligned} &\langle \{ \}, q, \alpha_0 \rangle \\ &\langle \{x\}, q \wedge true(x), \alpha_1 \rangle \\ &\langle \{x\}, q \wedge r(x), \alpha_2 \rangle \\ &\langle \{x\}, true(x), \alpha_3 \rangle \\ &\langle \{x\}, r(x), \alpha_4 \rangle \\ &\langle \{x, y\}, q \wedge true(x) \wedge true(y), \alpha_5 \rangle \\ &\langle \{x, y\}, q \wedge r(x) \wedge true(y), \alpha_6 \rangle \\ &\langle \{x, y\}, q \wedge r(x) \wedge r(y), \alpha_7 \rangle \end{aligned}$$

In RLR and in MLNs, if the truth value of $r(x)$ for every individual x is observed:

$$P(q|obs) = sigmoid(\alpha_0 + n\alpha_1 + n_T\alpha_2 + n^2\alpha_5 + n_Tn\alpha_6 + n_T^2\alpha_7)$$

³Here $true(x)$ is true of every x . This notation is redundant. For the traditional MLN notation, one can remove the explicit set of logical variables and keep the $true(\cdot)$ relations. If you are happy with the explicit logical variables, you can remove the $true(\cdot)$ predicates. Removing both is incorrect. Keeping both is harmless. You could add formulae that involve negations or disjunctions, but they are redundant (Kazemi et al. 2014).

where obs has $r(x)$ true for n_T individuals out of a population of n . The use of two different logical variables of the same population in a formula (e.g. x and y in the last formula of this example) allows a sigmoid-of-a-quadratic dependency on the population sizes (n_T, n_F, n).

This example uses only positive conjunctions (but needs to have the “extra” logical variables). This is one of the canonical representations for RLR. Using the canonical forms for RLR, Kazemi et al. (2014) proved that in RLR, each conditional probability is the sigmoid of a polynomial of counts. Moreover, any conditional probability that is a sigmoid of polynomials of counts can be represented by RLR.

Example 4 Suppose we have a set of movies and the ratings people gave to these movies in a star-rating system. We define a movie to be “popular” if the average of its ratings is more than 3.5. In this case, we have a parametrized random variable $popular(m)$ which is a child of a parametrized random variable $rate(p, m)$. The following weighted formulae can approximate the conditional dependence of $popular(m)$ on its parent (the second weighted formula is repeated 5 times for different values of $i \in \{1, 2, 3, 4, 5\}$):

$$\begin{aligned} &\langle \{m\}, popular(m), -\epsilon \rangle \\ &\langle \{m, p\}, popular(m) \wedge rate(p, m) = i, i - 3.5 \rangle \end{aligned}$$

RLR sums over the above weighted formulae and takes the *sigmoid* resulting in:

$$P(popular(m) = True | \Pi) = sigmoid(-\epsilon + s - 3.5n)$$

where s denotes the sum of the ratings and n represents the number of ratings for this movie. The value inside the sigmoid is positive if $mean = \frac{s}{n} > 3.5$ and is negative otherwise (ϵ is used to ensure the the value inside the sigmoid is negative when $mean = 3.5$). This example demonstrates how the aggregator *mean* can be represented by RLR. It has been demonstrated in (Kazemi et al. 2014) that many other popular aggregators such as OR, AND, noisy-OR, noisy-AND, max, more-than- t trues, more-than- $t\%$ trues, and mode can be represented in terms of RLR.

Related Work

Logistic regression has been previously used in relational domains for learning purposes. Popescul et al. (2002) use inductive logic programming (ILP) to generate first-order rules for a target relation, create features by propositionalizing the rules, and then use logistic regression to learn a classifier based on these features.

Similar ideas to RLR have been used for discriminative learning in MLNs. Huynh and Mooney (2008) use an ILP technique to generate discriminative clauses for a target relation and then use logistic regression with L1-regularizer to learn the weights with automatic feature selection (automatic structure learning). They empirically demonstrate how their discriminative model outperforms MLNs when there is a single target relation. Their work can be considered as one approach for learning an RLR conditional probability for a relation when all its parents are observed.

Conclusion

The paper introducing relational logistic regression (RLR) (Kazemi et al. 2014):

- Defines RLR in terms of weighted formulae
- Investigates how conditional probabilities are defined based on the number of (tuples of) individuals for which each weighted formula holds
- Presents non-redundant canonical forms for the formulae, e.g., positive conjunctions
- Proves that a conditional dependency of child relation on population sizes of its parents can be represented by RLR if and only if it can be represented as the sigmoid of a polynomial of these population sizes
- Shows how other aggregators can be represented by RLR, including OR, AND, noisy-OR, noisy-AND, $\text{mean} > t$, max, more-than- t trues, more-than- $t\%$ trues, and mode.

References

- de Raedt, L.; Frasconi, P.; Kersting, K.; and Muggleton, S. 2008. *Probabilistic Inductive Logic Programming: Theory and Applications*. Springer-Verlag.
- Domingos, P.; Kok, S.; Lowd, D.; Poon, H.; and Richardson, M. 2008. *Markov Logic*. In Raedt, L.D., Frasconi, P., Kersting, K. and Muggleton, S., eds., *Probabilistic Inductive Logic Programming*. Springer.
- Getoor, L., and Taskar, B. 2007. *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.
- Huynh, T. N., and Mooney, R. J. 2008. Discriminative structure and parameter learning for markov logic networks. In *Proc. of the international conference on machine learning*.
- Kazemi, S. M.; Buchman, D.; Kersting, K.; Natarajan, S.; and Poole, D. 2014. Relational logistic regression. In *Proc. 14th International Conference on Principles of Knowledge Representation and Reasoning (KR-2014)*.
- Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA.
- Popescul, A.; Ungar, L. H.; Lawrence, S.; and Pennock, D. M. 2002. Towards structural logistic regression: Combining relational and statistical learning. In *KDD Workshop on Multi-Relational Data Mining*.
- Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 42:107–136.

Classification from One Class of Examples for Relational Domains

Tushar Khot and **Sriraam Natarajan*** and **Jude Shavlik**

University of Wisconsin-Madison, {tushar,shavlik}@cs.wisc.edu

* Indiana University, natarasr@indiana.edu

Abstract

One-class classification approaches have been proposed in the literature to learn classifiers from examples of only one class. But these approaches are not directly applicable to relational domains due to their reliance on a feature vector or a distance measure. We propose a non-parametric relational one-class classification approach based on first-order trees. We learn a tree-based distance measure that iteratively introduces new relational features to differentiate relational examples. We update the distance measure so as to maximize the one-class classification performance of our model. We also relate our model definition to existing work on probabilistic combination functions and density estimation. We experimentally show that our approach can discover relevant features and outperform three baseline approaches.

1 Introduction

Traditional classification approaches rely on labeled examples from all classes to learn a model. But in several real world problems, labeled examples are available for only one class. For example, in information extraction from text, a common annotation approach is to mark the true instances in the text (UzZaman et al. 2012; Kim et al. 2009). Since human annotation in these tasks can be quite costly, the negative examples are not annotated or in rare cases very few negative examples are annotated. For instance, presidents of countries will be marked in an article for training, there is no explicit mention of what the negative examples of presidents are. One possible approach to handle this problem is to assume that the unmarked instances are negative examples for the task at hand. Since the number of unmarked examples can be very large, this can increase the learning time for standard classification approaches. A bigger problem is that this can possibly lead to class imbalance where the number of instances of the negative class can dominate the positive instances. Finally, this simplistic assumption could be incorrect as all instances of positive examples may not be annotated (e.g., all mentions of presidents need not be annotated).

Consequently, a lot of research has focused on developing one-class classification (OCC) methods (Khan and Madden 2009) that directly learn using examples from only one class

or using very few examples of the other class. For example, Tax and Duin (1999) fit a hypersphere with the smallest radius around the labeled examples. All the examples inside this hypersphere are assumed to belong to the labeled class. Anomaly/Novelty detection tasks can also be viewed as an OCC task where many examples are available for the normal class and few or no examples of the anomalous class are provided. One-class classification can also be viewed as a specific case of semi-supervised learning (Zhu and Goldberg 2009) with the labeled data being provided for only one class. Hence methods such as clustering and nearest-neighbors have also been used previously for OCC (de Ridder, Tax, and Duin 1998). These methods, in general, rely on a feature space representation (e.g., to fit a hypersphere) or distance measure (e.g., for clustering) to perform OCC.

While successful, these methods rely on a propositional representation of the data (i.e., as a flat feature vector). Data in the real world is inherently relational i.e., they consist of inter-related objects and dependencies. Such structured data cannot be easily represented using a fixed-length feature vector and doing so can result in loss of possible information (Jensen and Neville 2002). Recently, several algorithms have been developed that exploit the rich structure while handling uncertainty. Collectively called Statistical Relational Learning (Getoor and Taskar 2007) (SRL), these algorithms combine rich representations such as first-order or relational logic with probabilistic models.

While representations such as first-order logic provide a lossless way to model structured data, they introduce the additional complexity of having unbounded features associated with each example. E.g., consider the task of predicting the blood type of a person based on his lineage. We can potentially introduce a feature for the bloodtype of every ancestor of a person. As a result, standard distance metrics such as Euclidean norms cannot be naively used for relational problems. One potential approach for relational OCC is performing propositionalization followed by using the standard OCC techniques. Standard propositionalization may pick irrelevant features and as a result OCC techniques on these propositionalized datasets can underperform, as we show empirically. The propositionalization approach needs to select the features so as to optimize the one-class classification performance which can be computationally expensive with standard OCC methods.

We propose a non-parametric approach for OCC using a tree-based distance measure between relational examples which is learned by directly optimizing the OCC performance. We define this measure using the path similarity in relational decision trees. We combine the distances from multiple incrementally learned trees and labeled examples to calculate the probability of an example belonging to the labeled class. We present three interpretations of our approach: 1) based on combining rules for probability distributions (Natarajan et al. 2008), 2) based on using our distance with Kernel Density Estimation (Bishop 2006) and 3) based on using a similarity kernel with Support Vector Data Description (SVDD). Using this model definition, we derive a scoring method that allows for learning the trees so as to maximize the likelihood of the labeled examples. Although we use unlabeled examples in our approach to discover the relevant features, we do not require them to be specified (which we explain in Section 3). Another interesting interpretation of our approach is to view the method as a *feature selection* strategy for standard OCC methods. From this perspective, our approach can be seen as a way to propositionalize relational data for OCC methods.

We make several key contributions in this work: 1) We present a new distance metric for relational examples that can be learned incrementally. 2) We propose a relational one-class classification approach using this distance metric. 3) We relate and place this work in the context of existing work on combination functions, existing work on density estimation and OCC. 4) We show how we can perform feature selection for propositional OCC methods by evaluating our trees. 5) Finally, we demonstrate over multiple domains that our approach can learn more accurate models than standard relational learning and OCC methods.

2 Related Work

One-class classification (OCC) problem was first described by Moya et al. (1996) for an automatic target recognition task using neural networks. Following their work, Scholkopf et al. (2000) learn a Support Vector Machine (SVM) to separate the labeled examples from the origin. Tax and Duin (1999), on the other hand, learn a hypersphere with the smallest radius that encompasses the labeled examples. They have also shown that using the Gaussian kernel with a normalized dataset gives solutions comparable to the SVM approach. DeComite et al. (1999), on the other hand, use modified C4.5 to perform one-class classification. All these approaches rely on a propositional representation.

Anomaly/outlier detection methods can also be viewed as a one-class classification approach. Although there have been many outlier detection methods proposed in literature (Chandola, Banerjee, and Kumar 2009), outlier detection approaches generally assume that outliers are far and isolated (Liu, Ting, and Zhou 2012). This is not true for the one-class classification problem where the labeled examples can be clustered and close to other unlabeled examples. Semi-supervised learning approaches can also be used for OCC as shown by Ridder et al. (1998). Our approach can be viewed as semi-supervised relational learning as it leverages unlabeled examples for learning the relational distance function.

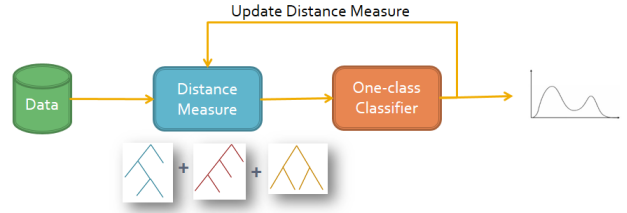


Figure 1: Relational one-class classification system design

One of the earliest works on relational one-class classification (Muggleton 1997) learns the smallest first-order hypothesis that can cover the labeled examples. Since this work only performed Boolean predictions, it was extended to perform relational density estimation (Cussens 1998). This approach relied on calculating proportions of examples captured by every hypotheses which can be prohibitively expensive for large domains.

Multiple propositionalization techniques and distance measures have been proposed for relational data which can potentially be used for OCC. One simple unsupervised approach to generate propositional features is by creating Boolean features corresponding to random first-order logic clauses (Anderson and Pfahringer 2008). Relational Instance-based Learning (RIBL) (Horváth, Wrobel, and Bohnebeck 2001) uses similarity between local neighborhoods to calculate the similarity between examples. Such unsupervised approaches may not capture long range information while we leverage the labeled examples to guide our approach. kFoil (Landwehr et al. 2010) uses a *dynamic* approach to learn clauses to propositionalize relational examples for SVMs. Each clause corresponds to one Boolean feature and is scored based on the improvement in the SVM learned from the propositionalized data. Rather than learning a new model for every candidate structure, we compute the error reduction on the fly locally.

3 Relational OCC

The high-level overview of our approach to Relational One-Class Classification (RelOCC) is shown in Figure 1. Our approach consists of the following steps:

1. Learn a first-order tree for calculating relational distances. (Section 3.1)
2. Use the distance function to perform OCC. (Section 3.2)
3. Learn the next distance tree to improve the classification. (Section 3.3)
4. Go to Step 2.

Table 1 presents the terminology used throughout the paper. We will now describe the individual steps of our approach.

3.1 Tree-based distance

Inspired by semantic similarity measures (D’Amato, Staab, and Fanizzi 2008), we propose a distance measure based on the paths taken by examples in a relational decision tree. For

Term	Description
x, x_i, y	Relational examples
$\{x_1, x_2, \dots, x_l\}$	Labeled examples
$lca(x_1, x_2)$	Lowest Common Ancestor of the leaves reached by x_1 and x_2
$depth(n)$	Depth of node n . $depth(Root) = 0$.
$d_i(x, y)$	Distance between x and y based on i^{th} tree
$D(x, y)$	Distance between x and y based on all trees
$P(y \in \mathcal{L})$	Probability of y being in labeled class
α_j	Weight of labeled example x_j
β_i	Weight of i^{th} tree
$I(condition)$	Indicator function that returns 1 if condition is true, 0 otherwise.
$P^t(y \notin \mathcal{L})$	$P(y \notin \mathcal{L})$ using t trees

Table 1: Description of terms used in this paper.

instance, consider two examples that reach the same node, n in a decision tree before taking separate paths down the tree i.e., n is the lowest common ancestor (LCA) of these examples. If the node n is at depth of three, the two examples share *at least* three common attribute values. If n is at depth of one, they share at least one attribute value. Intuitively, the two examples in the first case are more likely to be similar than in second case. Thus, our distance measure is inversely proportional to the depth of LCA of two example paths.

$$d(x_1, x_2) = \begin{cases} 0 & lca(x_1, x_2) \text{ is a leaf} \\ e^{-\lambda \cdot depth(lca(x_1, x_2))} & \text{otherwise} \end{cases}$$

Examples that differ at the root node i.e. have no common attribute will have a distance of 1. A major feature of this distance definition is that it can be used on any off-the-shelf tree learner (such as TILDE (Blockeel and Raedt 1998)). The key difference from a first-order decision tree is that, in our formulation, there are no values associated with the leaves since all we are interested in are the paths that determine the distance between examples. The parameter λ (set to 0.5 in experiments) ensures that the distance value decreases gradually as the depth increases.

Properties It can be easily shown that this distance function satisfies the following distance metric properties:

- $d(x_1, x_2) \geq 0$ (non-negative)
- $d(x_1, x_2) = d(x_2, x_1)$ (commutative)
- x_1 and x_2 are identical $\Rightarrow d(x_1, x_2) = 0^1$ (equality)
- $d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$. (triangle property)
Proof: If $l_{ij} = depth(lca(x_i, x_j))$, $l_{13} \geq \min(l_{12}, l_{23})$. WLOG, $l_{12} = \min(l_{12}, l_{23})$.
 $l_{13} \geq l_{12} \Rightarrow e^{-l_{13}} \leq e^{-l_{12}} \Rightarrow e^{-l_{13}} \leq e^{-l_{12}} + e^{-l_{23}} \Rightarrow d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$. This can also be proven if x_1 and x_3 reach the same leaf node i.e., $d(x_1, x_3) = 0$. ■

Using just one tree may not suffice when dealing with potentially infinite dimensions in relational data. A simple next step would be to learn multiple relational trees. However this step introduces an additional complexity –the need for a way

¹The reverse condition does not hold.

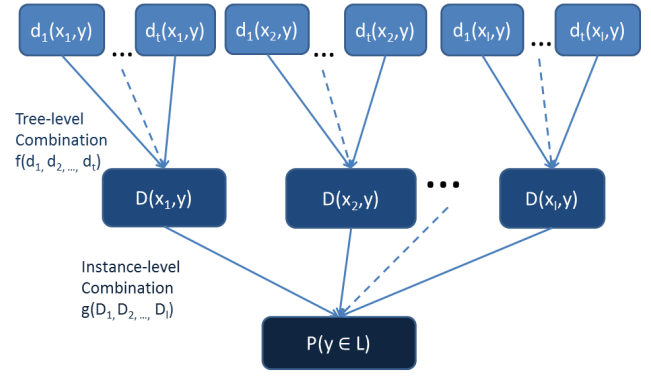


Figure 2: Density Estimation using Distance measure

to combine these trees without increasing the complexity of the already complex problem. Our solution is inspired by density estimation techniques that we present next.

3.2 Density Estimation Model

We now define a model (shown in Figure 2) to combine distances from multiple trees and then distances from multiple examples to perform relational one-class classification. Recall that to calculate the probability estimate of an example y to belong to the labeled class, we use the distances between y and the labeled examples. We denote the distance between the current example y and a labeled example x from the i^{th} tree as $d_i(x, y)$. $D(x, y)$ combines the tree distances to return the overall distance between x and y . We finally compute the density estimate $P(y \in \mathcal{L})$ using the distances from all the labeled examples $\{x_1, x_2, \dots, x_l\}$. The combination function that we use in both levels is weighted mean:

$$D(x_j, y) = \sum_i \beta_i d_i(x_j, y) \text{ s.t. } \sum_i \beta_i = 1, \beta_i \geq 0 \quad (1)$$

$$P(y \notin \mathcal{L}) = \sum_j \alpha_j D(x_j, y) \text{ s.t. } \sum_j \alpha_j = 1, \alpha_j \geq 0 \quad (2)$$

One of the key advantage of this function is that it allows for efficient learning of trees as we show in Section 3.3. Also, this approach is closely related to existing research in one-class classification thus making it theoretically well grounded. We first present the learning of the trees followed by the relation to existing methods.

3.3 Model Learning

Tree Learning As shown in Figure 1, we update the distance measure by iteratively adding to the set of trees. Assume that we have learned t trees already and are now learning the $(t+1)^{th}$ one. We can use any off-the-shelf tree learner and we employ the method described by TILDE tree learning (Blockeel and Raedt 1998). Since our distance metric only relies on the LCA position, we can make local scoring decisions at every node. If a pair of examples are already split by the tree, any further splits at lower levels will have no impact on the distance between these examples i.e., adding

a node to the tree only affects the distance between the pair of examples that are split at that node. Specifically, splitting example x_i and x_j at node n , increases their distance $d_{t+1}(x_i, x_j)$ from zero to $\Delta_{t+1}(x_i, x_j) = e^{-\text{depth}(n)}$.

We use a modified splitting criterion which is the squared error over the examples i.e., $\sum_y [I(y \notin \mathcal{L}) - P(y \notin \mathcal{L})]^2$. $I(y \notin \mathcal{L})$ is the indicator function which returns 1 if y is an unlabeled example. Hereafter we use $I(y)$ to compactly represent $I(y \notin \mathcal{L})$. As can be seen here, our scoring function does rely on using the unlabeled examples in the dataset. Even if unlabeled examples are not provided, these can be generated by using constants of matching types (e.g. $\{\text{friends}(x, y) | \forall x, y \in \text{People}, \text{friends}(x, y) \notin \mathcal{L}\}$). Relying on only the labeled examples for learning a distance function will result in trivial distances of $d(x, y) = 0$.

Consider $\mathbf{x}$ to be the set of examples that reach node n . $\mathbf{x}_l$ and $\mathbf{x}_r$ are the set of examples that take the left and right branch respectively. Since introducing the node has no impact on the distances (and in turn probabilities) for examples not reaching node n , the squared error is

$$\begin{aligned} & \sum_{y \in \mathbf{x}} [I(y \notin \mathcal{L}) - P(y \notin \mathcal{L})]^2 \\ &= \sum_y [I(y) - \sum_j \alpha_j \{\sum_i \beta_i d_i(x_j, y) + \beta_{t+1} \Delta_{t+1}(x_j, y)\}]^2 \\ &= \sum_y [I(y) - P^t(y \notin \mathcal{L}) - \sum_j \alpha_j \beta_{t+1} \Delta_{t+1}(x_j, y)]^2 \end{aligned}$$

At every node in the tree, we minimize this squared error. If y is close to the labeled examples ($P^t(y \notin \mathcal{L}) \approx 0$) for an unlabeled example ($I(y) = 1$), increase in the distances to the labeled examples will reduce the squared error. If y is already spread apart from the labeled examples (i.e. $P^t(y \notin \mathcal{L}) \approx 1$), the distances to the labeled examples will not be increased any further. As a result, minimizing the squared error *introduces new features* to spread out the examples and *prevents redundant features* being added.

Since introducing node n only increases the distance between the pair of examples split at the node, $\Delta_{t+1}(x_j, y) \neq 0$ for $x_j \in \mathbf{x}_l, y \in \mathbf{x}_r$ or $x_j \in \mathbf{x}_r, y \in \mathbf{x}_l$. Hence the splitting criterion can be modified to minimizing

$$\begin{aligned} & \sum_{y \in \mathbf{x}_r} [I(y) - P^t(y \notin \mathcal{L}) - \sum_{x_j \in \mathbf{x}_l} \alpha_j \beta_{t+1} \Delta_{t+1}(x_j, y)]^2 \\ &+ \sum_{y \in \mathbf{x}_l} [I(y) - P^t(y \notin \mathcal{L}) - \sum_{x_j \in \mathbf{x}_r} \alpha_j \beta_{t+1} \Delta_{t+1}(x_j, y)]^2 \end{aligned}$$

We use a greedy search procedure for learning the tree where we pick the feature whose split reduces this squared error. The key insight is that the use of the mean squared error along with our relational path based distance function allows us to efficiently make these local scoring decisions. Note that our approach is non-parametric as the only parameter is the number of trees that increases as more data is obtained. Thus we can potentially update the distances in online fashion.

Weight Learning We set uniform weights on the examples and trees initially. After learning each tree, we update the weights α and β for weighted mean function. We use a

co-ordinate gradient descent approach where we iteratively update α and β to minimize the squared error. Since there is no closed form solution, we update the weights only after learning an entire tree instead of updating after learning every node in the tree. The gradient of error w.r.t. α is

$$\begin{aligned} & \frac{\partial}{\partial \alpha_j} \sum_y [I(y) - \sum_i \alpha_j \beta_i d_i(x_j, y)]^2 \\ &= -2 \sum_y [I(y) - P(y \notin \mathcal{L})] \sum_i \beta_i d_i(x_j, y) \end{aligned}$$

and the gradient w.r.t. β is

$$-2 \sum_y [I(y) - P(y \notin \mathcal{L})] \sum_j \alpha_j d_j(x_j, y)$$

To ensure that weights are always greater than 0 and sum to 1, we project them as described by Natarajan et al. (2008).

3.4 Model Interpretations

We now place our work in context to existing work on learning parameters in SRL and propositional OCC methods.

Combining Rules: Since $d_i(x, y) \leq 1$, it can be interpreted as the probability of example x being not equal to y based on the i^{th} tree, i.e., $P_i(x \neq y) = d_i(x, y)$. Similarly $D(x, y)$ can be viewed as overall probability of example x being not equal to y , i.e., $P(x \neq y) = D(x, y)$. This can be viewed as using two-level combining rules (Natarajan et al. 2008) in directed SRL models with the weighted mean combination function. At first level, we combine probabilities from the set of trees (Eqn. 1). At second level, we combine probabilities based on individual examples (Eqn. 2).

Kernel Density Estimation: An alternate interpretation of our model is that of using Kernel Density Estimation (Bishop 2006) with a triangular kernel² and bandwidth 1.

$$\begin{aligned} 1 - \sum_j \alpha_j \sum_i \beta_i d_i(x_j, y) &= \sum_j \frac{1}{n} (1 - \sum_i \alpha_j \beta_i d_i(x_j, y)) \\ &= \sum_j \frac{1}{n} |1 - D'(x_j, y)| = \sum_j \frac{1}{nh} K\left(\frac{D'(x_j, y)}{h}\right) \text{ for } h=1 \end{aligned}$$

where $D'(x_j, y) = \sum_i \alpha_j \beta_i d_i(x_j, y)$. Since the term $|x_j - y|$ cannot be defined easily for relational examples, we use D' function to represent the absolute distance needed by the kernel density estimator. The value of D' is always below 1 and as a result is within the bounds of the triangular kernel.

SVDD: SVDD (Tax and Duin 1999) approach learns a hypersphere around the labeled one-class examples. An example z is classified as belonging to the labeled class if $(z \cdot z) - 2 \sum_i \alpha_i (z \cdot x_i) + \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \leq R^2$. Since the classification only depends on the dot product, SVDD also uses kernel functions. Using the function $1 - D$ in our case as the kernel functions³ changes the function used for classification to $1 - D(z, z) - 2 \sum_i \alpha_i (1 - D(z, x_i)) + \sum_{i,j} \alpha_i \alpha_j (1 - D(x_i, x_j)) \leq R^2$. Since $D(z, z) = 0$, we can rewrite the decision boundary as $1 - \sum_i \alpha_i D(z, x_i) \geq$

² $K(x, y) = (1 - |x - y|) \cdot I(|x - y| < 1)$

³ this is possible as kernel functions are similarity functions

$1 - \sum_i \alpha_i D(x_k, x_i)^4$ where x_k is any labeled example. We use the left hand side of the equation as the probability estimate for z belonging to the labeled class. Our definition allows to return probabilistic estimates, but we can now also use $1 - \sum_i \alpha_i D(x_k, x_i)$ as a decision boundary to predict the example class. Since we optimize a different function, it is not necessary for all examples x_k to result in the same boundary. But we can approximate it by using the average distance to all labeled examples.

Feature Selection for OCC: It is also possible to view our proposed approach as a feature selection strategy for standard one-class classification methods. More specifically, we can convert every path from root to leaf into a Boolean feature since these paths can be represented as a horn clause. Since the resulting feature is Boolean, doing this for every path will yield a propositionalized data set that can then employ standard OCC methods such as SVM-OCC. We call this approach SVMOCC-Rel and show the importance of this feature selection strategy in our experiments. Propositionalization of relational data can possibly result in infinite features as a walk through the relational graph can lead to several resulting features. For instance, when reasoning about a particular genetic disposition, it is possible to use generations of ancestors' information. Our method on the other hand, can be seen as selecting these relational features (i.e., identifying the most important set of ancestors) for a propositional classifier making it possible for the propositional methods to work with relational data more seamlessly.

Implementation details: Our approach begins with the distance of 0 between every pair of examples. During every tree learning step, we use only a subsample of examples for efficiency which has shown to improve performance in highly skewed datasets (Chan and Stolfo 1998). We further subsample the examples based on their current predictions. Based on active learning (Settles 2012) techniques, we pick unlabeled examples based on their proximity to the decision boundary as well as the misclassified examples. For the weight learning step, we use a step length of $\eta = 0.001$.

4 Experiments

We evaluate the following different algorithms:

- **RelOCC:** This approach is our relational OCC method using the predictions from the combination function.
- **RND:** We modify the tree learner to randomly pick features at every node. To ensure a strong baseline, we do not pick any feature that results in more than 95% examples going to one branch.
- **O-Rel:** This refers to the strategy described above for using SVM-OCC with features generated from RelOCC.
- **O-Rnd:** This refers to SVM-OCC with features from RND trees (i.e, random features).
- **SVM:** Instead of using only labeled examples, we use all the examples for training a standard SVM on the propositionalized data. We use *S-Rel* and *S-Rnd* to denote the SVM learned from RelOCC and RND trees respectively.

⁴after expanding R^2

- **RPT:** As we are learning conditional discriminative models, we learn relational probability trees (Neville et al. 2003) for the target class as a relational baseline.

We use area under the Precision-Recall curve (AUC-PR) and accuracy to compare our approaches. These datasets have extremely skewed distributions, so using all the examples for testing can achieve a very high precision when predicting the most common label. We use a subset of examples with a reduced skew (twice the negatives to positives) for testing to avoid this issue. The same subset is used for all approaches within each domain. Since one-class SVM methods return only binary predictions we compare the accuracies, while we compare the AUC-PR values for the relational approaches.

We aim to explicitly evaluate the following questions: (Q1) Does RelOCC compare favorably to the baseline relational learning methods for the OCC tasks? and (Q2) Do the features extracted out of RelOCC help the propositional one-class classifier (O-Rel)?

4.1 UW-CSE

UW-CSE dataset (Richardson and Domingos 2006) is a standard SRL dataset developed at University of Washington. The goal is to predict the `advisedBy` relationship between a student and a professor. The data set contains information about professors, students and courses from five different sub-areas of computer science. It includes relations such as `professor`, `student`, `publication`, `advisedBy`, `hasPosition` etc. Since the primary motivation of our approach is one-class classification where all the positive examples are not marked, we use only a subset of positive examples and assume they were the only ones marked by annotators. We pick 20%, 40% and 60% of the labeled examples for training and use all the positives for testing. The results for five-fold cross-validation are shown in Table 2 and Table 3.

% marked	RelOCC	RND	RPT
20%	0.93	0.87	0.76
40%	0.93	0.81	0.81
60%	0.92	0.83	0.88

Table 2: AUC-PR on UW dataset. Bold shows the best approach.

% marked	RelOCC	O-Rel	O-Rnd	S-Rel	S-Rnd
20%	89.75	77.39	75.24	60.56	60.58
40%	89.97	75.55	74.94	60.56	60.56
60%	89.60	76.02	74.43	60.56	60.56

Table 3: Accuracy on UW dataset

As can be seen from the AUC-PR values, our approach outperforms both OCC with random feature selection and discriminative learning using RPTs thus answering Q1 affirmatively. Our approach also has a higher accuracy than both using SVM-OCC and standard SVMs on the propositionalized dataset. The training dataset is highly skewed and

as a result basic SVM approaches (S-Rel and S-Rnd) predict the most common label. Thus Q2 can also be answered affirmatively in that using our method for feature selection helps SVM over the other feature selection methods. For all other datasets, S-Rel and S-Rnd show similar behavior where they predict all examples belonging to the negative class and hence we do not show them in other data sets.

4.2 IMDB

IMDB is another popular SRL dataset (Mihalkova and Mooney 2007) containing information about actors, directors and movies obtained from imdb.com. We focused on the task of predicting the `workedUnder` relation in this dataset. We performed five-fold cross-validation and the results are presented in Table 4 and 5. The results are similar to UW-CSE where RelOCC outperforms the SVM-based approaches. But for this dataset, RPT are also able to learn with few positive examples and have similar performance to RelOCC. Both these approaches still outperform RND.

% marked	RelOCC	RND	RPT
20%	0.98	0.80	0.99
40%	0.98	0.88	0.97
60%	0.97	0.91	0.97

Table 4: AUC-PR on IMDB dataset.

% marked	RelOCC	RND	O-Rel	O-Rnd
20%	93.88	56.76	81.15	84.10
40%	94.68	60.49	81.27	78.90
60%	94.36	62.14	78.69	77.53

Table 5: Accuracy on IMDB dataset

4.3 NFL

One of the primary motivations of our work was to learn models for information extraction data sets with only positive labels. To this end, we evaluate our approach on the task of extracting winners of National Football League (NFL) games from sports articles. The annotations only provided few positive examples. We use the Stanford NLP toolkit ⁵ to convert the text into NLP structures such as parse trees and dependency graphs. Since this dataset only has few annotated positives, we do not drop any labeled examples in our experiments. The results for ten-fold cross-validation on this task are shown in Table 6 and 7.

RelOCC outperforms the other relational approaches thus answering Q1. But as compared to the SVM-based approaches, using the RelOCC features for SVM-OCC (O-Rel) does as well as the relational OCC approach in terms of the accuracy. This shows clearly that our proposed approach serves as a good feature selection method as well (answering Q2). Since this dataset has a large number of possible features, random feature selection is not able to pick the relevant attributes and both RND and O-Rnd are much worse.

⁵<http://nlp.stanford.edu/software/corenlp.shtml>

RelOCC	RND	RPT	RelOCC	RND	O-Rel	O-Rnd
0.63	0.35	0.59	69.95	34.92	70.22	47.70

Table 6: AUC-PR (NFL) Table 7: Accuracy results (NFL)

4.4 Heart

To evaluate the quality of feature selection, we use the Heart dataset ⁶ which is a multivariate data set with 13 attributes. The task is to predict the presence of heart disease in patients. We performed four-fold cross-validation and show the results in Tables 8 and 9.

% marked	RelOCC	RND	RPT
20%	0.47	0.38	0.39
40%	0.44	0.37	0.36
60%	0.43	0.38	0.43

Table 8: AUC-PR on Heart dataset

While RelOCC is better than both RND and RPT, using SVM-OCC with RelOCC as a feature selection procedure does much better than using the RelOCC model. Since this dataset is originally a propositional dataset, the complex interactions captured by the distance function in RelOCC are not required and can possibly overfit the data. Hence with fewer examples marked (20%), RelOCC does as well as SVM-OCC and better than all the other approaches. The results here show that SVM benefits from the feature selection as it is better than both O-Rnd and the baseline SVM methods answering Q2 affirmatively.

% marked	RelOCC	RND	O-Rel	O-Rnd
20%	63.85	38.71	62.73	50.64
40%	50.34	37.67	62.59	49.50
60%	46.38	39.31	60.55	51.53

Table 9: Accuracy on Heart dataset

5 Conclusion

We presented a non-parametric approach for one-class classification from relational data. We defined a new distance metric based of first-order decision forest and a density estimation model using the distance metric. We can efficiently update the distance metric to improve the classifier’s performance. Our approach as a side effect introduces novel and relevant features which can also be used to augment propositional OCC methods. The next step is to apply our approach to other information extraction tasks such as TempEval (Uz-Zaman et al. 2012) and BioNLP (Kim et al. 2009). Integrating other kernels and combination functions into our method might help in further improving the performance.

⁶<http://archive.ics.uci.edu/ml/datasets/Heart+Disease>

6 Acknowledgments

The authors gratefully acknowledge support of the DARPA DEFT Program under the Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0039. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, ARO, or the US government.

References

- Anderson, G., and Pfahringer, B. 2008. Exploiting propositionalization based on random relational rules for semi-supervised learning. In *PAKDD*, 494–502.
- Bishop, C. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.
- Blockeel, H., and Raedt, L. D. 1998. Top-down induction of first-order logical decision trees. *Artificial Intelligence* 101:285–297.
- Chan, P., and Stolfo, S. 1998. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *KDD*.
- Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly detection: A survey. *ACM Computing Surveys* 41(3):15:1–15:58.
- Cussens, J. 1998. Using prior probabilities and density estimation for relational classification. In *ILP*.
- D’Amato, C.; Staab, S.; and Fanizzi, N. 2008. On the influence of description logics ontologies on conceptual similarity. In *EKA*.
- de Ridder, D.; Tax, D.; and Duin, R. 1998. An experimental comparison of one-class classification methods. In *Conference of the Advanced School for Computing and Imaging, Delft*.
- DeComité, F.; Denis, F.; Gilleron, R.; and Letouzey, F. 1999. Positive and unlabeled examples help learning. In *ALT*.
- Getoor, L., and Taskar, B., eds. 2007. *Introduction to Statistical Relational Learning*. MIT Press.
- Horváth, T.; Wrobel, S.; and Bohnenbeck, U. 2001. Relational instance-based learning with lists and terms. *Machine Learning* 43:53–80.
- Jensen, D., and Neville, J. 2002. Linkage and autocorrelation cause feature selection bias in relational learning. In *ICML*.
- Khan, S., and Madden, M. G. 2009. A survey of recent trends in one class classification. In *Irish conference on Artificial Intelligence and Cognitive Science*.
- Kim, J.; Ohta, T.; Pyysalo, S.; Kano, Y.; and Tsujii, J. 2009. Overview of BioNLP’09 shared task on event extraction. In *BioNLP*.
- Landwehr, N.; Passerini, A.; De Raedt, L.; and Frasconi, P. 2010. Fast learning of relational kernels. *Machine Learning* 78:305–342.
- Liu, F.; Ting, K.; and Zhou, Z. 2012. Isolation-based anomaly detection. *TKDD* 6(1):3.
- Mihalkova, L., and Mooney, R. 2007. Bottom-up learning of Markov logic network structure. In *ICML*.
- Moya, M., and Hush, D. 1996. Network constraints and multi-objective optimization for one-class classification. *Neural Networks* 9(3):463–474.
- Muggleton, S. 1997. Learning from positive data. In *ILP*.
- Natarajan, S.; Tadepalli, P.; Dietterich, T.; and Fern, A. 2008. Learning first-order probabilistic models with combining rules. *Annals of Mathematics and AI* 54(1-3):223–256.
- Neville, J.; Jensen, D.; Friedland, L.; and Hay, M. 2003. Learning relational probability trees. In *KDD*.
- Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.
- Schölkopf, B.; Williamson, R.; Smola, A.; Shawe-Taylor, J.; and Platt, J. 2000. Support vector method for novelty detection. In *NIPS*.
- Settles, B. 2012. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool.
- Tax, D., and Duin, R. 1999. Support vector domain description. *Pattern Recognition Letters* 20:1191–1199.
- UzZaman, N.; Llorens, H.; Allen, J.; Derczynski, L.; Verhaagen, M.; and Pustejovsky, J. 2012. Tempeval-3: Evaluating events, time expressions, and temporal relations. *CoRR* abs/1206.5333.
- Zhu, X., and Goldberg, A. 2009. *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Applying Marginal MAP Search to Probabilistic Conformant Planning: Initial Results

Junkyu Lee and Rina Dechter
University of California, Irvine
junkyu@uci.edu, dechter@ics.uci.edu

Radu Marinescu
IMB Research, Ireland
radu.marinescu@ie.ibm.com

Abstract

In this position paper, we present our current progress in applying marginal MAP algorithms for solving the conformant planning problems. Conformant planning problem is formulated as probabilistic inference in graphical models compiled from relational PPDDL domains. The translation from PPDDL into Dynamic Bayesian Network is developed by mapping the SAT encoding of the ground PPDDL into factored representation. We experimented with recently developed AND/OR branch and bound search algorithms for marginal MAP over instances from the international planning competition domains, and we show that several domains were solved efficiently.

Introduction

We present our current progress in applying a general purpose Marginal MAP search algorithms (Marinescu, Dechter, and Ihler 2014) for solving conformant planning problems by encoding their relational formulae into DBN (Dynamic Bayesian Network) up to fixed time horizon, into a factored representation.

The probabilistic planning problem $P = \langle S, I, G, A \rangle$ we address is based on the probabilistic extension to sequential STRIPS with negation and conditional effects, where the S is the set of states grounded from function free first order predicates, the I is the probabilistic distribution over initial states, the G is the goal that must be satisfied, and the A is a set of actions which are instances of action schemata.

Probabilistic conformant planning is the task of generating a sequence of actions achieving the goal without sensing. In other words, a planner should find a sequence of actions in the belief state space that is resulting from the uncertainty of initial states and action effects. The query to the planner can be formulated in two different ways: $\langle P, \theta \rangle$, which ask for a plan achieving the goal with probability that exceeds a threshold θ , or $\langle P, T \rangle$, which fixes the time horizon T and ask for a plan with maximum probability.

We define an optimal probabilistic conformant planner as a planner which always finds the shortest length plan that exceeds a threshold. Note that a planner which returns a

plan from the query $\langle P, T \rangle$ can be transformed into an optimal planner by increasing the time horizon incrementally. For example, COMPLAN (Huang 2006) and CPPlan (Hyafil and Bacchus 2004) answer to the query $\langle P, T \rangle$. On the other hand, Probabilistic Fast Forward (Domshlak and Hoffmann 2007) returns some sequence of actions that exceeds the threshold.

We address probabilistic planning problems expressed in Probabilistic Planning Domain Definition Language (Younes and Littman 2004) with minor extensions to express probability distribution over the initial states. Such problem domains consist of function free first order predicates to encode state variables, and parameterized action schemata with a precondition and an effect, expressed in the first order syntax. The precondition of an action was treated like a condition of conditional effect such that executing an inapplicable action in a state that violates the precondition results in no-op. For simplicity, we don't deal with quantifiers and disjunctions in the condition, and we assumed that there is only single goal state in the goal.

In this paper we consider the optimal probabilistic conformant plan that maximizes the probability of achieving the goal given a fixed time horizon. We show that the optimal plan can be found by marginal MAP inference over the DBN that compiles a planning problem properly. Such a translation of PPDDL into DBN was developed based on the SAT encoding of planning problems (Kautz, McAllester, and Selman 1996), where the translation is applied after grounding the lifted expressions by grounding each predicate and action schema into state and action variables. Benchmark problems were selected from the international planning competitions to assess the applicability of marginal MAP inference for planning.

Conformant Planning as Marginal MAP

Our task is to solve the conformant planning problem $P = \langle s_0..s_T, a_0..a_{T-1}, T \rangle$. Namely to find a length T sequence of action assignments $a_0..a_{T-1}$ that maximizes the conditional probability of reaching the goal s_T given the initial state s_0 and the action assignments as follows:

$$\arg \max_{a_0..a_{T-1}} \sum_{s_1..s_{T-1}} P(s_1..s_{T-1} | s_0, s_T, a_0..a_{T-1}) \quad (1)$$

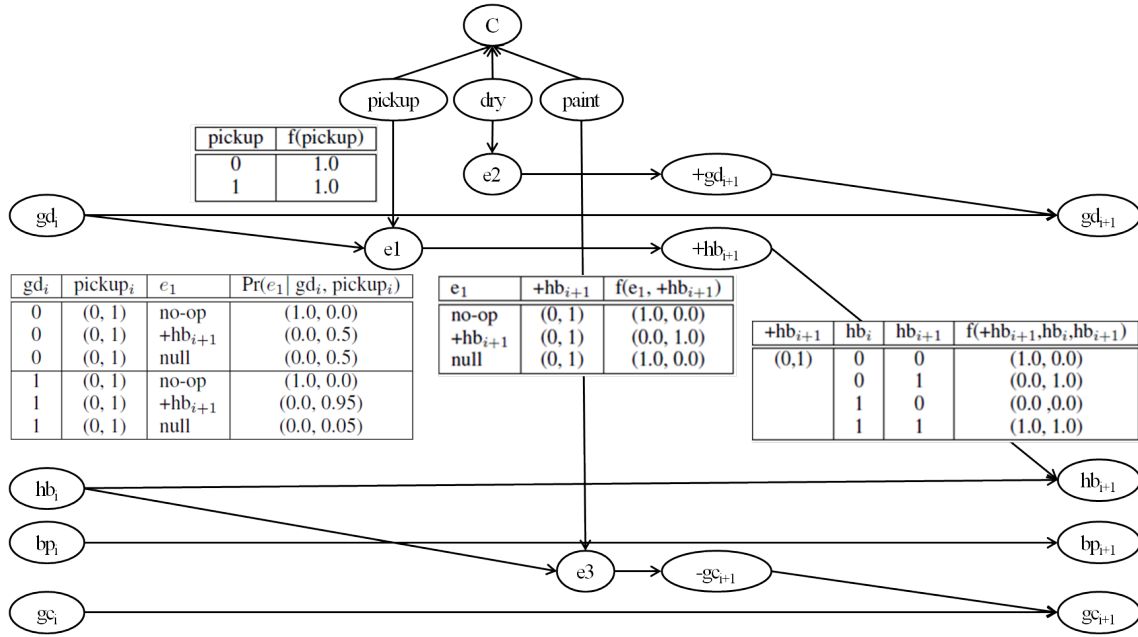


Figure 1: 2TDBN for the slippery gripper domain

where the $\mathbf{a}_i$ is the vector of action variables at the i -th stage, and the $\mathbf{s}_i$ is the vector of state variables at the i -th stage. If multiple goal states were allowed in the goal, we would add additional layer r_{T+1} qualifying the final state s_T , and the summation would subsume the final state s_T .

Encoding PPDDL into DBN

We developed translation from PPDDL domain into 2 stage DBN (2TDBN) encoding state transitions in a single time step. Our 2TDBN encoding extends the translation given in (Younes and Littman 2004) by introducing additional variables and functional relations to incorporate action variables and bound the maximum number of parents of a node.

The semantics of 2TDBN is based on the linear encoding of SATPLAN with additional multivalued effect variables to address probabilistic outcomes of a probabilistic effect. The j -th boolean state variables s_i^j of the state s_i , and the k -th boolean action variables a_i^k of the action $\mathbf{a}_i$ are obtained by grounding the lifted predicates and action schemata, where the i is the time step.

A ground action consists of a precondition ϕ , where the ϕ is a conjunction of equality predicates on ground objects and literals of state variables, and an effect which can be either a simple effect, a conditional effect, a probabilistic effect or a conjunction of other effects. We assumed that a planner would execute a no-op action if the precondition didn't meet. Thus, an action is implicitly converted to a conditional effect, which is the so called weak precondition. From now on, we do not distinguish between a precondition of an action and a condition in the conditional effect.

A simple effect is a conjunction of state variables where positive literals corresponds to add list and negative literals corresponds to delete list of STRIPS formalism. A condi-

tional effect is a pair (ϕ, e) , where the ϕ is a condition defined earlier, and the e is an effect. A probabilistic effect is a list of pairs (p_i, e_i) , where the p_i is the probability value for the i -th effect e_i . Thus, a PPDDL effect can be nested in arbitrary depth forming a tree of which simple effects placed at the leaf. For each effect we introduce an effect variable capturing the outcomes of the effect. Specifically, a probabilistic effect having n outcomes maps to an effect variable with $n + 1$ values where the additional value is reserved for the no-op. Finally, additional variables $+s_{i+1}^k$ and $-s_{i+1}^k$ are defined to indicate whether the state variable s_{i+1}^k will be added or deleted.

A 2TDBN $G = \langle V, F \rangle$ with a set of variables V and a set of conditional probability table F for each variable can be obtained by converting following clauses:

- $a_i^j \wedge \phi_i^j \rightarrow (e_i^j \neq \text{no-op}), \neg(a_i^j \wedge \phi_i^j) \rightarrow (e_i^j = \text{no-op})$, where ϕ_i^j is the precondition of a_i^j
- $\forall_{k \in K} (e_i^j = v) \equiv +/ -s_{i+1}^k$, where $K = \{k \mid s_{i+1}^k \in \cup_v \text{add/del}(e_i^k = v)\}$
- $\forall_{k_1 \neq k_2} (e_i^{k_1} = v_1) \wedge (e_i^{k_2} = v_2) \rightarrow \neg +/ -s_{i+1}^k$, if $s_{i+1} \in \text{add/del}(e_i^{k_1} = v_1)$ and $s_{i+1} \in \text{add/del}(e_i^{k_2} = v_2)$
- $\forall_j \vee a_i^j, \forall_{j \neq k} a_i^j \rightarrow \neg a_i^k$
- $\neg +s_{i+1}^j \wedge \neg -s_{i+1}^j \rightarrow (s_i^j \wedge s_{i+1}^j) \vee (\neg s_i^j \wedge \neg s_{i+1}^j)$

Note that above clauses encode a planning domain where each action has a single probabilistic effect without nesting. So, an outcome v of the j -th effect variable at the i -th time stage ($e_i^j = v$) corresponds to a simple effect. For the nested effects, we consult in normal forms in (Rintanen 2003). The first clauses correspond to the CPT for an effect variable, where each outcome occurs only if the action triggering the

effect was executed and the precondition was satisfied. The second and the third clauses encode the CPT for an indicator variable to indicate the result of executing some action as well as to deprecate inconsistent combination of effects. The last two clauses encode the mutual exclusivity on action variables and the frame axioms. Concurrent actions can be easily adopted by modifying the previous propositional encoding.

Consider slippery gripper domain (Kushmerick, Hanks, and Weld 1995) which consists of four state variables, gripper-dry (gd), holding-block (hb), block-painted (bp), and gripper-clean (gc), and three actions, “pick up”, “dry”, and “paint”. Initially, the gc is true, the gd is true with probability 0.7, and other two state variables are false. The goal is a conjunction of literals, $gd \wedge hb \wedge bp$. If gripper-dry were true, the “pick up” action would add hold-block with probability 0.95. Otherwise hold-block would be added with probability 0.5. The “dry” action would add gripper-dry with probability 0.8. Lastly, If holding-block were true, gripper-clean will be deleted by “paint” action, and if it were false, gripper-clean would be deleted with probability 0.1. The “paint” action also adds block-painted.

Fig. 1 shows the 2TDBN for the slippery gripper domain. Each value of the CPT for “pick up” is normalized to 1.0 because the joint conditional distribution is conditioned on the action variable. The effect variable e_1 takes three values, the no-op for the case where the “pick up” action was not executed, the $+hb_{i+1}$ for adding the state variable hb_{i+1} , and the null effect derived from the semantics of probabilistic effect, i.e., when $\sum_i p_i < 1$, the probabilistic effect takes an additional null effect with probability $1 - \sum_i p_i$. The probability of each outcome is mapped to the corresponding row of the CPT. Note that the effect CPTs from probabilistic effects are only probabilistic. The rest of the two tables in Fig. 1 are simply rewritten in tabular form of the propositional encoding. Finally, the CPT for the node C in Fig. 1 is also in tabular form of the mutual exclusivity clauses.

The DBN with time horizon T will be obtained by replicating the 2TDBN over T times. The probability distribution over the initial state variables and the single goal state can be assigned to the state variables s_0 , and s_T .

In practice, in-degree of the nodes for the constraint on action variables, and the indicator variables $\pm s_{i+1}^j$ grows prohibitively for storing the tabular form of the CPT, so additional hidden variables were introduced to bound the maximum in-degrees.

The complexity of the resulting 2TDBN is similar to that of SATPLAN, and it can be summarized as follows. Let the number of action schemata in the planning domain be $|as|$, the number of predicates be $|pre|$, the maximum arity of the action schemata be p , the maximum arity of the predicates be q , and the number of ground objects in the instance be k . The total number of the action variables is $O(|as|k^p)$, and the total number of the state variables is $O(|pre|k^q)$. Assuming that we have only flat effects, the number of effect variables are the same as the number of action variables. Thus, the total number of variables in a 2TDBN is $O(2|as|k^p + 3|pre|k^q)$. In case of introducing hidden variables to bound the maximum in-degree, we would introduce

$|as|k^p - 1$ hidden variables to bound the maximum in-degree of the constraint variable to be 2. Similarly, if the maximum number of parents for the indicator variables was $|E|$, we would introduce $|E| - 1$ hidden variables. In such a case, the maximum arity of the CPTs is $\max(3, s + 1)$, where the s is the maximum number of state variables shown in a precondition which contributes to the arity of the CPT for an effect variable. The maximum domain size is $n + 1$, where the n is the maximum number of outcomes of a probabilistic effect. Finally, the total number of variables is $O(3|as|k^p + (1 + 2|E|)|pre|k^q)$.

Marginal MAP Algorithms

We are using recently proposed AND/OR Branch and Bound (AOBB) search algorithms for marginal MAP (Marinescu, Dechter, and Ihler 2014), where heuristics were created by Weighted Mini Bucket elimination (WMB) from the constrained variable ordering. AOBB search heuristics are:

- Pure MB (Dechter and Rish 2003)
- MB with moment matching (Ihler et al. 2012)
- WMB with join graph cost shifting (Ihler et al. 2012)

Constraint propagation also employed on the conditioned subproblems by CNF encoding of zero probability tuples and applying unit resolution or full satisfiability (Marinescu and Dechter 2009). In addition, we tested our other algorithm BBOT, which is an OR branch and bound search guided by MCTE (Mini Cluster Tree Elimination) heuristic as well as Yuan’s algorithm (Yuan and Hansen 2009).

Experiment Result

The benchmark instances were selected from the planning domains provided by past international planning competitions: “slippery gripper”, “comm”, and “blocks world”. For the blocks world domain the instances are from probabilistic track of IPC 2006, and we removed three “tower actions” that can move more than two blocks at once. The problem instances for the blocks world domain were recreated such that there exists shorter length plan. The slippery gripper instances and “comm” domain were not modified from the original release. Note that the “comm” domain has only uncertain initial state. In contrast, blocks world domain has only probabilistic transitions, and the slippery gripper domain contains uncertain initial state and probabilistic state transitions.

Table 1 summarizes our benchmark results from three marginal MAP algorithms: 1. AOBB search with the WMB-JG heuristic ran for 10 iterations and full constraint propagation for deterministic tuples in the CPTs (AOBB-WMB-JG), 2. BBOT search with the incremental MCTE heuristic (BBOTi), and 3. Yuan’s algorithm. The tuple $(n, k, s, |S|, |A|)$ is problem statistics showing the total number of variables in the 2TDBN, the maximum domain size, the maximum scope size, and the number of the states and action variables per time step.

The number of variables n is much higher than sum of grounded state/action variables since we introduced hidden

Table 1: Experiment results from marginal MAP search algorithms. Time/Memory limit 6 Hr/4GB.

Domain Name (n, k, s, S , A)	Solution	AOBB-JG	BBBTi	YUAN	Solution	AOBB-JG	BBBTi	YUAN	Solution	AOBB-JG	BBBTi	YUAN
	max prob (L, w, h)	time (i, c)	time (i, c)	time -	max prob (L, w, h)	time (i, c)	time (i, c)	time -	max prob (L, w, h)	time (i, c)	time (i, c)	time -
slippery gripper (23, 3, 3, 4, 3)	0.7335 (2, 6, 3)	0.01 (6, 6)	0.01 (6, 6)	0.01 -	0.9 (14, 44, 58)	210 (22, 22)	54 (30, 30)	12 -	0.9 (21, 65, 82)	oot (24, 24)	8906 (26, 26)	3734 -
	0.8985 (6, 20, 29)	8.33 (20, 20)	0.08 (20, 20)	0.03 -	0.9 (19, 59, 74)	8840 (26, 26)	2329 (30, 30)	881 -	0.9 (22, 68, 88)	oot (24, 24)	oot (26, 26)	7334 -
	0.8999 (10, 32, 44)	5.49 (20, 20)	1.99 (30, 30)	0.65 -	0.9 (20, 62, 78)	19262 (26, 26)	7481 (30, 30)	2121 -	0.9 (23, 71, 86)	oot (24, 24)	oot (26, 26)	19607 -
	0.25 (5, 269, 324)	11 (8, 8)	oom (2, 2)	oom -	0.25 (7, 363, 436)	74 (2, 2)	oom -	oom -	0.25 (9, 467, 540)	54 (2, 2)	oom (2, 2)	oom -
bw-24 (73, 3, 5, 9, 8)	0.1406 (3, 32, 55)	0.57 (10, 10)	0.35 (12, 1)	7 -	0.7031 (5, 51, 80)	5 (10, 10)	8.93 (10, 10)	92 -	0.8701 (7, 69, 101)	879 (16, 16)	9513 (14, 14)	18809 -
	0.5625 (4, 40, 66)	1 (10, 10)	1.95 (12, 12)	10 -	0.8085 (6, 57, 90)	67 (12, 12)	242 (12, 12)	1029 -	0.9162 (8, 72, 115)	9930 (18, 18)	oot (22, 22)	oot -
	0.0351 (4, 101, 142)	1.71 (4, 4)	2.19 (8, 8)	oom -	0.1406 (6, 135, 191)	225 (8, 8)	217 (8, 8)	oom -	0.5273 (7, 156, 221)	1572 (8, 8)	7681 (8, 8)	oom -
bw-34 (170, 3, 5, 16, 18)	0.0351 (4, 101, 142)	1.71 (4, 4)	2.19 (8, 8)	oom -	0.1406 (6, 135, 191)	225 (8, 8)	217 (8, 8)	oom -	0.5273 (7, 156, 221)	1572 (8, 8)	7681 (8, 8)	oom -

variables for high in-degree CPTs due to the action constraints and the frame axioms. The solution is reported for the maximum probability with length L , the induced width w , and the pseudo tree height h for the topological variable ordering. AOBB-WMB-JG and BBBTi shares the same algorithmic parameters, the i bound for the WMB and the c bound for the caching, which are shown below the running time for each algorithms.

slippery gripper The optimal plan for the slippery gripper domain should satisfy two partial order relations: the “paint” action precedes the “pick up” action, and the “dry” action precedes the “pick up” action. One such valid plan is (paint, dry ^{i} , pick-up ^{j}), where the $i \geq 0$ and the $j \geq 1$ denotes the number of “dry” and “pick up” actions executed in the plan, and $T = 1 + i + j$. The probability of success for the plans satisfying above partial order relations is $0.9(d_i p_j + (1 - d_i) q_j)$, where $d_0 = 0.7$, $d_i = d_{i-1} + 0.8(1 - d_{i-1})$, $p_1 = 0.95$, $p_j = p_{j-1} + 0.95(1 - p_{j-1})$, $q_1 = 0.5$, and $q_j = q_{j-1} + 0.5(1 - q_{j-1})$. For example, the optimal length 3 plan is (paint, pick up, pick up) with the probability 0.830925. Thus, we can assure that the plans found by marginal MAP search algorithms were truly optimal by checking above equations on various i and j values.

Yuan’s algorithm performed best among three algorithms, and it was able to find conformant plans up to 23 time horizon in 6 hour time limit. On the other hand, BBBT search with incremental MCTE heuristics was able to find up to 21 time horizon.

comm For the “p01” problem from the IPC 2006 competition, AOBB-WMB-JG solved problems up to 9 time horizon. We suspect that the full constraint propagation incorporated in AOBB-WMB-JG played a significant role for solving the problems. Compared to other domains, induced width of the constrained ordering was prohibitively huge, for example the induced width was 467 for the 9 time horizon instance.

simple blocks world Two blocks world domain were reported. Each of them has 2 and 3 blocks with 4 action schemata. AOBB-WMB-JG was able to find plans for up to 8 time horizon for bw-24 and 7 time horizon for bw-34. Since the number of state/action variables is exponentially increasing, the algorithms couldn’t find a solution for the problems having more than 3 blocks.

Comparison with Other Planners

In this section, we compare the marginal MAP search algorithms with COMPLAN and Probabilistic-FF. The performance of Probabilistic-FF was evaluated in the same benchmark environment.

To the best of our knowledge, COMPLAN is the most recent probabilistic conformant planning algorithm that maximizes the probability of success for a plan with a fixed time horizon. COMPLAN extends linear encoding of SAT-PLAN with chance variables that encode the uncertainty of a planning problem, and compiles CNFs into deterministic decomposable negation normal form (Darwiche 2004), and then apply the marginal MAP query. The conformant plan is found by depth first branch and bound search guided by heuristics generated by a relaxed marginal MAP query on the d-DNNF compilation. Note that the induced width of constrained variable orderings for a planning problem is usually prohibited as shown in the Table 1. AOBB-WMB-JG relaxes the problem based on the weighted mini-buckets with cost shifting, whereas COMPLAN allows an unconstrained variable ordering to yield lower induced width.

Probabilistic-FF finds a conformant plan that achieves the goal higher than a desired threshold by heuristic forward search in a belief state space. The belief states $b_{\bar{a}}$ is a probability distribution over states reachable from initial belief state b_I by applying a sequence of actions $\bar{a}$. Probabilistic-FF also represents belief states by DBN $N_{b_{\bar{a}}}$ (Boutilier, Dean, and Hanks 1999), and the DBN is further compiled into weighted CNFs, to retrieve probabilistic queries via weighed model counting (Sang, Beame, and Kautz 2005).

In our representation, all the action variables were nodes of the DBN such that the probabilistic inference directly query over the action variables after marginalizing the rest of the variables. On the other hand, each DBN for the belief states $b_{\bar{a}}$ of the Probabilistic-FF unrolls only portion of the entire DBN that can be instantiated by applying a specific sequence of actions $\bar{a}$, and the weighted model counting computes only posterior probability so that the heuristic function evaluates the probability of achieving the goal.

Table 2 compares Probabilistic-FF and AOBB-WMB-JG on the planning problems: “slippery gripper”, and “simple blocks world”. We set threshold values of the Probabilistic-

Table 2: Comparison between Probabilistic-FF and AOBB-WMB-JG. Time/Memory limit 6 Hr/4GB.

s-gripper	θ	0.7335	0.830925	0.884385	0.895077	0.898539	0.899618	0.899859	0.899967	0.899989	0.899999	0.9
pff	time	0.04	0.03	0.04	0.05	0.04	0.05	0.04	0.04	0.07	0.11	oot
(h1, w0)	length	3	4	5	6	8	10	11	13	15	18	-
pff	time	0.03	0.19	0.42	1.22	1.27	3.05	6.29	13.89	31.56	156.86	oot
(h2, w1)	length	2	4	5	6	6	7	8	9	10	12	-
AOBB	pre, tot	0.01, 0.01	0.02, 0.02	0.13, 0.13	0.98, 0.98	8.33, 8.33	66.23, 66.23	3.37, 4.13	4.27, 5.19	46.94, 49.29	9.07, 37.23	90.59, 126.41
WMB-JG	L, w, i	2, 6, 6	3, 10, 10	4, 14, 14	5, 17, 17	6, 20, 20	7, 23, 10	8, 27, 22	9, 29, 22	10, 32, 28	12, 38, 24	13, 41, 24
bw-24	θ	0.140625	0.5625	0.703125	0.808594	0.870117	bw-34	θ	0.140625	0.2	0.4	0.527344
pff	time	0.04	0.05	oom	oom	oom	pff	time	0.09	0.09	0.1	error
(h1, w0)	length	4	4	-	-	-	(h2, w0)	length	6	6	6	-
AOBB	pre, tot	0.56, 0.57	0.9, 1.06	1.28, 5	3.62, 67	55.6, 879	AOBB	pre, tot	5, 225	-	-	7.4, 1572
WMB-JG	L, w, i	3, 32, 10	4, 40, 10	5, 48, 10	6, 57, 12	7, 67, 16	WMB-JG	L, w, i	6, 135, 8	-	-	7, 156, 8

FF from optimal conformant planning solutions for comparison. For the Probabilistic-FF we present the best result from 2 heuristic functions and 3 weight propagation schemes. For the detail about Probabilistic-FF planner, we refer to the Probabilistic-FF codes available at the author’s website. For the AOBB-WMB-JG, we present preprocessing time for compiling WMB-JG static heuristics with 10 iterations for the join graph cost shifting, the total time that covers both preprocessing time and AOBB search time, and triplet (L, w, i) for the shortest plan length, the induced width of a constrained variable ordering, and the i-bound parameter for WMB.

slippery gripper Probabilistic-FF solved all the instances around 0.1 seconds except the one with threshold 0.9. The length of the plan is typically longer than the optimal, but plans that are closer to the optimal could be found with increased time. For the slippery gripper problem, the threshold 0.9 is the maximum probability of success achievable.

Considering the length of the plan, the performance of AOBB-WMB-JG is comparable to Probabilistic-FF in many problem instances. The WMB relaxation was exact, i.e., the i-bound was same as the induced width when the time horizon was less than 7. Thus, AOBB-WMB-JG was solved by inference.

simple blocks world In two problem domains, bw-24 and bw-34, Probabilistic-FF was not able to find a plan longer than 4 and 6 due to the 4GB memory limit. In our experiments, Probabilistic-FF showed cut-off thresholds for both domains, 0.6 and 0.5 for each. As we discussed in previous sections, AOBB-WMB-JG ran out of the time when the time horizon grows higher than 8 and 7 for each domain.

Discussion and Future Work

We applied state of the art marginal MAP solver to probabilistic conformant planning problems by converting SAT encoding of a planning instance into DBN. We were able to solve instances from three domains that are originated from the past planning competitions. When the size of the factored representation at the propositional level was manageable, our marginal MAP solver could find the optimal plan given a fixed time horizon. On the other hand, the heuristics based on the approximate probabilistic inference becomes inaccurate as we extend the time horizon since the induced width of constrained variable ordering increases linearly.

Benchmark on harder problems The three planning domains we addressed in this paper are easy domains; “slippery gripper” domain was already defined at the proposi-

tional level and it contains 4 state variables and 3 action variables. “Blocks world” domains were simplified by removing three action schemata in the original release to bound the number of ground variables at each time stage. The problem instances solved by marginal MAP algorithms were limited to contain at most three blocks and 7 time horizon. Therefore, our goal is to translate the planning domains in more compact way, and improve the probabilistic inference to exploit symmetry and determinism present in the planning problem.

Compact translation More compact translation at the propositional level is desired to push the marginal MAP search algorithms to solve planning instances leaving a larger number of constant objects and longer time horizon at the ground level. The SAS+ based SAT encoding for the classical planning is known to produce more compact translation compared to the linear encoding of SATPLAN (Huang, Chen, and Zhang 2010). Thus, the straightforward extension of our current work is to convert a STRIPS like PPDDL domain into FDR (finite domain representation) (Helmert 2009), and translate the new SAT encoding into the graphical model with proper extension addressing the uncertainty.

Need for lifting Planning problems have two source of lifting: the time horizon which implies replicated structure over the time axis, and the state/action representation parameterized by object variables. It was shown that MDP/POMDP planning can be reformulated as a mixture model of unbounded number finite length DBNs (Toussaint 2009) with flat state variables. Still, lifted inference for the parameterized random variables declared in both STRIPS like PPDDL and SAS+ based FDR has not gained much attention in the literature. We leave improving current algorithms in conjunction with lifted inference on parfactor graphs possibly translated from lifted planning problem representations as future work.

References

- Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 1:1–93.
- Darwiche, A. 2004. New advances in compiling cnf to decomposable negation normal form. In *Proc. of ECAI*, 328–332. Citeseer.
- Dechter, R., and Rish, I. 2003. Mini-buckets: A general

scheme for bounded inference. *Journal of the ACM (JACM)* 50(2):107–153.

Domshlak, C., and Hoffmann, J. 2007. Probabilistic planning via heuristic forward search and weighted model counting. *J. Artif. Intell. Res.(JAIR)* 30:565–620.

Helmert, M. 2009. Concise finite-domain representations for pddl planning tasks. *Artificial Intelligence* 173(5):503–535.

Huang, R.; Chen, Y.; and Zhang, W. 2010. A novel transition based encoding scheme for planning as satisfiability. In *AAAI*.

Huang, J. 2006. Complan: A conformant probabilistic planner. *ICAPS 2006* 63.

Hyafil, N., and Bacchus, F. 2004. Utilizing structured representations and cps in conformant probabilistic planning. In *ECAI*, volume 16, 1033.

Ihler, A.; Flerova, N.; Dechter, R.; and Otten, L. 2012. Join-graph based cost-shifting schemes. In *UAI*.

Kautz, H.; McAllester, D.; and Selman, B. 1996. Encoding plans in propositional logic. *KR* 96:374–384.

Kushmerick, N.; Hanks, S.; and Weld, D. S. 1995. An algorithm for probabilistic planning. *Artificial Intelligence* 76(1):239–286.

Marinescu, R., and Dechter, R. 2009. And/or branch-and-bound search for combinatorial optimization in graphical models. *Artificial Intelligence* 173(16):1457–1491.

Marinescu, R.; Dechter, R.; and Ihler, A. 2014. Improving marginal map search in graphical models. In *submitted to UAI*.

Rintanen, J. 2003. Expressive equivalence of formalisms for planning with sensing. In *ICAPS*, 185–194.

Sang, T.; Beame, P.; and Kautz, H. 2005. Solving bayesian networks by weighted model counting. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, volume 1, 475–482.

Toussaint, M. 2009. Probabilistic inference as a model of planned behavior. *Künstliche Intelligenz* 3(9):23–29.

Younes, H. L., and Littman, M. L. 2004. Ppddl1. 0: The language for the probabilistic part of ipc-4. In *Proc. International Planning Competition*.

Yuan, C., and Hansen, E. A. 2009. Efficient computation of jointree bounds for systematic map search. In *IJCAI*, 1982–1989.

Towards Adversarial Reasoning in Statistical Relational Domains

Daniel Lowd and Brenton Lessley and Mino De Raj

Dept. of Computer and Information Science
University of Oregon
Eugene, Oregon 97403

Abstract

Statistical relational artificial intelligence combines first-order logic and probability in order to handle the complexity and uncertainty present in many real-world domains. However, many real-world domains also include multiple agents that cooperate or compete according to their diverse goals. In order to handle such domains, an autonomous agent must also consider the actions of other agents. In this paper, we show that existing statistical relational modeling and inference techniques can be readily adapted to certain adversarial or non-cooperative scenarios. We also discuss how learning methods can be adapted to be robust to the behavior of adversaries. Extending and applying these methods to real-world problems will extend the scope and impact of statistical relational artificial intelligence.

Introduction

Statistical relational models provide a powerful framework for artificial intelligence by combining first-order logic and probabilistic models. With some notable exceptions (e.g., (Poole 1997)), previous work in statistical relational artificial intelligence has mainly focused on representation, learning, and non-adversarial reasoning in statistical relational models. However, intelligent agents also need adversarial reasoning in order to be robust to worst-case situations or handle environments with multiple agents.

In this position paper, we discuss the problem of adversarial statistical relational reasoning and the related problem of adversarially robust statistical relational learning. We use the term “adversarial” informally to refer to both zero-sum games and general-sum non-cooperative games. We show that, when the domain is represented as a log-linear model such as a Markov logic network, certain adversarial reasoning tasks can be reduced to standard maximum a posteriori (MAP) inference. A similar transformation can convert learning methods that are based on MAP inference into adversarially robust learning methods.

Background

A number of statistical relational representations have been introduced over the years, either by extending probabilistic

graphical models to handle relational domains or by extending logic programming to handle uncertainty. In this paper, we focus on the former case in general, and Markov logic networks (MLNs) (Domingos and Lowd 2009) in particular. An MLN uses first-order formulas as the features in a log-linear model. Together with a finite set of constants (and assuming known functions), this defines a probability distribution over possible worlds:

$$\log P(\mathcal{X} = \mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) - \log Z,$$

where $\mathcal{X} = \mathbf{x}$ is a possible world, consisting of the truth values of all ground atoms; $\phi_i(\mathbf{x})$ is the number of satisfied groundings of the i th formula; w_i is its weight; and Z is a normalization constant.

One notable extension of Markov logic is Markov logic decision networks (MLDNs), which adapt MLNs for the problem of utility maximization in uncertain relational domains (Nath and Domingos 2009). An MLDN distinguishes between evidence atoms, $\mathbf{e}$; action atoms controlled by agent, $\mathbf{a}$; and other state atoms, $\mathbf{x}$. An MLDN assigns a utility weight u_i to each formula. The utility of a world is the total utility weight of the satisfied formulas:

$$U(\mathbf{x}, \mathbf{a}, \mathbf{e}) = \mathbf{u}^T \phi(\mathbf{x}, \mathbf{a}, \mathbf{e})$$

The agent’s goal is to select a configuration of the action atoms to maximize the expected utility, defined as follows:

$$E[U(\mathbf{x}, \mathbf{a}, \mathbf{e}) | \mathbf{a}, \mathbf{e}] = \sum_{\mathbf{x}} P(\mathbf{x} | \mathbf{a}, \mathbf{e}) U(\mathbf{x}, \mathbf{a}, \mathbf{e})$$

This can naturally model many decision-theoretic problems, but it does not consider the actions of other rational agents in the environment.

Adversarial Relational Reasoning

MLDNs use an expected utility over many possible outcomes. In general, this expectation is very difficult to compute. We begin by considering the simpler case, in which there are no state atoms, and then extend this to include multiple agents. Efficiently incorporating state atoms with multiple agents remains an important direction for future work.

If there are no state atoms, then maximizing the utility is a combinatorial optimization problem:

$$\mathbf{a}_o = \arg \max_{\mathbf{a}} U(\mathbf{a}, \mathbf{e}) = \arg \max_{\mathbf{a}} \mathbf{u}^T \phi(\mathbf{a}, \mathbf{e}).$$

This problem has the same form as standard maximum a posteriori (MAP) inference in a log-linear model and can be solved with the same (approximate) inference algorithms.

Now we extend this model to consist of two agents: s (“self”) with utility weights $\mathbf{u}_s$, and o (“opponent”) with utility weights $\mathbf{u}_o$. We model the game as follows: First, Agent s selects an action, $\mathbf{a}_s$. Agent o observes this action and selects a response, $\mathbf{a}_o$. Agents s and o then receive rewards $\mathbf{u}_s^T \phi(\mathbf{a}_o, \mathbf{a}_s, \mathbf{e})$ and $\mathbf{u}_o^T \phi(\mathbf{a}_o, \mathbf{a}_s, \mathbf{e})$, respectively. Assuming that o is rational, the optimal action of s is given by the following optimization problem:

$$\begin{aligned} & \underset{\mathbf{a}_s}{\text{maximize}} \quad \mathbf{u}_s^T \phi(\mathbf{a}_o, \mathbf{a}_s, \mathbf{e}) \\ & \text{where } \mathbf{a}_o = \arg \max_{\mathbf{a}_o} \mathbf{u}_o^T \phi(\mathbf{a}_o, \mathbf{a}_s, \mathbf{e}). \end{aligned}$$

This corresponds to a Stackelberg game where s is the leader. For mathematical convenience, we assume the $\arg \max$ is unique; if not, we can break ties arbitrarily. We can equivalently state that $\mathbf{a}_o$ has higher utility than all alternate actions, $\mathbf{a}'_o$:

$$\begin{aligned} & \underset{\mathbf{a}_s, \mathbf{a}_o}{\text{maximize}} \quad \mathbf{u}_s^T \phi(\mathbf{a}_o, \mathbf{a}_s, \mathbf{e}) \\ & \text{s.t.} \quad \mathbf{u}_o^T \phi(\mathbf{a}_o, \mathbf{a}_s, \mathbf{e}) \geq \mathbf{u}_o^T \phi(\mathbf{a}'_o, \mathbf{a}_s, \mathbf{e}) \quad \forall \mathbf{a}'_o \neq \mathbf{a}_o. \end{aligned}$$

Solving this optimization problem in either form is typically intractable: the first is a bilevel optimization problem, and the second involves an exponential number of constraints – one for each of the exponentially many possible actions $\mathbf{a}_o$. We can make the second form more tractable by reducing the number of constraints. For example, suppose we only consider a single alternate action $\mathbf{a}'_o$. Taking the Lagrangian relaxation,

$$\begin{aligned} & \underset{\mathbf{a}_s, \mathbf{a}_o}{\text{maximize}} \quad \min_{\lambda \geq 0} \quad \mathbf{u}_s^T \phi(\mathbf{a}_o, \mathbf{a}_s, \mathbf{e}) + \\ & \quad \lambda \mathbf{u}_o^T (\phi(\mathbf{a}_o, \mathbf{a}_s, \mathbf{e}) - \phi(\mathbf{a}'_o, \mathbf{a}_s, \mathbf{e})). \end{aligned} \quad (1)$$

For a fixed value of λ , Equation 1 can be represented as a single MAP inference problem:

$$\underset{\mathbf{a}_s, \mathbf{a}_o}{\text{maximize}} \quad \mathbf{w}'^T \phi'(\mathbf{a}_o, \mathbf{a}_s, \mathbf{e}) \quad (2)$$

where $\mathbf{w}' = [(\mathbf{u}_s - \lambda \mathbf{u}_o); \lambda \mathbf{u}_o]$ and $\phi'(\mathbf{a}_o, \mathbf{a}_s, \mathbf{e}) = [\phi(\mathbf{a}_o, \mathbf{a}_s, \mathbf{e}); \phi(\mathbf{a}'_o, \mathbf{a}_s, \mathbf{e})]$. We can search over different values of λ as necessary to obtain tighter bounds.

To summarize, this optimization problem finds actions $\mathbf{a}_s$ to maximize the utility of agent s , assuming that agent o will respond with an action $\mathbf{a}_o$ that has higher utility than reference action $\mathbf{a}'_o$. If most of the opponent’s high-utility actions are similar, then this could be a reasonable approximation. If not, then we can iteratively introduce more reference actions and Lagrange multipliers, obtaining a better and better approximation of the true objective.

Example: Consider a webspam domain, in which the spammer’s goal is to create a set of webpages with many advertisements and have them appear among the results of popular search engines. We wish to predict the actions of

spammers in order to understand their behavior, develop better defenses, and evaluate the robustness of these defenses.

The spammer’s opponent is the search engine’s anti-spam component, which uses an MLN to label web pages as spam or non-spam, after they have been modified by the spammer. We view the MLN weights as utility weights, which makes the search engine’s MAP labeling its maximum utility action. We can represent the spammer’s utility as the number of spam web pages that are not detected by the search engine, minus a penalty for the number of words and links modified in order to disguise these web pages (representing the effort or cost of evading detection). To make the problem more tractable, we use the true spam/non-spam labels as the alternate reference action, $\mathbf{a}'_o$. Thus, the task is to find web page modifications, $\mathbf{a}_s$, and alternate labeling, $\mathbf{a}_o$, to maximize the spammer’s utility, under the constraint that $\mathbf{a}_o$ is a “better” labeling than the true labels. Following Equation 2, this can be solved by standard MAP inference in an MLN.

Adversarial relational reasoning can also be used to develop adversarially robust learning methods. For example, suppose we wish to learn the parameters of a webspam classification system that will be robust to adaptive spammers. In standard max-margin learning, this is done by enforcing a margin between the true labels and all alternate labelings. With cutting plane methods, these margin constraints are added one at a time using MAP inference (Tsochantaridis et al. 2005). For adversarially robust learning, we instead enforce a margin between true and alternate labelings for *any* adversarially selected action. The most violated adversarial margin constraints can be found by solving a problem similar to Equation 2. In previous work (Torkamani and Lowd 2013), we have solved a special case of this problem, but the more general case has a lot of potential.

Ongoing Work and Future Directions

Adversarial reasoning is a core component of artificial intelligence that has mostly been neglected by research in statistical relational methods. In ongoing work, we are exploring how statistical relational models can detect spam and fraud in relational domains such as Twitter (Yang, Harkreader, and Gu 2011) and YouTube (O’Callaghan et al. 2012), how adversarial reasoning can automatically predict spammer strategies and model weaknesses, and how adversarial learning can construct provably robust models. As a first step, we are extending RockIt (Noessner, Niepert, and Stuckenschmidt 2013) to perform adversarial reasoning in any MLN with any linear utility model.

However, the general problem of adversarial and multi-agent reasoning in relational domains is much larger. To cope with the complexities of real-world environments, agents must handle rational adversaries, boundedly rational or irrational adversaries, multiple cooperating or competing adversaries, and random (non-adversarial) effects from the environments. Moving forward, we need to identify the most promising domains for exploration, extend statistical relational representations to efficiently model these additional factors, and scale up learning and inference methods to handle these problems.

Acknowledgments

We thank the anonymous reviewers for useful comments. This research was partly funded by ARO grant W911NF-08-1-0242. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO or the U.S. Government.

References

- Domingos, P., and Lowd, D. 2009. *Markov Logic: An Interface Layer for AI*. San Rafael, CA: Morgan & Claypool.
- Nath, A., and Domingos, P. 2009. A language for relational decision theory. In *Proceedings of the International Workshop on Statistical Relational Learning*.
- Noessner, J.; Niepert, M.; and Stuckenschmidt, H. 2013. RockIt: Exploiting parallelism and symmetry for MAP inference in statistical relational models. In *Proceedings of the Twenty-Seventh National Conference on Artificial Intelligence*. Bellevue, WA: AAAI Press.
- O’Callaghan, D.; Harrigan, M.; Carthy, J.; and Cunningham, P. 2012. Network analysis of recurring YouTube spam campaigns. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media*. AAAI Press.
- Poole, D. 1997. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence* 94:7–56.
- Torkamani, M. A., and Lowd, D. 2013. Convex adversarial collective classification. In *Proceedings of the Thirtieth International Conference on Machine Learning*. Atlanta, Georgia: JMLR: W&CP 28.
- Tsochantaridis, I.; Joachims, T.; Hofmann, T.; and Altun, Y. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research* 6(2):1453–1484.
- Yang, C.; Harkreader, R. C.; and Gu, G. 2011. Die free or live hard? Empirical evaluation and new design for fighting evolving Twitter spammers. In *Recent Advances in Intrusion Detection*, 318–337. Springer.

Automated Debugging with Tractable Probabilistic Programming

Aniruddh Nath and Pedro Domingos

Department of Computer Science & Engineering
University of Washington
Seattle, WA 98195, U.S.A.
{nath, pedrod}@cs.washington.edu

Abstract

Probabilistic programming languages allow domain experts to specify generative models in a high-level language, and reason about those models using domain-independent algorithms. Given an input, a probabilistic program generates a distribution over outputs. In this work, we instead use probabilistic programming to explicitly reason about the distribution over *programs*, rather than outputs. We propose *Tractable Probabilistic Programs* (TPP), a language to represent rich probabilistic dependencies between different parts of a program; we make use of the recent work on sum-product networks to ensure that inference remains tractable. We explain how TPP can be applied to the problem of automated program debugging; given a corpus of buggy programs, a TPP model can be learned to capture a probability distribution over the location of the bug. The model can also incorporate additional sources of information, such as coverage statistics on test suites. We also briefly outline how TPP can be used to solve the more ambitious problem of *fault correction*, i.e. predicting the most probable true program conditioned on a buggy one. The ability to learn common patterns of bugs and incorporate multiple sources of information potentially makes TPP useful as a unifying framework for automated program debugging.

Introduction

The term ‘probabilistic programming’ refers to the practice of specifying probabilistic models in a high-level language, and reasoning about them using domain-independent algorithms. This paradigm decouples model design from algorithm design, simplifying the job of the domain expert (who no longer needs to be involved in algorithm design) and making algorithmic advances more widely applicable.

Existing probabilistic programming languages are designed to answer probabilistic queries about the program’s output variables, conditioned on some input (evidence). A common design for probabilistic programming languages is to add stochastic primitive functions to an existing deterministic language (e.g. sampling from Bernoulli or Gaussian distributions). This allows users to easily define sophisticated probability distributions over the output variables.

However, probabilistic programs can also be seen as distributions over deterministic programs (e.g. Church

(Goodman et al. 2008) programs are distributions over Scheme programs, and ProbLog (De Raedt, Kimmig, and Toivonen 2007) programs are distributions over Prolog programs). Seen through this lens, existing probabilistic programming languages define trivial distributions over deterministic languages, typically modeling a probabilistic program as a set of independent distributions over statements.

In this paper, we consider the design of probabilistic programming languages that allow probabilistic dependencies among statements. Specifically, we define *Tractable Probabilistic Programs* (TPP), a language for defining rich probability distributions over programs. TPP uses Sum-Product Networks (SPNs; Poon and Domingos 2011) to model these rich dependencies, while ensuring that probabilistic queries about the program can be computed in polynomial time. The design of TPP is not tied to the choice of a particular deterministic language; TPP allows the user to define a probabilistic version of any deterministic language.

This view of probabilistic programming is well-suited to tackling the problem of automated debugging, a challenging task of great practical importance. Automated debugging has been an active research area for several decades (Shapiro 1983); the bulk of the effort has gone towards the problem of *fault localization*. A well-established approach in the software engineering community is to use coverage statistics on test suites to compute ‘suspiciousness scores’ independently for each statement (e.g. Jones and Harrold 2005). In the following section, we outline how TPPs can incorporate this coverage data into a rich probabilistic model that captures dependencies between statements, rather than predicting the suspiciousness of each line independently.

Tractable Probabilistic Programming

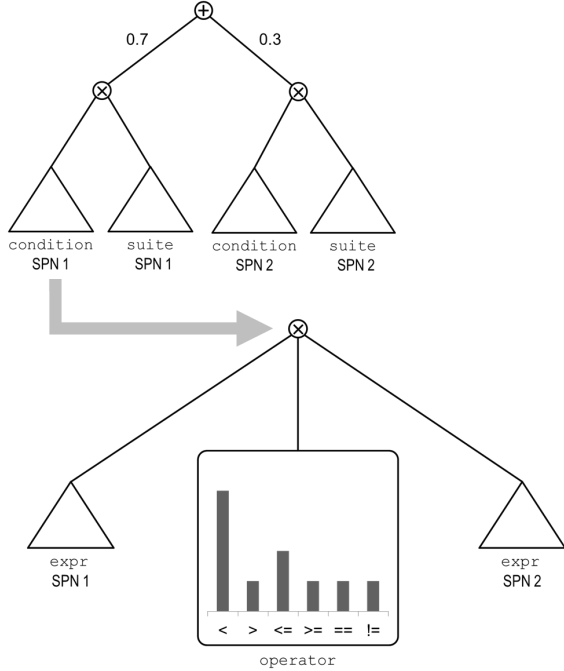
A TPP for a deterministic language L defines a probability distribution over programs in L . The TPP contains a *rule SPN* for each production rule $\alpha \rightarrow \beta$ in L ’s grammar (α is a non-terminal symbol; β is a string of terminals and non-terminals). The leaves of this SPN are univariate distributions over terminals in β , and sub-SPNs for each non-terminal in β . The SPN may also include leaf distributions over additional variables, which we refer to as *rule attributes*. These rule attributes can be used to represent variables of interest that are not part of the program itself. In the fault localization setting, the important attribute is the vari-

able ‘Buggy’, which is true iff the line contains a bug. Other informative features may also be included as attributes; for instance, one or more coverage-based suspiciousness metrics (Jones and Harrold 2005) may be included for each line.

Intuitively, the sum nodes in the SPN can be thought of as choices between alternative subprogram distributions. The product nodes capture context-specific independence between subprograms.

Example 1. The following production rules are a fragment of the grammar of a Python-like language:
`while_stmt → ‘while’ condition ‘:’ suite`
`condition → expr operator expr`

The following are simple SPNs for these two rules:



Like many statistical relational representations, a TPP is a lifted specification of a probabilistic model, and must be *grounded* for a specific mega-example. In this case, a mega-example is the parse tree of a program, along with the corresponding rule attributes. Sub-SPNs for non-terminals are grounded recursively. The resulting ground SPN is linear in the size of the parse tree, and inference on the SPN is guaranteed to be tractable.

Learning

TPPs build on ideas similar to Relational Sum-Product Networks (Nath and Domingos 2014), and their structure can be learned similarly, using an extension of the LearnSPN algorithm (Gens and Domingos 2013). Alternatively, the rule SPN structure may be fixed, and the parameters trained by EM. A simple option is to use a PCFG-like structure, where each non-terminal is modeled by a sum over a fixed number of classes, where the sum node can have different parameters in each production rule that the symbol appears in. TPPs are not restricted to PCFG-like structures, and can compactly represent some high-treewidth models (due to their ability to compactly represent context-specific independence).

An implementation of the above learning algorithm is currently in development, and we are in the process of running experiments on a corpus of buggy Python programs. We use the TARANTULA score (a coverage-based metric; Jones and Harrold 2005) as a rule attribute. We hypothesize that the learned TPPs will capture context-specific dependencies between the coverage-based metrics and the probability of a bug (e.g. a high TARANTULA score is more informative inside a top-level `if` statement than inside a `for` loop).

Current Directions and Future Work

The above discussion deals with the use of TPPs for fault localization. TPPs can enrich existing fault localization methods by learning recurring patterns of errors from a corpus of buggy programs, and combining multiple sources of information (e.g. different suspiciousness metrics).

TPPs can also be used for the more ambitious problem of fault *correction*. In the simplest case, the correct and buggy program have identical parse trees, and TPPs can be used to correct local bugs in the terminal symbols (e.g. incorrect operators or variable names). In this case, the buggy terminals are included as rule attributes, and the correct program is predicted by querying the MAP state of the true terminals.

A more challenging case is when the parse tree of the true program is unknown a priori. In this setting, fault correction can be done by searching over parse trees when grounding a TPP, instead of taking a fixed parse tree as input. To retain tractability, the search space must be restricted (e.g. re-ordering blocks of code, or allowing insertions or deletions of bounded size).

Combining these (and potentially other) approaches in a single unifying framework for automated debugging could have significant practical importance, given the large economic cost of software errors. TPPs may also have applications in natural language processing, and other domains that use grammars.

Acknowledgments

This research was partly funded by ARO grant W911NF-08-1-0242, ONR grants N00014-13-1-0720 and N00014-12-1-0312, and AFRL contract FA8750-13-2-0019.

References

- De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. Problog: A probabilistic Prolog and its application in link discovery. In *IJCAI*.
- Gens, R., and Domingos, P. 2013. Learning the structure of sum-product networks. In *ICML*.
- Goodman, N. D.; Mansinghka, V. K.; Roy, D. M.; Bonawitz, K.; and Tenenbaum, J. B. 2008. Church: a language for generative models. In *UAI*.
- Jones, J. A., and Harrold, M. J. 2005. Empirical evaluation of the Tarantula automatic fault-localization technique. In *ASE*.
- Nath, A., and Domingos, P. 2014. Learning tractable statistical relational models. In *StarAI*.
- Poon, H., and Domingos, P. 2011. Sum-product networks: A new deep architecture. In *UAI*.
- Shapiro, E. Y. 1983. *Algorithmic Program Debugging*. MIT Press.

Learning Tractable Statistical Relational Models

Aniruddh Nath and Pedro Domingos

Department of Computer Science & Engineering
University of Washington
Seattle, WA 98195, U.S.A.
{nath, pedrod}@cs.washington.edu

Abstract

Intractable inference has been a major barrier to the wide adoption of statistical relational models. Existing exact methods suffer from a lack of scalability, and approximate methods tend to be unreliable. Sum-product networks (SPNs; Poon and Domingos 2011) are a recently-proposed probabilistic architecture that guarantees tractable exact inference, even on many high-treewidth models. SPNs are a propositional architecture, treating the instances as independent and identically distributed. In this paper, we extend SPNs to the relational setting, resulting in *Relational Sum-Product Networks* (RSPNs). Previous tractable statistical relational models (Domingos and Webb 2012; Webb and Domingos 2013) defined their models over a pre-determined set of objects, and therefore could not be generalized to new mega-examples. In contrast, RSPNs can be learned and applied to previous unseen examples. We present a learning algorithm for RSPNs; in preliminary experiments, RSPNs outperform Markov Logic Networks (Richardson and Domingos 2006) in both running time and predictive accuracy.

Introduction

Graphical probabilistic models compactly represent a joint probability distribution among a set of variables. Unfortunately, inference in graphical models is intractable. In practice, using graphical models for most real-world applications requires either using approximate algorithms, or restricting oneself to a subset of graphical models on which inference is tractable. A common restriction that ensures tractability is to use models with low treewidth (e.g. Bach and Jordan 2001; Checheta and Guestrin 2007). However, not all tractable models have low treewidth. Poon and Domingos (2011) recently proposed *Sum-Product Networks* (SPNs), a tractable probabilistic architecture that guarantees efficient exact inference. SPNs subsume most known tractable probabilistic models, and can compactly represent some high-treewidth distributions. SPNs can be seen as a deep architecture with alternating layers of sum nodes and product nodes. Since their introduction, several SPN learning algorithms have been proposed, and SPNs have been applied to a variety of problems (Delalleau and Bengio 2011; Gens and Domingos 2012; Dennis and Ventura 2012; Amer

and Todorovic 2012; Peharz, Geiger, and Pernkopf 2013; Gens and Domingos 2013).

Besides intractability, one other key restriction of most widely used graphical models is their reliance on the i.i.d. assumption. In many real-world applications, training instances are not truly independent, and can be better modeled if their interactions are taken into account. This is one of the key insights of the Statistical Relational Learning (SRL) community. In recent years, SRL techniques have been applied to a wide variety of tasks, including collective classification, link prediction, vision, natural language processing, etc. However, most SRL methods are built on top of graphical models (e.g. Markov logic networks; Richardson and Domingos 2006), and suffer from the same computational difficulties; these are compounded by the additional problem of modeling interactions between instances.

In this paper, our goal is to combine these two lines of research: tractable probabilistic models (in this case, SPNs) and relational learning. One piece of related research is Flach and Lachiche’s (2004) work on Naïve Bayes classification in structured domains. Their work was limited to classification, and was very limited in expressiveness (being a Naïve Bayes model). The first non-trivial tractable probabilistic relational representation was Tractable Markov Logic (TML; Domingos and Webb 2012), a subset of Markov logic on which efficient inference could be guaranteed. TML is surprisingly expressive, subsuming most previous tractable models. However, a TML knowledge base determines the set of possible objects in the domain, and the relational structure among them. This limits the applicability of TML to learning; a TML knowledge base cannot be learned on one set of objects and applied to another mega-example with different size or structure.

To address this, we propose *Relational Sum-Product Networks* (RSPNs), a new tractable relational probabilistic architecture. RSPNs generalize SPNs by modeling a set of instances jointly, allowing them to influence each other’s probability distributions, as well as modeling the probabilities of relations between objects. We also introduce *LearnRSPN*, the first algorithm for learning tractable statistical relational models. Intractable inference has historically been a major obstacle to the wider adoption of statistical relational methods; the development of learning and inference algorithms for tractable relational models could go a long way towards

making SRL more widely applicable.

Background

Sum-Product Networks

A sum-product network (SPN) is a rooted directed acyclic graph with univariate distributions at the leaves; the internal nodes are (weighted) sums and (unweighted) products.

Definition 1. (Gens and Domingos 2013)

1. A tractable univariate distribution is an SPN.
2. A product of SPNs with disjoint scopes¹ is an SPN.
3. A weighted sum of SPNs with the same scope is an SPN, provided all weights are positive.
4. Nothing else is an SPN.

A univariate distribution is tractable iff its partition function and mode can be computed in $O(1)$ time.

Intuitively, an SPN can be thought of as an alternating set of mixtures (sums) and decompositions (products) of the leaf variables. If the values at the leaf nodes are set to the partition functions of the corresponding univariate distribution, then the value at the root is the partition function (i.e. the sum of the unnormalized probabilities of all possible assignments to the leaf variables). This allows the partition function to be computed in time linear in the size of the SPN.

Similarly, if some of the leaves are known, setting their values to their probabilities (according to the corresponding univariates) yields the unnormalized probability of the evidence. This can be divided by the partition function to obtain the normalized probability. The most probable state of the SPN, viewing sums as marginalized-out hidden variables, can also be computed in polynomial time.

The first learning algorithms for sum-product networks used a fixed network structure, and only optimized the weights (Poon and Domingos 2011; Amer and Todorovic 2012; Gens and Domingos 2012). The network structure is domain-dependent; applying SPNs to a new problem required manually designing a suitable network structure.

More recently, several algorithms have been proposed that learn both the weights and the network structure, allowing SPNs to be applied out-of-the-box to new domains. Dennis and Ventura (2012) suggested an algorithm that builds an SPN based on a hierarchical clustering of variables. Gens and Domingos (2013) construct SPNs top-down, recursively partitioning instances and variables. Peharz, Geiger, and Pernkopf (2013) construct SPNs bottom-up, greedily merging SPNs into models of larger scope. These algorithms have been shown to perform well on a variety of domains, making more accurate predictions than conventional graphical models, while guaranteeing tractable inference.

Statistical Relational Learning

SPNs are a propositional representation, modeling instances as independent and identically distributed (i.i.d.). Although the i.i.d. assumption is widely used in statistical machine learning, it is often an unrealistic assumption. In practice,

objects usually interact with each other; Statistical Relational Learning algorithms can capture dependencies between objects, and make predictions about relationships between them.

Markov Logic Networks (MLNs; Richardson and Domingos 2006) are a widely-used representation used for relational learning. An MLN is a first-order representation of the dependencies between objects in a domain. Given a *mega-example* (a set of related objects), an MLN can be *grounded* into a propositional graphical model representing a joint probability distribution over the attributes and relations among those objects. Unfortunately, the resulting graphical model is typically high treewidth, and inference is intractable. In practice, users of SRL methods typically resort to approximate inference algorithms based on MCMC or loopy belief propagation, resulting in long runtimes and unreliable predictions. Like propositional graphical models, statistical relational models can be trivially restricted to the low-treewidth case, but this comes at great cost to the representational power of the model.

Tractable Markov Logic (TML; Domingos and Webb 2012) is a subset of Markov logic that guarantees polynomial-time inference, even in certain cases where the ground propositional model would be high in treewidth. TML is expressive enough to capture several cases of interest, including junction trees, non-recursive PCFGs, hierarchical mixture models etc. A TML knowledge base is a generative model that decomposes the domain into parts, with each part drawn probabilistically from a class hierarchy. Each part is further probabilistically decomposed into subparts (according to its class). Inference in TML is carried out using probabilistic theorem proving (Gogate and Domingos 2011). The key limitation of TML is that the knowledge base fully specifies the set of possible objects in the domain, and their class and part structure. A TML knowledge base cannot be learned from one mega-example and applied to another example with different size or structure.

This paper uses some first-order logic terminology. For our purposes, ‘*predicate*’ refers to a first-order logic predicate. (Our representation and algorithm support numeric attributes and relations as well, but for simplicity we focus on the Boolean case.) A *grounding* of a predicate (or a *ground atom*) is a replacement of all its variables by constants.

Relational Sum-Product Networks

Exchangeable Distribution Templates

Before we define RSPNs, we first define the notion of an *Exchangeable Distribution Template* (EDT).

Definition 2. Consider a finite set of variables $\{X_1, \dots, X_n\}$ with joint probability distribution P . Let $S(n)$ be the set of all permutations on $\{1, \dots, n\}$. $\{X_1, \dots, X_n\}$ is a *finite exchangeable set* with respect to P if and only if $P(X_1, \dots, X_n) = P(X_{\pi(1)}, \dots, X_{\pi(n)})$ for all $\pi \in S(n)$. (Diaconis and Freedman 1980).

Note that finite exchangeability does not require independence: a set of variables can be exchangeable despite having strong dependencies. (e.g. binary variables $X_1, \dots, X_n$,

¹The scope of an SPN is the set of variables that appear in it.

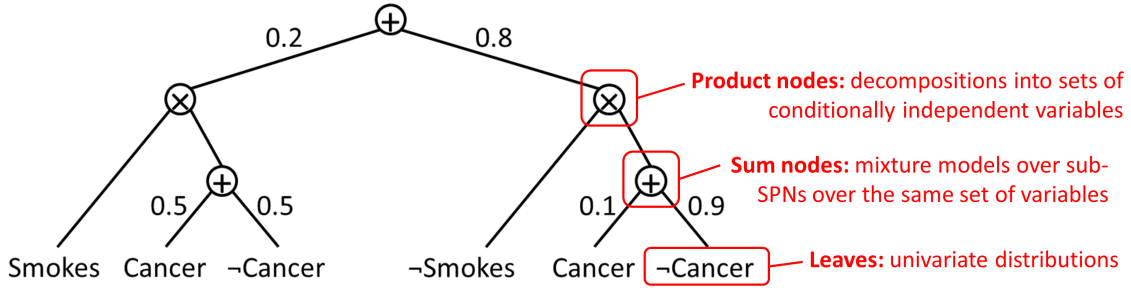


Figure 1: Example sum-product network. *Smokes* and *Cancer* are Bernoulli distributions with probability 1; $\neg$ *Smokes* and $\neg$ *Cancer* are distributions over the same two variables, with probability 0.

with a uniform probability distribution over value assignments with an even number of non-zero variables.)

Definition 3. An *Exchangeable Distribution Template* (EDT) is a function that takes a set of variables $\{X_1, \dots, X_n\}$ as input (n is unknown a priori), and returns a joint probability distribution P with respect to which $\{X_1, \dots, X_n\}$ are exchangeable. We refer to the probability distribution P returned by the EDT for a given set of variables as an *instantiation* of that EDT.

Example 1. The simplest family of EDTs simply returns a product of univariate distributions over each of $X_1, \dots, X_n$. For example, if the variables are binary, then an EDT might model them as a product of Bernoulli distributions with some probability p .

Example 2. Consider an EDT over a set of binary variables $X_1, \dots, X_n$ (with n unknown a priori), returning the following distribution:

$$P(X_1, \dots, X_n) \propto \frac{\lambda^k}{k!} e^{-\lambda} \quad \text{where} \quad k = \sum_{X_i} 1[X_i]$$

λ is a parameter of the EDT. This is an EDT with a Poisson distribution over the number of variables in the set with value 1. (The probabilities must be renormalized, since the set of variables is finite.) Note that this EDT does not assume independence among variables.

Intuitively, EDTs can be thought of as probability distributions that depend only on aggregate statistics, and not on the values of individual variables in the set.

Relational Sum-Product Networks

Relational Sum-Product Networks (RSPNs) jointly model the attributes and relations among a set of objects. RSPNs inherit TML’s notion of parts and classes. Unlike TML, an RSPN class’s parts may be *unique* or *exchangeable*. An object’s unique parts are those that play a special role, e.g. the commander of a platoon, the queen of a bee colony, or the hub of a social network. The exchangeable parts are those that behave interchangeably: soldiers in a platoon, worker bees in a colony, spokes in a network, and so on.

Definition 4. A *class definition* for RSPN class C consists of:

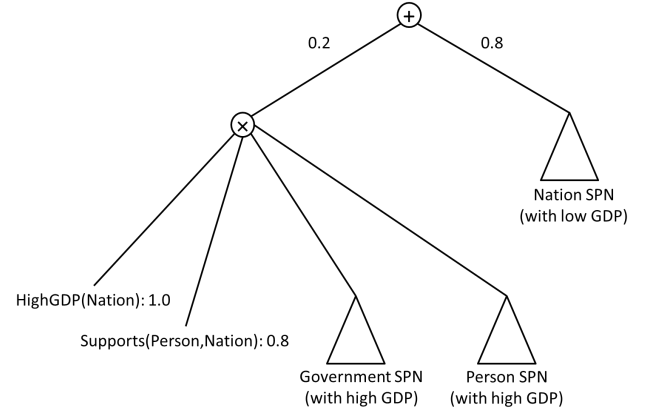


Figure 2: Partial SPN for the ‘Nation’ class (example 3), modeling all a nation’s parts as independent conditioned on the ‘HighGDP’ attribute.

- A set of *attributes*: unary predicates of the form $A(x)$, where $x \in C$.
- A set of *unique parts*, each associated with a class.
- A set of *exchangeable parts*, each associated with a class.
- A set of *relations* between subparts: predicates of the form $R_1(P_1, P_2)$ or $R_2(C, P_1)$, where P_1 and P_2 are either unique or exchangeable parts of C . (Predicates may be of any arity.)
- A *class SPN* whose leaves are in one of three sets:
 - $U_A^{(C)}$, univariate distributions over unary predicates involving C (i.e. attributes of the form $A(C)$);
 - $U_R^{(C)}$, EDTs over groundings of binary (or higher-order) predicates involving C and/or its subpart types (e.g. relations of the form $R_1(P_1, P_2)$ or $R_2(C, P_1)$, where P_1 and P_2 are parts of C);
 - $U_P^{(C)}$, sub-SPNs for subparts.

Example 3. The following is a partial class specification for a simple political domain. A ‘Region’ consists of an arbitrary number of nations, and relationships between nations are modeled at this level. A ‘Nation’ has a unique government and an arbitrary number of people. National properties

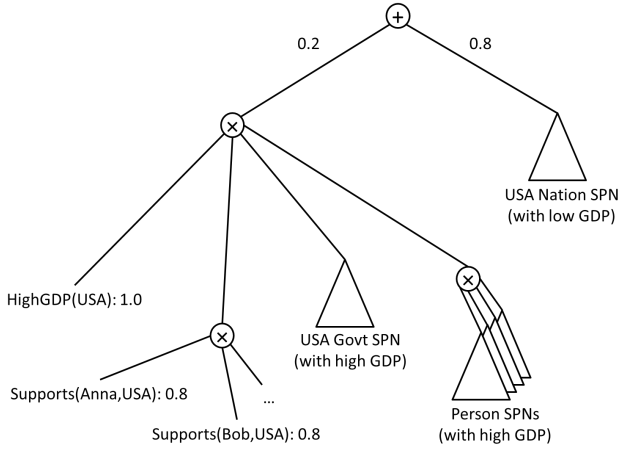


Figure 3: Example grounding of the ‘Nation’ class, with the class SPN from figure 2.

such as ‘High GDP’ are modeled here. The ‘Supports’ relation can capture a distribution over the number of people in the nation who support the government.

```
class Region:
    exchangeable part Nation
    relation Adjacent(Nation, Nation)
    relation Conflict(Nation, Nation)

class Nation:
    unique part Government
    exchangeable part Person
    attribute HighGDP
    relation Supports(Person, Government)
```

See figure 2 for an example class SPN.

Grounding an RSPN

Like MLNs, RSPNs are templates for propositional models. To generate a ground SPN from an RSPN, we take as input a *part decomposition*, which is a tree of typed objects. A part decomposition D is consistent with RSPN class C iff:

- The root object of D is of class C ;
- The root object has one child o_u for each unique part C_u of C , and the subtree rooted at o_u is consistent with C_u ;
- Each remaining child object o_e is of a type matching some exchangeable part C_e of C , and the subtree rooted at o_e is consistent with C_e .

To *ground* a class SPN is to instantiate the template for a specific set of objects. Given a class C and a part decomposition D (rooted at object o), grounding C ’s SPN yields a propositional SPN whose leaf distributions are over attributes and relations involving the objects in D . This is done recursively as follows:

- For leaves in $U_A^{(C)}$: replace the univariate distribution over predicate $A(C)$ in the class SPN with a univariate distribution over $A(o)$ in the ground SPN.

- For leaves in $U_R^{(C)}$: replace the EDT over $R(X, Y)$ in the class SPN with an instantiation of that EDT over the groundings of R .
- For leaves in $U_P^{(C)}$: recursively ground P ’s class SPN over each type- P child of object o . Replace the sub-SPN in C ’s SPN with a product over groundings of P .

See figure 3 for an example of a ground SPN.

Representational Power

The main limitation of the RSPN representation is that individual relational atoms are not modeled directly, but through aggregations. In this respect, it is similar to Probabilistic Relational Models (PRMs, Friedman et al. 1999)—though RSPNs guarantee tractable inference, unlike PRMs. Aggregations are extremely useful for capturing relational dependencies (Natarajan et al. 2012). For example, a person’s smoking habits may depend on the number of friends she has who smoke, and not the smoking habits of each individual friend. Nevertheless, aggregations are not well-suited to capturing relational patterns that depend on specific paths of influence, such as Ising models.

It is important to note that RSPNs (and SPNs) are not simply tree-structured graphical models. The graph of an RSPN is not a conditional dependency graph, but a graphical representation of the computation of the partition function. A ground RSPN can be converted into an equivalent graphical model, but the resulting model may be high-treewidth, and computationally intractable.

In effect, RSPNs are a way to compactly represent context-specific independences in relational domains: different children of a sum node may have different variable decompositions in their product nodes. These context-specific independences are what give RSPNs more expressiveness than low-treewidth tractable models.

Learning RSPNs

The learning task for RSPNs is to determine the structure and parameters of all the class SPNs in the domain. In this work, we assume that the part relationships among classes are known, i.e. the user determines what types of unique and exchangeable parts are allowed for each class. The input for the learning algorithm is a set of part decompositions (trees of objects), and an evidence database specifying the values of the attributes and relations among those objects.

Our learning algorithm is based on the top-down LearnSPN algorithm (Gens and Domingos 2013). LearnSPN is a propositional SPN learning algorithm, taking as input a set of training instances T and variables V . It attempts to decompose V into independent subsets $V_1, \dots, V_k$ (using pairwise statistical independence tests); if successful, LearnSPN recurses over each subset, returning a product node over the recursively learned sub-SPNs. If V does not decompose into independent subsets, LearnSPN instead clusters the instances T , recursively learns a sub-SPN over each subset $T_1, \dots, T_k$, returning a sum-node over the sub-SPNs weighted by the mixture proportions. Under certain assumptions, LearnSPN can be seen as a greedy search maximizing the likelihood of the learned SPN.

```

input:  $C$ , a class
          $T$ , a set of instances of  $C$ 
          $V$ , a set of class attributes, relation aggregates,
         and subparts

output: an RSPN for class  $C$ 

if  $|V| = 1$  then
    if  $v \in V$  is an attribute then
        return univariate estimated from  $v$ 's values in  $T$ 
    else if  $v$  is a relation aggregate then
        return EDT estimated from  $v$ 's values in  $T$ 
    else
        //  $v$  is a subpart
         $C_{child} \leftarrow$  class of  $v$ 
         $T_{child} \leftarrow$  subparts of  $t \in T$  of type  $C_{child}$ 
         $V_{child} \leftarrow$  attributes, relations and subparts of  $C_{child}$ 
        return LearnRSPN( $C_{child}, T_{child}, V_{child}$ )
    end if
else
    partition  $V$  into approximately independent subsets  $V_j$ 
    if success then
        return  $\prod_j$  LearnRSPN( $C, T, V_j$ )
    else
        partition  $T$  into sets of similar subsets  $T_i$ 
        return  $\sum_i \frac{|T_i|}{|T|} \cdot \text{LearnRSPN}(C, T_i, V)$ 
    end if
end if

```

LearnRSPN is a top-down algorithm similar to LearnSPN; it attempts to find independent subsets from among the object’s set of attributes, relations and subparts; if multiple subsets exist, the algorithm learns a sub-SPN over each subset, and returns a product over the sub-SPNs. If independent subsets cannot be found, LearnRSPN instead clusters the instances, returning a sum node over the components, weighted by the mixture proportions.

The base case of LearnRSPN (when $|V| = 1$) varies depending on what v is. When v is an attribute, the RSPN to be returned is simply a univariate distribution over the attribute, as in the propositional version of LearnSPN. When v is a subpart of class C_{child} , LearnRSPN returns an SPN for class C_{child} . Crucially, different SPNs are learned for C_{child} in different children of a sum node in parent class C (since the recursive call is made with a different set of instances). The final base case is when v is an aggregate over an exchangeable relation. In this case, the RSPN to be returned is an EDT over the relation.

Like LearnSPN, LearnRSPN can be seen as an algorithm schema rather than a single learning algorithm; the user is free to choose a clustering algorithm for instances, a dependency test for variable splitting, an aggregate statistic, and a family of EDTs for exchangeable relations. Note that different families of EDTs may require different aggregate statistics for parameter estimation. The fraction of true groundings is sufficient for the two EDTs described earlier.

Methodology

To cluster instances in LearnRSPN, we used the EM implementation in SCIKIT-LEARN (Pedregosa et al. 2011), with two clusters. To test dependence between variables, we fit a Gaussian distribution (for aggregate variables) or Bernoulli distribution (for binary attributes), and computed the pairwise mutual information (MI) between the variables. The test statistic used was $G = 2N \times MI$, which in the discrete case is equivalent to the G-test used by Gens and Domingos (2013). We used a threshold of 0.5 for dependence. For EDTs, we used the independent Bernoulli form, as described in example 1. All Bernoulli distributions were smoothed with a pseudocount of 0.1.

We report results in terms of area under the precision-recall curve (AUC; Davis and Goadrich 2006) and the average conditional marginal likelihood (CMLL) of the test atoms. AUC is a measure of prediction quality that is insensitive to imbalance in the number of true and false atoms. CMLL directly measures the quality of the probability estimates. For LearnRSPN, we also report the test set likelihood normalized by the number of queries, as an alternate measure of prediction quality. (ALCHEMY does not compute this quantity.)

66

Table 1: UW-CSE results. The five areas are AI, Graphics, Languages, Systems and Theory.

Area	Q	Learning time (s)		Inference time (s)		AUC-PR			CMLL			LL/ Q
		RSPN	MLN	RSPN	MLN	RSPN	MLN	BS	RSPN	MLN	BS	RSPN
AI	1414	8.85	12,137.74	0.10	5.88	0.73	0.25	0.25	-0.09	-0.19	-0.16	-0.09
Grph.	171	8.60	6,464.82	0.03	0.55	0.63	0.44	0.26	-0.25	-0.31	-0.30	-0.24
Lang.	17	8.46	20,294.36	0.01	0.04	1.00	0.42	0.43	-0.10	-1.35	-0.37	-0.07
Sys.	1120	10.90	9,106.62	0.09	6.99	0.54	0.30	0.13	-0.09	-0.17	-0.14	-0.09
Th.	308	6.93	17,435.36	0.03	1.11	0.75	0.30	0.33	-0.13	-0.37	-0.21	-0.12
Avg.	606	8.75	13,087.80	0.05	2.91	0.73	0.34	0.27	-0.13	-0.47	-0.23	-0.12

Experiment

The UW-CSE database (Richardson and Domingos 2006) has been used to evaluate a variety of statistical relational learning algorithms. The dataset describes the University of Washington Compute Science & Engineering department, and includes advising relationships, paper authorships etc. The database is divided into five non-overlapping mega-examples, by research area.

To generate a part structure for this domain, we separated the people into one research group per faculty member, with students determined using the *AdvisedBy* and *TempAdvisedBy* predicates (breaking ties by number of coauthored papers). Publications are also divided among groups: each paper is assigned to the group of the professor who wrote it, voting by the number of student authors in the group in the event of a tie. The prediction tasks are to infer the roles of faculty (Professor, Associate Professor or Assistant Professor) and students (Pre-Quals, Post-Quals, Post-Generals), as well as paper authorships. The part structure is also made available to ALCHEMY in the form of predicates *Has(Area, Group)*, *Has(Group, Professor)*, *Has(Group, Student)*, *Has_Group(Group, Paper)*, and *Has_NonGroup(Group, Paper)*.

We performed leave-one-out testing by area, testing on each area in turn using the model trained from the remaining four. 80% of the groundings of the query predicates were provided as evidence, and the task was to predict the remaining atoms. Table 1 shows the results on all five areas, and the average. LearnRSPN is orders of magnitude faster than ALCHEMY at both training and inference time, and considerably more accurate than either ALCHEMY or the baseline.

Discussion

Statistical relational learning algorithms have been successfully applied to several problems, but the difficulty, cost and unreliability of approximate inference has limited their wider adoption. In practice, applying statistical relational methods to a new domain requires substantial engineering effort in choosing and configuring the approximate learning and inference algorithms. The expressiveness of languages like Markov logic is both a boon and a curse: although these languages can compactly represent sophisticated probabilistic models, they also make it easy for practitioners to unintentionally design models too complex even for state-of-the-art inference algorithms.

In the propositional setting, several approaches have been recently proposed for learning high-treewidth tractable mod-

```

class Area:
    exchangeable part Group

class Group:
    unique part Professor
    exchangeable part Student
    exchangeable part GroupPaper
    exchangeable part NonGroupPaper
    relation Author(Professor, GroupPaper)
    relation Author(Professor, NonGroupPaper)
    relation Author(Student, GroupPaper)
    relation Author(Student, NonGroupPaper)

class Professor:
    attribute Position_Faculty
    attribute Position_Adjunct
    attribute Position_Affiliate

class Student:
    attribute InPhase_PreQuals
    attribute InPhase_PostQuals
    attribute InPhase_PostGenerals

```

Figure 4: Part structure for UW-CSE domain

els (Lowd and Domingos 2008; Roth and Samdani 2009; Gogate, Webb, and Domingos 2010; Poon and Domingos 2011). To our knowledge, LearnRSPN is the first algorithm for learning high-treewidth tractable relational models. (TML is also a high-treewidth relational representation, but is unsuitable for learning, as explained earlier.) In preliminary experiments, LearnRSPN outperforms conventional statistical relational methods in accuracy, inference time and training time.

A limitation of RSPNs is that they require a known, fixed part decomposition for all training and test mega-examples. Applying RSPNs to a new domain requires the user to specify the part decomposition; this is analogous to specifying the class structure in object-oriented programming. Many domains have a natural part structure that can be exploited (like the UW-CSE database); in other cases, part structure can be created using existing graph-cut or community detection algorithms. An important direction for future work is to develop an efficient, principled method of finding part structure in a database.

Acknowledgments

This research was partly funded by ARO grant W911NF-08-1-0242, ONR grants N00014-13-1-0720 and N00014-12-1-0312, and AFRL contract FA8750-13-2-0019. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, ONR, AFRL, or the United States Government.

References

- Amer, M. R., and Todorovic, S. 2012. Sum-product networks for modeling activities with stochastic structure. In *CVPR*.
- Bach, F., and Jordan, M. I. 2001. Thin junction trees. In *NIPS*.
- Checheta, A., and Guestrin, C. 2007. Efficient principled learning of thin junction trees. In *NIPS*.
- Davis, J., and Goadrich, M. 2006. The relationship between precision-recall and ROC curves. In *ICML*.
- Delalleau, O., and Bengio, Y. 2011. Shallow vs. deep sum-product networks. In *NIPS*.
- Dennis, A., and Ventura, D. 2012. Learning the architecture of sum-product networks using clustering on variables. In *NIPS*.
- Diaconis, P., and Freedman, D. 1980. De Finetti's generalizations of exchangeability. *Studies in Inductive Logic & Probability* 2.
- Domingos, P., and Webb, W. A. 2012. A tractable first-order probabilistic logic. In *AAAI*.
- Flach, P., and Lachiche, N. 2004. Naive Bayesian classification of structured data. *Machine Learning* 57.
- Friedman, N.; Getoor, L.; Koller, D.; and Pfeffer, A. 1999. Learning probabilistic relational models. In *IJCAI*.
- Gens, R., and Domingos, P. 2012. Discriminative learning of sum-product networks. In *NIPS*.
- Gens, R., and Domingos, P. 2013. Learning the structure of sum-product networks. In *ICML*.
- Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *UAI*.
- Gogate, V.; Webb, W. A.; and Domingos, P. 2010. Learning efficient Markov networks. In *NIPS*.
- Kok, S., and Domingos, P. 2005. Learning the structure of Markov logic networks. In *ICML*.
- Kok, S.; Sumner, M.; Richardson, M.; Singla, P.; Poon, H.; Lowd, D.; Wang, J.; and Domingos, P. 2008. The Alchemy system for statistical relational AI. Technical report, University of Washington. <http://alchemy.cs.washington.edu>.
- Lowd, D., and Domingos, P. 2008. Learning arithmetic circuits. In *UAI*.
- Natarajan, S.; Khot, T.; Kersting, K.; Gutmann, B.; and Shavlik, J. 2012. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning* 86.
- Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12.
- Peharz, R.; Geiger, B. C.; and Pernkopf, F. 2013. Greedy part-wise learning of sum-product networks. In *ECML-PKDD*.
- Poon, H., and Domingos, P. 2011. Sum-product networks: A new deep architecture. In *UAI*.
- Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62.
- Roth, D., and Samdani, R. 2009. Learning multi-linear representations of distributions for efficient inference. *Machine Learning* 76.
- Webb, W. A., and Domingos, P. 2013. Tractable probabilistic knowledge bases with existence uncertainty. In *StaR-AI*.

Tractable Probabilistic Knowledge Bases: Wikipedia and Beyond

Mathias Niepert and **Pedro Domingos**

Computer Science & Engineering
University of Washington
Seattle, WA 98195-2350, USA

Abstract

Building large-scale knowledge bases from a variety of data sources is a longstanding goal of AI research. However, existing approaches either ignore the uncertainty inherent to knowledge extracted from text, the web, and other sources, or lack a consistent probabilistic semantics with tractable inference. To address this problem, we present a framework for tractable probabilistic knowledge bases (TPKBs). TPKBs consist of a hierarchy of classes of objects and a hierarchy of classes of object pairs such that attributes and relations are independent conditioned on those classes. These characteristics facilitate both tractable probabilistic reasoning and tractable maximum-likelihood parameter learning. TPKBs feature a rich query language that allows one to express and infer complex relationships between classes, relations, objects, and their attributes. The queries are translated to sequences of operations in a relational database facilitating query execution times in the sub-second range. We demonstrate the power of TPKBs by leveraging large data sets extracted from Wikipedia to learn their structure and parameters. The resulting TPKB models a distribution over millions of objects and billions of parameters. We apply the TPKB to entity resolution and object linking problems and show that the TPKB can accurately align large knowledge bases and integrate triples from open IE projects.

Introduction

A knowledge base continuously acquiring more knowledge and answering complex queries is a vision as old as the field of AI itself. Such a knowledge base would address numerous pressing problems such as data integration, information extraction, and entity resolution. However, there are three mutually conflicting properties that pertain to all KBs: (a) the ability to populate the KB from (uncertain) data; (b) the tractability of reasoning; and (c) the expressiveness of the underlying logic. When the uncertainty is modeled with a probabilistic semantics, the problem of tractability and learnability is even more challenging. Recent advances in tractable probabilistic models, however, have shown that there is the possibility of new and better trade-offs between these conflicting properties.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

With this paper, we present a novel framework for tractable probabilistic knowledge bases (TPKBs) that addresses some of the aforementioned challenges. The TPKB facilitates reasoning over inheritance hierarchies and on the level of classes and relations; it provides tractable query answering via a mapping to queries in a relational database; and it features a complex query language. Instead of merely making the model more expressive, we focus on a model that is expressive while featuring a rich query language. We also emphasize the feasibility of parameter learning using large data sets and the ability to model uncertainty of class and relation hierarchies. The tractability is due to the TPKB's property that attributes of objects and object pairs are independent conditioned on classes of objects and object pairs.

Problem Statement and Contributions

In this paper, we say that a (probabilistic) knowledge base is *tractable* if computing the answer to queries takes time polynomial in the size of the PKB's data. Hence, we are concerned with polynomial data complexity (Abiteboul, Hull, and Vianu 1995). There is a large body of work on fragments of first-order logic with polynomial data complexity. Most notably, the databases literature identified numerous tractable fragments for which query processing has polynomial data complexity. The understanding of tractable fragments of first-order logic with probabilistic semantics, however, is not as thoroughly developed. More recently, work on probabilistic databases (Suciu et al. 2011) has characterized classes of unions of conjunctive queries for which query answering is possible in polynomial time. The assumption in probabilistic databases is that the instantiations of a relation (its tuples) are mutually independent. Unfortunately, even in this seemingly restrictive setting, numerous natural conjunctive queries are NP-hard. For instance, consider the query

$$\exists x, y. \text{student}(x, y), \text{bDate}(x, 1950), \text{almaMater}(y, \text{Yale})$$

that asks for the probability of there existing a pairs of objects (x, y) such that x was born in 1950, y 's alma mater is Yale, and x was a student of y . This is a natural query that could, for instance, occur as part of an entity disambiguation problem. However, it is known that this query is *not* generally tractable under the assumption of tuple independence (Suciu et al. 2011). Therefore, one ought to be careful when designing probabilistic knowledge bases that need to be both tractable and useful in practice. Contrary to probabilistic databases, a probabilistic knowledge base models

Type	PKB expression	FOL expression	RDF expression
Schema	$\text{sub}_c(C, D)$ $\text{sub}_r(r, s)$	$\forall x : C(x) \Rightarrow D(x)$ $\forall x, y : r(x, y) \Rightarrow s(x, y)$	$C \text{ rdfs:subclassOf } D$ $r \text{ rdfs:subPropertyOf } s$
Assertion	$\text{ins}_c(\text{Einstein}, \text{Person})$ $\text{ins}_r(\text{Einstein}, \text{Kleiner}, \text{advisor})$ $\text{attr}_o(\text{Einstein}, 1879, \text{birthYear})$ $\text{attr}_{oo}(\text{Eugene}, \text{Seattle}, 283, \text{distance})$	$\text{Person}(\text{Einstein})$ $\text{advisor}(\text{Einstein}, \text{Kleiner})$ $\text{birthYear}(\text{Einstein}, 1879)$ $\text{distance}(\text{Eugene}, \text{Seattle}, 283)$	$\text{Einstein rdfs:type Person}$ $\text{Einstein advisor Kleiner}$ $\text{Einstein birthYear 1879}$ reification

Table 1: Example expressions in PKB syntax with corresponding expressions in first-order logic and the resource description framework (RDF). Unlike probabilistic databases, PKBs allow queries involving schema expressions.

and reasons over uncertain inheritance hierarchies of classes and relations, that is, classes and relations can occur in the answer set of queries. Hence, while probabilistic databases are in some ways not restrictive enough they are too restrictive in other ways.

In this paper, we present a tractable probabilistic knowledge base (TPKB) that facilitates efficient query answering. Intuitively, the PKB is a probabilistic model that interconnects tractable local distributions pertaining to objects, classes, and relations in a consistent and tractable manner. The novel TPKB has the following properties

- tractable learning from existing, very large data sets;
- tractable answering of complex queries;
- employment of robust relational database technology;
- default reasoning in the presence of missing data.

We provide experimental results that demonstrate that TPKBs can be applied to common large-scale problems such as entity linking and entity resolution. This shows for the first time that a PKB over millions of objects and billions of parameters can be efficiently queried.

Related Work

PKBs are different from probabilistic databases (Suciu et al. 2011) in that PKBs represent class and relation hierarchies. Moreover, each query over the PKB corresponds to an efficient query in a (probabilistic) database. Hence, every query in the PKB is tractable and an efficient query plan does not have to be generated by a process that can take exponential time.

TPKBs are related to PROBLOG (Raedt, Kimmig, and Toivonen 2007) a probabilistic logic programming language. However, PROBLOG is not tractable and does not support existentially quantified conjunctive queries. URDF (Nakashole et al. 2012) is based on weighted MAX-SAT algorithms and is in principle intractable. It also does not support queries that ask for a distribution over objects, classes, and relations, given a subset of their attributes. Infinite relational models (IRMs) are non-parametric hierarchical models but are not tractable in general (Kemp et al. 2006). We leverage the hierarchy of TPKBs to estimate parameters on the level of classes and relation more robustly. This is related to shrinkage in text classification (McCallum et al. 1998), back off models (Katz 1987), and multilevel models in regression analysis (Gelman and Hill 2007).

Open information extraction (Etzioni et al. 2011) and other IE projects (Carlson et al. 2010; Lehmann et al. 2012) often use ad-hoc approaches and heuristics and do not provide a consistent joint distribution and query language. There exist several statistical relational systems employing relational database technology to facilitate queries over structured data (Wang et al. 2010; Noessner, Niepert, and Stuckenschmidt 2013). However, the proposed systems are intractable in general. Recent work on statistical relational learning has focused on tractable probabilistic graphical models, that is, probabilistic models for which inference is efficient by design. Examples are PRISM (Sato and Kameya 1997), tractable Markov logic (Domingos and Webb 2012), particular tractable fragments of probabilistic logics (Van den Broeck 2011; Niepert and Van den Broeck 2014), and probabilistic soft logic (Kimmig et al. 2012). None of these languages features complex query languages *and* uncertain inheritance hierarchies.

Probabilistic description logic programs (Lukasiewicz 2007) combine DL programs under the answer set and well-founded semantics with independent choice logic (Poole 2008). Particular variants of light-weight description logics with probabilistic semantics (Gutiérrez-Basulto et al. 2011; Niepert, Noessner, and Stuckenschmidt 2011; Noessner and Niepert 2011) have been proposed. However, these formalisms are too expressive to be tractable for the types of complex queries needed in large-scale applications, do not allow the modeling of numerical attributes, and do not address the problem of parameter learning.

There is related work in the context of information extraction and relation learning. Notable representative publications of this line of research are tensor factorizations of YAGO (Nickel, Tresp, and Kriegel 2012) and universal schemas (Riedel et al. 2013). These approaches do not facilitate a consistent probabilistic semantics and expressive query languages.

A Tractable Probabilistic Knowledge Base

A probabilistic knowledge base (PKB) models a probability distribution over possible worlds where each such possible world models an instantiation of a logical knowledge base. The design of a tractable PKB, therefore, involves (a) the logical characterization of a possible world and (b) a parameterization of possible worlds rendering probabilistic reasoning tractable. Table 1 lists several expressions in PKB syntax and in first-order logic and simplified RDF syntax.

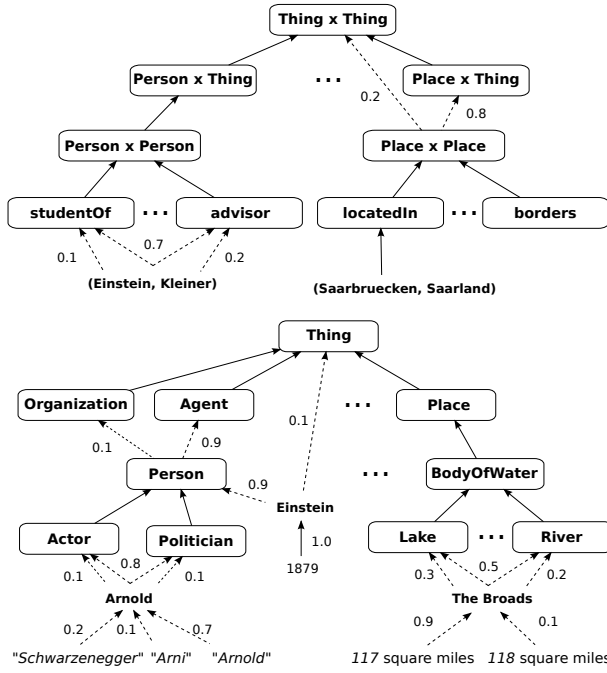


Figure 1: The TPKB models a distribution over possible worlds where each such world corresponds to a fully materialized hierarchical knowledge base modeling objects, classes, relations, and properties. Arrows correspond to *prior* conditional probabilities for subclass, subrelation, class, relation, and property assertions. In each possible world and for every entity, exactly one of the assertions modeled by outgoing dashed arrows holds. Solid arrows represent deterministic assertions.

The chosen syntax facilitates a seamless knowledge transfer from existing logical KB formalisms. We define the syntax and semantics of the novel TPKB, its parameterization, and prove tractability results.

Syntax

The domain of a PKB $\mathcal{K}$ is a set of symbols $\mathbf{K}$ defined as the union of a set of object names $\mathbf{O}$, a set of class names $\mathbf{C}$, a set of attribute names $\mathbf{A}$, and a set of binary relation names $\mathbf{R}$. Object names $\mathbf{O}$ represent individuals such as AlbertEinstein and MountEverest. Class names $\mathbf{C}$ represent sets of objects such as Person and Mountain. There is a unique class name Thing that represents the set of all objects. Binary relation names $\mathbf{R}$ represent subsets of $\text{Thing} \times \text{Thing}$, that is, classes of pairs of objects, such as studentOf and actedIn. Attribute names $\mathbf{A}$ represent properties of objects such as birthYear, properties of classes such as medianAge, and properties of relations such as surfaceForm. In addition, we write $\mathbf{A}(x)$ to denote the set of attributes of entity x and $\mathbf{A}(x, y)$ to denote the set of attributes of object pair $x, y \in \mathbf{O}$. Moreover, for every $\mathbf{A}$, we assume that the set of possible values $\text{Val}(\mathbf{A})$ is finite. The difference between attributes and relations is that the former relate entities to

1. The dsub_c and dsub_r assertions form trees on $\mathbf{C}$ and $\mathbf{R}$ with root nodes Thing and $\text{Thing} \times \text{Thing}$, respectively.
2. For all $x, y, z \in \mathbf{C}$:
 - (a) $\text{dsub}_c(x, y) \in \mathbf{W} \Rightarrow \text{sub}_c(x, y) \in \mathbf{W}$;
 - (b) $\text{sub}_c(x, y) \in \mathbf{W} \wedge \text{sub}_c(y, z) \in \mathbf{W} \Rightarrow \text{sub}_c(x, z) \in \mathbf{W}$;
 - (c) $\forall \mathbf{A} \in \mathbf{A}(x)$ there exists exactly one $\mathbf{V} \in \text{Val}(\mathbf{A})$ with $\text{attr}_c(x, \mathbf{V}, \mathbf{A}) \in \mathbf{W}$.
3. For all $x, y, z \in \mathbf{R}$:
 - (a) $\text{dsub}_r(x, y) \in \mathbf{W} \Rightarrow \text{sub}_r(x, y) \in \mathbf{W}$;
 - (b) $\text{sub}_r(x, y) \in \mathbf{W} \wedge \text{sub}_r(y, z) \in \mathbf{W} \Rightarrow \text{sub}_r(x, z) \in \mathbf{W}$;
 - (c) $\forall \mathbf{A} \in \mathbf{A}(x)$ there exists exactly one $\mathbf{V} \in \text{Val}(\mathbf{A})$ with $\text{attr}_r(x, \mathbf{V}, \mathbf{A}) \in \mathbf{W}$.
4. For all $x \in \mathbf{O}$:
 - (a) $\exists \mathbf{C} \in \mathbf{C} : \text{dins}_c(x, \mathbf{C}) \in \mathbf{W}$;
 - (b) $\forall \mathbf{C} \in \mathbf{C} : \text{dins}_c(x, \mathbf{C}) \in \mathbf{W} \Rightarrow \text{ins}_c(x, \mathbf{C}) \in \mathbf{W}$;
 - (c) $\forall \mathbf{C}, \mathbf{C}' : \text{ins}_c(x, \mathbf{C}) \in \mathbf{W} \wedge \text{sub}_c(\mathbf{C}, \mathbf{C}') \in \mathbf{W} \Rightarrow \text{ins}_c(x, \mathbf{C}') \in \mathbf{W}$;
 - (d) $\forall \mathbf{A} \in \mathbf{A}(x)$ there exists exactly one $\mathbf{V} \in \text{Val}(\mathbf{A})$ with $\text{attr}_o(x, \mathbf{V}, \mathbf{A}) \in \mathbf{W}$.
5. For all $x, y \in \mathbf{O}$:
 - (a) $\exists \mathbf{R} \in \mathbf{R} : \text{dins}_r(x, y, \mathbf{R}) \in \mathbf{W}$;
 - (b) $\forall \mathbf{R} : \text{dins}_r(x, y, \mathbf{R}) \in \mathbf{W} \Rightarrow \text{ins}_r(x, y, \mathbf{R}) \in \mathbf{W}$;
 - (c) $\forall \mathbf{R}, \mathbf{R}' : \text{ins}_r(x, y, \mathbf{R}) \in \mathbf{W} \wedge \text{sub}_r(\mathbf{R}, \mathbf{R}') \in \mathbf{W} \Rightarrow \text{ins}_r(x, y, \mathbf{R}') \in \mathbf{W}$;
 - (d) $\forall \mathbf{A} \in \mathbf{A}(x, y)$ there exists exactly one $\mathbf{V} \in \text{Val}(\mathbf{A})$ with $\text{attr}_{oo}(x, y, \mathbf{V}, \mathbf{A}) \in \mathbf{W}$.

Figure 2: Axiomatization of possible worlds.

values of a particular data type such as the integers or the set of all strings while the latter relate objects to objects. There are 8 predicate symbols $\text{ins}_c(x, y)$, $\text{ins}_r(x, y, z)$, $\text{attr}_o(x, y, z)$, $\text{attr}_{oo}(x, y, z, z')$, $\text{attr}_c(x, y, z)$, $\text{attr}_r(x, y, z)$, $\text{sub}_c(x, y)$, and $\text{sub}_r(x, y)$ representing class, relation, attribute, subclass, and subrelation assertions. Note that classes and relations are reified allowing us to express arbitrary relations between object pairs. Table 1 depicts some example expressions formed with these predicates. There are 4 additional predicates $\text{dsub}_c(x, y)$, $\text{dsub}_r(x, y)$, $\text{dins}_c(x, y)$, and $\text{dins}_r(x, y, z)$ whose sole purpose is to model *prior* conditional probabilities for subclass, subrelation, and instance assertions. A grounding of a predicate is obtained by replacing each variable with the appropriate symbols in $\mathbf{K}$. The Herbrand base of $\mathbf{K}$ is the set of all ground predicates. Each subset of the Herbrand base is a Herbrand interpretation specifying which ground predicates are true. We make the closed (active) domain assumption meaning that quantifiers range over elements in $\mathbf{K}$ (Abiteboul, Hull, and Vianu 1995).

Semantics

Every TPKB's joint distribution is fully characterized by two components. First, a set of first-order formulas $\mathbf{T}$ that axiomatize the Herbrand models (possible worlds) of the TPKB. A Herbrand model $\mathbf{W}$ is a Herbrand interpretation

that satisfies all formulas in $\mathbf{T}$. We use the words Herbrand model and possible world interchangeably. Second, a set of *prior* conditional probabilities for knowledge base assertions. Prior probabilities *might differ* from the posterior probabilities obtained via inference as the axiomatization introduces complex dependencies between assertions.

There are several possible ways to specify an axiomatization of possible worlds and a parameterization in form of prior probabilities. We focus on one particular combination that results in tractable probabilistic inference. First, we characterize a possible world $\mathbf{W}$ with the axioms in Figure 2. These axioms make every possible world corresponds to a fully materialized description logic based ontology with classes, roles (binary predicates), individuals (objects), and concrete domains (attributes) (Baader et al. 2003).

Second, we specify *prior* conditional probabilities for assertions of the knowledge base. For every $C \in \mathbf{C} \setminus \{\text{Thing}\}$, $P(\text{dsub}_c(C, C'))$ is the *prior* probability of an object being an instance of class C' given that it is an instance of class C . Since we specify conditional distributions, we must have that $\sum_{C' \in \mathbf{C}} P(\text{dsub}_c(C, C')) = 1$, for every $C \in \mathbf{C}$, and we write $P(\text{dsub}_c(C, \cdot))$ for this conditional distribution. For example, the probabilistic KB depicted in Figure 1 has the prior probabilities $P(\text{dsub}_c(\text{Person}, \text{Organization})) = 0.1$, $P(\text{dsub}_c(\text{Person}, \text{Agent})) = 0.9$, and $P(\text{dsub}_c(\text{Person}, C')) = 0.0$ for all other $C' \in \mathbf{C}$. In this example, the *prior* probability of $\text{dsub}_c(\text{Person}, \text{Thing})$ is 0.0 but the *posterior* probability of $\text{sub}_c(\text{Person}, \text{Thing})$ is 1.0 which is due to the axiomatization of a possible world in Figure 2. We have to impose a mild restriction on the *prior* probabilities to make them compatible with the axiomatization of possible worlds. We assume that we can partition the classes $\mathbf{C}$ into sets $C_1, \dots, C_k$ with $C_1 = \{\text{Thing}\}$ such that for all $1 \leq i \leq j \leq k$, if $C_i \in C_i$ and $C_j \in C_j$ then $P(\text{dsub}_c(C_i, C_j)) = 0$. Stratification ensures that, in every possible world, classes in stratum C_i can only be subclasses of classes in strata C_j with $j < i$ which, in turn, ensures that the dsub_c assertions form a tree. We refer to this restriction on the parameterization as *stratification* and to k as the stratification size. For every $R \in \mathbf{R} \setminus \{\text{Thing} \times \text{Thing}\}$, $P(\text{dsub}_r(R, \cdot))$ is the prior conditional distribution for relations with stratification restrictions identical to those made for classes.

For every $0 \in \mathbf{O}$, $P(\text{dins}_c(0, C_1) \wedge \dots \wedge \text{dins}_c(0, C_n))$ is the *prior* probability of object 0 being an instance of exactly the classes $C_1, \dots, C_n$. We must have that $\sum_{C' \subseteq \mathbf{C}} P(\bigwedge_{C' \in C'} \text{dins}_c(0, C')) = 1$ for every $0 \in \mathbf{O}$. For example, the probabilistic KB depicted in Figure 1 has the prior probabilities $P(\text{dins}_c(\text{Arnold}, \text{Actor})) = 0.1$, $P(\text{dins}_c(\text{Arnold}, \text{Politician}) \wedge \text{dins}_c(\text{Arnold}, \text{Actor})) = 0.8$, $P(\text{dins}_c(\text{Arnold}, \text{Politician})) = 0.1$, and $P(\bigwedge_{C' \in C'} \text{dins}_c(\text{Arnold}, C')) = 0$ for all other $C' \subseteq \mathbf{C}$. In order to ensure tractability, we have to impose two restrictions. First, we have to ensure that every object is an instance of at least one class. Hence, there must exists a $C' \subseteq \mathbf{C}$ with $P(\bigwedge_{C' \in C'} \text{dins}_c(0, C')) > 0$ for all $0 \in \mathbf{O}$. Moreover, we have to bound the number of sets of classes an object can be an instance of. That is, we assume there exists a constant $c \in \mathbb{N}$ which is bounded polynomially

in the size of the PKB and, for every $0 \in \mathbf{O}$, we have that $|\{C' \subseteq \mathbf{C} : P(\bigwedge_{C' \in C'} \text{dins}_c(0, C')) > 0\}| < c$. For every pair of objects $0, 0' \in \mathbf{O}$, $P(\text{ins}_r(0, 0', X))$ specifies the analogous prior probabilities for object pairs with restrictions identical to those made for objects.

For every $0 \in \mathbf{O}$ and every $A \in \mathbf{A}(0)$, $P(\text{attr}_o(0, X, A))$ is the probability of object 0 having value $X \in \text{Val}(A)$ for attribute A . For example, for the PKB depicted in Figure 1, we have $P(\text{attr}_o(\text{Arnold}, \text{"Arni"}, \text{surfaceForm})) = 0.1$, $P(\text{attr}_o(\text{Arnold}, \text{"Arnold"}, \text{surfaceForm})) = 0.7$, and $P(\text{attr}_o(\text{Arnold}, \text{"Schwarzenegger"}, \text{surfaceForm})) = 0.2$. The families of prior conditional distributions $P(\text{attr}_{oo}(0, 0', \cdot, A))$, $P(\text{attr}_c(C, \cdot, A))$, and $P(\text{attr}_r(R, \cdot, A))$ are the analogous conditional distributions for object pairs, classes, and relations.

The probability of a possible world now factorizes into the probability of the class hierarchy, the probability of the relation hierarchy, and the probability of instance and attribute assertions given these hierarchies:

$$P(\mathbf{W}) = \prod_{\text{dsub}_c(C, C') \in \mathbf{W}} P(\text{dsub}_c(C, C')) \prod_{\text{attr}_c(C, V, A) \in \mathbf{W}} P(\text{attr}_c(C, V, A)) \times \\ \prod_{\text{dsub}_r(R, R') \in \mathbf{W}} P(\text{dsub}_r(R, R')) \prod_{\text{attr}_r(R, V, A) \in \mathbf{W}} P(\text{attr}_r(R, V, A)) \times \\ \prod_{0, 0' \in \mathbf{O}} \left[P\left(\bigwedge_{C \in \mathbf{C}_0} \text{dins}_c(0, C)\right) \prod_{\text{attr}_o(0, V, A) \in \mathbf{W}} P(\text{attr}_o(0, V, A)) \right] \times \\ \prod_{0, 0' \in \mathbf{O}} \left[P\left(\bigwedge_{R \in \mathbf{R}_{0, 0'}} \text{dins}_r(0, 0', R)\right) \prod_{\text{attr}_{oo}(0, 0', V, A) \in \mathbf{W}} P(\text{attr}_{oo}(0, 0', V, A)) \right],$$

where $\mathbf{C}_0 = \{C \in \mathbf{C} : \text{dins}_c(0, C) \in \mathbf{W}\}$ and $\mathbf{R}_{0, 0'} = \{R \in \mathbf{R} : \text{dins}_r(0, 0', R) \in \mathbf{W}\}$. We can now state the following theorem.

Theorem 1. *Let $\mathcal{K}$ be a PKB, let $\mathcal{W}$ be the set of possible worlds for $\mathcal{K}$, and let P be the distribution over possible worlds. Then, $\sum_{\mathbf{W} \in \mathcal{W}} P(\mathbf{W}) = 1.0$.*

Probabilistic Reasoning

The joint distribution over possible worlds is a product of conditional distributions. The conditional distributions are representable with block-independent-disjoint (BID) tables of a probabilistic database. BID tables are table where the tuples can be partitioned into blocks such that tuples within the same block are mutually exclusive and tuples in different blocks are independent (Suciu et al. 2011). This opens the door to the application of known tractability results from the probabilistic database literature. However, for the types of queries typically performed with a PKB, the known results do not generally apply and we have to prove the tractability of certain queries only partially relying on known results.

We first define the query language and semantics. A formula is given by the following grammar:

$$Q := u = v \mid R(\bar{x}) \mid \exists x. Q_1 \mid Q_1 \wedge Q_2 \mid Q_1 \vee Q_2$$

where $R(\bar{x})$ is one of the TPKB's predicate symbols with variables and/or constants from $\mathbf{K}$. A union of conjunctive

$Q_1(x)$	$:= \text{attr}_o(x, \text{Val}_1, A_1), \dots, \text{attr}_o(x, \text{Val}_n, A_n)$
$Q_2(x)$	$:= \text{ins}_c(x, C_1), \dots, \text{ins}_c(x, C_n)$
$Q_3(x, y)$	$:= \text{ins}_c(x, y), Q_1(x), Q_2(x)$
$Q_4(x)$	$:= \text{sub}_c(x, C_1), \dots, \text{sub}_c(x, C_n)$
$Q_5(x)$	$:= Q_1(x), Q_2(x)$
$Q_6(x, y)$	$:= \text{ins}_r(x, y, R_1), \dots, \text{ins}_r(x, y, R_n)$
$Q_7(x, y, z)$	$:= \text{ins}_r(x, y, z), Q_6(x, y)$
$Q_8(x, y)$	$:= \text{sub}_r(x, y, R_1), \dots, \text{sub}_r(x, y, R_n)$
$Q_9(x, y)$	$:= Q_6(x, y), Q_5(x), Q_5(y)$
$Q_{10}(x, y)$	$:= Q_7(x, y), Q_5(x), Q_5(y)$

Table 2: A collection of tractable query families.

queries (UCQ) has the form $Q(\bar{x})$ where $\bar{x}$ are the free variables of formula Q . A Boolean query is a query without free variables. We write $Q[\bar{a}/\bar{x}]$ to refer to the query expression Q where the free variables $\bar{x}$ are substituted with constants $\bar{a}$. When a query has free variables $\bar{x}$, then for possible world $\mathbf{W}$, $Q(\mathbf{W}) = \{\bar{a} : \mathbf{W} \models Q[\bar{a}/\bar{x}]\}$ where $\mathbf{W} \models Q$ means Q is true in $\mathbf{W}$. We refer to the elements $Q(\mathbf{W})$ as tuples. Given a PKB $\mathcal{K}$ with possible worlds $\mathcal{W}$, the *marginal probability* of tuple t for a query Q is

$$P(t \in Q) = \sum_{\mathbf{W} \in \mathcal{W}: t \in Q(\mathbf{W})} P(\mathbf{W}).$$

We construct mappings from queries over the PKB to queries over a probabilistic database and exploit the hierarchical structure of possible worlds to derive tractability results and efficient query evaluation plans. Note that we can compute probabilities *conditioned* on attribute values by normalizing the query probabilities with the probability of the tractable Boolean query $\exists x. Q_1(x)$.

Consider the families of queries in Table 2. Queries of type Q_1 return, for a given set of attribute values, a list of objects with their marginal probabilities. For instance, the query “All entities born in 1950 and with surface form ‘Einstein’” can be expressed. Queries of type Q_2 , Q_3 , and Q_4 are queries one typically asks against a logical KB. For instance, the query “All classes/instances that are subclasses/instances of Politician and Actor” can be expressed. Queries of type Q_3 return a list of object class tuples given a set of attribute values and instance restrictions. For instance, the query “All classes whose instances were born after 1950 and are instances of the class Person.” Queries of type Q_9 return a list of object pairs that are related via a set of relations R_i given a set of attribute values and instance restrictions. For instance, the query “All object pairs (x, y) that are in a *studentOf* relation and where x was born in 1950 and y has surface form ‘Bach’” is a member of this query family. Query Q_{10} returns a list of object pairs and relations that could hold between objects x and y given a set of attribute values and instance restrictions for the two objects such as “All relations that could hold between a person and an organization where the person’s surface form is ‘Obama’ and the organization was founded after 2009.”

Theorem 2. *Query types $Q_1, \dots, Q_{10}$ are tractable.*

The tractability of most queries does not follow from the probabilistic database literature and the proofs in the appendix are quite involved. This is because the posterior probabilities for the ins_c , ins_r , sub_c , and sub_r assertions are not identical to the prior probabilities. Moreover, there are only few tractability results for conjunctive queries with self-joins. The TPKB’s hierarchical structure of the possible worlds and the stratification of the parameterization render the queries tractable. Table 2 lists classes of queries that are important for applications such as entity resolution and data integration, however, there are other classes of tractable queries and this is often easily verifiable.

Modeling and Learning

The most sophisticated PKB is of limited use if learning its parameters from existing data sets is not possible. Users of the TPKB only need to worry about learning (some of) the prior conditional distributions. *Both* structure and parameter learning is accomplished by estimating these parameters *independently* because the likelihood decomposes into a product of the local conditional distributions of the joint distribution. It is possible to incrementally add attribute and object distributions without having to relearn the entire PKB. Especially if the parameter values are learned by processing very large data sets, this property of incremental learning is crucial. Since there are no latent variables and all parameters to be estimated are those of conditional distributions, we can apply closed-form maximum-likelihood learning as long as the chosen parameterization of the random variables allows this. For instance, this is always possible if we use multinomials to model the distributions.

Let $\mathbf{W}_1, \dots, \mathbf{W}_N$ be sets of assertions of the PKB’s predicates. For instance, we estimate the parameter $P(\text{dsub}_c(C, C'))$ of the multinomial distribution with

$$\frac{1}{|N||O|} \sum_{i=1}^N \sum_{o \in O} \frac{[[\text{ins}_c(o, C') \in \mathbf{W}_i \text{ and } \text{ins}_c(o, C) \in \mathbf{W}_i]]}{[[\text{ins}_c(o, C) \in \mathbf{W}_i]]},$$

where $[[\cdot]]$ is the indicator function. Similarly, we can estimate the parameters for the other distributions. The only additional constraint is that the parameterization is stratified and that every object (object pair) is an instance of at least one class (relation) in a possible world.

We also learn histograms to estimate the densities of numerical attribute values on the level of classes and relations. These histograms can be leveraged for *default reasoning* when data on the level of objects is missing. For instance, consider the attribute *birthYear*. There are numerous persons without an assertion for this property in any of the information extraction projects. However, the TPKB can still compute the probability for those objects by using the distribution of *birthYear* on the class level. For instance, Arnold who is an instance of the class Actor might not have an assertion for the attribute *birthYear*. In this case, we use the probability of an actor being born in 1950 as a surrogate for $P(\text{attr}_o(\text{Arnold}, 1950, \text{birthDate}))$.

Experiments

For the experiments, we learned a TPKB’s direct subclass and instance distributions (both for objects and object pairs)

NELL relation	Precision@1			Recall@1		
	WL	TPKB1	TPKB2	WL	TPKB1	TPKB2
ActorStarredInMovie	0.81	0.85	0.95	0.82	0.81	0.31
AgentcollaboratesWithAgent	0.82	0.83	0.95	0.86	0.84	0.20
AnimalIsTypeofAnimal	0.86	0.86	1.00	0.86	0.85	0.00
AthleteLedSportsTeam	0.89	0.91	0.92	0.86	0.79	0.37
BankBankInCountry	0.82	0.87	0.92	0.76	0.69	0.10
CityLocatedInState	0.80	0.85	0.96	0.81	0.80	0.64
BookWriter	0.82	0.83	0.92	0.83	0.82	0.73
CompanyAlsoKnownAs	0.71	0.71	1.00	0.58	0.28	0.00
PersonLeadsOrganization	0.79	0.81	0.93	0.75	0.57	0.11
TeamPlaysAgainstTeam	0.81	0.81	1.00	0.81	0.73	0.00
WeaponMadeInCountry	0.88	0.91	1.00	0.88	0.84	0.00
LakeInState	0.90	0.91	1.00	0.92	0.90	0.84

System	Precision	Recall
PARIS	91.9	73.8
TPKB1@1	85.3	72.4
TPKB2@1	89.3	69.3
TPKB3@1	92.8	67.1
TPKB1@2	-	73.1
TPKB2@2	-	74.0
TPKB3@2	-	74.2

Table 3: Results for entity resolution experiments (left) and entity linking experiments (right). WL is the WikiLink baseline.

using the ontology and the data from DBPEDIA (Lehmann et al. 2014; Bizer, Heath, and Berners-Lee 2009) and we manually created a shallow relation hierarchy. For several attributes on the class and object level, such as `birthDate`, `elevation`, `geocoordinates`, etc. we also used data in DBPEDIA. To learn these attributes on the object level, we assumed a uniform distribution if, for one object, more than one value is given for a particular attribute. For instance, if Arnold has values 1947 and 1946 for attribute `birthDate` then, following the maximum-likelihood principle, we assume that both values have probability 0.5. For the class level, we pooled attribute values of the instances of each class and used histograms to model these distributions. For the object attribute `surfaceForm` we used the WIKIPREP tool (Gabrilovich and Markovitch 2006; 2007) to compute the conditional distributions using Wikipedia’s link structure. Based on object attributes it is possible to learn the parameters of attributes for object pairs. For instance, we introduced the attribute `diffBirthYear` which models a distributions over the absolute value of birth year differences. For instance, if Arnold was born in 1947 or 1946 and Einstein in 1879, then the value for this attribute would be 68 and 67 with probability 0.5 each. We also pool attributes for distributions on the level of relations. The number of parameters of the resulting TPKB exceeds 1 billion and we model more than 1 million objects and object pairs. We ran the respective conjunctive queries using a MYSQL database and *stored functions* for the computations of the extensional queries. Each query used for the experiments could be answered in less than one second.

We evaluated the learned TPKB on two important problem classes which we discuss in the following.

Entity Linking For the entity linking experiments we used an existing gold standard for aligning NELL triples (Carlson et al. 2010) to DBPEDIA (Dutta et al. 2013). We run two different queries to compute, for each triple, subject and object alignments using the same experimental set-up as previous work (Dutta et al. 2013). For each of the NELL predicates in Table 3(left) we manually aligned the domain and range with classes in the TPKB if possible. For instance, for the first relation, we know that subjects are

instances of class `Actor` and objects are instances of class `Film`. The queries are now $Q_5(x)$ where we used only the attribute `surfaceForm` and the aligned classes (TPKB1); and query $Q_{10} := \text{ins}_r(x, y, z), \text{attr}_o(x, \text{Val}_1, \text{surfaceForm}) \text{ins}_c(x, C), \text{attr}_o(y, \text{Val}_2, \text{surfaceForm}), \text{ins}_c(y, C)$ where we only use answer tuples with $z \neq \text{Thing} \times \text{Thing}$ (TPKB2). This query retrieves all object pairs that are instance of a relation other than `Thing` $\times$ `Thing` given attribute values and class memberships. The results are given in Table 3 (left) and compared with a baseline given in (Dutta et al. 2013). Precision@k and Recall@k is computed by retrieving the k most probable answer tuples.

Entity Resolution Entity resolution is the problem of determine whether two objects are equivalent. To evaluate the TPKB for entity linking we repeated the experiment of linking YAGO (Hoffart et al. 2013) to DBPEDIA conducted to evaluate the PARIS matching system (Suchanek, Abiteboul, and Senellart 2011). Both knowledge bases use Wikipedia identifiers for their objects which gives us a large set of gold standard pairs for evaluation purposes. We manually aligned a set of attributes (datatype properties) and classes between YAGO and the TPKB (DBPEDIA). We sampled 100000 objects in YAGO, retrieved the aligned attributes for each object (labels, numerical attributes, etc.) and ran, for each object, the query Q_1 only with attribute `surfaceForm` (TPKB1); and all other attributes for which a manual alignment existed (TPKB2). Moreover, we ran the query $Q_5(x)$ when an alignment between at least two classes existed (TPKB3). Table 3 shows that TPKBs are able to accurately link entities and compare favorably with specialized algorithms.

The TPKB is both efficient and performs comparable to existing problem specific and less versatile approaches.

Future Work

Directions for future work include more expressive variants of the presented TPKB framework, more applications, and learning TPKBs from multiple IE projects.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

- Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook*. Cambridge University Press.
- Bizer, C.; Heath, T.; and Berners-Lee, T. 2009. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.* 5(3):1–22.
- Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B., Jr., E. R. H.; and Mitchell, T. M. 2010. Toward an architecture for never-ending language learning. In Fox, M., and Poole, D., eds., *AAAI*. AAAI Press.
- Domingos, P., and Webb, W. A. 2012. A tractable first-order probabilistic logic. In Hoffmann, J., and Selman, B., eds., *AAAI*. AAAI Press.
- Dutta, A.; Meilicke, C.; Niepert, M.; and Ponzetto, S. P. 2013. Integrating open and closed information extraction: Challenges and first steps. In *Proceedings of NLP-DBPEDIA@ISWC*.
- Etzioni, O.; Fader, A.; Christensen, J.; Soderland, S.; and Mausam. 2011. Open information extraction: The second generation. In Walsh, T., ed., *IJCAI*, 3–10. IJCAI/AAAI.
- Gabrilovich, E., and Markovitch, S. 2006. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *AAAI*, 1301–1306. AAAI Press.
- Gabrilovich, E., and Markovitch, S. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In Veloso, M. M., ed., *IJCAI*, 1606–1611.
- Gelman, A., and Hill, J. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical Methods for Social Research. Cambridge University Press.
- Gutiérrez-Basulto, V.; Jung, J. C.; Lutz, C.; and Schröder, L. 2011. A closer look at the probabilistic description logic prob-el. In *Proceedings of AAAI*.
- Hoffart, J.; Suchanek, F. M.; Berberich, K.; and Weikum, G. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.* 194:28–61.
- Katz, S. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing* 35(3):400–401.
- Kemp, C.; Tenenbaum, J. B.; Griffiths, T. L.; Yamada, T.; and Ueda, N. 2006. Learning systems of concepts with an infinite relational model. In *Proceedings of AAAI*, 381–388.
- Kimmig, A.; Bach, S. H.; Broecheler, M.; Huang, B.; and Getoor, L. 2012. A short introduction to probabilistic soft logic. In *NIPS Workshop on Probabilistic Programming: Foundations and Applications*.
- Lehmann, J.; Gerber, D.; Morsey, M.; and Ngomo, A.-C. N. 2012. Defacto - deep fact validation. In *International Semantic Web Conference (I)*, 312–327.
- Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P. N.; Hellmann, S.; Morsey, M.; van Kleef, P.; Auer, S.; and Bizer, C. 2014. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*.
- Lukasiewicz, T. 2007. Probabilistic description logic programs. *Int. J. Appr. Reas.* 45.
- McCallum, A.; Rosenfeld, R.; Mitchell, T. M.; and Ng, A. Y. 1998. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of ICML*, 359–367.
- Nakashole, N.; Sozio, M.; Suchanek, F. M.; and Theobald, M. 2012. Query-time reasoning in uncertain rdf knowledge bases with soft and hard rules. In Brambilla, M.; Ceri, S.; Furche, T.; and Gottlob, G., eds., *VLDS*, volume 884 of *CEUR Workshop Proceedings*, 15–20. CEUR-WS.org.
- Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2012. Factorizing yago: Scalable machine learning for linked data. In *Proceedings of WWW*, 271–280.
- Niepert, M., and Van den Broeck, G. 2014. Tractability through exchangeability: A new perspective on efficient probabilistic inference. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*.
- Niepert, M.; Noessner, J.; and Stuckenschmidt, H. 2011. Log-linear description logics. In *Proceedings of IJCAI*, 2153–2158.
- Noessner, J., and Niepert, M. 2011. Elog: A probabilistic reasoner for owl el. In *Proceedings of the 5th Conference on Web Reasoning and Rule Systems (RR)*, 281–286.
- Noessner, J.; Niepert, M.; and Stuckenschmidt, H. 2013. Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. In *Proceedings of the 27th Conference on Artificial Intelligence (AAAI)*.
- Poole, D. 2008. The independent choice logic and beyond. In *Probabilistic inductive logic programming*. Springer-Verlag.
- Raedt, L. D.; Kimmig, A.; and Toivonen, H. 2007. Problog: a probabilistic prolog and its application in link discovery. In *Proceedings of 20th International Joint Conference on Artificial Intelligence*, 2468–2473.
- Riedel, S.; Yao, L.; Marlin, B. M.; and McCallum, A. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of HLT-NAACL*.
- Sato, T., and Kameya, Y. 1997. Prism: A language for symbolic-statistical modeling. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, 1330–1339.
- Suchanek, F. M.; Abiteboul, S.; and Senellart, P. 2011. Paris: Probabilistic alignment of relations, instances, and schema. *PVLDB* 5(3):157–168.
- Suciu, D.; Olteanu, D.; Christopher, R.; and Koch, C. 2011. *Probabilistic Databases*. Morgan & Claypool Publishers, 1st edition.
- Van den Broeck, G. 2011. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *Proceedings of NIPS*, 1386–1394.
- Wang, D. Z.; Franklin, M. J.; Garofalakis, M. N.; and Hellerstein, J. M. 2010. Querying probabilistic information extraction. *PVLDB* 3(1):1057–1067.

A Sparse Parameter Learning Method for Probabilistic Logic Programs

Masaaki Nishino^{†*} and Akihiro Yamamoto^{*} and Masaaki Nagata[†]

[†]NTT Communication Science Laboratories, NTT Corporation

2-4, Hikaridai, Seika, Souraku, Kyoto, 239-0153, Japan

^{*}Graduate School of Informatics, Kyoto University

Yoshida Honmachi, Sakyo, Kyoto, 606-8501, Japan

{nishino.masaaki,nagata.masaaki}@lab.ntt.co.jp, akihiro@i.kyoto-u.ac.jp

We propose a new parameter learning algorithm for ProbLog, which is an extension of a logic program that can perform probabilistic inferences. Our algorithm differs from previous parameter learning algorithms for probabilistic logic program (PLP) models on the point that it tries to reduce the number of probabilistic parameters contained in the estimated program. Since the amount of computation required for a probabilistic inference with a ProbLog program can be exponential with respect to the number of probabilistic parameters, programs with fewer parameters are preferable. Our algorithm tries to reduce the number of parameters by adding a penalty term to the objective function, and then minimizing it to encourage the estimated parameters to take either 0 or 1. If a parameter takes value 0 or 1, we can delete the corresponding probabilistic fact, or treat the corresponding fact as a deterministic one, to remove the parameter from the obtained model. We also show that the optimization problem can be solved efficiently by applying the projected gradient method to a compiled knowledge representation. We confirm experimentally that the proposed algorithm is comparable with the state-of-the-art algorithm, while it can reduce the number of probabilistic parameters contained in the estimated program.

Introduction

A probabilistic logic program (PLP) is an extension of a logic program that can perform probabilistic inferences. A PLP is a kind of statistical relational model (Getoor and Taskar 2007), and was developed for modeling complex and uncertain relationships. Many PLP models have been proposed (e.g., (Muggleton 1996; Sato and Kameya 2001; De Raedt, Kimmig, and Toivonen 2007)), and most of existing PLP models are based on Sato’s distribution semantics (Sato 1995), which defines a probability distribution over possible worlds by introducing probabilistic ground facts into a logic program.

A problem with these PLP models is the difficulty related to inferences. In the worst case, a PLP model that is based on the distribution semantics may require computational time exponential to the number of probabilistic parameters contained in the model. It prevents PLP models from being ap-

plied to large problems. Although some efficient exact and approximate inference algorithms have been proposed for these models (De Raedt, Kimmig, and Toivonen 2007), inference is still difficult and hence PLP models with fewer probabilistic parameters are preferable.

In this paper, we propose a parameter learning algorithm for probabilistic logic programs. The proposed algorithm can reduce the number of probabilistic factors contained in the estimated model. Parameter estimation algorithms for PLP models proposed in previous research (e.g., (Gutmann, Thon, and De Raedt 2011; Sato and Kameya 2001)) are not intended for estimating compact models, i.e., a model with fewer probabilistic parameters. In order to estimate a compact model, we add penalty terms to the negative log-likelihood function and then minimize it to estimate probabilistic parameters. Penalty terms are often introduced into machine learning algorithms to impose some restrictions on the estimated parameters. A well-known penalty term is ℓ_1 norm, which is applied for obtaining sparse solutions (Bach et al. 2009), but the ℓ_1 or other sparsity inducing norms cannot be directly applied to the parameter learning problem of PLP models. The new penalty term we propose in this paper induces the learned parameters to take either 0 or 1. When a probabilistic parameter is 0 or 1, we can remove it or treat it as deterministic to obtain a PLP model with fewer number of probabilistic parameters. We also give an efficient optimization algorithm that can run on a compiled knowledge representation. Given an objective function with penalty terms, we minimize it by applying a projected gradient algorithm (Bertsekas 1999). A projected gradient algorithm is an efficient method for solving an optimization problem with constraints that a solution must be contained in a convex set, and we present a method to run it with a compiled knowledge representation, which is obtained by transforming a logical model into another form, and reduces the computational time. As the transformation, we use a deterministic and decomposable negation normal form (d-DNNF) (Darwiche and Marquis 2002) so that we make optimization problems tractable.

In the following, we use ProbLog (De Raedt, Kimmig, and Toivonen 2007) as a concrete example of PLP model and we propose a parameter learning algorithm on it. However, with slight modification our method may also be applicable to other PLP models. We give a class of PLP models

to which our parameter learning algorithm can be applied.

Preliminaries

We first briefly introduce some basic notations used in this paper. A term is a variable, a constant, or a function applied to terms. Let $q(t_1, \dots, t_k)$ be an atom, where $t_1, \dots, t_k$ are terms and q is a predicate of arity k . Definite clauses are universally quantified expressions of the form $h :- b_1, \dots, b_n$, where $h, b_1, \dots, b_n$ are all atoms, and h is the head of a clause and $b_1, \dots, b_n$ are the body. A clause without a body is a fact. A substitution θ is an expression of the form $\{V_1/t_1, \dots, V_m/t_m\}$ where V_i are different variables and t_i are terms. If a substitution θ is applied to an expression e , then the instantiated expression $e\theta$ is made by simultaneously replacing the variables V_i in e with t_i . An expression is called ground if it has no variables. The semantics of a set of definite clauses is given by its least Herbrand model, i.e., the set of all ground facts entailed by the theory.

A ProbLog theory $\mathcal{T}$ consists of both a set of labeled facts F and a set of definite clauses KB . Let f_i be a fact contained in F and $w_i \in [0, 1]$ be the label of f_i . w_i represents the probability that each ground substitution $f_i\theta$ is true in the theory. We refer to an annotated fact $w_i :: f_i$ as a *probabilistic fact*.

Example 1. *The following is an example of ProbLog program.*

```
0.1::burglary.      0.2::earthquake.
0.7::hears_alarm(X) :- person(X).
person(mary).      person(john).
alarm :- burglary.  alarm :- earthquake.
calls(X) :- alarm, hears_alarm(X).
```

In this program, burglary. and earthquake. are facts and their probabilities are 0.1 and 0.2, respectively. 0.7 :: hears_alarm(X) :- person(X). is a notation that represents two probabilistic facts, 0.7::hears_alarm(mary) and 0.7::hears_alarm(john).

Given a finite number of possible ground substitutions $\{\theta_{i,1}, \dots, \theta_{i,K_i}\}$ for each probabilistic fact $w_i :: f_i$, a ProbLog program $\mathcal{T}$ defines probability distribution over total choices L , where L is a subset of the set of all ground facts $L_{\mathcal{T}} = \{f_1\theta_{1,1}, \dots, f_1\theta_{1,K_1}, \dots, f_N\theta_{N,1}, \dots, f_N\theta_{N,K_N}\}$.

$$P(L|\mathcal{T}) = \prod_{f_i\theta_{i,k} \in L} w_i \prod_{f_i\theta_{i,k} \in L_{\mathcal{T}} \setminus L} (1 - w_i). \quad (1)$$

Using the above definition of probability $P(L|\mathcal{T})$, we define the success probability of a query literal q as

$$P(q|\mathcal{T}) = \sum_{L \subseteq L_{\mathcal{T}}, L \cup KB \models q} P(L|\mathcal{T}) \sum_{L \subseteq L_{\mathcal{T}}} \delta(q, KB \cup L) \cdot P(L|\mathcal{T})$$

where $\delta(q, KB \cup L) = 1$ if there exists a substitution θ such that $KB \cup L \models q\theta$, and 0 otherwise.

Example 2. *For the program in Example 1, $L_{\mathcal{T}}$ contains four probabilistic facts, and hence there are $2^4 = 16$ possible $L \subseteq L_{\mathcal{T}}$. If $q = \text{alarm}$, $\delta(q, KB \cup L) = 1$ if*

either burglary or earthquake is contained in L , and its probability is $P(\text{alarm}|\mathcal{T}) = 1 - P(\neg \text{burglary} \wedge \neg \text{earthquake}|\mathcal{T}) = 1 - 0.9 \times 0.8 = 0.28$.

Parameter Learning

Motivating Examples

Before presenting our parameter learning algorithm, we first show what it aims to do. What we want is a compact ProbLog program, but a compact program is not just a program with fewer clauses. At this point, our algorithm differs slightly from sparse learning algorithms (Bach et al. 2009); sparse learning algorithms try to obtain sparse models by letting many parameters take zero. By contrast, our learning algorithm induces many parameters to take either 0 or 1. This setting is motivated by the following two examples.

$$w_1 :: q. \quad w_2 :: r. \quad p :- q. \quad p :- r.$$

Suppose that we are given training examples $\mathcal{D}$ that only contains literal p , and we want to set parameters w_1, w_2 so as to maximize the log-likelihood of $\mathcal{D}$. If training examples follow probabilistic distribution $P(p) = 0.5$, then any combination of parameters w_1 and w_2 that satisfies $1.0 - (1 - w_1)(1 - w_2) = 0.5$ will maximize log-likelihood. However, if we set $w_1 = 0.0$ and $w_2 = 0.5$ (or equivalently, set $w_1 = 0.5$ and $w_2 = 0.0$), then we can remove one probabilistic fact from the program. This clearly reduces the size of the obtained program. The above approach is equivalent to theory compression (De Raedt et al. 2008), which allows some parameters of a ProbLog program to be zero to obtain a more concise logic program.

We give another example.

$$w_1 :: q. \quad w_2 :: r. \quad p :- q, r.$$

With this program, the probability $P(p)$ is represented as $P(p) = w_1 w_2$. As same as the previous example, suppose that we are given training examples $\mathcal{D}$ that only contains literal p , and we want to set parameters w_1, w_2 so as to maximize log-likelihood of $\mathcal{D}$. If training examples follow probabilistic distribution $P(p) = 0.5$, many combinations of parameters are possible. However, if we set $w_1 = 1.0$ and $w_2 = 0.5$ (or $w_1 = 0.5$ and $w_2 = 1.0$), it means we treat the probabilistic fact q as a deterministic (i.e., non probabilistic) fact. Since the complexity of probabilistic inference with a ProbLog program depends on the number of probabilistic facts, treating some probabilistic facts as deterministic facts also helps to reduce the cost of probabilistic inference.

Our algorithm uses a penalty function to obtain these two types of reduction of probabilistic parameters, namely, (i) removing probabilistic facts and (ii) substituting probabilistic facts with deterministic ones.

Learning Algorithm

Our parameter learning algorithm follows the learning from interpretation setting that has been proposed in (Gutmann, Thon, and De Raedt 2011) since it is more general than the learning from entailment settings as used in other learning algorithms (Gutmann et al. 2008; Sato and Kameya 2001; Cussens 2001). Let I be a partial interpretation of ground

atoms in $L_{\mathcal{T}}$, which determines the truth values of some atoms in $L_{\mathcal{T}}$. We represent a partial interpretation as $I = (I^+, I^-)$ where I^+ contains all true atoms and I^- all false atoms. We define the probability of a partial interpretation I in a way that is similar to that for an atom q :

$$P(I|\mathcal{T}) = \sum_{L \subseteq L_{\mathcal{T}}} \delta(I^+, KB \cup L) \bar{\delta}(I^-, KB \cup L) P(L|\mathcal{T}),$$

where $\delta(I^+, KB \cup L) = 1$ if $KB \cup L \models q$ for all atoms $q \in I^+$, and $\bar{\delta}(I^-, KB \cup L) = 1$ if $KB \cup L \not\models q$ for all atoms $q \in I^-$.

Parameter learning is formalized as the task of finding a set of parameters $\hat{\mathbf{w}} = \{\hat{w}_1, \dots, \hat{w}_N\}$ that minimizes the objective function given training examples. Let a set of interpretation $\mathcal{D} = I_1, \dots, I_M$ be training examples. We make the objective function as a combination of the negative log-likelihood and a penalty function that encourages parameters to take either 0 or 1. We therefore define the objective function $g(\mathcal{T}(\mathbf{w}), \mathcal{D})$ as

$$g(\mathcal{T}(\mathbf{w}), \mathcal{D}) = \ell(\mathcal{T}(\mathbf{w}), \mathcal{D}) + \lambda h(\mathbf{w}),$$

where $\mathcal{T}(\mathbf{w})$ represents a ProbLog program whose parameters are $\mathbf{w}$, and $\ell(\mathcal{T}(\mathbf{w}), \mathcal{D})$ represents a negative log-likelihood function and $h(\mathbf{w})$ is a penalty function. A parameter λ controls the effect of the penalty term. We minimize the above objective function by considering that $0 \leq w_i \leq 1$ for all $1 \leq i \leq N$. We define the negative log-likelihood function $\ell(\mathcal{T}(\mathbf{w}), \mathcal{D})$ as

$$\ell(\mathcal{T}(\mathbf{w}), \mathcal{D}) = - \sum_{j=1}^M \log P(I_j|\mathcal{T}(\mathbf{w})).$$

It is simply defined as the logarithm of the product of the probabilities $P(I_j|\mathcal{T}(\mathbf{w}))$ for $j = 1, \dots, M$. We define the penalty function $h(\mathbf{w})$ as

$$h(\mathbf{w}) = \sum_{i=1}^N \{\log(w_i + \varepsilon) + \log(1 - w_i + \varepsilon)\}, \quad (2)$$

where ε is a small positive value that is added to avoid the function becoming $-\infty$ at $w_i = 0$ or $w_i = 1$. The penalty term takes smaller values when w_i is near 0 or 1, hence we can encourage w_i to take 0 or 1 by adding $h(\mathbf{w})$ to the negative log-likelihood $g(\mathcal{T}(\mathbf{w}), \mathcal{D})$. Hence this penalty term can contribute to reduce the number of parameters. We show an example of $h(\mathbf{w})$ when $N = 1$ in Fig.1.

The minimization problem of only negative log-likelihood function $\ell(\mathcal{T}(\mathbf{w}))$ is the same as the problem solved in the LFI-ProbLog algorithm shown in (Gutmann, Thon, and De Raedt 2011), and it can be solved efficiently by using the Expectation Maximization (EM) algorithm. The difference between the LFI-ProbLog algorithm and ours is the use of the penalty term $h(\mathbf{w})$. This addition makes EM algorithm inapplicable for our problem. We therefore introduce a new optimization method that is based on the projected gradient algorithm.

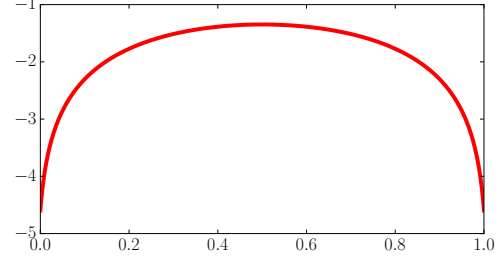


Figure 1: Shape of the penalty term $h(\mathbf{w})$, where $N = 1$ and $\varepsilon = 0.001$.

Algorithm 1 A projected gradient parameter learning algorithm

- 1: Initialize $\mathbf{w}^0$, set $k \leftarrow 0$.
 - 2: **while** k is less than the iteration limit **do**
 - 3: $\mathbf{x}^k \leftarrow \mathbf{w}^k - \alpha_k \nabla g(\mathbf{w}^k)$
 - 4: $\mathbf{w}^{k+1} \leftarrow \text{proj}(\mathbf{x}^k)$
 - 5: **if** converged($\mathbf{w}^k, \mathbf{w}^{k+1}$) **then**
 - 6: **return** $\mathbf{w}^k$
 - 7: $k \leftarrow k + 1$
 - 8: **return** $\mathbf{w}^k$
-

Projected Gradient Algorithm

The minimization problem we introduced in the previous section cannot be solved with the EM algorithm, we therefore propose a new method that is based on the projected gradient algorithm. The projected gradient algorithm (Bertsekas 1999) is used for minimization problems with the constraint that the variables must be contained in a convex set. In the present case, w_i must satisfy $0 \leq w_i \leq 1$, and hence the region in which $\mathbf{w}$ is contained is convex. We can therefore employ the projected gradient method for our problem.

The projected gradient algorithm is an extension of a gradient descent algorithm, and it solves optimization problems by repeating gradient computation and projection onto a convex set. Algorithm 1 shows the projected gradient algorithm. After initializing $\mathbf{w}^0$ (line 1), it repeatedly updates $\mathbf{w}^k$ until it converges (lines 2 to 7). First we compute $\mathbf{x}^k$ from the current $\mathbf{w}^k$ and the gradient $\nabla g(\mathbf{w}^k)$ (line 3). This step is the same as an ordinary gradient descent algorithm, then we project $\mathbf{x}^k$ into the domain that satisfies $0 \leq w_i \leq 1$ ($1 \leq i \leq N$) to obtain $\mathbf{w}^{k+1}$. Here, function $\text{proj}(\mathbf{x})$ is a projection function that maps $\mathbf{x}$ to $\mathbf{w}$ such that satisfies $0 \leq w_i \leq 1$ for all $1 \leq i \leq N$ and minimizes $\|\mathbf{x} - \mathbf{w}\|^2$. In this case, the i -th element $[\text{proj}(\mathbf{x})]_i$ is defined as

$$[\text{proj}(\mathbf{x})]_i = \begin{cases} 0 & \text{if } x_i \leq 0 \\ x_i & \text{if } 0 < x_i < 1 \\ 1 & \text{if } x_i \geq 1 \end{cases}, \quad (3)$$

i.e., we simply map x_i to 0 or 1 if it is not in the $0 \leq x_i \leq 1$ range.

α_k is the step size used for the k -th iteration. Setting an appropriate α_k is important since it determines the convergence of the projected gradient algorithm. We set α^k by using a simple line search based procedure called *the Armijo*

rule along the projection arc, as described in (Bertsekas 1999; 1976). It defines α_k as $\alpha_k = \beta^{t_k}$, where $\beta \in (0, 1)$, and t_k is the first non-negative integer t that satisfies

$$g(\mathbf{w}^k) - g(\mathbf{w}^k(\beta^{t_k})) \leq \sigma \nabla g(\mathbf{w}^k)^T (\mathbf{w}^k - \mathbf{w}^k(\beta^{t_k})), \quad (4)$$

where $\mathbf{w}^k(\beta^{t_k})$ is $\text{proj}(\mathbf{w}^k - \beta^{t_k} \nabla g(\mathbf{w}^k))$, and σ is a parameter that satisfies $\sigma \in (0, 1)$. In the experiments, we use parameters $\sigma = 0.2$ and $\beta = 0.5$, as these values are suggested in the textbook (Bertsekas 1999). By using α_k selected with this rule, it is proved that the projected gradient algorithm converges to a stationary point after several iterations (Bertsekas 1999; Calamai and Moré 1987) even if the objective function is nonconvex. Hence Alg. 1 converges after finite numbers of iterations.

Computation of gradient

To compute gradient $\nabla g(\mathcal{T}(\mathbf{w}), \mathcal{D})$, we must compute $\nabla \ell(\mathcal{T}(\mathbf{w}), \mathcal{D})$ and $\nabla h(\mathbf{w})$. Let $[\nabla \ell(\mathcal{T}(\mathbf{w}), \mathcal{D})]_i$ be the i -th element of the gradient $\nabla \ell(\mathcal{T}(\mathbf{w}), \mathcal{D})$, then it becomes

$$[\nabla \ell(\mathcal{T}(\mathbf{w}), \mathcal{D})]_i = - \sum_{j=1}^N \frac{\sum_{L \in \mathcal{L}_j} [\nabla P(L|\mathcal{T}(\mathbf{w}))]_i}{P(I_j|\mathcal{T}(\mathbf{w}))}. \quad (5)$$

Here we use $\mathcal{L}_j$ as the set that contains all possible assignments $L \subseteq L_{\mathcal{T}}$ that are consistent with I_j . We define $\nabla P(L|\mathcal{T}(\mathbf{w}))$ as

$$[\nabla P(L|\mathcal{T}(\mathbf{w}))]_i = \sum_{k=1}^{K_i} (2\delta(f_i\theta_{i,k} \in L) - 1) \prod_{f_n\theta_{n,k} \in L^{-i,k}} w_n \prod_{f_n\theta_{n,k} \in L_{\mathcal{T}}^{-i,k} \setminus L} (1 - w_n),$$

where $\delta(f_i\theta_{i,k} \in L) = 1$ if the condition is satisfied, otherwise 0, and $L^{-i,k}$ is $L \setminus \{f_i\theta_{i,k}\}$. $\nabla h(\mathbf{w})$ is easy to compute and $[\nabla h(\mathbf{w})]_i$ becomes

$$[\nabla h(\mathbf{w})]_i = \frac{1}{w_i + \varepsilon} - \frac{1}{1 - w_i + \varepsilon}.$$

The computation of gradient $\nabla \ell(\mathcal{T}(\mathbf{w}), \mathcal{D})$ involves summation over $L \in \mathcal{L}_j$. Since $\mathcal{L}_j$ consists of all the possible combinations of ground probabilistic facts $L_{\mathcal{T}}$, its size becomes $2^{L_{\mathcal{T}}}$ in the worst case and the naive computation of gradients is intractable with a large $L_{\mathcal{T}}$. Hence we use the knowledge compilation technique to compute them.

Knowledge compilation (Darwiche and Marquis 2002) is an approach that can be used for efficient computation involving propositional models. It first compiles a propositional model so as to represent it in a form that is suitable for specific operations such as probabilistic inference. Although knowledge compilation incurs the additional cost of compiling a model, once a compiled model is obtained, we can efficiently perform several operations by using it. In previous work, a ProbLog program was compiled into *binary decision diagrams (BDD)* (Bryant 1986), and a *deterministic, decomposable negation normal form (d-DNNF)* (Darwiche and Marquis 2002; Darwiche 2009). With a ProbLog program compiled into a BDD or a d-DNNF, we can compute the

probability $P(I|\mathcal{T}(\mathbf{w}))$ in time proportional to the size of the compiled representation. This computation is also used when employing EM style algorithms (Fierens et al. 2013; Gutmann, Thon, and De Raedt 2011).

We use compiled knowledge representations for computing the gradients of the negative log-likelihood function $\nabla \ell(\mathcal{T}(\mathbf{w}), \mathcal{D})$. Although both BDD and d-DNNF can be used for computing gradients, we chose d-DNNF as a compiled knowledge representation since it performed well in the previously reported parameter learning algorithms (Fierens et al. 2013).

We briefly introduce d-DNNF. d-DNNF is a kind of negation normal form (NNF) that satisfies decomposability and determinism. An NNF is a rooted directed acyclic graph in which each leaf node is labeled with a literal, true or false, and each internal node is labeled with a conjunction or disjunction. For any node n in an NNF graph, $\text{vars}(n)$ denotes all propositional variables that appear in the subgraph rooted at n , and $\Delta(n)$ denotes the formula represented by n and its descendants. We say an NNF satisfies decomposability if $\text{vars}(n_i) \cap \text{vars}(n_j) = \emptyset$ holds for any two children n_i and n_j ($i \neq j$) of an and-node n , and NNF satisfies determinism when $\Delta(n_i) \wedge \Delta(n_j)$ is logically inconsistent for any two children n_i and n_j ($i \neq j$) for an or-node n .

Given the set of interpretations $\mathcal{D}$, the process of compiling a ProbLog program into d-DNNFs is the same as for the parameter learning method proposed in (Fierens et al. 2013): We first make a ground ProbLog program and convert it into a conjunctive normal form (CNF), and then convert it into d-DNNF with a d-DNNF compiler that converts a CNF into the corresponding d-DNNF. After obtaining N different d-DNNFs that correspond to each interpretation $I_1, \dots, I_N$, we use them to compute the probability $P(I_j|\mathcal{T}(\mathbf{w}))$ and $\nabla P(L, \mathcal{T}(\mathbf{w}))$, which is the numerator of (5). We use the inference algorithm shown in (Darwiche 2009), which was originally used for computing conditional probabilities and gradients for graphical models. These algorithms can compute $P(I_j|\mathcal{T}(\mathbf{w}))$ by traversing all the nodes of a d-DNNF once, and can compute $\nabla P(L|\mathcal{T}(\mathbf{w}))$ by traversing all the nodes twice. As a result, we can efficiently compute $\nabla g(\mathcal{T}(\mathbf{w}), \mathcal{D})$.

Discussion

Our sparse parameter learning algorithm can be applied to other PLP models whose probabilistic parameters take values in $[0, 1]$. For example, PRISM, SLP, and ICL are in this class of PLP models. A major difference between ProbLog and these models is that these models employ the multinomial distribution, while ProbLog programs give probabilistic distribution based on the Bernoulli distribution defined on probabilistic facts.

With these multinomial distribution PLP models, the set of probabilistic parameters can be represented as $\mathbf{w} = \{\mathbf{w}_1, \dots, \mathbf{w}_N\}$, where $\mathbf{w}_i$ ($i = 1, \dots, N$) is a K_i dimensional vector $\mathbf{w}_i = (w_{i1}, \dots, w_{iK_i})$ and it satisfies $\sum_{j=1}^{K_i} w_{ij} = 1$ and $w_{ij} \geq 0$ for $j = 1, \dots, K_i$. Here we make an assumption that we can compute the gradient of the negative log-likelihood function $\ell(\mathbf{w}, \mathcal{D})$, given a set of

training examples $\mathcal{D}$. Then we apply our algorithm by only modifying the penalty function $h(\mathbf{w})$ defined in (2) as

$$h(\mathbf{w}) = \sum_{i=1}^N \sum_{j=1}^{K_i} \log(w_{ij} + \varepsilon).$$

If $K_i = 2$ for all $i = 1, \dots, N$, the above definition of $h(\mathbf{w})$ is equivalent to that in (2). It is also easy to compute $\nabla h(\mathbf{w})$.

we also need to modify the projection function $\text{proj}(\mathbf{x})$ to perform our projected gradient algorithm. We have to project a K_i dimensional real value vector $\mathbf{x}_i$ onto a probability simplex, i.e., project onto a K_i dimensional vector $\mathbf{w}_i$ that satisfies $\sum_{j=1}^{K_i} w_{ij} = 1$ and $w_{ij} \geq 0$ for all $j = 1, \dots, K_i$. This type of projection also can be efficiently performed (Parikh and Boyd 2014).

Experiments

Settings

We conducted experiments to evaluate our proposed learning algorithm. Our aim was to answer the following questions:

1. **(Q1)** Can we learn a compressed ProbLog program with our proposed algorithm?
2. **(Q2)** How do the estimated model changes when we change the parameter λ ?
3. **(Q3)** Can the proposed algorithm recover true distributions with a sufficient number of training examples?

We compare our proposed method with the LFI-ProbLog algorithm, which is an EM-style algorithm for estimating parameters from interpretations (Gutmann, Thon, and De Raedt 2011). We also compared our algorithm with an gradient projection algorithm, which simply minimizes the negative log-likelihood function.

Since our algorithm has much in common with the LFI-ProbLog algorithm, we implemented our algorithm on ProbLog2 (Renkens et al. 2012), a ProbLog implementation that supports several inference methods and parameter learning algorithms. We use grounding, CNF conversion, and a d-DNNF compilation algorithm implemented in ProbLog2, and run our projected gradient algorithm on a compiled d-DNNF to estimate the parameters. We also use the LFI-ProbLog algorithm implemented in ProbLog2.

We use two datasets, **WebKB** and **Smokers** for evaluating our parameter learning algorithm. The **WebKB** dataset¹ (Craven and Slattery 2001) is a real dataset that consists of labeled Web pages from the computer science departments of four universities. Every web page is marked with one of the following categories, *student*, *faculty*, *project*, *course*, *staff*, and *other*. The task is to predict the classes of pages given the words contained in each page and the link structure between pages. Following the setting used in (Fierens et al. 2013), we use the following ProbLog program.

```
p::link_class(P,P2,c1,c2) :- links_to(P,P2).
p::word_class(P,w1,c1) :- has_word(P,w1).
```

¹<http://www.cs.cmu.edu/~webkb/>

Table 1: Negative Log-Likelihood (lower is better) and the number of probabilistic parameters contained on the **WebKB** learning experiment.

Method	NLL	Num. params
LFI	1387.28	39.0
PG	1299.30	38.0
PG+P ($\lambda = 0.001$)	1318.96	25.0
PG+P ($\lambda = 0.01$)	1445.37	18.0

```
p::learnable_prior(P,c1) :- page(P).
0.001::fixed_prior(P,c1):-page(P),class(c1).
has_class(P,C) :- word_class(P,W,C).
has_class(P,C) :- has_class(P2,C2),
                    link_class(P,P2,C,C2).
has_class(P,C) :- fixed_prior(P,C).
has_class(P,C) :- learnable_prior(P,C).
```

Where the probabilistic fact `link_class/4` represents the effect of the link structure, and `word_class/3` represents the effect of words contained in a page. We also added probabilistic facts `learnable_prior/2` to represent the probabilistic distribution on labels that are independent with the link structure and words. Finally we add `fixed_prior/2` for avoiding log-likelihood to become infinity in the test data. For computational reasons, we selected 20 words that show the highest information gain with the class labels. We therefore have in total $6 \times 20 + 6 \times 6 + 6 = 162$ probabilistic parameters to be estimated from the data. We conduct four-fold cross validation by using the dataset for three universities as a training set and use the other university as the test set.

The **Smokers** dataset (Domingos and Lowd 2009) represents the relationships between people, and contains the following probabilistic facts and rules.

```
0.2::stress(P) :- person(P).
0.3::influences(P1,P2) :- friend(P1,P2).
0.1::cancer_spont(P) :- person(P).
0.3::cancer_smoke(P) :- person(P).
smokes(P) :- stress(P).
smokes(P) :- smokes(P2), influences(P2,P).
cancer(P) :- cancer_spont(P).
cancer(P) :- smokes(P), cancer_smoke(P).
```

In addition to the above program, we add some ground facts `person/1` and `friend/2` into the program. We add them by first deciding the number of people in the domain, and then randomly deciding friend relationships between people. We set the number of people to 4.

Results

Table 1 shows the average negative log-likelihood and the number of parameters for **WebKB** dataset. Here we use LFI, PG, PG+P to represent the results of LFI-ProbLog, projected gradient algorithm, and projected gradient algorithm with penalty term (the proposed method), respectively. We can see that PG+P shows comparable performance comparing with the state-of-the-art method LFI when $\lambda = 0.001$, while it can reduce the average number of parameters contained in

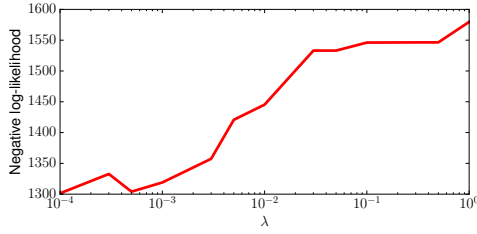


Figure 2: Negative log-likelihood for different λ .

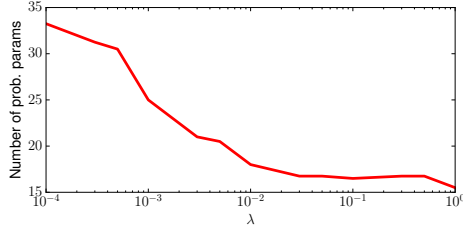


Figure 3: The number of estimated probabilistic parameters for different λ .

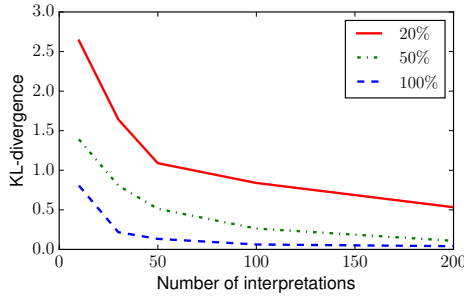


Figure 4: KL divergence on **Smokers** dataset with different numbers of evidences.

the learned model from 39 to 25. Following these results, we can answer **Q1** that the proposed method shows the inference performance that is comparable with the state-of-the-art algorithm, while it can reduce the number of parameters.

For answering **Q2**, we show the results of the proposed algorithm while changing the parameter λ in Fig.2 and Fig.3. From the result of Fig.3, we can see that the number of parameters contained in the learned program monotonically decreases when we use a large λ . This result reflects that the parameters tend to take 0 or 1 if we add more weight to the penalty term. Figure 2 shows that the performance decreases as we use a large λ . This result suggests the performance may decrease if we penalize too much.

To answer the **Q3**, we measured the KL divergence between the true probabilistic distribution and the distribution estimated by the proposed algorithm. We first make 10, 20, 50, 100, 200 different interpretations on **smokes/1** and **cancer/1** atoms of **Smokers** dataset, and then sample 20%, 50%, and 100% of them to make training data. We

evaluated the KL divergence between the true probabilistic distribution on **smokes/1** and **cancer/1** atoms and the distribution estimated from the data. Figure 4 shows the results. We can see that KL divergence decreases as the number of interpretations increases. We therefore can say that the proposed algorithm can estimate the true probabilistic distribution with a sufficient amount of training data (**Q3**).

Related Work

Many PLP models, such as ProbLog (De Raedt, Kimmig, and Toivonen 2007), PRISM (Sato and Kameya 2001), SLP (Muggleton 1996), ICL (Poole 2008), and parameter learning algorithms for these models have been proposed. Cussens proposed a parameter learning algorithm for SLP (Cussens 2001), Sato (Sato and Kameya 2001) proposed an EM learning algorithm for PRISM, and Gutmann et al. proposed two parameter learning algorithms for ProbLog (Gutmann, Thon, and De Raedt 2011; Gutmann et al. 2008). These algorithms exploit EM-learning or gradient descent methods to optimize an objective function for estimating parameters. Our proposed method differs in that we add penalty terms to induce parameters to take a zero or one probability. This feature resembles the sparse learning algorithms (Bach et al. 2009) used in many machine learning problems, but we believe that this paper is the first to apply a sparse learning method to a parameter learning problem for PLP models. ℓ_1 -regularization is used in structure learning for Markov Logic Networks (Huynh and Mooney 2011), however, the algorithm cannot be directly applied to PLP models like ProbLog.

Our work is also similar to probabilistic theory compression (De Raedt et al. 2008) and the theorem revision methods (Zelle and Mooney 1994) in that it tries to compress a theory into a more concise form. Our algorithm differs in that it simultaneously removes probabilistic facts and infers parameters in one operation.

Our algorithm can be seen as a kind of structure learning algorithm for PLP models, since it outputs a new PLP model given a prototype program and training examples. A previously reported structure learning algorithm for a PLP program is a beam search based algorithm, and it makes it necessary to solve the EM style parameter learning algorithm many times (Bellodi and Riguzzi 2012). Obviously we must conduct empirical comparisons, but we believe our algorithm can be more efficient than the previous structure learning approach since it can find a program by just performing projection gradient based optimization.

Conclusion

We proposed a novel parameter learning algorithm for PLP models that attempts to set the learned parameters so that they take either 0 or 1 by adding a penalty term to an optimization problem. With our algorithm, the learned ProbLog program will have fewer probabilistic components, and inference tasks performed with it become easier. We solved the optimization algorithm by combining the gradient projection algorithm and the computation of gradients in a d-DNNF based knowledge representation.

References

- Bach, F.; Jenatton, R.; Mairal, J.; and Obozinski, G. 2009. Optimization with sparsity-inducing penalties. *Foundations and Trends in Theoretical Computer Science* 3(2-3).
- Bellodi, E., and Riguzzi, F. 2012. Learning the structure of probabilistic logic programs. In *Inductive Logic Programming*, 61–75.
- Bertsekas, D. 1976. On the goldstein-levitin-polyak gradient projection method. *Automatic Control, IEEE Transactions on* 21(2):174–184.
- Bertsekas, D. P. 1999. *Nonlinear programming*. Athena Scientific.
- Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Trans. on* 100(8):677–691.
- Calamai, P. H., and Moré, J. J. 1987. Projected gradient methods for linearly constrained problems. *Mathematical Programming* 39(1):93–116.
- Craven, M., and Slattery, S. 2001. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning* 43(1-2):97–119.
- Cussens, J. 2001. Parameter estimation in stochastic logic programs. *Machine Learning* 44(3):245–271.
- Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *JAIR* 17:229–264.
- Darwiche, A. 2009. *Modeling and reasoning with Bayesian networks*. Cambridge University Press.
- De Raedt, L.; Kersting, K.; Kimmig, A.; Revoredo, K.; and Toivonen, H. 2008. Compressing probabilistic prolog programs. *Machine Learning* 70(2-3):151–168.
- De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, 2462–2467.
- Domingos, P., and Lowd, D. 2009. Markov logic: An interface layer for artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 3(1).
- Fierens, D.; Broeck, G. V. d.; Renkens, J.; Shterionov, D.; Gutmann, B.; Thon, I.; Janssens, G.; and De Raedt, L. 2013. Inference and learning in probabilistic logic programs using weighted boolean formulas. *arXiv preprint arXiv:1304.6810 (to appear in Theory and Practice of Logic Programming)*.
- Getoor, L., and Taskar, B. 2007. *Introduction to statistical relational learning*. The MIT press.
- Gutmann, B.; Kimmig, A.; Kersting, K.; and De Raedt, L. 2008. Parameter learning in probabilistic databases: A least squares approach. In *ECML/PKDD*, 473–488.
- Gutmann, B.; Thon, I.; and De Raedt, L. 2011. Learning the parameters of probabilistic logic programs from interpretations. In *ECML/PKDD*, 581–596.
- Huynh, T. N., and Mooney, R. J. 2011. Online structure learning for markov logic networks. In *ECML/PKDD*.
- Muggleton, S. 1996. Stochastic logic programs. *Advances in inductive logic programming* 32:254–264.
- Parikh, N., and Boyd, S. 2014. Proximal algorithms. *Foundations and Trends in Optimization* 1(3):123–231.
- Poole, D. 2008. The independent choice logic and beyond. In *Probabilistic inductive logic programming*. Springer. 222–243.
- Renkens, J.; Shterionov, D.; Van den Broeck, G.; Vlasselaer, J.; Fierens, D.; Meert, W.; Janssens, G.; and De Raedt, L. 2012. Problog2: From probabilistic programming to statistical relational learning. In *Proceedings of the NIPS Probabilistic Programming Workshop*.
- Sato, T., and Kameya, Y. 2001. Parameter learning of logic programs for symbolic-statistical modeling. *JAIR* 15:391–454.
- Sato, T. 1995. A statistical learning method for logic programs with distribution semantics. In *ICLP*, 715–729.
- Zelle, J. M., and Mooney, R. J. 1994. Inducing deterministic prolog parsers from treebanks: A machine learning approach. In *AAAI*, 748–753.

A Deeper Empirical Analysis of CBP Algorithm: Grounding Is the Bottleneck

Shrutika Poyrekar and Sriraam Natarajan and Kristian Kersting*

School of Computing and Informatics, Indiana University, USA

* Department of Computer Science, Technische Universitat Dortmund, Germany

Abstract

In this work-in-progress, we consider a lifted inference algorithm and analyze its scaling properties. We compare two versions of this algorithm – the original implementation and a newer implementation built on a database. Our preliminary results show that constructing the factor graph from the relational model rather than the construction of the compressed model is the key bottleneck for the application of lifted inference in large domains.

1 Counting Belief Propagation

Recent years have seen a surge of interest in lifted probabilistic inference algorithms that exploit redundancies to speed up inference, ultimately avoiding explicit state enumeration by manipulating first-order state representations directly. Lifted inference exploits the observation that ground models obtained by instantiating a set of logical models exhibit a large degree of symmetry, which leads to repeating the same set of computations multiple times. Accordingly, lifted algorithms operate on predicate level so that the inference is performed at a higher level instead of at the instance level. There have been several different algorithms proposed recently, ranging from exact algorithms (Choi, de Salvo Braz, and Bui 2011; de Salvo Braz, Amir, and Roth 2005; den Broeck et al. 2011; Milch et al. 2008; Poole 2003; Sen, Deshpande, and Getoor 2008), deterministic approximation of exact algorithms (Gogate and Domingos 2013; Ahmadi, Kersting, and Sanner 2011; Ahmadi, Kersting, and Hadji 2010; Kersting, Ahmadi, and Natarajan 2009; Nath and Domingos 2010; Singla and Domingos 2008), sampling algorithms (Milch and Russell 2006; Poon, Domingos, and Sumner 2008; Zettlemoyer, Pasula, and Kaelbling 2008), search-based algorithms (Gogate and Domingos 2011; Van den Broeck et al. 2011) and pre-processing (Shavlik and Natarajan 2009).

We had earlier developed *Counting Belief Propagation* based on message passing that is illustrated in Figure 1. The left-most graph in this figure shows a factor-graph representation of a graphical model. A factor graph is a bi-partite graph with random-variable nodes and factor nodes. For instance, in the first graph of the Figure 1, the circles with “A”,

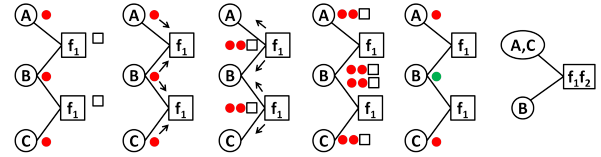


Figure 1: Counting Belief Propagation (CBP). From left to right, the steps of compressing a simple factor graph assuming no evidence. The shaded/colored small circles and squares denote the groups and signatures produced running CBP. On the right-hand side, the resulting compressed factor graph is shown.

“B”, and “C” in them represent random variables, and the squares represent the factors, which are essentially functions that encode the probabilistic relationships between two random variables. The belief-propagation (BP) algorithm works by passing real-valued functions called “messages” along the edges between the nodes. For the factor graph in the figure, let us assume that the nodes A , B , and C all send the same message to factors f_1 and f_2 , which send the message back to the nodes. As can be seen after first iteration, nodes A and C send the same message back while B sends two copies of the same message. This allows us to cluster the nodes A and C to a single node and the factors to a single factor. Hence, message passing can now take place on a smaller network. This is a simple example of a lifted inference algorithm from a message passing perspective.

This algorithm has now been widely applied to several tasks - from model counting and temporal inference (Kersting, Ahmadi, and Natarajan 2009) to computing personalized page ranks and Kalman filters (Ahmadi, Kersting, and Sanner 2011) to clustering on demand (Neumann, Ahmadi, and Kersting 2011) to solving linear programs (Mladenov, Ahmadi, and Kersting 2012). While this is successful in many real tasks, some important questions remain in understanding the scalability of the algorithm. Specifically, there is prevalent criticism that compression of the ground factor graph can be a bottleneck for this algorithm. In this paper, we explore this hypothesis in detail and investigate the relationship between construction of a ground factor graph and the compression of this factor graph in several domains. We show empirically that if there exists a robust, fast grounding technique, this is in fact a very practical algorithm. This

also opens up the potential research direction of constructing a fast (and possibly approximate) grounding algorithm. The results in this work are preliminary and help identify important future research directions.

We next present the experimental domains and the results of the evaluation in these domains. We then conclude by outlining some future research directions.

2 Empirical Analysis

To empirically analyze the CBP algorithm, we used two implementations: first is the original implementation of Kersting et al. (Kersting, Ahmadi, and Natarajan 2009) in Python. We focus here on the naive $\mathcal{O}(n^2)$ version of color-passing using flooding. Doing so is a sensible idea since flooding is easy to parallelize. There indeed are quasi-linear $\mathcal{O}((m+n)\log n)$ algorithms for color passing on graphs with n vertices and m edges due to asynchronous color updates, see e.g. (Berkholz, Bonsma, and Grohe 2013; Grohe et al. 2013) and references in there for more details, however, asynchronous updates are far more difficult to parallelize and the overhead may actually spoil the benefits of distributed computations. The second implementation is a java-based algorithm that uses Tuffy (Niu et al. 2012), a database implementation of Markov Logic networks that employs PostgreSQL underneath. In this implementation, we used Tuffy for grounding the MLN to a factor graph and implemented the compression step using the data structures (tables) provided by Tuffy. The aim was to answer the following two questions:

1. Does the use of a database system improve the performance of CBP? Does it scale well compared to the original implementation?
2. Which is the bottleneck? Construction of the factor graph or its compression?

To answer these two questions, we considered four standard MLNs that are used widely by the StaRAI community. They are (i) Smokers (ii) Cora entity resolution (iii) UWCSE and (iv) WebKB. These MLNs were obtained from the alchemy website¹. We kept the evidence to about 10% of the total groundings (this evidence is significantly high for many lifted inference algorithms to handle). We increased the number of objects in the domain to better understand the scaling properties of the algorithms. The reported results are averaged over 5 runs. It must be mentioned that the number of MLN clauses in Smokers and WebKB domain are significantly smaller than the number of clauses in the other two domains. Thus the algorithms scale better in these two domains.

The results of the (initial) empirical analysis are presented in Figure 2. Both the methods converged to the exact same marginals for the queries indicating that there is no difference in the quality of the results. The second row of the results are the results in the 4 domains of the original CBP implementation (we will call this as CBPP to denote CBP in Python) and the first row is the Tuffy based implementation (we will call this as CBPT) in Java. It can be observed in

all the domains of both the implementations that grounding of the MLN to a factor graph is the key bottleneck compared to that of the compression. This suggests that there is a necessity in improving the grounding of the network for efficient inference. Also, it can be observed that in some domains, CBPP was not able to compress the factor graph as it ran out of memory. Note that in all domains, the number of ground atoms is squared of the number of objects. Hence, in Cora domain for example, CBPP could not compress with 1M ground atoms while CBPT was able to handle 1B ground atoms. In WebKB, both methods were able to handle 1B ground atoms as the number of clauses was significantly small.

In both domains, CBPP is faster than CBPT due to its faster implementation in C++ and Python as the list operations inside Python are faster. But as mentioned earlier, the Tuffy based implementation is able to scale much better than CBPP (1B vs 10k in Cora and 100M vs 100 in UWCSE). This demonstrates that the use of a database underneath helps in scaling up the lifted inference algorithm by order of magnitude.

To summarize and answer the two questions (1) CBPT scales significantly better than CBPP algorithm although the latter is faster due to its implementation in Python and (2) Construction of the ground factor graph is the significant bottleneck in all the domains for both the algorithms.

3 Discussion

As far as we are aware, this is the first preliminary evaluation of the scalability of a lifted inference algorithm. Our goal was to evaluate whether the algorithms can handle a scale of millions of ground atoms and evidence that can be in millions of ground atoms. It is clear that with the use of underlying data base technology, BP algorithm can scale well.

The results while initially promising also raise several questions and challenges that need to be addressed before the algorithm can be applied on larger data sets:) (1) the full grounding must be avoided to form the factor graph. Approaches such as FROG (Shavlik and Natarajan 2009) and Tuffy’s default grounding require knowledge of the query to avoid fully grounding the network. (2) It would be an interesting research direction to determine how to incrementally ground the data for compression of the factor graph. (3) There is a necessity to parallelize the grounding operation to scale up inference. There has been prior work on using MapReduce for the compression step (Ahmadi et al. 2013) but as we show here, the key bottleneck is still the grounding process which needs to be sped up. (4) Finally, combining incremental grounding with compression in the same step would be an interesting direction for future research.

4 Acknowledgement

We like to acknowledge the work performed under LongView International Inc. AFRL contract FA8750-14-C-0022 under AFOSR STTR Topic AF13-AT11. The opinions and conclusions do not reflect the position of the Air Force, AFOSR or AFRL. The authors also gratefully acknowledge

¹<http://alchemy.cs.washington.edu/mlns/>

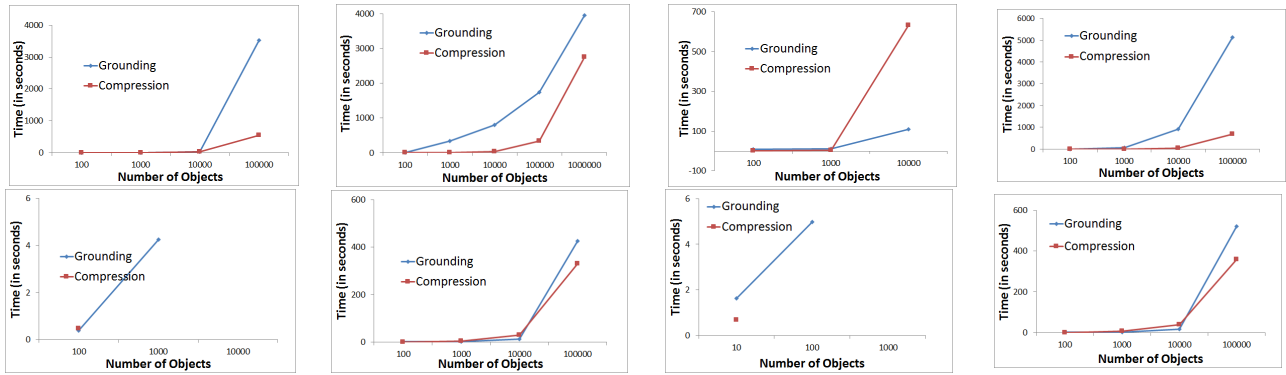


Fig. 2: Results of our evaluation. The top row presents the results of the CBPT algorithm while the bottom row presents the results of CBPP algorithm. Columnwise, the first column is the Cora data set, the second is the Smokers data set, third is UW-CSE and the last column is the WebKB data set.

discussions with the AFRL COTR James Nagy for his constructive feedback and support.

References

- Ahmadi, B.; Kersting, K.; Mladenov, M.; and Natarajan, S. 2013. Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training. *Machine Learning*.
- Ahmadi, B.; Kersting, K.; and Hadiji, F. 2010. Lifted belief propagation: Pairwise marginals and beyond. In P. Myllymaeki, T. Roos, T. J., ed., *Proceedings of the 5th European Workshop on Probabilistic Graphical Models (PGM-10)*.
- Ahmadi, B.; Kersting, K.; and Sanner, S. 2011. Multi-evidence lifted message passing with application to pagerank and the kalman filter. In *IJCAI*.
- Berkholz, C.; Bonsma, P.; and Grohe, M. 2013. Tight lower and upper bounds for the complexity of canonical colour refinement. In *21st Annual European Symposium on Algorithms (ESA)*, 145–156.
- Choi, J.; de Salvo Braz, R.; and Bui, H. 2011. Efficient methods for lifted inference with aggregate factors. In *AAAI 2011*.
- de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted First Order Probabilistic Inference. In *IJCAI*, 1319–1325.
- den Broeck, G. V.; Taghipour, N.; Meert, W.; Davis, J.; and Raedt, L. D. 2011. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, 2178–2185.
- Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *Proc. 27th Conf. Uncertainty in AI*.
- Gogate, V., and Domingos, P. 2013. Structured message passing. In *UAI13*.
- Grohe, M.; Kersting, K.; Mladenov, M.; and Selman, E. 2013. Dimension reduction via colour refinement. In *arXiv:1307.5697*.
- Kersting, K.; Ahmadi, B.; and Natarajan, S. 2009. Counting Belief Propagation. In *UAI*.
- Milch, B., and Russell, S. 2006. General-purpose mcmc inference over relational structures. In *Uncertainty in Artificial Intelligence*, 349–358. AUAI Press.
- Milch, B.; Zettlemoyer, L.; Kersting, K.; Haimes, M.; and Pack Kaelbling, L. 2008. Lifted Probabilistic Inference with Counting Formulas. In *AAAI*.
- Mladenov, M.; Ahmadi, B.; and Kersting, K. 2012. Lifted linear programming. In *AISTATS*, 788–797.
- Nath, A., and Domingos, P. 2010. Efficient lifting for online probabilistic inference. In *AAAI*.
- Neumann, M.; Ahmadi, B.; and Kersting, K. 2011. Markov logic sets: Towards lifted information retrieval using pagerank and label propagation. In *AAAI*.
- Niu, F.; Zhang, C.; Re, C.; and Shavlik, J. 2012. Scaling inference for markov logic via dual decomposition. In *ICDM*, 1032–1037.
- Poole, D. 2003. First-Order Probabilistic Inference. In *IJCAI*, 985–991.
- Poon, H.; Domingos, P.; and Sumner, M. 2008. A general method for reducing the complexity of relational inference and its application to mcmc. In *AAAI*, 1075–1080.
- Sen, P.; Deshpande, A.; and Getoor, L. 2008. Exploiting shared correlations in probabilistic databases. *Proc. VLDB Endow.* 1:809–820.
- Shavlik, J., and Natarajan, S. 2009. Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *IJCAI*.
- Singla, P., and Domingos, P. 2008. Lifted first-order belief propagation. In *AAAI*, 1094–1099.
- Van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*.
- Zettlemoyer, L. S.; Pasula, H. M.; and Kaelbling, L. P. 2008. Logical particle filtering. In *Probabilistic, Logical and Relational Learning - A Further Synthesis*, number 07161 in Dagstuhl Seminar Proceedings.

Explanation-Based Approximate Weighted Model Counting for Probabilistic Logics

Joris Renkens, Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt

KU Leuven
Celestijnenlaan 200A
3001 Heverlee - Belgium

Abstract

Probabilistic inference can be realized using weighted model counting. Despite a lot of progress, computing weighted model counts exactly is still infeasible for many problems of interest, and one typically has to resort to approximation methods. We contribute a new bounded approximation method for weighted model counting based on probabilistic logic programming principles. Our bounded approximation algorithm is an anytime algorithm that provides lower and upper bounds on the weighted model count. An empirical evaluation on probabilistic logic programs shows that our approach is effective in many cases that are currently beyond the reach of exact methods. (To be published at AAAI14)

1 Introduction

During the last few years there has been a significant interest in statistical relational learning (Getoor and Taskar 2007; De Raedt et al. 2008) and probabilistic databases (Dalvi and Suciu 2004), leading to the definition of many expressive probabilistic logical languages such as Markov Logic (Richardson and Domingos 2006), PRISM (Sato 1995) and ProbLog (De Raedt, Kimmig, and Toivonen 2007). It is by now well-known that probabilistic inference for such languages corresponds to weighted model counting (WMC) on weighted logical formulae (Darwiche 2009; Fierens et al. 2011). Knowledge compilation (Darwiche and Marquis 2002) is an effective and popular technique for WMC. It essentially compiles the weighted logical formulae into logical circuits on which inference can be performed efficiently.

Despite a lot of progress in knowledge compilation and other probabilistic inference techniques, inference remains the bottleneck when facing real-life problems and doing applications. One often has to resort to approximate inference methods, typically employing some form of sampling. While sampling is often effective, current sampling methods do not give strong guarantees (in the form of bounds on the computed probabilities).

The key contribution of this paper is the introduction of a novel anytime approximate inference method for WMC using knowledge compilation that provides bounds on the computed probabilities. The technique is inspired by probabilistic logic programming techniques due to (Poole 1993;

De Raedt, Kimmig, and Toivonen 2007; Renkens, Van den Broeck, and Nijssen 2012), where bounds were derived based on a subset of proofs (or explanations) instead of all proofs, with the actual subset depending on the search strategy employed. Rather than taking a logic programming perspective, however, this paper takes a weighted model counting and knowledge compilation perspective. This potentially opens the way to develop WMC approximation techniques for other types of probabilistic formalisms such as Markov Logic and Bayesian networks.

Our technique is empirically evaluated on a number of benchmark problems. The experiments provide evidence that it is effective for probabilistic languages working under Sato’s distribution semantics (Sato 1995), whose probability distribution over possible worlds is defined using a basic distribution over extensional probabilistic facts (sometimes called atomic choices) which then, together with a logic program, induces a distribution over the intensionally defined atoms. Thus, under Sato’s distribution semantics, the truth-values of a limited number of extensional probabilistic facts determine the probability of the possible worlds. This is unlike Markov Logic, where most formulae are soft and the truth-value of one atom therefore typically depends on that of almost all other atoms.

The structure of the paper is as follows. We first formalize the problem and discuss the basic solution idea in Section 2. Section 3 defines the central concept of *explanation*, and Section 4 introduces the algorithm as used in the experiments in Section 7. Section 5 positions our approach with respect to WMC in general, and Section 6 discusses related work.

2 Background

Many probabilistic programming languages, including PRISM (Sato 1995), ICL (Poole 1997), ProbLog (De Raedt, Kimmig, and Toivonen 2007) and LPADs (Vennekens, Verbaeten, and Bruynooghe 2004), are based on Sato’s distribution semantics (Sato 1995). They start from a set of probabilistic independent choices (ground atoms) and define further derived predicates using logic programs. In such a program, each ground atom corresponds to a random variable. We will call the probabilistic choices extensional variables E and the derived atoms intensional variables I .

Example 1. Let us define two extensional variables `head1`

and head2, representing two biased coin flips. The game is won when the two coin flips have the same outcome. We use the ProbLog notation where the extensional variables are ground facts annotated with their probability and the intensional variables are defined using a logic program.

```
0.4::head1.
0.7::head2.
twoHeads :- head1, head2.
twoTails :- not head1, not head2.
win :- twoHeads; twoTails.
```

The probability of each truth assignment to the extensional variables is obtained as the product of the probability of the individual assignments. If the fact is true, it contributes the corresponding probability; if it is false it contributes 1 minus the probability of the fact. The truth values for the intensional variables are derived by logical deduction using the logic program. This process results in a model of the theory. A typical query in probabilistic logic programming languages asks for the marginal probability of a query atom q given the theory Δ : $P(q | \Delta)$, which is defined as the sum of the weights of the models of the theory in which the query is true. It is well known that this probability can be computed using weighted model counting.

Definition 1. (Weighted Model Count) Let Φ be a propositional logic formula over variables V , w be a weight function that assigns a positive (real) weight value to each literal v and $\neg v$ (with $v \in V$), and $\mathcal{I}(V)$ the set of interpretations of V , that is, the set of all possible truth value assignments to variables in V . The *weighted model count* (WMC) of Φ is

$$\text{WMC}_V(\Phi) = \sum_{i \in \mathcal{I}(V), i \models \Phi} \prod_{l \in i} w(l).$$

The probability $P(q | \Delta)$ can be computed as $\text{WMC}(\Phi)$, where Φ is a propositional representation of $q \wedge \Delta$, and the weight function is $w(v) = p_v$ and $w(\neg v) = 1 - p_v$ for the variables $v \in E$, and $w(v) = w(\neg v) = 1$ for the variables $v \in I$, cf. (Fierens et al. 2011) for more details on this transformation. In the remainder of this paper, when the set of variables over which to do WMC is omitted, we assume it consists of the variables present in the formula.

Example 2. In our example, we obtain the following propositional formula (where we abbreviate names of propositional variables based on the names of logical atoms) and weight function for WMC.

$$\begin{aligned} th &\leftrightarrow h1 \wedge h2 \\ tt &\leftrightarrow \neg h1 \wedge \neg h2 \\ win &\leftrightarrow th \vee tt \end{aligned}$$

	<i>true</i>	<i>false</i>
<i>h1</i>	0.4	0.6
<i>h2</i>	0.7	0.3
<i>th</i>	1	1
<i>tt</i>	1	1
<i>win</i>	1	1

```
 $\psi \leftarrow \text{false}$ 
while  $\text{time} < \text{maxTime}$  do
  |  $e \leftarrow \text{nextExpl}(\Delta \wedge q)$ 
  |  $\psi \leftarrow \psi \vee e$ 
end
return  $\text{WMC}_V(\psi)$ 
```

Algorithm 1: Computing a lower bound on $\text{WMC}(\Delta \wedge q)$ given a time budget maxTime .

A popular and effective approach to WMC is by knowledge compilation: Φ is compiled into a graph structure on which the WMC can be computed efficiently (Darwiche and Marquis 2002). However, since WMC is a #P-complete problem, compilation often is infeasible in practice, and approximation methods are needed. The approximation technique proposed here is sketched in Algorithm 1. It is based on (De Raedt, Kimmig, and Toivonen 2007; Renkens, Van den Broeck, and Nijssen 2012) where a formula ψ in disjunctive normal form (DNF) is used to approximate $\text{WMC}(\Delta \wedge q)$. Each explanation e in this algorithm is a conjunction of extensional variables that satisfies $\Delta \wedge q$.

Since the intensional variables do not influence the WMC, they can be excluded from the calculations. And since every model of an explanation e represents a model in which q is true, its WMC has to be a lower bound: $\text{WMC}_E(e) \leq \text{WMC}(\Delta \wedge q)$. Furthermore, taking the disjunction of two explanations is equivalent to taking the union of the models of the two explanations. This means that the WMC of ψ will never decrease by adding an explanation and since the number of explanations is finite, it will converge to the actual WMC.

Example 3. In our running example, there are two possible explanations when calculating $\text{WMC}(\Delta \wedge \text{win})$: $e_1 = h1 \wedge h2$ and $e_2 = \neg h1 \wedge \neg h2$. Both explanations entail win relative to theory Δ , that is, $\Delta \wedge e_i \models \Delta \wedge \text{win}$. Therefore, $\text{WMC}(\Delta \wedge e_i) \leq \text{WMC}(\Delta \wedge \text{win})$. By taking the disjunction $e_1 \vee e_2$ of these two explanations, all models of $\Delta \wedge \text{win}$ are accounted for and we have $\text{WMC}(\Delta \wedge (e_1 \vee e_2)) = \text{WMC}(\Delta \wedge \text{win})$. A further important observation is that the weight of all the intensional variables is identical (here equal to 1) in all models, which allows us to simplify $\text{WMC}(\Delta \wedge (e_1 \vee e_2)) = c \cdot \text{WMC}_E(e_1 \vee e_2)$ (with $c = 1$).

The WMC of a disjunction of explanations is typically much easier to compute than that of the original formula since it has fewer variables. In the next section we present a formal framework for explanations that guarantees that the WMC can be simplified and bounded.

3 Explanations

Let us now formally introduce the notion of an *explanation* used in the algorithm sketched above. Our definitions are motivated by probabilistic logic programming languages (PLP) and databases. In such languages, the truth values of the extensional variables (or predicates) determine the truth values of the intensional ones (through some fixpoint computation), and the probability of a world is a function of only

its extensional variables. In this and the next section, we focus on WMC problems in this context; a discussion of the general case can be found in Section 5.

For WMC problems generated by probabilistic logic programs, this separation of variables translates into the notion of an intensional split. Although the formulae for WMC are generally provided in CNF, we will use equivalences for the sake of readability. It is trivial to transform them into CNF.

Definition 2 (Intensional Split). An *intensional split* of a WMC problem with theory Φ over variables V is a tuple (E, I, Δ, Γ) , consisting of an ordered set of *intensional variables* $I = \{d_1, \dots, d_k\} \subseteq V$, a set of *extensional variables* $E = V \setminus I$, a *defining theory* Δ and a *constraining theory* Γ , such that

- (i) $\Phi \equiv \Delta \wedge \Gamma$,
- (ii) $\Delta = \bigwedge_{j=1}^k (d_j \leftrightarrow \alpha_j)$
- (iii) all α_j range over the variables $E \cup \{d_{j+1}, \dots, d_k\}$, and
- (iv) all d_i have uniform weights $w(d_i) = w(\neg d_i) = 1$.

The intuition is that for every value assignment to the extensional variables that satisfies Γ , there is exactly one possible truth-assignment to the intensional variables that satisfies the formula Φ . In addition, the weight of a world is only a function of the extensional variables.

Our algorithm will assume that an intensional split of the theory is given. Starting from a probabilistic logic program, the intensional split of its WMC problem is immediately clear: the extensional variables are the groundings of probabilistic facts, and the intensional variables are the groundings of the heads of clauses. The constraining theory consists of queries and constraints, and the defining theory represents the grounding of Prolog clauses.

Example 4. In our running example, we have Δ exactly as used so far, and $\Gamma \equiv \text{win}$.

Definition 3. (Explanation) An *explanation* e of a formula Φ with extensional split (E, I, Δ, Γ) is a conjunction of literals, containing no variables from I , and at most one of v and $\neg v$ for every variable $v \in E$, such that

$$\Delta \wedge e \models \Gamma \quad (1)$$

Intuitively, an explanation fixes truth values of a subset of the extensional variables that is sufficient to ensure that the constraints Γ are satisfied by all models of the explanation together with the definitions, where the latter propagate truth values from those extensional variables to the intensional variables.

Example 5. In our example, an explanation has to satisfy $\Delta \wedge e \models \text{win}$, as already illustrated in Example 3.

Theorem 1. Given a theory Φ with intensional split (E, I, Δ, Γ) and a disjunction of explanations $e_1 \vee \dots \vee e_m$ for Φ , we have

$$\text{WMC}_V(\Phi \wedge (e_1 \vee \dots \vee e_m)) = \text{WMC}_E(e_1 \vee \dots \vee e_m) \quad (2)$$

For a single explanation e ,

$$\text{WMC}_E(e) = \prod_{l \in e} w(l) \cdot \prod_{v \in E \setminus V_e} (w(v) + w(\neg v)) \quad (3)$$

with V_e denoting those variables occurring in e .

Proof outline. $\text{WMC}_V(\Phi \wedge e)$ sums the weights of all interpretations i for which $i \models \Phi \wedge e$, or equivalently $i \models \Delta \wedge \Gamma \wedge e$. Such an interpretation has to agree with e on the extensional variables fixed by the latter. By construction, for every assignment to the extensional variables, there is a single assignment to the intensional variables such that the combination of the two is a model of Δ . All literals for intensional variables have weight 1, and thus can be omitted from the product (and the sum). By (1), every model of $\Delta \wedge e$ is also a model of Γ . Equation (3) then follows by distributivity. $\square$

This shows that we can efficiently compute the WMC of $\Phi \wedge e$ for any given explanation e . We will discuss how to find such explanations in the next section.

4 Algorithm

Given the notion of an explanation, we are now in a position to define the algorithm that finds explanations, and that iteratively computes bounds on our queries.

Explanation Finding

We now introduce the nextExpl method of Algorithm 1 to find the next best explanation, that is, the explanation maximizing the WMC in Equation (3). It is based on a reduction to a weighted partial max-SAT (WPMS) problem:

Given: a set of weighted clauses

$$WC = \{w_i : c_i \mid \text{with } w_i \text{ the weight of the clause } c_i\}$$

Find: $\arg \min_{i \in \mathcal{I}(V)} \sum_{i \models c_i} w_i$.

The goal of WPMS thus is to find an interpretation i with smallest (weighted) sum of violated clauses, where hard constraints can be imposed by setting $w = \infty$.

We transform a theory Φ with intensional split (E, I, Δ, Γ) into a WPMS $W(\Phi)$ as follows, where we introduce two WPMS variables v^+ and v^- for each variable $v \in V$, indicating that the value of v is fixed to true or false, respectively:

1. assign weight ∞ to each formula in Φ ;
2. for all variables $v \in V$:
 - replace all positive literals v in Φ by v^+ ,
 - replace all negative literals $\neg v$ in Φ by v^- ,
 - add $\infty : \neg v^- \vee \neg v^+$ to Φ ;
3. for all variables $v \in I$, replace $\infty : (v^+ \leftrightarrow \alpha)$ in Φ by $\infty : (v^+ \leftrightarrow \alpha) \vee (\neg v^+ \wedge \neg v^-)$
4. for all variables $v \in E$, add the following to Φ :
 - (a) $-\log(w(v)) : \neg v^+$
 - (b) $-\log(w(\neg v)) : \neg v^-$
 - (c) $-\log(w(v) + w(\neg v)) : v^+ \vee v^-$
5. rewrite Φ to clausal form

Theorem 2. A solution of the WPMS $W(\Phi)$ encodes an explanation of Φ with maximal WMC.

Proof outline. Step 1 turns Φ into hard constraints. Step 2 introduces the encoding which allows the WPMS solver to leave the truth value of a variable open (by setting both v^+ and v^- to false), but at the same time excludes a contradicting assignment (setting both v^+ and v^- to true). Step 3 states that for each intensional variable, if it has a truth value assigned, its definition needs to hold. All constraints up to here are hard, and thus have to be satisfied by every assignment. Step 4 adds the soft clauses influencing the minimization. If a clause of type (a) is violated, v^+ is true, and the corresponding positive literal v is part of the explanation, in which case the negative logarithm of its weight is added to the sum to be minimized. Clauses of type (b) achieve the same for negative literals. Clauses of type (c) cover variables that are not part of the explanation; these clauses are violated if the variable is set neither true nor false. Exactly one type of clause is violated for any legal assignment to v^+ and v^- , and the sum becomes the negative logarithm of Equation (3). Minimizing this logarithm is equivalent to maximizing Equation (3). $\square$

The solutions of the WPMS problem contain intensional variables, which can simply be dropped since their truth values are implied by the extensional variables and the definitions.

This reduction is the basis of procedure `nextExpl`, which calls a weighted maxSAT solver on $W(\Phi)$ to produce the next explanation. Once an explanation $v_1 \wedge \dots \wedge v_i \wedge \neg v_{i+1} \wedge \dots \wedge \neg v_n$ has been found, we simply extend $W(\Phi)$ with the hard constraint $\infty : \neg v_1^+ \vee \dots \vee \neg v_i^+ \vee \neg v_{i+1}^- \vee \dots \vee \neg v_n^-$ in order to avoid that the same explanation is found again, and iterate. Note that this constraint does not guarantee that the found explanations are mutually exclusive.

Computing Bounds

For arbitrary WMC tasks, we can use Algorithm 1 with an implementation of `nextExpl(.)` based on the WPMS approach discussed above, that is, prior to each call, we add a new hard clause corresponding to the last solution.

We can do better when $WMC(\Delta)$ represents a probability distribution ($WMC(\Delta) = 1$), as is the case in the PLP setting. In this case we can calculate an upper bound by calculating a lower bound on $WMC(\Delta \wedge \neg q)$ and by using $WMC(\Delta \wedge q) = WMC(\Delta) - WMC(\Delta \wedge \neg q)$. Algorithm 2 sketches an anytime algorithm based on these observations, using the WPMS reduction to implement `nextExpl(.)`. It keeps updating the lower and upper bounds, in each iteration adding a new explanation to the bound whose last update resulted in the largest change in value.

5 Beyond Probabilistic Logic Programs

Of all existing WMC algorithms, exact and approximate, our algorithm takes a rather unique position. Existing algorithms, such as `c2d` (Darwiche 2004) and `Cachet` (Sang et al. 2004), are intended to solve arbitrary WMC problems (in CNF). This is partly their strength, as this permits their application to Bayesian network, PLP, and Markov logic inference equally. Our algorithm applies to general WMC problems (as we discuss later). Yet, its main strength comes from

```

improveTop  $\leftarrow$  0.5
improveBot  $\leftarrow$  0.5
top  $\leftarrow$  true
bot  $\leftarrow$  false
up  $\leftarrow$  1
low  $\leftarrow$  0
while time < maxTime do
  if improveTop > improveBot then
    e  $\leftarrow$  nextExpl( $\Delta \wedge \neg \Gamma$ )
    next  $\leftarrow$  WMC(top  $\wedge \neg e$ )
    improveTop  $\leftarrow$  up - next
    top  $\leftarrow$  top  $\wedge \neg e$ 
    up  $\leftarrow$  next
  else
    e  $\leftarrow$  nextExpl( $\Delta \wedge \Gamma$ )
    next  $\leftarrow$  WMC(bot  $\vee e$ )
    improveBot  $\leftarrow$  next - low
    bot  $\leftarrow$  bot  $\vee e$ 
    low  $\leftarrow$  next
end
return [low, up]

```

Algorithm 2: Computing lower as well as upper bounds in the PLP setting given a time budget *maxTime*.

exploiting the presence of intensional variables and definitions. This type of logical structure is found in WMC encodings of probabilistic logic programs. For example, in the formulae for the GENES dataset used in the experiments (Section 7), the number of intensional variables is on average three times that of extensional variables. This means that using the intensional split can allow significantly smaller explanations. However, intensional variables may not appear in arbitrary WMC problems, or encodings of other formalisms. The converse is also true: the generality of existing WMC solvers prohibits them from easily exploiting the presence of definitions and intensional variables, and our algorithm loses much of its appeal when applied to general WMC problems.

This distinction between general-purpose solvers and solvers that exploit structural properties of the WMC encoding opens up new questions. Can we develop WMC solvers that exploit structural properties of Bayesian network encodings, or encodings of Markov logic networks? For Bayesian networks alone, three different WMC encodings have been proposed in the literature (Darwiche 2009), each with their own intrinsic properties. In the future, we anticipate to see more WMC solvers that exploit the properties of a specific WMC encoding. To bridge the gap in the other direction, we will now investigate the application of our algorithm to general WMC problems.

Finding Intensional Splits

So far, we have assumed that an intensional split can be read off directly from the probabilistic logic program that generated the WMC problem at hand. When presented with an arbitrary WMC problem, not necessarily encoding a prob-

abilistic logic program, we can still use our algorithm. We only need to provide an intensional split, which can be found as follows. Starting with $\Delta = \text{true}$ and $I = \emptyset$, repeatedly delete a subformula ($d \leftrightarrow \alpha$) from theory Φ for which $w(d) = w(\neg d)$ and neither d nor any of the variables in α belong to I , conjoin this subformula to Δ and add the variable d to I . Proceed until no further definitions can be found in Φ . Then $\Gamma = \Phi$ and $E = V \setminus I$.

It is desirable to obtain the split that maximizes $|I|$, so as to minimize the length of our explanations. In this regard, the effectiveness of the split-finding procedure above strongly depends on the order in which subformulae are selected. In fact, we can show the following.

Theorem 3. *Given a WMC problem, finding its intensional split that maximizes $|I|$ is NP-hard.*

Proof outline. The proof is by reduction from the maximum independent set problem. Given a graph $G = (V, \mathcal{E})$, an independent set $S \subseteq V$ contains no two vertices that are connected by an edge in $\mathcal{E}$. The maximum independent set problem is concerned with finding the independent set that maximizes $|S|$, and is NP-hard (Karp 1972).

We can encode a maximum independent set problem into the problem of finding an optimal intensional split as follows. Let there be a set of variables $V = \{v_1, \dots, v_n\}$ denoting the vertices V . Suppose that for every variable v whose neighbors in G are denoted by variables $n_1, \dots, n_k$, theory Φ contains the formula $v \leftrightarrow n_1 \vee \dots \vee n_k$. Suppose further that $w(\ell) = 1$ for all literals ℓ . There is now a one-to-one mapping between intensional splits of Φ and independent sets of G , where variables I of the split denote an independent set of vertices. Indeed, no two can have an edge between them, as this would violate property (iii) of Definition 2. Conversely, every independent set has a corresponding intensional split whose $I = S$ and $E = V \setminus S$, which can be verified by checking properties (i)-(iv) of Definition 2. In this mapping, $|S| = |I|$, and we can solve the NP-hard maximum independent set problem by finding the intensional split that maximizes $|I|$. $\square$

Nonetheless, our algorithm works with any intensional split, not only the one minimizing $|I|$, and approximately optimal splits can be used in practice.

6 Related Work

Our approximate WMC algorithm is closely related, and inspired by several approximation techniques developed in the PLP community (Poole 1993; Kimmig et al. 2008; Renkens, Van den Broeck, and Nijssen 2012). Their key observation is that the probability of a probabilistic Prolog query equals the probability of its proofs, and that this probability can be approximated, and bounded below by a subset of the proofs. Whereas Kimmig et al. (2008) search for the k proofs with the highest individual probability, Renkens, Van den Broeck, and Nijssen (2012) search for the k proofs whose joint probability is maximized. They show that this optimization problem is NP-hard. Luckily it is submodular, and therefore easy to optimize greedily within bounds.

The key difference between the PLP techniques and our approach is that we search for explanations of WMC problems, not for proofs. This has several advantages. Firstly, while the PLP techniques work with probabilistic logic programs that have no negation or loops, our approach is oblivious to which program generated the WMC problem it operates on. The reductions from PLP to WMC of Fierens et al. (2011) support negation and loops, and so does our work. In fact, the need for approximate inference with such programs is what motivated us. Moreover, we can directly benefit from any future extensions of these conversions, for example supporting new PLP language concepts. Secondly, support for negation directly allows us to approximate the negation of a query. Our algorithm can therefore also compute upper bounds on queries, which is not a feature of the earlier algorithms. Thirdly, our approach obviates the need to find individual Prolog proofs. Instead, it requires a PLP-to-WMC conversion, which is very different in nature from proof-finding, and an off-the-shelf WPMS solver.

This work follows a tradition of *approximate knowledge compilation* and *theory approximation* (Selman and Kautz 1996). There, the goal is to approximate a theory Φ below and above such that $\Phi_{lb} \models \Phi \models \Phi_{ub}$, and Φ_{lb}, Φ_{ub} allow efficient logical query answering. We follow a similar approach to approximate WMC problems.

Finally, by reducing approximate inference to a WMC task, our algorithm exploits local structure and determinism. As such, it follows a long line of research on *approximate model counting* (Wei and Selman 2005; Gomes, Sabharwal, and Selman 2006), more general approximate probabilistic inference algorithms that exploit local structure and determinism (Poon and Domingos 2006; Gogate and Dechter 2011) and bounded inference in graphical models (Mooij and Kappen 2008; Bidyuk and Dechter 2006; Dechter and Rish 2003; Wainwright, Jaakkola, and Willsky 2003; Gogate and Domingos 2012).

7 Experiments

We experimentally evaluate our approximation algorithm as given in Algorithm 2, focusing on two questions:

- Q1:** How does approximate inference compare to exact inference in general?
- Q2:** How does approximate inference perform when exact inference is not feasible?

Both approximate and exact inference use the PLP-to-WMC conversion of Fierens et al. (2011) to obtain the theory Φ , and compilation to SDDs (Darwiche 2011) to compute WMCs. The WPMS solver used in approximate inference is WPM1 (Ansótegui, Bonet, and Levy 2009). Experiments are run on a 3.4 GHz machine with 16 GB of memory.

Our comparison uses two datasets. The first is WEBKB¹, where university webpages have to be tagged with classes. The tag of a page depends on both its textual content as well as the classes of the pages it links to. Following Fierens et al. (2011), we select 150 pages, and we use the 100 most frequent words only. Probabilities are chosen randomly from

¹<http://www.cs.cmu.edu/~webkb/>

[0.1, 0.4]. This dataset contains 62 queries, and we give each algorithm a time budget of five minutes per query. The second dataset is the biological application of (Ourfali et al. 2007), where the regulatory effect of a gene on another gene has to be predicted based on combinations of different types of molecular interactions. We refer to this dataset as GENES. This dataset contains 500 queries, and we give each algorithm a time budget of ten minutes per query.

For **Q1**, we focus on the queries for which exact inference did return a result within the allocated time. On WEBKB, exact inference solved 15 of the 62 queries, and approximate inference converged to the exact value for four of these. On GENES, exact inference solved 258 of the 500 queries, and approximate inference converged to the exact value for 173 of these. We further consider three different thresholds (0.05, 0.01, 0.0001) on the quality of approximation as measured by the maximal error (i.e., half of the difference between the bounds). For each threshold, in Figures 1a (WEBKB) and 1b (GENES), we plot the number of queries solved by exact inference at any point within the time budget against the number of queries approximated to at least the threshold quality at that time. For a maximal error of 0.05, in both cases the approximate inference outperforms the exact inference for any timeout. For smaller error thresholds, the results for the two datasets differ. In the case of WEBKB, approximate inference outperforms exact inference initially, but exact inference solves more queries if allowed more time. On GENES, exact inference solves more queries initially, but is outperformed by approximate inference if more time is given. As an answer to **Q1**, we thus conclude that although approximate inference is competitive with exact inference, even when the allowed error is small, it is hard to predict when approximate inference will do better. A lot depends on the structure of the given program.

For **Q2**, we focus on the queries on which exact inference did not return a result within the allocated time (47 out of 62 for WEBKB, 242 out of 500 for GENES). Figures 2a (WEBKB) and 2b (GENES) show for each such query the returned interval plotted around the average of the lower and upper bound, ordered by increasing difference between the bounds. Here, the trend is the same on both datasets. Our algorithm achieves close approximations for most queries, though there are also queries that are hard for this approach. As an answer to **Q2**, we thus conclude that approximate inference performs well on most of the queries, even when exact inference is not feasible.

8 Conclusions

We have proposed an anytime approximation algorithm for WMC motivated by approximation techniques from probabilistic logic programming. The explanation-based approach searches for conjunctions of literals whose WMC is easy to calculate and can be used to approximate the WMC of the original problem. Furthermore, explanations can be combined to obtain a better approximation. The approach has been used in an approximation algorithm that can calculate hard bounds on the marginal probability for probabilistic logic programs which has been tested empirically on two datasets. The results show that the approximate inference

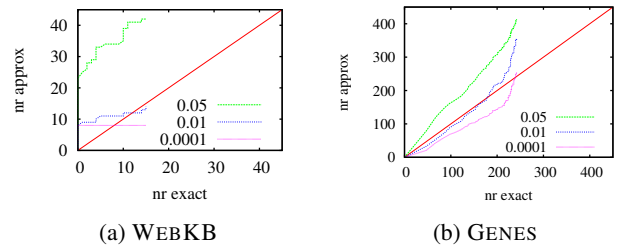


Figure 1: Number of queries solved by exact inference (x-axis) and by approximate inference with different thresholds on approximation error (y-axis) over time.

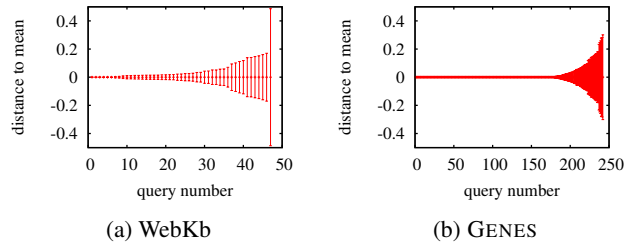


Figure 2: Interval sizes obtained by approximate inference on queries not solved by exact inference within the time budget.

is competitive with exact inference and performs well, even when exact inference is not feasible.

9 Acknowledgments

Joris Renkens is supported by the IWT (gentschap voor Innovatie door Wetenschap en Technologie). Angelika Kimmig and Guy Van den Broeck are supported by the Research Foundation-Flanders (FWO-Vlaanderen).

References

- Ansótegui, C.; Bonet, M. L.; and Levy, J. 2009. Solving (weighted) partial MaxSAT through satisfiability testing. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT)*.
- Bidyuk, B., and Dechter, R. 2006. An anytime scheme for bounding posterior beliefs. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 21, 1095. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
- Dalvi, N. N., and Suciu, D. 2004. Efficient query evaluation on probabilistic databases. In *Proceedings of the 13th International Conference on Very Large Data Bases (VLDB)*.
- Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research* 17(1):229–264.
- Darwiche, A. 2004. New advances in compiling CNF to decomposable negation normal form. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*.

- Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press. Chapter 12.
- Darwiche, A. 2011. SDD: A new canonical representation of propositional knowledge bases. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*.
- De Raedt, L.; Frasconi, P.; Kersting, K.; and Muggleton, S., eds. 2008. *Probabilistic Inductive Logic Programming — Theory and Applications*, volume 4911 of *Lecture Notes in Artificial Intelligence*. Springer.
- De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: a probabilistic Prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*.
- Dechter, R., and Rish, I. 2003. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM (JACM)* 50(2):107–153.
- Fierens, D.; Van den Broeck, G.; Thon, I.; Gutmann, B.; and De Raedt, L. 2011. Inference in probabilistic logic programs using weighted CNFs. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Getoor, L., and Taskar, B., eds. 2007. *An Introduction to Statistical Relational Learning*. MIT Press.
- Gogate, V., and Dechter, R. 2011. SampleSearch: Importance sampling in presence of determinism. *Artificial Intelligence* 175(2):694–729.
- Gogate, V., and Domingos, P. 2012. Approximation by quantization. *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Gomes, C. P.; Sabharwal, A.; and Selman, B. 2006. Model counting: A new strategy for obtaining good bounds. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*.
- Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Springer. 85–103.
- Kimmig, A.; Santos Costa, V.; Rocha, R.; Demoen, B.; and De Raedt, L. 2008. On the efficient execution of ProbLog programs. In *Proceedings of the 24th International Conference on Logic Programming (ICLP)*.
- Mooij, J. M., and Kappen, H. J. 2008. Bounds on marginal probability distributions. In *NIPS*, volume 4, 3.
- Ourfali, O.; Shlomi, T.; Ideker, T.; Rupp, E.; and Sharan, R. 2007. SPINE: a framework for signaling-regulatory pathway inference from cause-effect experiments. *Bioinformatics* 23(13):i359–366.
- Poole, D. 1993. Logic programming, abduction and probability. *New Generation Computing* 11:377–400.
- Poole, D. 1997. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence* 94(1-2):7–56.
- Poon, H., and Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*.
- Renkens, J.; Van den Broeck, G.; and Nijssen, S. 2012. k-Optimal: a novel approximate inference algorithm for ProbLog. *Machine Learning* 89:215–231.
- Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.
- Sang, T.; Bacchus, F.; Beame, P.; Kautz, H. A.; and Pitassi, T. 2004. Combining component caching and clause learning for effective model counting. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT)*.
- Sato, T. 1995. A statistical learning method for logic programs with distribution semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP)*.
- Selman, B., and Kautz, H. 1996. Knowledge compilation and theory approximation. *Journal of the ACM* 43(2):193–224.
- Vennekens, J.; Verbaeten, S.; and Bruynooghe, M. 2004. Logic programs with annotated disjunctions. In *Proceedings of the 20th International Conference on Logic Programming (ICLP)*.
- Wainwright, M. J.; Jaakkola, T. S.; and Willsky, A. S. 2003. Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching. In *Workshop on Artificial Intelligence and Statistics*, volume 21, 97. Society for Artificial Intelligence and Statistics Np.
- Wei, W., and Selman, B. 2005. A new approach to model counting. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT)*.

A Proposal for Statistical Outlier Detection in Relational Structures

Fatemeh Riahi, Oliver Schulte, Qing Li

{oschulte, sriahi, liQing}@sfu.ca

School of Computing Science

Simon Fraser University

Vancouver-Burnaby, Canada

Abstract

This paper extends unsupervised statistical outlier detection to the case of relational data. For nonrelational data, where each individual is characterized by a feature vector, a common approach starts with learning a generative statistical model for the population. The model assigns a likelihood measure for the feature vector that characterizes the individual; the lower the feature vector likelihood, the more anomalous the individual. A difference between relational and nonrelational data is that an individual is characterized not only by a list of attributes, but also by its links and by attributes of the individuals linked to it. We refer to a relational structure that specifies this information for a specific individual as the individual's database. Our proposal is to use the likelihood assigned by a generative model to the individual's database as the anomaly score for the individual; the lower the model likelihood, the more anomalous the individual. As a novel validation method, we compare the model likelihood with metrics of individual success. An empirical evaluation reveals a surprising finding in soccer and movie data: We observe in the data a strong correlation between the likelihood and success metrics.

Introduction

Outlier detection is an important data analysis task in many domains (Hodge and Austin 2004). Statistical approaches to unsupervised outlier detection are based on a generative model of the data. The generative model represents normal behavior. An individual entity is deemed an outlier if the model assigns sufficiently low likelihood to generating its features. We propose a new method for extending statistical outlier detection to the case of relational data. While in this paper we employ Bayes nets as our generative model class, the anomaly detection method can be used with any statistical-relational model (for example, Markov Logic Networks, Probabilistic Relational Models, Bayes Logic Programs etc.) Our approach opens up outlier detection as a promising new application area for statistical-relational models.

A generic Bayes net (BN) structure is learned with data for the entire population. The nodes in the BN represent fea-

tures for links, of multiple types, and attributes of individuals, also of multiple types. *A difference between relational and nonrelational data is that an individual is characterized not only by a list of attributes, but also by its links and by attributes of the individuals linked to it.* We refer to the substructure comprising this information as the individual database. The *model likelihood* is computed by using the model parameter values that assign the highest possible likelihood to the individual database (Vuong 1989). The model likelihood defines an outlier score: the lower the model likelihood, the more anomalous the individual. A low model likelihood indicates that there are edges missing from the generic BN that represent an association or correlation that is present in the individual database.

As an alternative anomaly measure, we also consider a variant where a generic class-level BN model is compared with a specific BN model learned for the individual database. The *model log-likelihood ratio* (LR) is the log-ratio of the individual model likelihood to the generic model likelihood. A high model log-likelihood ratio indicates that the individual BN contains edges that are missing from the generic BN, but represent an association that is present in the individual database. Figure 1 shows a flow chart for computing the model likelihood and likelihood ratio. Note that the likelihood ratio is an inherently relational concept: if an entity is characterized by a flat feature vector, the individual database would contain just a single data point, which would not support learning a individual BN model.

Evaluation We analyze two real-world data sets, from the UK Premier Soccer League and the Internet Movie Database (IMDB). Unsupervised anomaly detection methods are notoriously difficult to evaluate because of the lack of ground truth labellings of anomalous cases (Cansado and Soto 2008; Xu and Shelton 2010). In this paper we take a new approach, that compares likelihood-based anomaly metrics with other meaningful and independent metrics for ranking individuals. Our reference metrics are success rankings of individuals, e.g. League Standing of a team. Success rankings are one of the most interesting features to users, and therefore correlations to success rankings are of special interest. Strong correlations between likelihood metrics and meaningful independent success metrics provide evidence that the likelihood metric is meaningful as well. What we discover in the

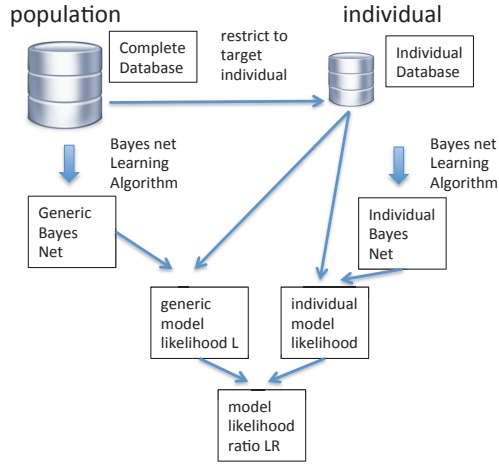


Figure 1: Computation of likelihood metrics for scoring anomalies. **Generic Model Likelihood L** : Learn a generic Bayes net on the complete database. Score the generic Bayes net on the individual database. **Model Likelihood Ratio LR** : Learn an individual Bayes net on the individual database. Compute the ratio of the generic BN/individual BN scores on the individual database.

data are surprisingly strong correlations between typicality and success.

Contributions The generative modelling approach to outlier detection is well established for nonrelational data. The novelty diagram in Figure 2 illustrates conceptual similarities and differences between that previous work and our new proposal. Our main contributions may be summarized as follows.

1. Two new statistical outlier metrics for relational data.
2. A new application of relational learning methods, for outlier detection.

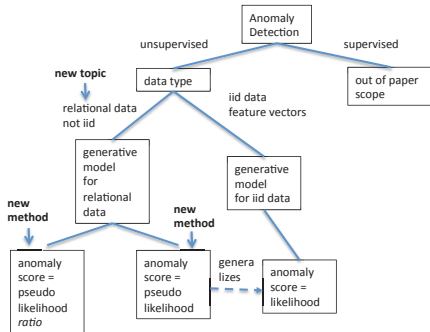


Figure 2: Novelty Diagram for conceptually positioning the new work in comparison with previous work.

Related Work on Outlier Detection

To our knowledge, this is the first work that uses a Bayes net generative model for relational outlier detection. Our work applies two new advances in statistical-relational learning, a recent field that combines AI and machine learning (Getoor and Taskar 2007; Domingos and Lowd 2009). (1) We use a tractable definition of the likelihood function for a Bayes net given a (sub)database that generalizes the standard definition for the nonrelational case (Schulte and Khosravi 2012; Alsanie and Cussens 2012). (2) We apply the learn-and-join algorithm (LAJ), a state-of-the-art Bayes net structure learning method for relational data (Schulte and Khosravi 2012).

Relational Outlier Detection. Many approaches to relational outlier detection are based on discovering rules that represent the presence of anomalous associations for an individual or the absence of normal associations (Maervoet et al. 2012; Gupta et al. 2013). Such rules are informative, but they are not based on a statistical model and do not provide a single aggregate outlier measure for each individual.

The community detection approach uses the relational structure to identify communities (Gao et al. 2010; Sun et al. 2013). Given a (soft) assignment of individuals to communities, outliers can be defined similar to the nonrelational feature vector setting, where normal behavior is measured with respect to a community. Our model also aggregates information from individual features and links of different types, but does not require the identification of communities.

Nonrelational Outlier Detection. Bayesian networks have been successfully applied for unsupervised anomaly detection (Hill, Minsker, and Amir 2007; Xu and Shelton 2010; Babbar and Chawla 2010) with nonrelational i.i.d. data. For nonrelational data, previous evaluation methods include the following. (i) synthetic models with synthetic normal and abnormal data points that provide ground-truth labels for comparison. (ii) Real-world datasets where domain expertise was used to evaluate whether an outlier identification was correct. In future work we will extend our evaluation to include this methodology.

Bayes Nets for Relational Data

We adopt the Parametrized Bayes net (PBN) formalism (Poole 2003) that combines Bayes nets with logical syntax for expressing relational concepts.

Functor Terms A functor is a function or predicate symbol. Each functor has a set of values (constants) called the **domain** of the functor. The domain of a predicate is $\{T, F\}$. A predicate of arity at least two is a **relationship** functor. Relationship functors specify which individuals are linked. Other functors represent **features** or **attributes** of an individual or a tuple of individuals (i.e., of a relationship). A **population** is a set of individuals. A **term** is of the form $f(\sigma_1, \dots, \sigma_k)$ where f is a functor and each σ_i is a first-order variable or a constant denoting an individual. In the context of Bayes nets, we refer to terms as **Parametrized**

Random Variables (PRVs). The grounding concept represents moving from the population-level to the individual level. A **grounding** replaces each first-order variable in a term by a constant; the result is a ground term. A grounding may be applied simultaneously to a set of terms. Relational data can be represented in terms of data tables that show the value of ground terms.

Example The Opta dataset represents information about premier league data (MCFC Analytics 2012). The basic populations are teams, players, matches, with corresponding first-order variables T, P, M . Table 1 specifies values for ground terms.

Table 1: Sample Population Data Table (Soccer). The first three column headers show first-order variables for different populations. The remaining columns represent features.

MatchId M	TeamId T	PlayerId P	First_goal(P, M)	TimePlayed(P, M)	ShotEff(T, M)	result(T, M)
117	WA	McCarthy	0	90	0.53	win
147	WA	McCarthy	0	90	0.35	loss
148	WA	McCarthy	0	85	0.57	loss
15	MC	Silva	1	90	0.59	win
175	MC	Silva	0	90	0.61	win
...	...	...	...	...	...	...

Table 2: Sample Individual Data Table, for team $T = WA$.

MatchId M	TeamId $T = WA$	PlayerId P	First_goal(P, M)	TimePlayed(P, M)	ShotEff(WA, M)	result(WA, M)
117	WA	McCarthy	0	90	0.53	win
147	WA	McCarthy	0	90	0.35	loss
148	WA	McCarthy	0	85	0.57	loss
...	WA	...	...	...	...	...

A novel aspect of our paper is that we learn generative models for specific individuals as well as for the entire population. The appropriate data table for a target individual is formed from the population data table by restricting the relevant first-order variable to the target individual. For example, the individual database for target individual Team *WiganAthletic*, forms a subtable of the data table of Table 1 that contains only rows where TeamID = *WA*; see Table 2.

Bayesian Networks A **Bayesian Network (BN)** is a directed acyclic graph (DAG) whose nodes comprise a set of random variables and conditional probability parameters (Pearl 1988). A **Parametrized Bayes Net (PBN)** is a Bayesian network structure whose nodes are PRVs. For most of the paper we refer to PBNs simply as Bayesian networks, and to PRVs simply as random variables.

The relationships and features in an individual database define a set of nodes for Bayes net learning. Figure 3 shows a sample Bayes net for the entire Team population and another one for Wigan Athletic. We use the following notation.

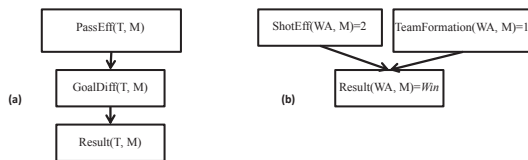


Figure 3: (a) Generic Bayes Net. (b) Individual Bayes net for Wigan Athletic (WA). The nodes in the individual BN are derived from the generic BN by setting $T = WA$.

- $\mathcal{D}_{\mathcal{P}}$ is the database for the entire population; cf. Table 1.
- $\mathcal{D}_t$ is the restriction of the input database to the target individual; cf. Table 2.
- $M_{\mathcal{P}}$ is a model (e.g., Bayesian network) learned with $\mathcal{D}_{\mathcal{P}}$ as the input database; cf. Figure 3(a).
- M_t is an individual model (e.g., Bayesian network) learned with $\mathcal{D}_t$ as the input database; cf. Figure 3(b).

Datasets

In this paper, data tables are prepared from Opta data (MCFC Analytics 2012) and IMDB (Internet Movie Database). Our datasets are available on-line (Khosravi et al. 2014). Table 3 lists the populations and features. Table 4 shows summary statistics for the datasets.

Soccer Data The Opta data were released by Manchester City. It lists all the ball actions within each game by each player, for the 2011-2012 season. For each player in a match, our data set contains eleven player features. For each team in a match, there are five features computed as player feature aggregates, as well as the team formation and the result (win, tie, loss). There are two relationship random variables, *Appears_Player*(P, M), *Appears_Team*(T, M). We store the data in a relational database, with a table for each base population and a table for each relationship.

Table 3: Attribute Features. μ = average, $\sum$ = sum. For relationships please see text.

Individuals	Features
Soccer-Player per Match	<i>TimePlayed, Goals, SavesMade, ShotEff, PassEff, WinningGoal, FirstGoal, PositionID, TackleEff, DribbleEff, ShotsOnTarget</i>
Soccer-Team per Match	<i>Result, TeamFormation, $\sum$ Goals, μShotEff, μPassEff, μTackleEff, μDribbleEff.</i>
IMDB-Actor	<i>Quality, Gender</i>
IMDB-Director	<i>Quality, avgRevenue</i>
IMDB-Movie	<i>year, isEnglish, Genre, Country, RunningTime, Rating by User</i>
IMDB-User	<i>Gender, Occupation.</i>

Table 4: Summary Statistics for the IMDB and Soccer data sets.

Premier League Statistics		IMDB Statistics	
Number Teams	20	Number Movies	3060
Number Players	484	Number Directors	220
Number Matches	380	Number Actors	98690
avg player per match	26.01	avg actor per movie	36.42

IMDB Data The Internet Movie Database (IMDB) is an on-line database of information related to films, television programs, and video games. The IMDB website offers a dataset containing information on cast, crew, titles,

technical details and biographies into a set of compressed text files. We preprocessed the data (Peralta 2007) to obtain a database with seven tables: one for each population and one for the three relationships $Rated(User, Movie)$, $Directs(Director, Movie)$, and $ActsIn(Actor, Movie)$.

Likelihood-Based Anomaly Metrics

In principle our approach can be used with any generative model class, such as Bayesian Networks, Markov Logic Networks, Probabilistic Relational Models, Bayes Logic Programs etc. All that is required is that a generative model define a **relational model log-likelihood function**

$$L(M, \mathcal{D})$$

that assigns a real number to a structure database $\mathcal{D}$ given a model M . A common approach for defining the likelihood function to use a template semantics (knowledge-based model construction) for assigning a likelihood to a relational dataset (Getoor and Taskar 2007). To emphasize the general applicability, we state our definitions abstractly, then discuss how to implement them using Bayesian networks in the next section.

A generative model assigns a likelihood to a single data point. Low likelihood of a data point indicates an anomaly. Defining a likelihood for relational data is more complicated, however, because an individual is characterized not only by a feature vector, but by an individual database that lists the individual’s links and the attributes of linked entities. Our proposal in this paper is to use as an anomaly score the model likelihood assigned to the individual database. The individual database $\mathcal{D}_t$ is defined by instantiating (grounding) the value of a population variable with the target individual, and extracting from the entire population database the facts that satisfy this instantiation; cf. Table 2. Every statistical-relational model defines a log-likelihood function $L(M, \mathcal{D})$ for any structured database, so once the individual database $\mathcal{D}_t$ is constructed, we can apply the log-likelihood function to obtain an anomaly score.

We examine two definitions of an anomaly metric that are based on the relational model likelihood function $L(M, \mathcal{D})$. Both metrics are defined so that a *higher value indicates a more anomalous individual*. The **generic likelihood metric** is the magnitude of the log-likelihood of a model learned for the general population, evaluated on an individual database (see Figure 1). Using the notation above, this quantity is denoted by

$$-L(M_P, \mathcal{D}_t), \quad (1)$$

which we typically abbreviate as $-L$.

The **log-likelihood ratio** LR is the difference between the log-likelihood of the individual BN on the individual database and the log-likelihood of the generic BN on the individual database:

$$L(M_t, \mathcal{D}_t) - L(M_P, \mathcal{D}_t), \quad (2)$$

which we typically abbreviate as LR .

Anomaly Score for Bayesian Networks

We now illustrate how the above recipe can be applied with Bayesian networks, which we utilize in this paper. For the model likelihood function $L(M, \mathcal{D})$, where M denotes a Bayesian network, we employ the random selection pseudo log-likelihood (Schulte 2011). This pseudo log-likelihood can be computed as follows for a given Bayesian network structure and database.

1. For each parent-child configuration, use the maximum-likelihood estimate of the conditional probability of the child given the parent.
2. Multiply the logarithm of this estimate by the instantiation proportion of the parent-child configuration. The instantiation proportion is the number of instantiating groundings, divided by the total number of possible instantiations.
3. Sum this product over all parent-child configurations and all nodes.

Example. In the BN structure of Figure 3, consider the parent-child configuration as shown in Figure 3(b). Suppose that the data show that team *WA* played 20 matches in formation 1 with team shot efficiency at level 2 and won 12 of these. Then the maximum likelihood estimate of winning given the parent values is $12/20 = 0.6$. The instantiation count is 10, so the instantiation proportion is $10/20 = 0.5$. This parent-child configuration contributes $\ln 0.5 \cdot 0.6 = -0.415$ to the aggregate pseudo-likelihood.

The combination of Bayesian network and the pseudo-likelihood has several attractive properties as an anomaly metric. (1) It can be computed in closed form given the sufficient database statistics (the instantiation counts). This is faster than computing conditional likelihoods for each individual ground term. (2) Instantiation counts are standardized to be proportions. This avoids giving exponentially more influence to features with more groundings (Schulte 2011).

Bayes Net Learning. We need to learn a Bayes net structure for the entire population database as well as to each individual database. We apply the learn-and-join algorithm to obtain these Bayes net models. Any structure learning method can be used, the details of the method do not matter here. Table 5 shows the learning times for applying the learn-and-join algorithm to the whole database (generic BN) and the average for the individual databases (individual BN). The learning algorithms were executed with 64-bit CentOS with 4GB RAM and an Intel Core i5-480 M processor. Learning on the soccer dataset is fast. Learning on the IMDB database is feasible at around 50 minutes. It takes the longest because of the large number of individuals (over 98,000 Actors) and fairly complex relationships. The subdatabase for an individual movie is small, so individual learning is fast, and it is possible to learn 3,060 individual models.

Empirical Distribution of Likelihood-Metrics

We show the distribution of the $-L$ and LR values on the soccer and the IMDB datasets. We observe that subtracting

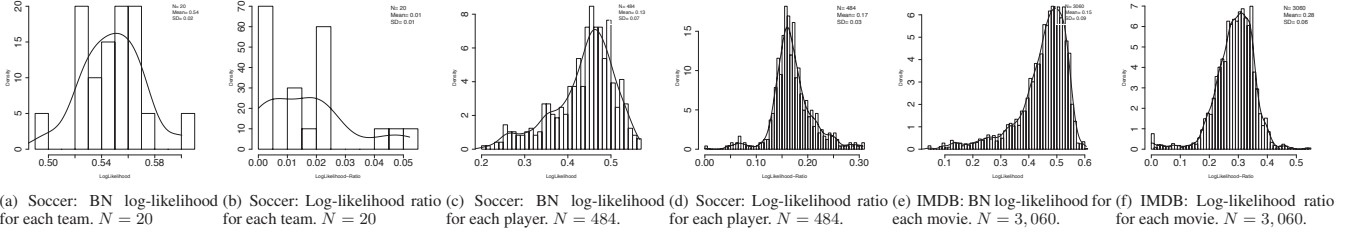


Figure 4: Generic BN model log-likelihood $-L$ (magnitude) vs. individual BN model log-likelihood ratio. The x-axis shows the generic model log-likelihood $-L$ resp. the log-likelihood ratio LR . Higher x-values indicate more anomalous individuals. The y-axis shows densities derived from the counts of x-values divided by total number of individuals. The plots include density estimation curves. N = number of individuals.

Database	Generic BN	Individual BN
Soccer-Team	10 min	0.33 min
Soccer-Player	10 min	0.25 min
IMDB-Movie	50 min	1.00 min

Table 5: Bayes Net structure learning time for different datasets.

the common trend in the population, tends to make individual deviations become more pronounced: The tail of the LR distribution is longer than that of the $-L$ distribution. This is because the log-likelihood ratio cancels out areas of agreement between the generic and the individual BN models.

Figure 4 presents six histograms, two for each type of individual, comparing the generic BN likelihood $-L$ to the LR . Density estimation curves smooth the empirical distribution. The density curves were produced by R’s default density function (the S3 Gaussian kernel estimator). If we think of the anomaly metric as measuring a distance from the generic individual (e.g., the generic team), we observe a pronounced central tendency for that distance. We may interpret this mean as an average degree of atypicality for an individual’s pattern of associations. There is a fairly large tail to the left of the mean that represents individuals who are more typical than most, and a smaller tail to the right that represents individuals who are definite outliers in their class. We next relate the likelihood and likelihood ratio metrics to metrics that identify successful individuals.

Likelihood Metrics vs. Success Metrics

The aim of this section is to compare the LR metric with other meaningful metrics for comparing individuals. Our reference metrics are success rankings of individuals, shown in Table 6. Success rankings are one of the most interesting features to users. Strong correlations between an anomaly metric and meaningful success metrics provide evidence that the anomaly metric is meaningful as well. We measure correlation strength by the standard correlation coefficient ρ . The coefficient ranges from -1 to 1, where 0 means no correlation and 1 or -1 indicate maximum strength (Fisher 1921).

Our main observation is that typicality correlates with success. These correlations are remarkable in at least two respects. (1) The strength of the correlation is surprisingly high: magnitudes range from 0.3 to 0.6 where 1 is the max-

Dataset	Success Metric	Min	Max	Standard Dev.	Mean
IMDB	Rating	1.8	9	1.14	6.3
Soccer-Player	TimePlayed	5.0	3420	1015.69	1484.0
Soccer-Team	Standing	1.0	20	5.91	10.5

Table 6: Success metrics.

	$\rho(\#Parameters, Standing)$	$\rho(LR, Standing)$	$\rho(-L, Standing)$
All Teams	-0.10	0.19	0.52
Top Teams	0.44	0.62	0.45
Bottom Teams	0.33	0.41	0.38

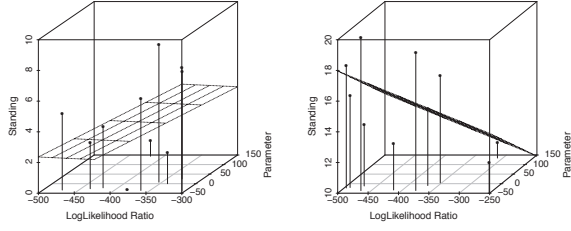
Table 7: Teams: Correlations of team-standing with team-#parameters, team LR ratio and team log-likelihood $-L$

imum strength. (2) The trend holds across two different domains, different types of individuals, and different success metrics. The correlation between typicality and success becomes most apparent when we divide the population into different subgroups or strata.

Team Standing The reference metric is *Standing* in the final Premier League table. The most successful team has *Standing* = 1 and the least successful team has *Standing* = 20. Table 7 shows the correlation between the model likelihood metrics and Team Standing. For teams the number of parameters in the individual BN correlates with *Standing* as well so we include it. If we distinguish Top Teams (*Standing* ≤ 10) from Bottom Teams (*Standing* > 10), a strong positive correlation emerges between $-L$ resp. LR and *Standing*, meaning that *unusual teams tend to have worse (i.e., higher) standing*.

The number of parameters correlates positively with *Standing* for Top Teams and Bottom Teams. We interpret this result as saying that teams with complex correlation patterns tend to perform worse. The plane of best fit is shown in the 3-D scatter plots of Figure 5.

Player Time Played The reference metric is *TimePlayed*, the total time that a player played over all matches in the season. This metric was shown to correlate strongly with other success metrics, such as salary, on MLS soccer data (T.Swartz 2013). Table 8 shows the correlations between the



(a) For bottom half teams. (b) For top half teams. The dependent variable is 10-standing.

Figure 5: Teams: Regression plane of Standing on the LR ratio and #parameters.

	$\rho(\text{Parameter}, \text{TimePlayed})$	$\rho(LR, \text{TimePlayed})$	$\rho(-L, \text{TimePlayed})$
All Players	0.00	0.34	0.20
Goalkeepers	-0.37	0.58	0.60
Defenders	0.09	0.26	0.54
Midfielders	-0.20	0.42	0.63
Strikers	-0.15	0.17	0.60

Table 8: Players: Correlations of TimePlayed, with player-#parameters, player LR ratio and player log-likelihood $-L$.

likelihood metrics and TimePlayed. For model likelihood (magnitude) $-L$, there is a strong positive correlation with TimePlayed. The correlation is even greater for subpopulations defined by the player positions Goalkeepers, Defenders, and Midfielders. For the log-likelihood ratio LR , there is a substantive correlation as well. *Thus unusual players tend to play more minutes.*

Movie Ratings The reference metric is the average user rating of the movie, called *Rating*. Table 9 shows the correlations between the likelihood metrics and Rating. For model likelihood (magnitude) $-L$, there is a positive correlation with Rating. This trend becomes stronger for strata defined by genres; similarly for the LR metric. The same movie may fall into different genres so the strata are not disjoint. *Thus unusual movies tend to receive higher ratings.*

In summary, we observe strong correlations between the likelihood metrics and the success metrics. Among players and movies, less typical individuals tend to be more successful, whereas among teams, more typical individuals tend to be more successful. Thus the direction of the correlation appears to be domain-specific. The fact that success can be pre-

Genre	$\rho(LR, \text{Rating})$	$\rho(L, \text{Rating})$
All Genres	0.28	0.13
Drama	0.28	0.29
Action	0.42	0.29
Sci-Fi	0.35	0.32
Adventure	0.34	0.37
Film-Noir	0.49	0.5

Table 9: Movies: Correlations of Average Rating with movie LR ratio and movie log-likelihood $-L$, by Genre. Correlations with #Parameters are weak (not shown).

dicted from our proposed anomaly metrics is an interesting and potentially useful feature of these metrics. Our argument is that a correlation with a meaningful quantity provided evidence that our proposed metrics are meaningful as well.

Conclusion

We presented a new approach for applying Bayes nets to relational outlier detection. The key idea is to learn a graphical model to represent population-level associations, and compare how well the generic model fits individual-level associations. A variant learns another graphical model to represent individual-level associations, and compare how well each model fits the relational data that characterize the individual, using the model likelihood ratio.

We applied efficient state-of-the-art relational learning methods to construct one generic Bayes net for a given domain and many Bayes nets for individuals (e.g., 3060 models for IMDB movies). The likelihood ratio highlights individual deviations more than the simple model likelihood. The data show a surprisingly strong correlation between our proposed anomaly metrics and success metrics. We believe that closer investigation of the correlation between success and typicality is a fruitful topic for future work.

Instead of estimating parameters for the generic BN from the individual database, an option is to estimate parameters from the population database. The likelihood score then reflects not only individual deviations in qualitative model structure, but also in event frequencies. The difficulty is that individual frequency deviations conflate group membership features with causal connections. For instance, relatively few soccer players are goalies, so a BN with frequency estimates derived from the general population is biased towards viewing goalies as anomalous. Combining individual deviations in terms of qualitative structure, as discussed in this paper, and quantitative frequency into a single meaningful metric is a challenging and fruitful topic for future work.

Overall, our new model likelihood metrics for statistical-relational outlier detection showed promising results on the Soccer and IMDB dataset. They provide a new approach for applying AI techniques to a challenging and practically important topic.

References

- Alsanie, W., and Cussens, J. 2012. Learning recursive prism programs with observed outcomes. In *Proceedings of the ICML-2012 Workshop on Statistical-Relational Learning*.
- Babbar, S., and Chawla, S. 2010. On bayesian network and outlier detection. In Kumar, P. S.; Parthasarathy, S.; and Godbole, S., eds., *COMAD*, 125. Allied Publishers.
- Cansado, A., and Soto, A. 2008. Unsupervised anomaly detection in large databases using bayesian networks. *Appl. Artif. Intell.* 22(4):309–330.
- Domingos, P., and Lowd, D. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool Publishers.
- Fisher, R. A. 1921. On the probable error of a coefficient of correlation deduced from a small sample. *Metron* 1:3–32.

- Gao, J.; Liang, F.; Fan, W.; Wang, C.; Sun, Y.; and Han, J. 2010. On community outliers and their efficient detection in information networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, 813–822. New York, NY, USA: ACM.
- Getoor, L., and Taskar, B. 2007. *Introduction to statistical relational learning*. MIT Press.
- Gupta, M.; Gao, J.; Yan, X.; Cam, H.; and Han, J. 2013. On detecting association-based clique outliers in heterogeneous information networks. In *ASONAM*. ACM.
- Hill, D. J.; Minsker, B. S.; and Amir, E. 2007. Real-time bayesian anomaly detection for environmental sensor data. In *In 32nd Conference of IAHR, the International Association of Hydraulic Engineering & Research*.
- Hodge, V., and Austin, J. 2004. A survey of outlier detection methodologies. *Artif. Intell. Rev.* 22(2):85–126.
- Internet Movie Database. Internet movie database. URL = <http://www.imdb.com/>.
- Khosravi, H.; Man, T.; Hu, J.; Gao, E.; and Schulte, O. 2014. Learn and join algorithm code. URL = <http://www.cs.sfu.ca/~oschulte/jbn/>.
- Maervoet, J.; Vens, C.; Vanden Berghe, G.; Blockeel, H.; and De Causmaecker, P. 2012. Outlier detection in relational data: A case study in geographical information systems. *Expert Syst. Appl.* 39:4718–4728.
- MCFC Analytics. 2012. The premier league dataset. URL = <http://www.mcfc.co.uk/Home/The%20Club/MCFC%20Analytics>.
- Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.
- Peralta, V. 2007. Extraction and Integration of MovieLens and IMDb Data. Technical report, APDM project, Laboratoire PRISM, Université de Versailles, Versailles, France.
- Poole, D. 2003. First-order probabilistic inference. In *IJCAI*, 985–991.
- Schulte, O., and Khosravi, H. 2012. Learning graphical models for relational data via lattice search. *Machine Learning* 88(3):331–368.
- Schulte, O. 2011. A tractable pseudo-likelihood function for Bayes nets applied to relational data. In *SIAM SDM*, 462–473.
- Sun, Y.; Han, J.; Zhao, P.; Yin, Z.; Cheng, H.; and Wu, T. 2013. Community distribution outlier detection in heterogeneous information networks. In *ECML/PKDD (1)*, volume 8188 of *Lecture Notes in Computer Science*, 557–573. Springer.
- T.Swartz, Adriano Arce, M. P. 2013. Assessing value of the draft positions in major league soccer's superdraft. *The Sport Journal*.
- Vuong, Q. H. 1989. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica* 57(2):307–333.
- Xu, J., and Shelton, C. R. 2010. Intrusion detection using continuous time bayesian networks. *J. Artif. Intell. Res. (JAIR)* 39:745–774.

WOLFE: Strength Reduction and Approximate Programming for Probabilistic Programming

Sebastian Riedel

University College
London, UK
s.riedel@cs.ucl.ac.uk

Sameer Singh

University of Washington,
Seattle, USA
sameer@cs.washington.edu

Vivek Srikumar

Stanford University,
Stanford, USA
svivek@cs.stanford.edu

Tim Rocktäschel

University College
London, UK
t.rocktaschel@cs.ucl.ac.uk

Larysa Visengeriyeva

Technische Universität
Berlin, Germany
visenger@gmail.com

Jan Noessner

University of Mannheim,
Germany
jan@informatik.uni-mannheim.de

Abstract

Existing modeling languages lack the expressiveness or efficiency to support many modern and successful machine learning (ML) models such as structured prediction or matrix factorization. We present WOLFE, a probabilistic programming language that enables practitioners to develop such models. Most ML approaches can be formulated in terms of scalar objectives or scoring functions (such as distributions) and a small set of mathematical operations such as maximization and summation. In WOLFE, the user works within a functional host language to declare scalar functions and invoke mathematical operators. The WOLFE compiler then replaces the operators with equivalent, but more efficient (strength reduction) and/or approximate (approximate programming) versions to generate low-level inference or learning code. This approach can yield very concise programs, high expressiveness and efficient execution.

Introduction

Existing probabilistic programming languages face a trade-off between the expressivity of the models they represent and complexity of the modeling process for the user. Towards one end of the complexity spectrum, we have languages that are limited (by design and often intentionally) in the types of machine learning models and approaches they support. For example, Church (Goodman et al. 2008) is a powerful tool for generative models, but discriminatively trained structured prediction models are difficult to realize. Languages such as Markov Logic Networks (Richardson and Domingos 2006) can train discriminative models, but fail to support paradigms such as matrix or tensor factorization. In contrast, libraries such as FACTORIE (McCallum, Schultz, and Singh 2009) can be used to create arbitrarily rich models, but the burden is often on the user to provide efficient algorithms and data structures to operate such models, requiring machine learning expertise. There is a need for an extensible probabilistic programming language that can

express a variety of models and algorithms while still retaining the simplicity and conciseness of existing paradigm-specific, declarative languages.

In this paper, we introduce WOLFE,¹ a functional probabilistic programming language that allows users to define rich models in a concise way (akin to mathematical expressions seen in machine learning papers) which are then compiled into efficient implementations, thus combining ease of use with expressiveness and efficient computation. WOLFE is based on a formulation that is common to many machine learning algorithms and models: real-valued/scalar functions (for example corresponding to density functions, energy functions, discriminative scores, and training objectives) and mathematical operators that operate upon them (most prominently maximization, summation, and integration). The user constructs the relevant scalar functions in a functional host programming language,² and uses mathematical operators such as `argmax` to operate upon them, thus enabling a rich set of concisely defined ML models. The semantics of a WOLFE program is defined by the mathematical operators, and further, each WOLFE program is an executable program in the host language (albeit with brute force default implementations of the operators that are often intractable).

To ensure efficiency, the WOLFE compiler analyzes the user provided scalar function definitions and generates optimized source code in the host language. WOLFE performs two types of optimizations: *strength reduction* to replace brute-force operator implementations (such as exhaustive search) with equivalent but more efficient versions (such as Max-Product in a junction tree); and *approximate programming* where operator calls are replaced by efficient approximations (for example parallelized stochastic optimization). This separation of program semantics and implementation provides a number of crucial benefits: (1) The user's focus is on the mathematical formulation using the host lan-

¹<http://www.wolfe.ml>

²Currently, we use Scala (Odersky, Spoon, and Venners 2008).

guage data structures, without distractions from efficiency or implementation details. (2) It allows inspection of the program structure that can lead to significant optimizations, (3) It enables reuse of existing inference and learning implementations from other probabilistic language implementations, for example we use the FACTORIE optimization package (Passos, Vilnis, and McCallum 2013), (4) By generating source code, WOLFE utilizes further specific optimizations from the host language compiler, and last, (5) The WOLFE interface, consisting of a few mathematical operators, is likely to remain fixed, and thus development will focus on further compiler optimizations, resulting in full backwards-compatibility of user programs with future WOLFE versions.

Wolfe

WOLFE models consist of a definition of the sample space and a scalar function that assigns probabilities (or, more generally, unnormalized scores) to each sample. A WOLFE program additionally consists of a set of WOLFE operators that are applied to these two components of the WOLFE model.

Sample Space

A *sample space* in WOLFE is a collection of objects of some type `T`, which represent the set of possible worlds for the model. In Scala we use objects of type `Iterable[T]` to represent this collection. WOLFE provides helper methods to compose rich sample spaces over primitive objects and algebraic data types (ADTs).³ For example, for the `Sentence` type in listing 1 the function `space` defines the set of all `Sentence` objects as the cross product of all possible word and tag sequences. Notice that this sample space may have word and tag sequences of different lengths. However, using a `filter` on the collection we can constrain sentences to be consistent.

Scalar Function

The second part of a WOLFE model is a scalar function that can be used to define probabilities, densities, loss functions, etc. over the sample space. In Scala, such functions are defined by `T => Double`, allowing use of existing or user-written functions. In listings 1 and 3 the functions denoted by `s` are (parametrized) examples.

Operators

WOLFE provides several operators that take the sample space and the scalar functions as curried arguments, i.e. `(space:Iterable[T])(obj:T=>Dist)`. The `query` operator, similar to that in Church, generates a sample from `space` using the (unnormalized) distribution as defined by `obj`. The `argmax` operator returns an argument in `space` that maximizes `obj`. The `sum` operator applies `obj` to every element in `space` and sums up the results. As we will see in the examples, these operators can be used to represent a variety of inference and learning objectives.

³In Scala ADTs are defined using case classes and sealed traits; they can represent complex structures such as maps and lists

Transformations

Each operator comes with a default brute-force implementation in Scala, for example the `argmax` performs a linear search over the `space` to find the object that maximizes `obj`. However, for most models of interest these default implementations will not be tractable. Instead, WOLFE replaces the implementation of these operators with tailor-made algorithms that leverage structure (by examining the definitions of the sample space and the model). We currently perform two types of transformations (implemented using Scala macros).

Factor Graph: For operators and sample spaces that represent inference, WOLFE represents the operator as inference on a factor graph (Kschischang, Frey, and Loeliger 2001). WOLFE first analyzes the Scala Abstract Syntax Tree (AST) that defines the sample space (`space`) to discover the random variables of the graph. It finds a compositional isomorphism between the sample space and a factor graph in which each value in the space corresponds to one assignment of the variables in the factor graph, and vice versa. Next the AST of the `obj` function is divided into *factors* according to occurrence of either `sum` or `prod` operators (or quantified versions of these). For each factor we find the nodes that affect the value of the factor, using the aforementioned isomorphism. Given this factor graph that represents the `obj` over the `space`, we can run message passing or sampling algorithms that exploit the graph's structure, providing an efficient (but often approximate) implementation of the original operator.

Gradient Optimizer: Many important scalar functions in machine learning are defined over continuous vector spaces, for example learning the parameters of a model. In WOLFE we provide support for efficient `argmax` execution over such functions, provided that we can analytically derive their gradients. For `argmax` calls that involve continuous `space`, WOLFE takes the AST of the `obj` function and applies derivation rules to find a gradient or sub-gradient. To perform the `argmax` operation WOLFE then uses the efficient FACTORIE optimization library to perform gradient-based search such as SGD, LBFGS or Perceptron.

For both of the above optimizations, WOLFE automatically selects the algorithm to use, and chooses the various hyper-parameters, by performing cursory inspection of the objective and the sample space. However, to allow expert users to inform this selection, WOLFE provides a library of Scala *annotations* that can be used in the WOLFE program to select the algorithm, and to specify the various parameters associated with it, for each call to the operators.

Examples

We give a few examples of WOLFE models and sketch how strength reduction and approximate programming can be lead efficient learning and inference. Note that the code snippets here are primarily for illustration of how a variety

```

case class Sentence(words: Seq[String], tags: Seq[String])
//sample space
def space = all(Sentence){seqs(strings) x seqs(tagSet)}
// feature vector for a sentence
def f(s:Sentence):Vector = {
  val n = s.words.size
  sum(0 until n) { i => oneHot(s.words(i)->s.tags(i)) } +
  sum(0 until n-1) { i => oneHot(s.tags(i)->s.tags(i+1)) }
}
// model to score a sentence
def s(w:Vector)(s:Sentence):Double = w dot f(s)

```

Listing 1: **Linear-chain CRF**: `oneHot (key)` is a vector where the component at index `key` is 1 and 0 elsewhere.

```

// MAP inference
def h(w:Vector)(x:Sentence):Sentence =
  argmax(space) (obs(_) == obs(x)) { d => s_w(d) }
// Loss over training data
def loss(data:Seq[Sentence])(w:Vector):Double =
  sum(data) { d => s_w(h_w(d)) - s_w(d) }
// Parameter estimation
def learn(data:Seq[Sentence]) =
  argmin(vectors) { w => loss_data(w) }

```

Listing 2: **Inference and Learning** for the model in Listing 1. `obs (x)` is the sequence of observations (words) in `x`.

of models may be expressed in WOLFE; the details of the syntax are not crucial for understanding, and may, in fact, be subject to change.

Linear Chains Listing 1 shows the definition of a linear chain Part-of-Speech tagger in WOLFE, together with inference and learning code necessary to use the model in Listing 2. The code is equivalent to the following formulation. We first define a feature function over a sentence using one-hot vectors e_i (vector where component at index i is 1, 0 elsewhere) with a rich index set:

$$f(d) = \sum_i^n e_{d_{word_i}, d_{tag_i}} + \sum_i^{n-1} e_{d_{tag_i}, d_{tag_{i+1}}}$$

With a weight vector w this yields a linear model:

$$s_w(d) = \langle w, f(d) \rangle.$$

In listing 2 maximum a-posteriori (MAP) inference and learning are defined in terms of `argmax` and `argmin` operators. For example, the MAP predictor `h` corresponds to

$$h_w(x) = \arg \max_{d: d_x = x} s_w(d)$$

i.e. it finds the highest scoring structure d that agrees with the observed part of the evidence x we condition on. Before execution of the program, WOLFE transforms the linear model into a factor graph isomorphic to the `Sentence` data structure, and to the factorization of the model `s`, during compilation time. Then WOLFE replaces each call to `h` with message passing on this factor graph that computes the exact solution, thus demonstrating strength reduction.

```

case class UserItem(user: User, item: Item, rating: Double)
// model
def s(w:Vector)(u:UserItem) =
  sum(0 until k) { i => w(u.item->i)*w(u.user->i) } +
  sum(u.user.items) { i => w(i->u.item) }
// training loss over observed cells
def loss(data:Seq[UserItem])(w:Vector) =
  sum(data) { d => pow2(d.rating - s_w(d)) }

```

Listing 3: **Matrix Factorization**: A collaborative filtering model including a neighborhood term.

The function `loss` in Listing 2 defines a perceptron style training loss. The function `learn` finds a minimizer of this loss. In this case WOLFE generates code that calculates the gradient of the loss and then performs gradient descent on it, demonstrating approximate programming. To calculate the gradient at any point of optimization, WOLFE performs MAP inference as described above.

Matrix Factorization Listing 3 shows a recommender system that combines a matrix factorization model (first line of `s`) with a neighborhood model (Koren 2008) that captures item-item correlations directly. WOLFE replaces calls to `argmin (loss (data))` with code that calculates the gradient of the loss and runs stochastic gradient descent.

Generative Models: Latent Dirichlet Analysis Listing 4 illustrates a Wolfe program for a topic model using Latent Dirichlet Analysis (Blei, Ng, and Jordan 2003).⁴ Since the underlying model is generative, the model (as defined by `lda`) is easier to write as the probability of the assignment to all the variables, instead of specifying the unnormalized energy functions as in the examples so far. The additional functions used in this example are `dir` for the Dirichlet density, `cat` for the categorical distribution, and `prod` is the product over the arguments (as opposed to the `sum`). WOLFE replaces inference calls code that integrates out the unobserved variables (φ and θ) with an efficient Gibbs sampling implementation.

Related Work

WOLFE shares a number of aspects of different probabilistic programming languages. Church (Goodman et al. 2008), its recently introduced extensions Venture (Mansinghka, Selam, and Perov 2014) and Fun (Borgström et al. 2011) are similar in that they make the separation between user program and efficient implementations via program (and execution) analysis; however they are limited to generative models defined through generative stories, not scalar functions. Infer.NET (Minka et al. 2010) combines program analysis and host compiler optimizations for efficient sampling and variational inference, but is restricted to generative models that are specified using custom data structures. Work on MLNs (Richardson and Domingos 2006) and ProbLog (Raedt, Kimmig, and Toivonen 2007) provide

⁴Note that generative models are not supported in the current version, but will be included in the future versions of WOLFE.

```

case class Token(topic:Int,word:String)
case class Doc(theta:Map[Int,Double],tokens:Seq[Token])
case class World(phi:Seq[Map[String,Double]],docs:Seq[Doc])

// model computes the probability of the input
def lda(w: Vector)(world:World):Double = {
  import world._
  prod(phi) { dir(_,w('beta))} *
  prod(docs) { d =>
    dir(d.theta, w('alpha)) *
    prod(d.tokens) { t =>
      cat(t.topic, d.theta) *
      cat(t.word, phi(t.topic))
    }
  }
}

```

Listing 4: **Topic Model:** The model here computes the probability given values for all the variables. Inference (not shown) computes distribution over topic assignments (`t.topic`) given the documents and the hyperparameters `w` by summing out `phi` and `theta`.

a useful logic-based declarative language that unfortunately cannot concisely express many of the models used in practice. Learning Based Java (Rizzolo and Roth 2010) provides a declarative language for specifying discriminative models, but restricts the expressiveness of the models that can be defined. Although FACTORIE (McCallum, Schultz, and Singh 2009) is able to express a wide variety of models, it often requires the user to additionally define data structures and implementations for efficient inference. WOLFE, on the other hand, allows the user to specify arbitrarily complex models (concisely by using host language syntax) while still providing efficient implementation via strength reduction and approximate programming.

Conclusions

We have presented WOLFE, a rich language for declarative ML modeling. WOLFE uses two ingredients: a functional host language to define real-valued functions in, and a compiler that transforms mathematical operations on these functions into efficient inference and learning code by strength reduction and approximate programming. WOLFE thus uses mathematics as the unifying machine learning language, and aims to make code look similar to the “model” section of papers that use machine learning. This approach enables users to define expressive models declaratively and shields them from implementation details.

Acknowledgments

This work was supported in part by Microsoft Research through its PhD Scholarship Programme, and in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

- Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *the Journal of machine Learning research* 3:993–1022.
- Borgström, J.; Gordon, A. D.; Greenberg, M.; Margetson, J.; and Van Gael, J. 2011. Measure transformer semantics for bayesian machine learning. In *Programming Languages and Systems*. Springer. 77–96.
- Goodman, N. D.; Mansinghka, V. K.; Roy, D.; Bonawitz, K.; and Tenenbaum, J. B. 2008. Church: a language for generative models. In *Uncertainty in Artificial Intelligence*.
- Koren, Y. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD ’08, 426–434. New York, NY, USA: ACM.
- Kschischang, F. R.; Frey, B. J.; and Loeliger, H. A. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions of Information Theory* 47(2):498–519.
- Mansinghka, V.; Selsam, D.; and Perov, Y. 2014. Venture: a higher-order probabilistic programming platform with programmable inference. *ArXiv e-prints*.
- McCallum, A.; Schultz, K.; and Singh, S. 2009. Factorie: Probabilistic programming via imperatively defined factor graphs. In Bengio, Y.; Schuurmans, D.; Lafferty, J.; Williams, C. K. I.; and Culotta, A., eds., *Advances in Neural Information Processing Systems* 22. 1249–1257.
- Minka, T.; Winn, J. M.; Guiver, J. P.; and Knowles, D. A. 2010. Infer.NET 2.4. Microsoft Research Cambridge. <http://research.microsoft.com/infernet>.
- Odersky, M.; Spoon, L.; and Venners, B. 2008. *Programming in Scala*. artima.
- Passos, A.; Vilnis, L.; and McCallum, A. 2013. Optimization and learning in factorie. In *NIPS Workshop on Optimization for Machine Learning (OPT)*.
- Raedt, L. D.; Kimmig, A.; and Toivonen, H. 2007. Problog: A probabilistic prolog and its application in link discovery. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2462–2467.
- Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.
- Rizzolo, N., and Roth, D. 2010. Learning based java for rapid development of nlp systems. In *Language Resources and Evaluation Conference (LREC)*.

Hierarchical Reasoning with Probabilistic Programming

Brian E. Ruttenberg

Charles River Analytics
625 Mt Auburn St.
Cambridge MA 02140
bruttenberg@cra.com

Matthew P. Wilkins

Applied Defense Solutions
10440 Little Patuxent Pkwy #600
Columbia, MD 21044
mwilkins@applieddefense.com

Avi Pfeffer

Charles River Analytics
625 Mt Auburn St.
Cambridge MA 02140
apfeffer@cra.com

Abstract

Hierarchical representations are common in many artificial intelligence tasks, such as classification of satellites in orbit. Representing and reasoning on hierarchies is difficult, however, as they can be large, deep and constantly evolving. Although probabilistic programming provides the flexibility to model many situations, current probabilistic programming languages (PPL) do not adequately support hierarchical reasoning. We present a novel PPL approach to representing and reasoning about hierarchies that utilizes *references*, enabling unambiguous access and referral to hierarchical objects and their properties.

Introduction

Tracking, detecting and identifying resident space objects (RSOs) is an important task for the US government and other space agencies. Any of the estimated 500,000 objects greater than 1 cm orbiting the earth can severely impair or destroy vital satellites upon collision (NASA 2013). Yet positively identifying the type, orbit and function of an RSO (e.g., a communications satellite in low earth orbit) is a difficult task given the limited and uncertain observations available and the extensive number of orbital objects.

One approach to manage the identification of RSOs is to utilize a space object taxonomy to classify RSOs (Früh et al. 2013; Wilkins et al. 2013). Similar to classical Linnean taxonomy in biology, this hierarchical representation of RSOs can organize space objects into different functional or physical categories that increase in specificity at lower levels of the hierarchy. In uncertain conditions (e.g., poor telescope observations), this representation has the advantage that identifying an object's label with low specificity is better than an incorrect identification; labeling a remote sensing satellite as an "earth observation" satellite instead of "remote sensing" is much more useful than a "navigation" label. This hierarchical classification process requires reasoning about the label of an unknown RSO given some specific observable attributes.

Effective hierarchical representation and reasoning on RSOs, and in other domains, raises a number of challenges:

(1) Hierarchies can be large and deep. (2) Hierarchies can be evolving; it should be possible to add labels or attributes to existing hierarchies. (3) In some domains, hierarchies do not form a tree but exhibit convergence. (4) There may be interactions between hierarchies; for example, a satellite hierarchy might interact with a hierarchy of lift vehicles. (5) There may be multiple instances of a hierarchy in the same situation, such as several RSOs clustered in one location. While ad hoc solutions can be provided for many of these challenges, a general-purpose solution that can help create new hierarchical reasoning systems would be beneficial.

Probabilistic programming has the potential to provide such a general-purpose solution, as probabilistic programming languages (PPL) have reduced the time and effort needed to build and use large, complex and highly relational probabilistic models (Goodman 2013). PPLs provide native probabilistic constructs and methods that encode the model complexity in the language, facilitating a more natural model building procedure. Finally, because the probabilistic constructs are interwoven into the language, arbitrary data structures can be incorporated into models, and the object-oriented nature of some PPLs provides a natural means to express hierarchical models.

Unfortunately, current PPLs have limited support for hierarchical generative processes and reasoning about posterior distributions of entities from the hierarchy. A typical encoding requires the definition of a large conditional probability distribution (CPD) for attributes over all labels in the hierarchy. Such a CPD is impractical for large hierarchies, and lacks the modularity needed to add new labels to the hierarchy. This process also cannot easily support domains that contain multiple hierarchies.

In this paper, we demonstrate that with the addition of several novel constructs, the full power, flexibility and richness of PPLs can be applied to hierarchical models. We extend the Figaro PPL (Pfeffer 2012), where hierarchical domains can be naturally encoded as a probabilistic object-oriented class hierarchy. However, there are several technical challenges that must be overcome, such as unambiguously identifying random variables in different inherited classes, allowing different attribute sets for classes, and consistent application of evidence. We overcome these challenges by using Figaro's *reference* system, which allows us to apply evidence to a hierarchical model that is conditioned upon the labels of an en-

tity. We demonstrate that we can build a large RSO hierarchy and effectively reason using the hierarchy for identification of real satellites.

Background

Hierarchical structures can be found in many domains, such as social networks (Gupte et al. 2011). In vision tasks, hierarchical models such as latent Dirichlet allocation (LDA) (Blei, Ng, and Jordan 2003) are frequently used to model the hierarchical generation of features, regions, or objects in an image (Bakhtiari and Bouguila 2010; Fei-Fei and Perona 2005). Once learned, the hierarchy can be used to classify an object into labels with specific semantic meaning, or general labels if such specificity is not possible (Sadovnik and Chen 2012; Marszalek and Schmid 2007). This type of hierarchical reasoning and classification can be found in other domains as well, such as text (Sun and Lim 2001) or audio (Burred and Lerch 2003) classification.

Probabilistic programming has recently developed as a potential solution to the problems associated with the creation of large and complex models. Models created with these languages tend to be highly modular, reusable, and can be reasoned with using a built-in suite of algorithms that work on any model created in the language. These languages take many different forms; some are functional languages (IBAL (Pfeffer 2001), Church (Goodman et al. 2008)), imperative (FACTORIE (McCallum, Schultz, and Singh 2009)), embedded in an existing language (Infer.NET (Microsoft Research 2013), Figaro) or implement their own domain specific language (BLOG (Milch et al. 2007)).

There has, however, been little effort expended so far to support hierarchical reasoning in PPLs. Hierarchical representations are implicitly supported in these languages via traditional conditional dependence. While PPLs provide flexible means to create chains of conditional distributions, the size of these hierarchies can make such a method even impractical for PPLs, and does not address situations where entities that share a common label may contain varied attributes. Work by Kuo et al. (2013) and prior work on hierarchical relational models (Newton and Greiner 2004) has tried to address some of these issues for ontology-based models. These works mostly proposed representations and reasoning on the model when relationships in the ontology are dependent on attributes that may not exist in some entities. While these works presented some solutions to these problems, they do not provide a PPL framework for reasoning about hierarchies.

Figaro Programming Language

Figaro is an open source, object-oriented, functional programming language implemented in Scala. In Figaro, models are Scala objects that define a stochastic or non-stochastic process. The basic unit in Figaro is the `Element[V]` class, parameterized by `V`, the output type of the element. An element is basically a unit that contains some value based on a random process. For example, `Geometric(0.9)` is an `Element[Int]` that produces an integer output according to a geometric process whose

parameter is 0.9. Elements can be passed to other elements as arguments, and a program typically consists of some number of element definitions.

One essential element in Figaro, the `Chain` element, captures sequencing of a probabilistic model by chaining the generation of one element with the generation of a subsequent element that depends on the first element. In Bayesian network terms, a `Chain`'s first argument represents a distribution over the parent of a node, while the second argument is essentially a conditional probability distribution over the `Chain`'s output type given the parent. Using `Chain`, many elements can be nested together, allowing a user to quickly create complex conditional probability distributions.

Hierarchies

We first formally define a probabilistic hierarchy, and then subsequently provide examples of several problems with hierarchical representation in PPLs.

Formal Definition

Let $\mathbb{L} = l_1, \dots, l_n$ be a set of n category labels. A hierarchy $\mathcal{H}$ is a partial ordering of $\mathbb{L}$ that defines the hierarchical structure of the labels, where l_n is the top label in the hierarchy, and $l_i \leq l_j$ means that l_i is a sub-label of l_j . We denote a global set of m attributes $\mathbb{A} = \{a_1, \dots, a_m\}$ for all labels in the hierarchy. Each label l_i , however, contains only a subset, $\mathbb{A}_i \subseteq \mathbb{A}$, of attributes, where $a_{k|i}$ is a random variable that defines a distribution for the k^{th} attribute from $\mathbb{A}$ given label l_i . If a label l_i contains an attribute a_k , all of its sub-labels must contain a_k . However, the support or distribution of a_k in different labels may be different.

We also define a generative process for each label l_j . We denote a random variable g_j for each l_j , where

$$P(g_j = l_i) = \begin{cases} p_i & \text{if } l_i \leq l_j \\ 0 & \text{otherwise} \end{cases}$$

As $l_i \leq l_j$, this means that the generative process for a label can also have non-zero probability of generating itself as a label instead of any of its sub-labels. In other words, this process allows instances of internal labels of a hierarchy, as opposed to just the leaf labels. Since l_n is the top level label in the hierarchy, we can denote g_n as the random variable that samples any label from the hierarchy.

An instance $\mathcal{I} = (y, x_1, \dots, x_k)$ represents a sample from the label generative process and subsequently from the attribute generative process, where $y \sim g_n$, $x_i \sim a_{i|y}$. This definition of a hierarchy is highly flexible and allows for a diverse representations (e.g., we do not enforce that $P(a_{k|j} = x) = \int_{l_i: l_i \leq l_j} P(a_{k|i} = x) P(l_i | l_j)$).

Hierarchies in PPLs

Consider the example hierarchy shown in Fig. 1 that shows a possible hierarchical organization of RSOs. The top label is RSO, which is divided into Geosynchronous Earth Orbit (GEO) and Low Earth Orbit (LEO). GEO is divided into two additional labels for communications and observation satellites. The generative process of the hierarchy is also shown.

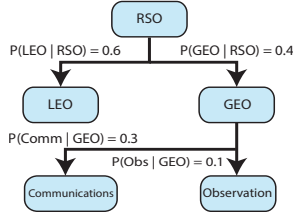


Figure 1: An example RSO classification hierarchy. The probabilities shown are conditioned upon the parent label.

Note that it is implied that a generic GEO label is generated with 0.6 probability (since the sub-labels of GEO do not account for all the probability mass).

It is easy to create a naive version of this hierarchy in a PPL. First, we encode the generative process of the labels as Figaro functions:

```

def rso() = Dist(0.6 -> leo(), 0.4 -> geo())
def geo() = Dist(0.6 -> Constant('geo'),
  0.3 -> communications(), 0.1 -> observation())
def communications() = Constant('communications')
def observation() = Constant('observation')
def leo() = Constant('leo')

```

where `Dist` is a categorical distribution over elements. Note Scala uses type inference, so explicit declaration of types can usually be omitted.

Next, we create a process for the attributes of an RSO, in this case the attitude of the object:

```

def attitude(v: Element[Symbol]) = CPD(v,
  'leo -> Constant('spinStable'),
  'communications -> Constant('nadir'),
  'observation -> Constant('stable'),
  'geo -> Select(0.25 -> 'uncontrolled, 0.75 -> 'stable))

```

`Select` is Figaro's notation for a categorical distribution. Reasoning about an instance from the hierarchy based on observed evidence is quite simple. We create an instance from this hierarchy by generating a label and an attribute distribution:

```

val myRSO: Element[Symbol] = rso()
val myAttitude: Element[Symbol] = attitude(myRSO)

```

where `myRSO` is an element representing generative process of the RSO label. At this point, we can apply evidence to the `myAttitude` element and perform inference using one of Figaro's built-in inference algorithms.

However, this representation is ill suited to represent more complex hierarchies. There is one large CPD with a case for each label. This results in a large and unwieldy CPD, especially with multiple non-independent attributes, where the dependencies between attributes in different labels must all be encoded in the same CPD. Furthermore, with each new label, the CPD must be rewritten, so a user cannot extend the hierarchy without modifying its core. In addition, the CPD of an attribute must define a distribution for every label, even if the label does not contain the attribute.

Clearly, an object-oriented representation with inheritance would be a more modular solution. We could define a set of classes, each of which contains its attributes. An

attribute could be defined for the first time in a class, or it could inherit or override the distribution of its parent. For example,

```

abstract class RSO { val attitude: Element[Symbol] }
class LEO extends RSO {
  val attitude = Constant('spinStable) }

```

creates a class for LEO by extending the RSO class, and we can change the generative process accordingly (i.e., to generate classes instead of symbols). This encoding also poses challenges, however. The problem is that now, instead of a single random variable (RV) representing the attitude of an RSO, all instances of RSO contain attitude RVs, so there are four RVs in this model (one for each possible value of `myRSO`).

It then becomes problematic to try to refer to the attitude of `myRSO`. A naive approach would be to use `myRSO.attitude`, but this is not correct. `myRSO` is an `Element[RSO]`, i.e., a RV. It does not have an attitude attribute; only the *values* of `myRSO` do. In addition, consider if the GEO class also defines a longitude attribute, but the LEO class does not. A system using this hierarchy might observe an RSO at a specific longitude, but this information cannot be asserted on `myRSO` because the RSO class does not have a longitude attribute (only the sub-class GEO does).

Figaro's `Chain` construct provides a possible solution to both these problems. We could write `Chain(myRSO, (v: RSO) => v.attitude)` to refer to the attitude of `myRSO`. However, this solution has the obvious problem that the entire hierarchy has to be hardcoded, losing the modularity benefits we have worked to achieve. In addition, this solution is highly sub-optimal for inference, as the `Chain` couples together the RVs for all the different classes. This has negative implications for both factor and sampling-based inference algorithms.

References

To solve this problem, we use *references*. References are an abstraction to access, modify and reason about Figaro elements in a model. A reference can refer to different RVs depending on the situation, enabling them to overcome the previous challenges.

Element Collections and Names

To enable references, we enforce that every Figaro element must have a name, which is just a string handle for the element. String names do not need to be unique, and when none is provided at element creation, the empty string is used. We also add element collections (EC) to Figaro. ECs are simply a way to organize Figaro elements into coherent groups. Every Figaro element is part of an EC; if an element is not explicitly placed into an EC at creation time, it is placed into the default EC.

To further explain this concept, we modify the RSO hierarchy with names and ECs as shown below:

```

abstract class RSO extends ElementCollection {
  val attitude: Element[Symbol]
}
class LEO extends RSO {

```

```

val attitude =
  Constant('spinStable')('attitude', this)
}
val myRSO = Constant(new LEO)('myRSO', default)

```

The RSO class extends the EC trait. Any elements (attributes) defined inside a RSO class are placed into an EC defined by the newly created class. When we define a LEO class, we also give its attributes a name. When a LEO is instantiated, it will create an element named “attitude” inside a LEO, and this means that the element belongs to the EC of the instantiated LEO. Finally, we create a specific instance of a RSO and give it a name.

Using References

A Figaro reference is a string that, when resolved, refers to the name of an element within some EC. A reference can be resolved into one of more elements using a recursive procedure. For instance, each EC defines a function called `get` that returns the referred element given the current values of all other elements. So, we can write

```

val rsoAttitude: Element[Symbol] =
  default.get('myRSO.attitude')

```

To resolve the reference and assign `rsoAttitude`, the element representing the attitude of the RSO, we first retrieve the element named “myRSO” from the `default` EC. There is only one value of this element (a LEO class), which is an instance of an EC, so we then recursively call `get` with the next part of the reference. On the second call, we retrieve the element named “attitude” in the LEO EC and return it.

In this example, `myRSO` is a constant, so any reference to “myRSO” has only one resolution path. If we use the generative process of RSOs as defined earlier, then `myRSO` could take on a several values, where each possible value is also an EC. Resolving the reference “myRSO.attitude” may refer to multiple elements, each defining their own random process. In this case, the reference itself defines a distribution of values. That is, $P(\text{myRSO.attitude} = x)$ is

$$\int_{y \in \text{myRSO}} P(y.\text{attitude} = x | \text{myRSO} = y) \cdot P(\text{myRSO} = y)$$

References in Hierarchies

References are a natural means to build, access and reason on probabilistic hierarchies because they address many of the issues touched upon in the previous section. First, using ECs and naming, we do not need to define abstract label attributes for all attributes where $l_i \leq l_j$. Meaning, in our example, the RSO class does not need to contain abstract elements that refer to all of the attributes of sub-class of RSO.

This feature is a powerful tool for building hierarchies; adding a new sub-class to the hierarchy need not modify any other class definitions or propagate any information “up” the class hierarchy. Yet accessing the attributes can still be accomplished in a top-down manner using references. That is, the reference to “myRSO.attitude” does not require that the RSO class have any encoded knowledge of the sub-classes where $l_i \leq \text{RSO}$. This allows anyone to extend the hierarchy without changing any other class definitions, greatly

enhancing the modularity and reusability of hierarchies for multiple purposes.

Using references, it is also not required that every class representing a label contains a definition for every attribute. Each class need only to define the attributes that the label actually contains, regardless of the other attributes in the hierarchy. So, for example, we can add a longitude attribute for only GEO objects as

```

class GEO extends RSO {
  val attitude = ...
  val longitude = Uniform(-180, 180)('longitude', this)
}

```

Calling `default.get('myRSO.longitude')` using the declaration of `myRSO` from above will create a distribution of the longitude of RSOs using only the labels that have defined a longitude attribute (i.e., only GEOs). Note that we did not need to modify the LEO or RSO classes to use this reference.

Handling Evidence

Evidence and References

In Figaro, evidence can be applied to elements as constraints or conditions (hard constraints). These are functions from the value of an element to either a double value (constraints) or a Boolean value (conditions). Conditions can also be applied using observation notation, which just asserts that the value of the element must be the observed value.

Figaro allows evidence to be applied to a reference, and ultimately, to the elements resolved from the reference. Evidence can be applied using the `assertEvidence` function in the EC trait, where the user must define the type of evidence to apply to the resolved elements. For example, the statement:

```

default.assertEvidence(
  NamedEvidence('myRSO.attitude', Observation('stable'))
)

```

asserts the observation that any element that is resolved by the reference “myRSO.attitude” should have a value of ‘stable’.

In this situation, however, it is unclear as to how the evidence should be applied to the resolved elements. A naive solution is to apply the evidence to the currently referred element, i.e., the element returned by `get`, but is incorrect. As the Figaro model represents a random process, the current state of the model may change and the element the reference refers to may change. For example, if the current value of `myRSO` is an instance of LEO class, we cannot apply the evidence only to the LEO class attributes, since the value of `myRSO` may change to an instance of GEO, thus changing the current resolution of the reference.

It seems then that the evidence should be applied to all possible elements that could be resolved by the reference. This too, is incorrect. While there are many possible values of `myRSO`, at any one time, the element’s value is a single instance of RSO. Hence it is incorrect to always condition the attributes of other values of `myRSO` on the evidence. To see this, consider the observation above that the

RSO's attitude is 'stable. 'stable is not a possible value for the attitude of a LEO. So if we apply the evidence to `LEO.attitude`, the evidence will have probability zero, even if the current value of `myRSO` is an instance of `GEO`.

Contingent Evidence

To solve this problem, we extend Figaro with contingent evidence. The idea behind contingent evidence is that evidence should be applied to a resolvable element of a reference only when the reference is currently resolved to the element. In other words, contingent evidence applies evidence to an element conditioned upon another set of elements taking on specific values.

Consider the example evidence from above where we observe that the attitude of an instance of RSO is 'stable. In this case, we want to assert that `LEO.attitude == 'stable` only when `myRSO.isInstanceOf[LEO] == true`. In this manner, we only assert the evidence on references contingent upon the values needed to resolve the reference.

When evidence is asserted using a reference, contingent evidence is automatically created and applied by Figaro, without the need for the user to manually specify how to create the contingent evidence (e.g., as above). Creating contingent evidence is a multistep process. First, we have to find all possible resolutions of the reference. Figaro will recursively expand the ECs in the reference and find all possible resolvable elements. Next, for each possible element, Figaro will create a list of contingent elements and the values they need to take to make the reference resolve to the element. The list is stored, along with the evidence, in the resolved element. Finally, for all elements along a resolvable path, Figaro will assert that they are only allowed to take on values that will result in a resolvable element. If no value of an element in the resolution path results in a valid resolution, then assert the element's value as null.

The last step is needed to ensure that elements only take on values that are consistent with the evidence. For example, if we observe that a RSO's longitude is 100, and only `GEO` classes have a longitude, then we must ensure that `myRSO` is never a `LEO`. We cannot apply the evidence of the longitude to a `LEO` (since they do not have one), but the semantics of the evidence state that the RSO cannot be a `LEO`, since the RSO we did observe has a longitude attribute.

During inference on the model, the value of an element is checked against any evidence and the list of contingent elements stored for that evidence. For a condition (Boolean evidence), a value of the element x satisfies the condition if

$$\neg(\bigwedge_{e,v \in Cont} e.value == v) \vee x.value == condition.value$$

where $Cont$ is a list of contingent elements and the values needed to resolve the reference to element x . These semantics enforce that whenever the values of the contingent elements are not equal to the stored values (needed to resolve the reference to this element), the condition on the element is always true. Because of this feature, we can avoid the problem as previously described: Evidence can only be dissatis-

fied when an element's value dissatisfies the evidence *and* the element is the current resolution of a reference.

Using contingent evidence, we can now reason about the label of a RSO given its observed attributes. For example, let us assume that an optical telescope detects a RSO and determines that the RSO's mass is within a certain range, and we wish to determine the label of the unknown RSO. Assuming the generative process and hierarchy previously described, we can accomplish this now:

```
val myRSO = rso("r1")
default.assertEvidence(NamedEvidence("r1.mass",
  Condition((d: Double) => d > 6700 && d < 7300)))
val alg = MetropolisHastings(50000, myRSO)
alg.start()
```

We simply create a new instance of an RSO, assert the condition that the RSO's mass is between 6700 and 7300 kg, and run the Metropolis-Hastings inference algorithm. Once the algorithm is complete, we can query the posterior distribution of the RSO instance.

Experiments

We implemented a large RSO hierarchy to test the utility of references in probabilistic programming. The Figaro code below has been modified for space, but the hierarchy and full code can be found in the supplement.

Hierarchy and Data Set

The RSO hierarchy is based on a recently proposed taxonomy for space objects (Wilkins et al. 2013). At the top level of the taxonomy is the RSO class, followed by classes that define the orbit of the object (`GEO`, `LEO`, `Medium Earth Orbit` and `Ellipse`). Subsequent to the orbit, RSOs are further broken down by the general mission type of the object: `Observation`, `Communications`, `Navigation` or `Debris`. `Observation` RSOs are further broken down into sub-missions (e.g. `Meteorology`). The top level RSO class is defined as

```
abstract class RSO extends ElementCollection
with Missions {
  val attitude: Element[Symbol] = getAttitude(this)
  val mass = FromRange(minMass, maxMass)("mass", this)
  val distance =
    FromRange(minDist, maxDist)("distance", this)
  val brightness: Element[Int]
  val brightnessRate: Element[Double]}
```

The attitude, mass and distance of an RSO are defined in the RSO class, but each sub-class of RSO is free to define their own distributions of these attributes. The attitude attribute is created from the `getAttitude` function, which is defined per the mission of the object (in the `Missions` trait). `FromRange` is Figaro notation for a discrete uniform distribution.

Unfortunately, these attributes are not directly observable. When observing an RSO via optical telescopes, one can only observe the brightness of the object and the rate at which the brightness is observed (e.g., rapid glints or no glints). These attributes are conditioned upon the attitude, mass and distance of the object. For instance, objects that are farther from earth have less intense brightness, and objects that are

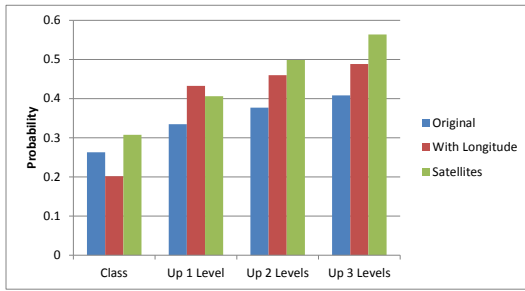


Figure 2: Testing results using the RSO taxonomy

spin stable have more glints than objects that are nadir. Since these attributes depend upon other attributes, they are not defined in the top level class (i.e., each sub-class of RSO must define them).

The RSO hierarchy just provides a taxonomical classification of RSOs. However, to highlight the benefits of using probabilistic programming for hierarchical reasoning, we also added class definitions for specific satellites in orbit. Using the database of 1047 known satellites compiled by the Union of Concerned Scientists (Grimwood 2011), we added class definitions for several satellites by extending existing hierarchy classes. For instance, the GOES13 satellite is defined as

```
class GOES13 extends GEO_Observation_Earth_Meteorology{
  override val mass = Constant(32)("mass", this)
  ...
  val longitude = Constant(-104.77)("longitude", this)
}
```

where the satellite definition overrides the parent class values for mass (and other attributes not shown). Note that this class also defines an attribute `longitude` that was not defined in the RSO class, as fixed longitudes are only properties of GEO satellites.

Testing

Currently, sensor resources are a significant limitation to accurate classification of RSOs. Given the large number of satellites in orbit, sensor resources must be used judiciously to ensure that observed data is valuable. For example, collecting low quality observations of a poorly lit but identified object instead of potentially high quality observations of a brightly lit but unknown object is a waste of resources. Hence, even low-specificity classifications using the RSO taxonomy can be extremely useful if the classification can prevent the misuse of further sensor resources. At present, no other RSO detection and classification scheme uses hierarchical reasoning as a means to triage sensor resources.

As a consequence, we tested the ability of the RSO hierarchy to detect and classify RSOs at different taxonomical classifications. We selected 50 random satellites from the UCS database, generated observation data from each satellite, and applied it as a constraint on the brightness and brightness rate of an unknown RSO. After applying evidence, posterior probabilities of the unknown class label were computed using the Metropolis–Hastings algorithm.

Fig. 2 shows the results of the random testing. First, we computed the expected posterior probability of the correct satellite label, shown in the first column. Then we computed the expected posterior probability of each parent label of the correct satellite label, up to three levels. As expected, the probability of each parent label increases at lower specificity classifications of an unknown RSO. It is clear, however, that this hierarchical reasoning can be combined with a decision rule to classify RSOs and possibly acquire more sensor data. For instance, if an RSO is an instance of an LEO with high probability, then additional sensor measurements can be taken to determine the sub-label of LEO more accurately; or if the probability of sub-labels are nearly equally likely, we can classify the RSO as the parent label instead of a more specific label.

To highlight the modularity and flexibility of using PPLs, we performed two additional tests, also shown in Fig. 2. First, we tested a situation where additional observation capabilities may be available, and if the longitude of an observed RSO can be determined, it is also applied as evidence to the model. Note that only GEO objects have a longitude, but application of this evidence to the appropriate objects is handled automatically. In this scenario, our posterior parent probabilities increased, reflecting the ability of the hierarchical model to reason with additional information when available, while requiring no changes in representation or inference. We also took observations from the 15 satellites encoded into the hierarchy and computed the posterior distributions for those observations. Not surprisingly, the distributions for this test were the highest. As adding new satellites to the hierarchy is extremely easy, this is an effective means to reason with an expanding catalog of satellites.

Conclusion

We presented a novel approach to representing and reasoning on probabilistic hierarchies in PPLs. Figaro references and contingent evidence enable flexible and modular probabilistic hierarchies. Hierarchies are easily expanded with new information, yet still accessed and manipulated without complete knowledge of the hierarchy. We also demonstrated the benefits of our approach for RSO classification, which has the potential to significantly impact the domain.

Due to the benefits of PPLs and references, creation of many powerful models is possible. One can easily create models with multiple RSOs that interact with each other, or create models with interacting hierarchies; the label of an RSO could also depend upon a hierarchy of body types, for example. In addition, the modularity and flexibility of this PPL representation enables easy creation of dynamic models, where we could use the hierarchy to track RSO observations over time. These types of models are targets for future research in probabilistic programming and RSO detection.

Acknowledgements

This work was supported by DARPA contract FA8750-14-C-0011.

References

- Bakhtiari, A. S., and Bouguila, N. 2010. A hierarchical statistical model for object classification. In *IEEE International Workshop on Multimedia Signal Processing*, 493–498. IEEE.
- Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.
- Burred, J. J., and Lerch, A. 2003. A hierarchical approach to automatic musical genre classification. In *Proceedings of the 6th International Conference on Digital Audio Effects*.
- Fei-Fei, L., and Perona, P. 2005. A bayesian hierarchical model for learning natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, 524–531. IEEE.
- Früh, C.; Jah, M.; Valdez, E.; Kervin, P.; and Kelecý, T. 2013. Taxonomy and classification scheme for artificial space objects. In *Advanced Maui Optical and Space Surveillance Technologies Conference*.
- Goodman, N.; Mansinghka, V.; Roy, D.; Bonawitz, K.; and Tenenbaum, J. 2008. Church: a language for generative models with non-parametric memoization and approximate inference. In *Uncertainty in Artificial Intelligence*.
- Goodman, N. D. 2013. The principles and practice of probabilistic programming. In *Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 399–402. ACM.
- Grimwood, T. 2011. Ucs satellite database. *Union of Concerned Scientists* 31.
- Gupte, M.; Shankar, P.; Li, J.; Muthukrishnan, S.; and Iftode, L. 2011. Finding hierarchy in directed online social networks. In *Proceedings of the 20th international conference on world wide web*, 557–566. ACM.
- Kuo, C.-L.; Buchman, D.; Katiyar, A.; and Poole, D. 2013. Probabilistic reasoning with undefined properties in ontologically-based belief networks. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (To appear)*.
- Marszalek, M., and Schmid, C. 2007. Semantic hierarchies for visual object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1–7. IEEE.
- McCallum, A.; Schultz, K.; and Singh, S. 2009. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Advances in Neural Information Processing Systems*, 1249–1257.
- Microsoft Research. 2013. Infer.net api documentation. <http://research.microsoft.com/en-us/um/cambridge/projects/infernet/>.
- Milch, B.; Marthi, B.; Russell, S.; Sontag, D.; Ong, D. L.; and Kolobov, A. 2007. Blog: Probabilistic models with unknown objects. In Getoor, L., and Taskar, B., eds., *Introduction to Statistical Relational Learning*. The MIT press. 373.
- NASA. 2013. Space debris and human spacecraft.
- Newton, J., and Greiner, R. 2004. Hierarchical probabilistic relational models for collaborative filtering. In *Proc. Workshop on Statistical Relational Learning, 21st International Conference on Machine Learning*.
- Pfeffer, A. 2001. IBAL: A probabilistic rational programming language. In *International Joint Conference on Artificial Intelligence*.
- Pfeffer, A. 2012. Creating and manipulating probabilistic programs with Figaro. In *2nd International Workshop on Statistical Relational AI*.
- Sadovnik, A., and Chen, T. 2012. Hierarchical object groups for scene classification. In *IEEE International Conference on Image Processing (ICIP)*, 1881–1884. IEEE.
- Sun, A., and Lim, E.-P. 2001. Hierarchical text classification and evaluation. In *IEEE International Conference on Data Mining*, 521–528. IEEE.
- Wilkins, M. P.; Pfeffer, A.; Schumacher, P. W.; and Jah, M. K. 2013. Towards an artificial space object taxonomy. In *Advanced Maui Optical and Space Surveillance Technologies Conference*.

Scalable Learning for Structure in Markov Logic Networks

Zhengya Sun, Zhuoyu Wei, Jue Wang, Hongwei Hao

Institute of Automation, Chinese Academy of Sciences

{zhengya.sun, zhuoyu.wei, jue.wang, hongwei.hao}@ia.ac.cn

Abstract

Markov Logic Networks (MLNs) provide a unifying framework that incorporates first-order logic and probability. However, learning the structure of MLNs is a computationally hard task due to the large search space and the intractable clause evaluation. In this paper, we propose a random walk-based approach to learn MLN structure in a scalable manner. It uses the interactions existing among the objects to constrain the search space of candidate clauses. Specifically, we obtain representative subset of simple paths by sampling from all sequences of distinct objects. We then transform each sampled path into possible ground atoms, and use them to form clauses. Based on the resulting ground network, we finally attach a set of weights to the clauses by optimizing ℓ_1 -constrained conditional likelihood. The experimental results demonstrate that our approach performs favorably compared to previous approaches.

Introduction

Markov logic networks (MLNs) have gained wide attention in recent years due to their ability to unify the strengths of first-order logic and probabilistic graphical models (Richardson and Domingos 2006). An MLN typically attaches weights to first-order clauses and views these as templates for Markov network features. Learning MLN structure is the process of generating clauses together with their weights (Kok and Domingos 2005), which allows us to capture uncertain dependencies underlying the relational data. Recent advances have revealed its superiority in dealing with real world data, such as biochemistry (Huynh and Mooney 2008), social network and text (Domingos and Lowd 2009).

The structure learning problem, although important, can be computationally prohibitive with the continuous increase in data size. Not only is the search space super-exponentially large, the clause evaluation also produces an overwhelming number of groundings. That is, the exhaustive search and full grounding may fail to finish in a reasonable amount of time. Most efforts are channeled towards constraining the space of candidate structures in a top-down or bottom-up manner.

Top-down approaches (Kok and Domingos 2005; Biba, Ferilli, and Esposito 2008) follow a generate-and-test strategy in which each clause is generated recursively, requiring that the newly added atom shares at least one variable with the existing ones, and then tested for the empirical efficacy. Such approaches are inefficient as they test many potential revisions with no support in the data, and are susceptible to local optima. Bottom-up approaches (Mihalkova and Mooney 2007) overcome these limitations by using the training data to guide their search for clause candidates. The relational pathfinding serves this purpose, which finds paths of true ground atoms, and then generalizes them into first-order clauses. The number of possible paths is often decreased by inducing high level representation (Kok and Domingos 2009; 2010), which still do not scale well to the ever-growing datasets. Meanwhile, evaluating each clause requires producing a grounded Markov network with the variables fully instantiated. This may be infeasible in the presence of numerous objects, which prevents a scalable structure learner.

In this paper, we present *Learning via ground Network Sampling (LNS)*, a novel MLN structure learning approach based on Markov Chain Monte Carlo (MCMC) simulation. The basic idea is to obtain representative groundings of logical statements for clause generation and evaluation. LNS starts by converting the unary atoms (i.e., an atom that involves only one argument) into binary ones, so that the already-present ground atoms can all be treated as interactions between their arguments. The resulting argument pairs that share common objects are then concatenated in order of increasing length, hence forming a partial order graph (POG). Every node in the POG corresponds to a simple path, i.e., a sequence of distinct connected objects, which may describe whether or not the initial and terminal objects interact with each other. Subsequently, LNS performs a random walk on the partial order graph with a prescribed transition probability matrix, and returns every unique simple path it visits. Furthermore, LNS creates the ground clauses from each sampled path by transforming the meaningful interactions into ground atoms. Based on the sampled ground network, LNS finally performs discriminative weight learning with ℓ_1 -norm constraint to encourage sparse solutions.

LNS combines the benefits of random walk and subgraph pattern mining in formulating the problem of structure learn-

ing for MLNs. In the clause generation stage, the approach constrains the search space according to the connections existing among objects. In this way, LNS only constructs clauses that have support in the data, while retaining both true and false instantiated clause samples. In the clause evaluation stage, the approach directly exploits these samples to learn a set of weights, avoiding enumeration of every grounding of the candidate clauses.

The remainder of this paper is organized as follows. In the next section, we begin by providing some background on MLNs and briefly defining some terms we use. Then we present our structure learning approach (Section 3) and report the experimental results (Section 4). Finally, we summarize the conclusion of this work (Section 5).

Markov Logic Networks

This section provides some background on representation, structure and discriminative weight learning for Markov logic networks. Let us briefly review some basic terminology which will be used throughout this paper. Predicates represent relations among objects or attributes of objects. An atom is a predicate symbol applied to a tuple of arguments. A positive literal is an atom, and a negative literal is a negated atom. A ground atom is an atom all of whose arguments are constants. A formula in clausal form, also called a clause, is a disjunction of positive/negative literals. A world is an assignment of truth values to all possible ground atoms.

Representation

Markov logic can be viewed as a probabilistic extension of first-order logic by attaching weights to formulas (clauses). The magnitude of each weight reflects the relative strength or importance of the corresponding clause. The higher weight indicates the greater reward to a world that satisfies the clause, with other things being equal. In the infinite-weight limit, Markov logic becomes equivalent to standard first-order logic.

Definition 1. (Richardson and Domingos 2006) A Markov logic network (MLN) L is a set of pairs (f_i, w_i) , where f_i is a formula in first-order logic and w_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, \dots, c_{|C|}\}$, it defines a Markov network $M_{L,C}$ as follows:

1. $M_{L,C}$ contains one binary node for each possible grounding of each atom appearing in L . The value of the node is 1 if the ground atom is true, and 0 otherwise.
2. $M_{L,C}$ contains one feature for each possible grounding of each formula f_i in L . The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with f_i in L .

More formally, let X be the set of all ground atoms, $\mathcal{F}$ be the set of all first-order clauses in the MLN, w_i be the weight associated with clause $f_i \in \mathcal{F}$, $\mathcal{G}_{f_i}$ be the set of all possible groundings of clause f_i with the constants in the domain. Then, the probability of a possible world x is defined as:

$$\begin{aligned} P_w(X = x) &= \frac{1}{Z} \exp\left(\sum_{f_i \in \mathcal{F}} w_i \sum_{g \in \mathcal{G}_{f_i}} g(x)\right) \\ &= \frac{1}{Z} \exp\left(\sum_{f_i \in \mathcal{F}} w_i n_i(x)\right) \end{aligned} \quad (1)$$

where the subscript w represents a set of clause weights, $g(x)$ equals 1 if g is satisfied and 0 otherwise, $n_i(x)$ denotes the number of true groundings of f_i in x , and $Z = \sum_{x' \in \mathcal{X}} \exp\left(\sum_{f_i \in \mathcal{F}} w_i n_i(x')\right)$ is the normalizing constant.

Structure Learning

The main task in MLN structure learning is to find a set of potentially good clauses. This allows us to represent complex dependencies among atoms, avoiding the assumption of i.i.d. (independent and identically distributed) data made by most statistical learners. Current solutions primarily focus on connected atoms that share common arguments (e.g., variables or constants) and produce all possible clauses consistent with them. In most cases, there is an exponential number of paths that connect the arguments of the given atoms. Thus, approximate search techniques are further required, including scaling down the size of input atoms by clustering (Kok and Domingos 2009; 2010) or restricting the candidate paths to a particular mode (Huynh and Mooney 2011). All candidate clauses are then fully instantiated and evaluated using weighted pseudo-log-likelihood (WPLL). Candidates that do not improve the overall WPLL of the learned structure are discarded.

Discriminative Weight Learning

In discriminative learning, we know a priori which atoms will be evidence and which ones will be queried, and the goal is to correctly predict the latter given the former. Given a set of clauses, weight learners try to maximize the conditional likelihood of query atoms Y given evidence ones X ,

$$P_w(Y = y|X = x) = \frac{1}{Z_x} \exp\left(\sum_{i \in \mathcal{F}_Y} w_i n_i(x, y)\right) \quad (2)$$

where $\mathcal{F}_Y$ denotes the set of all MLN clauses with at least one grounding involving a query atom, $n_i(x, y)$ is the number of true groundings of the i th clause involving query atoms. When non-recursive clauses are considered, the query atoms become independent given the evidence. The conditional log-likelihood (CLL) in this case becomes

$$\begin{aligned} \log P_w(Y = y|X = x) &= \log \prod_{j=1}^n P_w(Y_j = y_j|X = x) \\ &= \sum_{j=1}^n \log P_w(Y_j = y_j|X = x) \end{aligned} \quad (3)$$

and,

$$P_w(Y_j = y_j | X = x) = \frac{\exp(\sum_{i \in \mathcal{F}_{Y_j}} w_i n_i(x, y_{[Y_j=y_j]}))}{\exp(\sum_{i \in \mathcal{F}_{Y_j}} w_i n_i(x, y_{[Y_j=0]})) + \exp(\sum_{i \in \mathcal{F}_{Y_j}} w_i n_i(x, y_{[Y_j=1]}))} \quad (4)$$

where $n_i(x, y_{[Y_j=y_j]})$ is the number of true groundings of the i th clause when all the evidence atoms in X and the query atom Y_j are set to their truth values, and similarly for $n_i(x, y_{[Y_j=0]})$ and $n_i(x, y_{[Y_j=1]})$ when Y_j is set to 0 and 1 respectively. Then the derivative of the CLL with respect to the i th clause is

$$\begin{aligned} \frac{\partial}{\partial w_i} \log P_w(Y = y | X = x) = \\ \sum_{j=1}^n [n_i(x, y_{[Y_j=y_j]}) - P_w(Y_j = 0 | X = x) n_i(x, y_{[Y_j=0]}) \\ - P_w(Y_j = 1 | X = x) n_i(x, y_{[Y_j=1]})] \end{aligned}$$

Notice that the i th component of the gradient is simply the difference between the total count of true groundings of the i th clause and the expected count according to the current model. When the closed world assumption is made, all the counts can be computed exactly without approximate inference, since there is no evidence atom with unknown value.

Learning via Network Sampling

We now describe LNS (Learning via ground Network Sampling), a scalable approach for MLN structure learning. LNS performs a random walk search by starting with each observed ground atom as interactions between its arguments. It then creates a clause to match the sequence of connected objects by labeling predicates. In this paper, we only consider definite clauses containing exactly one positive atom, but the extension to more general case is straightforward.

The four crucial components of LNS, introduced in the next subsections, are: (i) how to construct the state space for random walks, (ii) how to obtain simple path samples, (iii) how to generalize the clauses, and (iv) how the overall approach operates.

State Space of the Random Walk

The partial order graph (POG) of simple paths works as the state space of the Markov Chain on which LNS runs its simulation. We do this by first converting unary atoms (e.g., Student(Bob)) into binary ones augmented by new predicates (e.g., IsA(Bob, Student)). These transformations allow already-present ground atoms to be formulated as interactions between their arguments. Let $\mathcal{I}$ denote the set of all the possible pairwise interactions from the training data. Intuitively, the pairwise interactions that share common objects could be concatenated into paths describing whether or not the initial and terminal objects interact with each other. In our case, we are only interested in simple paths, i.e., a sequence of distinct connected objects, which are also used to restrict every node in the POG. This restriction is actually

reasonable to avoid the propagation of redundant information. The length of a simple path is the number of edges in it. We then construct the partial order graphs locally around the current node. If the current node represents the simple path $p(s, t)$ joining the vertex s to t , its neighbors consist of nodes corresponding to all simple super-paths that expand $p(s, t)$ with one more vertex t' such that (t', s) or $(t, t') \in \mathcal{I}$, and sub-paths that removes the initial vertex s or the terminal one t from $p(s, t)$. The random walker chooses one neighbor (super- or sub- path) according to its transition probability (see below for details). If the current node is a new simple path that was not visited before, LNS returns the corresponding simple path for ground clause generation. The local construction of POG is essential as it circumvents the construction of the entire POG, which would require finding all the simple paths.

Simple Path Sampling

To achieve uniform distribution for all the simple paths, LNS exploits ideas from Metropolis-Hastings (MH) algorithm to perform random walks on the graph (Al Hasan and Zaki 2009). Consider a partial order graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, with a set of nodes $\mathcal{N}$ corresponding to distinct simple paths and a set of edges $\mathcal{E}$ indicating possible extensions of simple paths to larger simple paths. Let $N(u) = \{v \in \mathcal{N} | (u, v) \in \mathcal{E}\}$ denote the set of neighbors adjacent to the node $u \in \mathcal{N}$, and $d(u) = |N(u)|$ denote the degree of the node u . For any arbitrarily given target stationary distribution $\pi = (\pi(u), u \in \mathcal{N})$, the Metropolis-Hastings random walk (MHRW) $\{X_t \in \mathcal{N}, t = 0, 1, \dots\}$ on the graph $\mathcal{G}$ operates as follows. At the current state u of X_t , the next state X_{t+1} is proposed with a proposal probability $Q(u, v) (u \neq v)$. Then, the proposed transition to v is accepted with an acceptance probability $A(u, v)$, that is,

$$A(u, v) = \min \left\{ 1, \frac{\pi(v)Q(v, u)}{\pi(u)Q(u, v)} \right\}, \quad (5)$$

and rejected with probability $1 - A(u, v)$ in which case the state X_{t+1} remains unchanged. Thus, the transition probability $P(u, v)$ becomes, for $u \neq v$,

$$\begin{aligned} P(u, v) &= Q(u, v)A(u, v) \\ &= \min \left\{ Q(u, v), Q(v, u) \frac{\pi(v)}{\pi(u)} \right\}, \end{aligned} \quad (6)$$

with $P(u, u) = 1 - \sum_{u \neq v} P(u, v)$.

For unbiased graph sampling, the target stationary distribution should be set to the uniform distribution $\pi = (1/n, 1/n, \dots, 1/n)$, where $n = |\mathcal{N}|$ represents the number of possible states. Assuming that $N_+(u)$ and $N_-(u)$ denote the set of super- and sub- path neighbors of the node u , with $d_+(u) = |N_+(u)|$ and $d_-(u) = |N_-(u)|$ being the set size correspondingly, LNS then chooses the proposal probabilities for all $(u, v) \in \mathcal{E}$ as follows:

$$Q(u, v) = \begin{cases} \frac{1}{2d_+(u)}, & \text{if } v \in N_+(u) \\ \frac{1}{2d_-(u)}, & \text{if } v \in N_-(u) \end{cases} \quad (7)$$

Observe that any node in the POG as defined earlier has no more than two sub-path neighbors, while with a larger

number of super-path neighbors in most cases. For this, the proposal probability given by (7) can reduce the bias towards the nodes with longer simple paths, and increase the convergence rate of the sampling procedure. The resulting transition probability is thus given as follows:

$$P(u, v) = \begin{cases} \min\{\frac{1}{2d_+(u)}, \frac{1}{2d_-(v)}\}, & \text{if } (u, v) \in \mathcal{E}, v \in N_+(u), \\ \min\{\frac{1}{2d_-(u)}, \frac{1}{2d_+(v)}\}, & \text{if } (u, v) \in \mathcal{E}, v \in N_-(u), \\ 0, & \text{if } (u, v) \notin \mathcal{E}, u \neq v. \end{cases}$$

This ensures that $\mathbf{P} = \{P(u, v)\}_{(u,v) \in \mathcal{E}}$ is reversible with respect to π , i.e., $\pi(u)P(u, v) = \pi(v)P(v, u)$ for any u, v . Since there exists a positive probability to reach from one simple path to another via the $\emptyset$ path, $\mathbf{P}$ is irreducible. In this way, the uniqueness of π is granted due to the finite state space.

Note that when the POG of sampling simple path generators is large, the random walker may get stuck in local regions of the graph. Following the strategy validated by (Li and Zaki 2012), we set a non-zero probability that makes the walker jump back to the root node (empty path) at each step.

Clause Creation and Pruning

LNS constructs the ground clauses from each sampled path by transforming the meaningful interactions into ground atoms. Note the fact that a definite clause is logically equivalent to an implication where the antecedents are a conjunction of positive literals, and the conclusion is a single positive literal (e.g., the clause $\neg R \vee \neg S \vee T$ can be rewritten as the implication $R \wedge S \Rightarrow T$). By analogy, a path here can be interpreted as a statement about whether a conjunction of interactions between two adjacent objects logically implies the interaction between the initial and terminal objects. In addition, as already pointed out, the pairwise interaction corresponds to a set of ground atoms which share the two interacting objects in their arguments. Here we assume that the ground atom contains at least two arguments. It is thus intuitive to convert a path into a set of ground clauses. More specifically, the antecedents are formed as a conjunction of ground atoms that sequentially connect adjacent objects, the ground atom connecting the initial and terminal objects becomes the conclusion.

To control the size of the ground network, LNS imposes restrictions on the search space. On one side, the algorithm starts with the already-present ground atoms, and creates the true antecedents for each clause. Note that any false antecedents alone can make a clause trivially satisfied, and need not to be considered here. On the other side, the algorithm draws conclusions from all the possible atoms with both true and false groundings provided. Also note that the majority of conclusions are false due to the sparseness of relational data (i.e., the great majority of atoms are false), leading to the severe imbalance between the true and false clauses. The algorithm then tackles the imbalance ratio by randomly sampling a predefined number of false clauses. When discriminative learning is concerned, the algorithm variabilizes the resulting ground clauses that have specific query atoms, and discards those without any support in their groundings (i.e., there is at least one true grounding in the

network). In our case, the query atom may appear in the antecedents or the conclusion of a clause. Following Kok and Domingos (2009), the clauses containing variables that appear only once are ignored, since they might not be useful.

Given a set of clauses and a sampled ground network, one can easily find a set of weights that maximizes the CLL of query atoms. More precisely, the query atoms here are the sampled groundings of the target predicate. However, to prevent overfitting and enforce sparsity, LNS performs the optimization subject to an ℓ_1 constraint on the norm of the solution. The final objective function can be written as

$$\min_w \sum_{i=1}^n -\log P_w(Y_i = y_i | X = x), \quad (8)$$

subject to $\|w\|_1 \leq \beta$.

where $\beta > 0$ is a tunable parameter and need to be fixed in advance. In order to solve (8), LNS applies a state-of-the-art ℓ_1 -ball projection algorithm (Duchi et al. 2008) which implements the first-order projected gradient descent update. The clauses with zero weights are finally pruned away.

LNS Overview

The LNS approach proceeds in two phases. The first step, summarized in Algorithm 1, samples a set of simple paths. The relational database $\mathcal{D}$ is converted to the state space for random walks. Then a simple path limited to l is randomly selected. The procedure terminates after KT iterations.

Algorithm 1 Simple Path Sampling($\mathcal{D}, K, l, \gamma$)

Input: $\mathcal{D}$, a relational database

K , the number of initial paths

l , maximal length of each path

γ , restart probability to the empty path

Repeat for $k = 1, 2, \dots, K$

Start with any simple path p_0^k as the current state

Repeat for iteration $t = 1, 2, \dots, T$

· Construct the neighbor paths of p_{t-1}^k , with each path no longer than l

· Propose the next path $p_{cand}^k \sim Q(p_{t-1}^k, p_t^k)$

· Generate $q \sim U(0, 1)$

· Compute the acceptance probability

$$\alpha = \min\left\{1, \frac{Q(p_{cand}^k, p_{t-1}^k)}{Q(p_{t-1}^k, p_{cand}^k)}\right\}$$

· Set the next path p_t^k as

(a) if $q \leq \alpha$, then $p_t^k = p_{cand}^k$

(b) else if $\alpha < q \leq 1 - \gamma$, then $p_t^k = p_{t-1}^k$

(c) else $p_t^k = \emptyset$

Output: Set of simple paths PS

The second step, summarized in Algorithm 2, creates a set of weighted clauses. Under certain constraints, the PS is translated into a set of clauses together with their groundings. The time-complexity in the worst case is $O(mKT)$. LNS finally attaches a set of sparse weights to the clauses based on ℓ_1 -constrained optimization.

Algorithm 2 Clause Creation($\mathcal{D}, PS, m$)

Input: $\mathcal{D}$, a relational database
 PS , a set of simple paths
 m , maximal number of sampled false clauses
 for a certain path
Initialize a set of ground clauses $GS = \emptyset$
Repeat for all simple paths p in PS
 $GS' \leftarrow \text{CreateTrueClauses}(\mathcal{D}, p)$
 if $GS' = \emptyset$, then
 $GS' \leftarrow \text{SampleFalseClauses}(\mathcal{D}, p, m)$
 Remove clauses without query atoms GS'
 Add GS' to GS
 $VS \leftarrow \text{Variabilize}(GS)$
 $CS \leftarrow \text{LearnSparseWeights}(VS, GS)$
Output: Set of weighted clauses CS

Experimental Evaluation

Datasets

We use three publicly available datasets to evaluate the proposed approach. Table 1 shows the statistics of each dataset. The UW-CSE dataset, used by Richardson and Domingos (2006), describes facts about people in an academic department (e.g., Student, Professor) and their relationships (e.g., AdvisedBy, hasPosition). It is divided into five folds based on the areas of computer science. The WebKB dataset, processed by Mihalkova and Mooney (2007), contains information about the webpage categories and links between these pages. It includes four folds, each of which corresponds to one university. The Wordnet dataset, filtered by Richard Socher et al. (2013), provides ontological and lexical knowledge in common domains. It is given in one predefined split into training and test triplets.

Table 1: Statistics on datasets.

Dataset	Predicates	True literals	Total literals
UW-CSE	12	2112	260,254
WebKB	6	2065	688,193
Wordnet	11	133,669	1,488,146,352

Methodology

We performed two sets of experiments. The first experiment compared our approach with two baseline approaches, i.e., LSM-S and LSM-L, which provide the state-of-the-art solutions on relatively large datasets. The suffix '-S' and '-L' indicate the approaches with the short and long limit on clause length. We set the short limit to 5, and the long to 10. In the second experiment, we examined the influence of several sampling parameters on the overall performance of LNS.

The LNS parameter values were: $K = 20000$, $T = 50$, $\gamma = 0.1$, $m = 5$, $\beta = 10$. For LSM, we used the parameter settings provided by Kok and Domingos (2010). We run all the experiments on the identically configured servers (2.0GHz, 1T RAM, 18432KB CPU cache) for a maximum of 24-hour time limit during training.

To evaluate the performance of different structure learners, we implemented Alchemy’s MC-SAT inference in

MLNs over the groundings of each query predicate in the test set, using the groundings of all other predicates as evidence. Note that one MLN is tailored to each prediction task. We then chose the AUC-PR (Area under the Precision-Recall curve), CLL and runtime as the evaluation criteria. Since the CLL can be misleading when the number of negative examples overwhelms the number of positives (which happens in most cases), we kept twice the number of negative examples as positives in each test fold. For the calculation of AUC we used the package of (Davis and Goadrich 2006).

Results

Table 2 and 3 report the AUCs, CLLs and runtimes for different query predicates: advisedBy and tempAdvisedBy for the UW-CSE dataset, courseProf and courseTA for WebKB. The AUC and CLL results are averaged over all atoms in the test sets, while the runtimes are averaged over all the folds.

The tables below show that LNS consistently outperforms LSM in terms of the runtime when learning both short and long clauses. Specifically, the improved efficiency of LNS over LSM is by at least two orders of magnitude, and is independent of the clause length. As can be seen, LNS-S (with short clause limit) and LNS-L (with long clause limit) have similar runtimes on both datasets, with the latter even marginally faster than the former.

In addition to the runtime benefits, LNS also obtains competitive (often the best) AUC and CLL values. Comparing LNS-S to LNS-L, we see that long clauses are effective in lifting both evaluation criteria for most query predicates (with the exception of slightly lower CLL for advisedBy). This is not surprising because long clauses can capture more complex dependencies than short ones. By comparison with the LNS, LSM shows no better performance on all the predicates. For advisedBy, LSM-S has comparable AUC values to our structure learning approaches, while it is significantly worse on all the other predicates.

We examine the influence of the sampling parameters K and m for LNS on Wordnet, a much larger scale dataset. We do not report LSM on this testbed as its runtime exceeds the maximal time limit. Figure 1 plots the curves on AUC and runtime versus logarithmic number of initial paths K . As can be seen, the AUC values increase quickly within a certain interval, and then seem to attain a local optimum. Figure 2 plots the curves on CLL and runtime versus logarithmic number of false clauses m . This shows that the CLL values improve as the maximal number of sampled false clauses increases. The two figures both validate the fact that the time-complexity of LNS is linear related to either K or m .

Conclusion

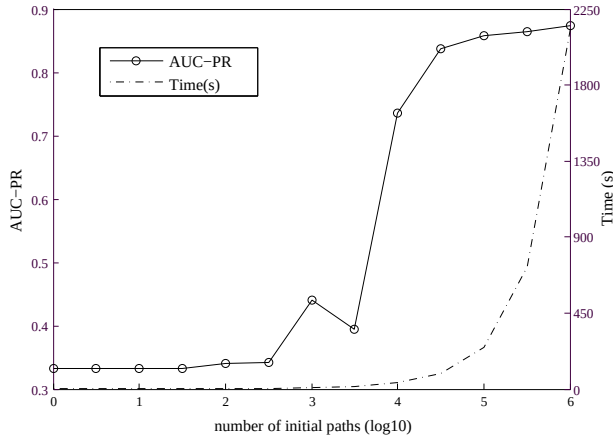
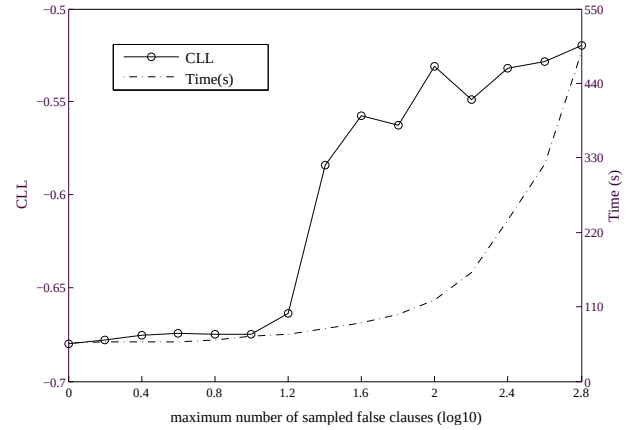
In this paper, we developed a scalable MLN structure learning approach by combining the random walk with sub-graph pattern mining. LNS restricts its search for clauses to within the sampled ground network. LNS then learns a set of sparse weights discriminatively. Our empirical studies on three datasets justified the effectiveness of LNS. Directions for future work include: sampling simple paths discriminatively, integrating more general clause forms, extending the

Table 2: Results on the UW-CSE datasets.

Approach	AUC-PR		CLL		Runtime	
	advisedBy	tempAdvisedBy	advisedBy	tempAdvisedBy	advisedBy	tempAdvisedBy
LNS-S	0.745 $\pm$ 0.159	0.746 $\pm$ 0.164	-0.602 $\pm$ 0.015	-0.665 $\pm$ 0.075	37.960 $\pm$ 0.838 sec	38.960 $\pm$ 0.783 sec
LNS-L	0.752 $\pm$ 0.152	0.749 $\pm$ 0.166	-0.606 $\pm$ 0.015	-0.639 $\pm$ 0.040	35.380 $\pm$ 0.217 sec	33.680 $\pm$ 0.572 sec
LSM-S	0.685 $\pm$ 0.070	0.398 $\pm$ 0.112	-1.590 $\pm$ 0.017	-2.438 $\pm$ 0.509	1.532 $\pm$ 0.461 hr	1.532 $\pm$ 0.461 hr
LSM-L	0.585 $\pm$ 0.196	0.398 $\pm$ 0.112	-1.676 $\pm$ 0.181	-2.438 $\pm$ 0.509	8.540 $\pm$ 0.900 hr	8.540 $\pm$ 0.900 hr

Table 3: Results on the WebKB datasets.

Approach	AUC-PR		CLL		Runtime	
	courseProf	courseTA	courseProf	courseTA	courseProf	courseTA
LNS-S	0.447 $\pm$ 0.070	0.467 $\pm$ 0.219	-0.919 $\pm$ 0.168	-1.675 $\pm$ 1.047	25.525 $\pm$ 0.126 sec	26.450 $\pm$ 0.507 sec
LNS-L	0.506 $\pm$ 0.125	0.497 $\pm$ 0.291	-0.837 $\pm$ 0.133	-0.700 $\pm$ 0.069	22.950 $\pm$ 0.404 sec	23.500 $\pm$ 1.065 sec
LSM-S	0.379 $\pm$ 0.039	0.338 $\pm$ 0.042	-2.315 $\pm$ 0.007	-2.320 $\pm$ 0.020	1.008 $\pm$ 0.145 hr	1.008 $\pm$ 0.145 hr
LSM-L	0.379 $\pm$ 0.039	0.338 $\pm$ 0.042	-2.315 $\pm$ 0.007	-2.320 $\pm$ 0.020	1.008 $\pm$ 0.145 hr	1.008 $\pm$ 0.145 hr

Figure 1: The curves of AUC-PR and runtime versus number of initial paths K on the Wordnet datasetFigure 2: The curves of CLL and runtime versus maximal number of sampled false clauses m on the Wordnet dataset

sampling framework to scalable inference, and further applying LNS to larger domains.

Acknowledgments

The authors thank the anonymous reviewers for their valuable comments. This research work was funded by the National Natural Science Foundation of China under Grant No. 61303179 and No. U1135005.

References

- Al Hasan, M., and Zaki, M. J. 2009. Output space sampling for graph patterns. 2(1):730–741.
- Biba, M.; Ferilli, S.; and Esposito, F. 2008. Discriminative structure learning of markov logic networks. In *Inductive Logic Programming*. Springer. 59–76.
- Davis, J., and Goadrich, M. 2006. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, 233–240. ACM.
- Domingos, P., and Lowd, D. 2009. Markov logic: an interface layer for artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 3(1):1–155.
- Duchi, J.; Shalev-Shwartz, S.; Singer, Y.; and Chandra, T. 2008. Efficient projections onto the ℓ_1 -ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, 272–279. ACM.
- Huynh, T. N., and Mooney, R. J. 2008. Discriminative structure and parameter learning for markov logic networks. In *Proceedings of the 25th International Conference on Machine Learning*, 416–423. ACM.
- Huynh, T. N., and Mooney, R. J. 2011. Online structure learning for markov logic networks. In *Machine Learning and Knowledge Discovery in Databases*. Springer. 81–96.
- Kok, S., and Domingos, P. 2005. Learning the structure of markov logic networks. In *Proceedings of the 22 International Conference on Machine Learning*, 441–448. ACM.
- Kok, S., and Domingos, P. 2009. Learning markov logic network structure via hypergraph lifting. In *Proceedings of*

the 26 *International Conference on Machine Learning*, 505–512. ACM.

Kok, S., and Domingos, P. 2010. Learning markov logic networks using structural motifs. In *Proceedings of the 27 International Conference on Machine Learning*, 551–558.

Li, G., and Zaki, M. J. 2012. Sampling minimal frequent boolean (dnf) patterns. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 87–95. ACM.

Mihalkova, L., and Mooney, R. J. 2007. Bottom-up learning of markov logic network structure. In *Proceedings of the 24th International Conference on Machine Learning*, 625–632. ACM.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.

Evidence-Based Clustering for Scalable Inference in Markov Logic

Deepak Venugopal

Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080, USA
dxv021000@utdallas.edu

Vibhav Gogate

Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080, USA
vgogate@hlt.utdallas.edu

Abstract

Lifted inference algorithms take advantage of symmetries in first-order probabilistic logic representations such as Markov logic networks (MLNs), and are naturally more scalable than propositional inference algorithms which ground the MLN. However, lifted inference algorithms have an “evidence problem” – evidence breaks symmetries, and the performance of lifted inference algorithms is the same as propositional inference algorithms (or sometimes worse, due to overhead). In this paper, we propose a general method for addressing this problem. The main idea in our method is to approximate the given MLN having, say, n objects by an MLN having k objects such that $k \ll n$ and the results obtained by running potentially much faster inference on the smaller MLN are as close as possible to the ones obtained by running inference on the larger MLN. We achieve this by finding clusters of “similar” groundings using standard clustering algorithms (e.g., K-means), and replacing all groundings in the cluster by their cluster center. To this end, we develop a novel distance (or similarity) function for measuring the similarity between two groundings, based on the evidence presented to the MLN. We evaluated our approach on different benchmarks utilizing various clustering and inference algorithms. Our experiments clearly show the generality and scalability of our approach.

Introduction

Markov Logic Networks (MLNs) (Richardson and Domingos 2006; Domingos and Lowd 2009) unify first-order logic and probabilistic models and are arguably the most popular representation for statistical relational learning. They have been used in a wide variety of application domains including NLP (Poon and Domingos 2008), computer vision (Tran and Davis 2008) and planning (Singla and Mooney 2011). As with other probabilistic models, the key challenge in MLNs is to develop scalable inference algorithms. The scalability requirement is even more crucial in MLNs because, even a seemingly simple MLN can yield an arbitrary large (propositional) probabilistic model as the number of objects in the real-world domain increases.

Existing MLN inference algorithms can be broadly classified into two categories, propositional algorithms such as Gibbs Sampling (Geman and Geman 1984) and Belief

Wins(A,A)	0.56			
Wins(A,B)	0.56			
Wins(A,C)	0.56			
Wins(B,A)	0.56			
Wins(B,B)	0.56			
Wins(B,C)	0.56			
Wins(C,A)	0.56			
Wins(C,B)	0.56			
Wins(C,C)	0.56			

(a)

Strong(C)		Wins(A,A)	0.6
Wins(A,C)		Wins(A,B)	0.6
Wins(B,C)		Wins(B,A)	0.63
Wins(B,C)		Wins(C,B)	0.85
Wins(C,A)		Wins(C,C)	0.85

(b)

Wins(A,A)	0.56
Wins(A,B)	0.56
Wins(A,C)	0.56
Wins(B,A)	0.56
Wins(B,B)	0.56
Wins(B,C)	0.56
Wins(C,A)	0.56
Wins(C,B)	0.56
Wins(C,C)	0.56

(c)

Figure 1: Effect of evidence on an MLN with one formula, $1.75 \text{ Strong}(x) \Rightarrow \text{Wins}(x,y)$. The marginal probabilities which were equal in (a) become unequal in (c) due to evidence shown in (b).

Propagation (Yedidia, Freeman, and Weiss 2001; Wainwright, Jaakkola, and Willsky 2003) which operate on the Markov network obtained by grounding the MLN and lifted algorithms (Poole 2003; de Salvo Braz 2007; Gogate and Domingos 2011; van den Broeck et al. 2011; Singla and Domingos 2008; Gogate, Jha, and Venugopal 2012; Venugopal and Gogate 2012; Niepert 2012; Bui, Huynh, and de Salvo Braz 2012; Bui, Huynh, and Riedel 2013), which operate directly on the first-order representation, grounding only as necessary. Propositional algorithms do not scale well because the Markov network representing the MLN is huge for large domain-sizes. On the other hand, all lifted inference algorithms, in principle, can scale significantly better than propositional inference.

Lifted inference algorithms typically suffer from two problems. First, they require MLNs to have a specific symmetric structure (de Salvo Braz 2007; Jha et al. 2010; van den Broeck 2011), which is not always the case in real-world applications. For example, to apply certain inference operations, the MLN needs to be composed of purely singleton atoms (Jha et al. 2010). Second, a far more serious problem is that, in the presence of evidence most MLNs are not liftable because evidence breaks symmetries. As a concrete example, the symmetrical marginal probabilities in Fig. 1 (a) are broken as shown in (c) with evidence (b). Therefore, a lifted algorithm that could potentially exploit the symmetry in (a) can no longer do so in (c) and needs to ground the MLN. To solve this evidence problem, recently, (van den Broeck and Darwiche 2013) suggest approximating the evi-

dence to make the MLN liftable. While this work is specific to binary evidence, our main contribution in this paper is a general approach for scalable inference on any MLN given arbitrary evidence.

Our main idea is to reduce the number of objects in the domain of the MLN, thereby approximating it by a much smaller MLN such that the results obtained by performing inference on the smaller MLN are as close as possible to the ones obtained by running an expensive inference algorithm on the original MLN. For this, we utilize standard clustering algorithms such as K-means to merge together several “similar/related” groundings in an MLN. Importantly, this allows us to plug-in existing grounded/lifted inference algorithms where the sampling-space (for sampling-based inference) or search-space (for search-based inference) can be controlled, which makes inference feasible even on very large domains.

In order to obtain a domain-reduced approximation of the original MLN, we specify a novel distance function that measures similarity based on the evidence presented to the MLN. The distance function helps cluster together objects in the real-world domain that have similar evidence-structure. The inherent symmetry in MLN representation makes it more likely that the marginal probabilities corresponding to objects in a common cluster are approximately close to each other. Thus, we compute the marginal probability for a single element of the cluster and project the same results to all elements in the cluster.

We evaluate our approach on benchmark MLNs available on the Alchemy website (Kok et al. 2008). In our experiments, we leverage clustering algorithms from data-mining/machine learning literature implemented in Weka (Hall et al. 2009) to scale-up to large domain-sizes. We experimented with two inference algorithms, a propositional sampling-based algorithm, Gibbs sampling and a lifted message-passing algorithm, Lifted Belief Propagation, to show the generality of our approach. Our results clearly illustrate the scalability and accuracy of our approach.

Preliminaries

First-order logic (FOL) consists of predicates (e.g. Friends) that represent relations between objects, logical connectives (e.g. $\vee$, $\neg$ etc.) and quantifiers ($\forall$, $\exists$). Each predicate has a parenthesized list of arguments which can be substituted by a term which can either be a logical variable (x), a constant (X) or a function. A formula in first order logic is a predicate (atom), or any complex sentence that can be constructed from atoms using logical connectives and quantifiers. A *ground* atom corresponding to a predicate is one where each term is substituted by a constant symbol. In contrast to full FOL, we make two relaxations here. First, we make the assumption of a *closed universe*, i.e., there is a one-to-one mapping between the constant symbols and objects. This means that any possible *world* is simply an assignment of True or False to every distinct *ground* atom. Secondly, we assume a function-free language where each term is universally quantified and the number of constant symbols is finite. Therefore, for any variable x , we can define a finite set Δ_x (domain of x) which consists of all the constants that can be substituted for x . A ground formula is

a formula obtained by substituting all of its variables with a constant.

Markov logic (Domingos and Lowd 2009) extends FOL by softening the hard constraints expressed by the formulas. A soft formula or a weighted formula is a pair (f, w) where f is a formula in FOL and w is a real-number. A MLN denoted by $\mathcal{M}$, is a set of weighted formulas (f_i, w_i) . Given a set of constants that represent objects in the domain, an MLN defines a Markov network or a log-linear model. The Markov network is obtained by grounding the weighted first-order knowledge base and represents the following probability distribution.

$$P_{\mathcal{M}}(\omega) = \frac{1}{Z(\mathcal{M})} \exp \left(\sum_i w_i N(f_i, \omega) \right) \quad (1)$$

where ω is a world, $N(f_i, \omega)$ is the number of groundings of f_i that evaluate to True in the world ω and $Z(\mathcal{M})$ is a normalization constant or the partition function.

The two main inference problems in MLNs are computing the partition function and the marginal probabilities of query atoms given evidence. In this paper, we focus on the latter.

Domain Clustering

Problem Formulation

We first begin with some notation. Let $\mathcal{M}$ represent an MLN with M predicates $R_1, R_2, \dots, R_M$, and N weighted formulas $f_1, f_2, \dots, f_N$. Let $G_{\mathcal{M}}$ represent the ground KB of the formulas in $\mathcal{M}$. Let $\mathbf{E} = \{E_k\}_{k=1}^S$ be the set of *evidences*. Each $E_k \in \mathbf{E}$ represents a single ground atom that is known to be either True or False. Let $\mathbf{I}$ be a set of indices of the form (i, j) such that $1 \leq i \leq M, 1 \leq j \leq A_i$, where A_i is the arity of the i^{th} predicate. In other words, (i, j) is an index to the j^{th} argument of the i^{th} predicate in $\mathcal{M}$. We now define a partition of $\mathbf{I}$ and its domain as follows.

Definition 1. Let $\mathcal{I} = \{\mathcal{I}_1 \mathcal{I}_2 \dots \mathcal{I}_P\}$ be a partition of $\mathbf{I}$ such that, $\{(i', j'), (i'', j'')\} \subseteq \mathcal{I}_k$ if and only if there exists a formula $f_m \in \mathcal{M}$ where (i', j') and (i'', j'') refer to the same logical variable in f_m . The domain (or groundings) of $\mathcal{I}_k = \{(i_1, j_1), (i_2, j_2), \dots, (i_{k'}, j_{k'})\}$ represented as $\Delta_{\mathcal{I}_k}$ is equal to $\Delta_{(i_1, j_1)} = \Delta_{(i_2, j_2)} = \dots \Delta_{(i_{k'}, j_{k'})}$, where $\Delta_{(i, j)}$ represents the domain of (i, j) .

Example 1. Let $\mathcal{M}$ contain exactly one formula $R_1(x, y) \wedge R_2(y, z) \Rightarrow R_3(z, x)$. Let $\Delta_x = \Delta_y = \Delta_z = \{A, B\}$. $\mathcal{I} = \{\{(1, 1), (3, 2)\}, \{(1, 2), (2, 1)\}, \{(2, 2), (3, 1)\}\}$. $\Delta_{\mathcal{I}_1} = \{A, B\}$ and grounding $\mathcal{I}_1$ with A , yields the partially ground formula, $R_1(A, y) \wedge R_2(y, z) \Rightarrow R_3(z, A)$.

To reduce the total number of formulas in $G_{\mathcal{M}}$, we reduce the number of groundings in each $\mathcal{I}_k \in \mathcal{I}$ independently. Specifically, for each $\Delta_{\mathcal{I}_k}$, we learn a new domain, $\hat{\Delta}_{\mathcal{I}_k}$ and a mapping ζ , where $|\hat{\Delta}_{\mathcal{I}_k}| \leq |\Delta_{\mathcal{I}_k}|$, for each $\mu \in \hat{\Delta}_{\mathcal{I}_k}$, $\zeta(\mu) = \mathbf{C}' \subseteq \Delta_{\mathcal{I}_k}$ and $\forall \mu', \mu'', \mu' \neq \mu'' \Rightarrow \zeta(\mu') \cap \zeta(\mu'') = \emptyset$. We formulate this domain-reduction problem as a standard clustering problem below.

Definition 2. Given a distance measure d between any two groundings of $\mathcal{I}_k \in \mathcal{I}$ and the number of clusters for $\mathcal{I}_k$ equal

to r_k , we define the clustering problem as,

$$\min_{\mathbf{C}_1 \dots \mathbf{C}_P} \sum_{k=1}^P \sum_{j=1}^{r_k} \sum_{C_{kj} \in \mathbf{C}_{kj}} d(C_{kj}, \mu_{kj}) \quad (2)$$

where $\mathbf{C}_{kj}$ corresponds to groundings of $\mathcal{I}_k$ in cluster j , μ_{kj} is the cluster-center of $\mathbf{C}_{kj}$, i.e., it represents the ‘‘average grounding’’ for that cluster and $\zeta(\mu_{kj}) = \mathbf{C}_{kj}$.

The modified (reduced) domain for $\mathcal{I}_k$, $\hat{\Delta}_{\mathcal{I}_k}$ is equal to $\{\mu_{kj}\}_{j=1}^{r_k}$. The cluster-centers in some sense form a compressed representation of the original domain. From this compressed representation, we generate a modified MLN $\hat{\mathcal{M}}$ with a reduced number of predicate groundings. Formally,

Definition 3. For the i^{th} predicate in $\mathcal{M}$, its reduced domain in $\hat{\mathcal{M}}$ is given by,

$$\hat{\Delta}_{R_i} = \bigcup_{j=1}^{r_{i_1}} \mu_{i_1 j} \times \bigcup_{j=1}^{r_{i_2}} \mu_{i_2 j} \dots \bigcup_{j=1}^{r_{i_{A_i}}} \mu_{i_{A_i} j} \quad (3)$$

where $(i, 1) \in \mathcal{I}_{i_1}, (i, 2) \in \mathcal{I}_{i_2} \dots (i, A_i) \in \mathcal{I}_{i_{A_i}}$

The formulation in Eq. (2) allows us to leverage existing clustering algorithms in machine learning/data mining and makes inference feasible even when $G_{\mathcal{M}}$ is very large. This is important because, for arbitrary MLN structures or for inference with evidence, even state-of-the-art inference techniques end up working on a model whose size is comparable to $G_{\mathcal{M}}$ and in most cases, $G_{\mathcal{M}}$ grows rapidly with domain-size. For example, consider the MLN, $R(x, y) \wedge S(y, z) \Rightarrow T(z, x)$, if $\Delta_x = \Delta_y = \Delta_z = 100$, $G_{\mathcal{M}}$ has a million formulas. Further, the search space (for search-based algorithms) or the sampling space (for sampling-based algorithms) is massive, i.e., exponential in the number of ground predicates. By clustering, we are essentially compressing this large space and now any existing inference algorithm can be used to solve large problems as they implicitly work in this reduced space. The key advantage is that this space complexity can now be controlled based on the number of clusters. Specifically,

Proposition 1. The number of ground predicates in $\hat{\mathcal{M}}$ is $O(Mr^A)$, where M is the number of predicates in $\mathcal{M}$, $r = \max_k r_k$ and A is the maximum arity of a predicate in $\mathcal{M}$.

From Eq. (3), it is clear that the domain of a predicate in $\hat{\mathcal{M}}$ is a set of a cluster-centers rather than a set of concrete objects in $\mathcal{M}$. Thus, one grounding of a predicate in $\hat{\mathcal{M}}$ implicitly corresponds to multiple groundings of that predicate in $\mathcal{M}$. This means that in $\hat{\mathcal{M}}$, the original evidence $\mathbf{E}$ is not valid anymore since $\mathbf{E}$ is specified on the original groundings of a predicate. Therefore, we approximate $\mathbf{E}$ with $\hat{\mathbf{E}}$ which specifies the evidence on predicates ground with cluster-centers instead of the original objects in $\mathcal{M}$. To specify this, we define the *expansion* of a predicate grounding in $\hat{\mathcal{M}}$ as the set of groundings that it represents in $\mathcal{M}$.

Definition 4. The *expansion* of the j^{th} grounding of the i^{th} predicate ($R_i(\mu_{i_1 j_1}, \dots, \mu_{i_{A_i} j_{A_i}})$) in $\hat{\mathcal{M}}$ is given by,

$$\pi_{ij} = R_i(\zeta(\mu_{i_1 j_1}) \times \dots \times \zeta(\mu_{i_{A_i} j_{A_i}})) \quad (4)$$

Clearly, if we assert in $\hat{\mathbf{E}}$ that a grounding of a predicate in $\hat{\mathcal{M}}$ is True (or False), this implicitly asserts that every grounding in the expansion of that predicate is True (or False). Given a clustering of the domains, in order to best approximate $\mathbf{E}$ for this clustering, we minimize the symmetric difference between the new evidence and the original evidence as represented below.

$$\min_{\hat{\mathbf{E}}} |\mathbf{E} \Delta \bar{\pi}(\hat{\mathbf{E}})| \quad (5)$$

where $\hat{\mathbf{E}}$ is a subset of predicate groundings in $\hat{\mathcal{M}}$ and each grounding is assigned a sign (positive/True or negative/False), $\bar{\pi}(\hat{\mathbf{E}})$ expands every grounding in $\hat{\mathbf{E}}$ and assigns each grounding in the expansion the same sign as its corresponding grounding in $\hat{\mathbf{E}}$. The Δ operator computes the symmetric difference between $\mathbf{E}$ and $\bar{\pi}(\hat{\mathbf{E}})$ (A grounding with different signs is treated as distinct elements).

We can easily show the following result to construct the optimal $\hat{\mathbf{E}}$ according to Eq. (5).

Proposition 2. Let π_{ij} be the expansion of one grounding ($\hat{E}$) in $\hat{\mathbf{E}}$. Let n_+ be the count of positive-sign elements and n_- , the count of negative-sign elements in $\pi_{ij} \cap \mathbf{E}$. Eq. (5) is optimized by assigning $\hat{E}$ as positive if $n_+ \geq \frac{|\pi_{ij}|}{2}$ and $\hat{E}$ as negative if $n_- \geq \frac{|\pi_{ij}|}{2}$.

Alg. 1 computes the marginal probabilities in an MLN given evidence. It needs three other methods as input namely, the distance function, clustering algorithm and the inference algorithm. The amount of reduction applied to each domain is specified as α . The algorithm starts by computing the partition $\mathcal{I}$ from the term dependencies in $\mathcal{M}$. Next, to each $\mathcal{I}_k \in \mathcal{I}$, the clustering algorithm $\mathcal{L}$ is applied which outputs the clustered domain $\Delta_{\mathcal{I}_k}$ as well as the mapping function ζ that maps each cluster-center to its original domain. $\Delta_{\mathcal{I}_k}$ is now replaced by its approximation in the new MLN $\hat{\mathcal{M}}$. Once all the domains are suitably reduced, the next step is to approximate the evidence based on the reduced domains. Using Proposition 2, for every grounding of every predicate in the approximate MLN $\hat{\mathcal{M}}$, we make a decision as to whether it is to be considered positive evidence, negative evidence or unknown. This yields the approximate evidence set $\hat{\mathbf{E}}$. $\hat{\mathcal{M}}$ is then re-weighted where each grounded formula’s weight is multiplied by the number of groundings it represents in $\mathcal{M}$. We then invoke the inference algorithm $\mathcal{F}$ to compute the marginals in $\hat{\mathcal{M}}$. Finally, we project the results obtained on $\hat{\mathcal{M}}$ back to the original domains. Specifically, if $\hat{\mathcal{M}}$ has a probability p , then each grounding in its expansion is assigned the same probability.

Distance Function

The distance function d is a key parameter that affects the clustering in Eq. (2) and in turn the inference results in Alg. 1. The advantage of our formulation is that it is easy to plug-in a new distance function and generate inference algorithms targeted towards specific applications or datasets. Here, we develop a distance function using the evidence, such that groundings within a cluster have similar marginals.

Algorithm 1: Compute-Marginals

Input: MLN $\mathcal{M}$, Evidence $\mathbf{E}$, Query $\mathbf{Q}$, Distance function d , Clustering function $\mathcal{L}$, Inference algorithm $\mathcal{F}$, cluster-bound α
Output: Marginal probabilities $\mathcal{P}$ for $\mathbf{Q}$

- 1 Compute the partition $\mathcal{I}$ from $\mathcal{M}$
- 2 $\hat{\mathcal{M}} = \mathcal{M}$
- 3 **for** $\mathcal{I}_k \in \mathcal{I}$ **do**
- 4 $numclusters = \alpha \times \Delta_{\mathcal{I}_k}$
- 5 $(\hat{\Delta}_{\mathcal{I}_k}, \zeta) = \mathcal{L}(numclusters, d)$
- 6 Replace $\Delta_{\mathcal{I}_k}$ with $\hat{\Delta}_{\mathcal{I}_k}$ in $\hat{\mathcal{M}}$
- 7 $\hat{\mathbf{E}} = \emptyset$
- 8 **for** $R_i \in \mathcal{M}$ **do**
- 9 Compute $\hat{\Delta}_{R_i}$ using Eqn. (3)
- 10 **for** Each j , where j indexes the groundings of $\hat{\Delta}_{R_i}$ **do**
- 11 Compute its expansion π_{ij} using Eqn. (4)
- 12 Add the j^{th} grounding of R_i to $\hat{\mathbf{E}}$ based on Proposition 2
- 13 $\hat{\mathcal{P}} = \mathcal{F}(\hat{\mathcal{M}}, \hat{\mathbf{E}}, \mathbf{Q})$
- 14 **for** Each $R_k \in \mathbf{Q}$ **do**
- 15 **for** Each j , where j indexes the groundings of R_k in $\hat{\mathcal{M}}$ **do**
- 16 **for** Each t , where t indexes the groundings of R_k in the expansion π_{kj} **do**
- 17 $\mathcal{P}(R_k, t) = \hat{\mathcal{P}}(R_k, j)$
- 18 **return** $\mathcal{P}$

Example 2. Consider the MLN with one formula $R(x) \Rightarrow S(x, y)$ with weight 1.75 and domain $\Delta_x = \{A, B, C\}$. Let the evidence $\mathbf{E} = \{R(A), R(B)\}$. The task is to compute the marginal probabilities of all groundings of $S(x, y)$ which we refer to as the query. The exact marginal probabilities for the query are, $S(A, y) = S(B, y) = 0.5$, $S(C, y) = 0.56$. Thus, an ideal distance function should give us a clustering of Δ_x where B and C are placed in the same cluster as they have the same marginals w.r.t the query variable. To do this, we observe that the evidence on $R(A) \Rightarrow S(A, y)$ and $R(B) \Rightarrow S(B, y)$ are “symmetrical” i.e. they satisfy the same number of groundings and consequently the number of groundings that are left unsatisfied in both the formulas is the same. In other words, when $x = A$, the relevant evidence yields MLN $\mathcal{M}'$ and when $x = B$, it yields $\mathcal{M}''$ and if $\mathcal{M}'$ is sufficiently close to $\mathcal{M}''$, we would want all the groundings where $x = A$ clustered together with the groundings where $x = B$ because they are likely to have the same marginal probabilities. We formalize this intuitive idea below.

Let $\mathcal{M}_{C_{kj}}$ represent the MLN obtained after grounding $\mathcal{I}_k$ with the j^{th} constant in $\Delta_{\mathcal{I}_k}$. Clearly, in the general case, for any two distinct j_1, j_2 , $\mathcal{M}_{C_{kj_1}}$ and $\mathcal{M}_{C_{kj_2}}$ are not necessarily independent MLNs as there may be atoms in $\mathcal{M}_{C_{kj_1}}$ that are also present in $\mathcal{M}_{C_{kj_2}}$. However, in our

Algorithm 2: Build-Query

Input: Clausal formula f_t
Output: Relational-Algebra expression $\mathcal{Q}$

- 1 $\mathcal{Q} = \emptyset$
- 2 **for** $R_i \in f_t$ **do**
- 3 $Rvalue = 1$
- 4 **if** R_i is positive **then**
- 5 $Rvalue = 0$
- 6 **if** $\mathcal{Q} = \emptyset$ **then**
- 7 $\mathcal{Q} = \mathcal{Q} + \sigma_{R_i.val=Rvalue}(R_i)$
- 8 **else**
- 9 $\mathcal{Q} = \mathcal{Q} \bowtie_{\theta} \sigma_{R_i.val=Rvalue}(R_i)$

distance function, we relax the constraints/dependencies between $\mathcal{M}_{C_{kj_1}}$, $\mathcal{M}_{C_{kj_2}}$ and assume these to be independent MLNs and compute the distance between these two MLNs. Specifically, we define a feature vector $\mathbf{U}_{C_{kj}} = c_{f_1}, c_{f_2} \dots c_{f_N}$, where c_{f_k} is the number of satisfied groundings in formula f_k of MLN $\mathcal{M}_{C_{kj}}$ due to the evidence $\mathbf{E}$. The distance is computed as $d(C_{kj_1}, C_{kj_2}) = \|\mathbf{U}_{C_{kj_1}} - \mathbf{U}_{C_{kj_2}}\|$.

Even though the above distance function seems like an intuitive and reasonable heuristic, it turns out that computing the distance function efficiently is infeasible in general because computing the counts in $\mathbf{U}_{C_{kj}}$ is a hard problem. In order to formalize this clearly, we specify the predicates and their groundings using a relational database. Further, we also assume that each formula is reduced to a clausal form. The i^{th} predicate R_i is stored as a table R_i with $A_i + 1$ columns (A_i is the arity), namely, $id_1 \dots id_{A_i}$ and val . The first A_i columns store the constants corresponding to a grounding of the predicate and the val column specifies the assignment. Given such a database, borrowing results from (Domingos and Lowd 2009),

Theorem 1. *Computing the number of satisfied groundings of a first-order clause in a database is #P-complete in the length of the clause.*

Thus, computing $\mathbf{U}_{C_{kj}}$ is hard as it requires queries that join an arbitrary number of tables. Therefore, we further relax the constraints/dependencies within the atoms in a formula to guarantee feasibility of computing $\mathbf{U}_{C_{kj}}$ as illustrated below.

Example 3. Let $\mathcal{M}$ contain one formula, $\neg R(x, y) \vee \neg S(y, z) \vee T(z, x)$, where $\Delta_x = \{A, B, C\}$. To compute the count of satisfied groundings for $x = A$, we compute its inverse, i.e., the number of unsatisfied groundings for $x = A$. The satisfied count is simply the difference between the total number of groundings and the number of unsatisfied groundings. As the total number of groundings $\Delta_y \times \Delta_z$ is a constant for all groundings of x , we simply ignore it. The unsatisfied groundings for $x = A$ is given by,

$$\sigma_{R.val=1 \wedge S.val=1 \wedge T.val=0}((\sigma_{R.id_0=A}(R) \bowtie_{R.id_1=S.id_0} S) \bowtie_{S.id_1=T.id_0 \wedge R.id_0=T.id_1} T) \quad (6)$$

where σ is the selection operator and $\bowtie$ is the join operator. Clearly, the above expression has two joins. However, if we

Algorithm 3: Compute-Features

Input: $\mathcal{M}$ and its associated relational DB, join-bound $J, \mathcal{I}_k \in \mathcal{I}$

Output: Feature vector set $\{\mathbf{U}_{C_{kj}}\}_{j=1}^{\Delta_{\mathcal{I}_k}}$

```

1  $\mathbf{U} = \emptyset$ 
2 for  $C_{kj} \in \Delta_{\mathcal{I}_k}$  do
3    $\mathbf{U}_{C_{kj}} = \emptyset$ 
4   for  $f_t \in \mathbf{F}$  do
5     if  $f_t$  is not relevant to  $\mathcal{I}_k$  then
6       continue
7      $Q = \text{Build-Query}(f_t)$ 
8     while  $Q$  do
9        $Q' = \text{Select a sub-query containing up to the}$ 
10         $\text{first } J \text{ joins in } Q$ 
11       for  $R_i \in Q'$  do
12         if  $\exists p$  such that  $(i, p) \in \mathcal{I}_k$  then
13           Wrap a select  $(\sigma_{R_i.id_p=C_{kj}})$  around  $R_i$ 
14          $\mathbf{U}_{C_{kj}}.\text{append}(\text{Count}(Q'))$ 
15         Let  $R_s$  be a table in  $Q'$  whose attribute
16         participates in the  $\theta$ -join after  $Q'$ 
17         if  $R_s = \emptyset$  then
18            $Q = (Q - Q')$ 
19         else
20           Relax the  $\theta$ -join and include only those
21           constraints involving  $R_s$ 
22            $Q = R_s \bowtie_{\theta} (Q - Q')$ 
23    $\mathbf{U}.\text{append}(\mathbf{U}_{C_{kj}})$ 
24 return  $\mathbf{U}$ 

```

impose a constraint that no joins are allowed during the computation of the feature vector, we approximate Eq. (6) by implicitly assuming that each predicate in the formula is independent i.e. we ignore the joins to obtain a vector of counts from 3 separate queries, $\sigma_{R.val=1 \wedge R.id_0=A}(R)$, $\sigma_{S.val=1}(S)$ and $\sigma_{T.val=0 \wedge T.id_1=A}(T)$. An alternate distance function can be obtained if we only allow exactly one join in a query. In this case, we can get a better approximation of Eq. (6) by considering two pairwise joins in the formula as,

$$\sigma_{R.id_0=A \wedge R.val=1}(R) \bowtie_{R.id_1=S.id_0} \sigma_{S.val=1}(S)$$

$$\sigma_{S.val=1}(S) \bowtie_{S.id_1=T.id_0} \sigma_{T.val=0 \wedge T.id_1=A}(T)$$

Alg. 3 generalizes the idea in the above example and computes the feature vectors for a specific $\mathcal{I}_k \in \mathcal{I}$. The algorithm generates multiple queries corresponding to each grounding of $\mathcal{I}_k$ such that the number of joins in each query is lesser than J . For this, we go over each formula f_t , and first check if f_t is *relevant* to $\mathcal{I}_k$ i.e. if f_t contains at least one predicate R_i such that for some p , $(i, p) \in \mathcal{I}_k$, then f_t is a relevant formula for clustering $\mathcal{I}_k$, otherwise, we ignore f_t . This is because, if f_t is not relevant to $\mathcal{I}_k$, then f_t yields identical counts for every grounding of x and therefore never affects the clustering. For every relevant f_t , we first build

the complete query which is a sequence of θ -joins on the tables corresponding to every predicate in f_t . Each predicate is wrapped with a “Select” to obtain tuples whose value is equal to the inverse of the predicate’s sign in f_t . The θ in the join specifies equality conditions on all the variables shared between the predicates in f_t . For e.g. In a formula $\neg R(x) \vee S(x)$, the θ -join is specified as $\sigma_{R.val=1}(R) \bowtie_{R.id_0=S.id_0} \sigma_{S.val=0}(S)$. Once we build the full query, we simply walk through the query and execute no more than J joins at a time. For each predicate which has a variable that corresponds to some element of $\mathcal{I}_k$, we ground the variable by enforcing the select condition in line 12 of the algorithm. We execute the partial query Q' with a maximum of J joins and store the result (count) in the feature vector. Next, we remove Q' from Q and relax the next θ -join condition as follows. Among all the tables mentioned in Q' , we select one table R_s , that participates in the next join operation in $Q - Q'$. We only retain the join conditions related to R_s in the next join in $Q - Q'$ and remove the rest of the conditions. We continue until we empty the original query Q . Finally, we return the accumulated set of feature vectors.

Experiments

We evaluate our approach on 3 benchmark MLNs available in the Alchemy website (Kok et al. 2008), namely Segmentation (Seg), WebKB, Protein and a new MLN, Student ($\text{Teaches}(i, c) \wedge \text{Prereq}(c, cl) \Rightarrow \text{Takes}(s, cl)$). In the inference subroutine, we used two algorithms from Alchemy, a lifted algorithm based on message passing, Lifted-BP (Singla and Domingos 2008) and a propositional algorithm based on sampling, Gibbs sampling (Geman and Geman 1984). For clustering, we used four algorithms available in Weka (Hall et al. 2009) namely, KMeans++ (KM), Expectation-Maximization (EM), Hierarchical clustering (HC) and XMeans (XM). We set the bound on the number of joins (J) as 1. We ran our experiments on a quad-core Ubuntu machine with 6GB RAM.

Approximation results on benchmarks

Fig. 2 illustrates the approximation error on each benchmark for all combinations of the inference and clustering algorithms. The x -axis plots $CR = \frac{N_c}{N_g}$, where N_c is the total number of ground formulas in $\hat{\mathcal{M}}$ and N_g is the total number of ground formulas in $\mathcal{M}$. The y -axis shows the approximation error calculated as follows. $Err = \frac{\sum_{g \in G} D_{KL}(P_g || P'_g)}{|G|}$, where D_{KL} is the standard KL-Divergence distance measure, G refers to all groundings of a query predicate, P_g is the marginal distribution of g computed from $\mathcal{M}$ and P'_g is computed from $\hat{\mathcal{M}}$. For fairness, both marginals are computed using the same inference algorithm. We set 50% of arbitrary groundings as evidence, where 25% are True and 25% are False. Fig. 2 illustrates the trade-off between accuracy and complexity. On all experiments, as CR increases, the complexity increases, however, the approximation error reduces as we map the original domains to a larger set thereby reducing the difference between the original MLN and the approximate MLN. In almost all cases

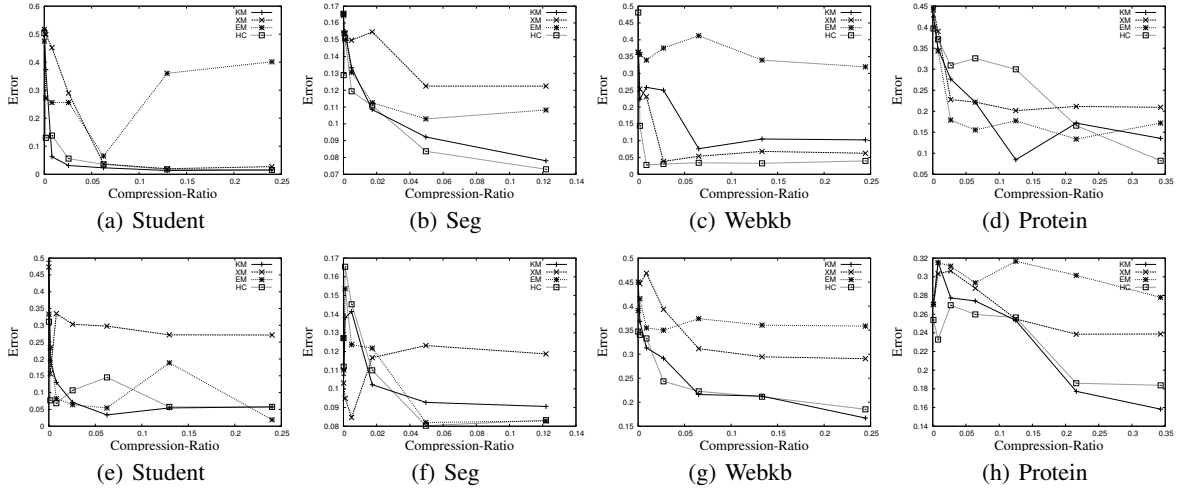


Figure 2: Approximation error for varying $CR = \frac{N_C}{N_G}$, where N_C is the number of ground formulas $\hat{\mathcal{M}}$ and N_G the number of ground formulas in $\mathcal{M}$. The y-axis shows the average KL-Divergence of the marginals computed from $\hat{\mathcal{M}}$ to those computed using $\mathcal{M}$ (smaller is better). (a) - (d) show the results using Lifted-BP, (e) - (h) using Gibbs sampling.

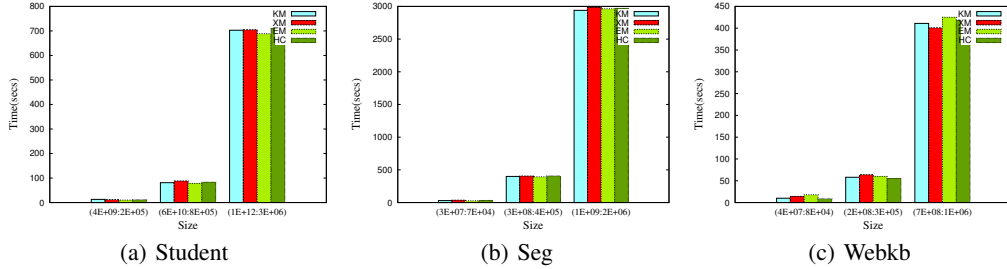


Figure 3: Scalability experiments. The y-axis shows the time in seconds to create the approximate MLN and the x-axis shows $(N_f : N_a)$, where N_f is the number of ground formulas and N_a is the number of ground predicates.

for Lifted-BP, the approximation error was below 0.2 for even small values of CR . For Gibbs sampling though, in some cases (g,h), it took a larger CR value for the approximation error to reduce significantly. One of the reasons for this could be that Gibbs sampling in general performs poorly in MLNs when there are hard constraints (evidence) in the model (Poon and Domingos 2006) and therefore yields worse approximation.

Scalability

Fig. 3 illustrates the scalability of our approach when handling large domain-sizes. For different domain-sizes, we show the time in seconds it takes to compute the approximate MLN after clustering. We used an α value of 0.25 for these experiments and introduced 50% random evidence with half of them True and the other half False. As expected, the time taken to compute the approximate MLN increases as the domain-size grows. However, we were able to complete processing the MLN in a reasonable amount of time even when the ground MLN had a trillion formulas (a). Also, the number of first-order formulas play a role in determining the complexity due to the distance function com-

putation. Recall that we compute a vector for every formula in the MLN. Therefore, a larger number of formulas mean more computations on the database. For instance, Fig. 3 (a) has just one formula while (b) has 7. Therefore, even though the number of ground formulas in (a) is a trillion while in (b) it is a billion, (b) takes longer to process. Further, for each MLN, the third (largest) instance takes a visibly longer time. This is expected because, the size of the database grows really large and query-processing requires more hard disk accesses. Finally, the type of clustering has minimal impact on the processing time.

Acknowledgements This research was partly funded by ARO MURI grant W911NF-08-1-0242, by the AFRL under contract number FA8750-14-C-0021 and by the DARPA Probabilistic Programming for Advanced Machine Learning Program under AFRL prime contract number FA8750-14-C-0005. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, AFRL, ARO or the US government.

References

- Bui, H.; Huynh, T.; and de Salvo Braz, R. 2012. Exact lifted inference with distinct soft evidence on every object. In *AAAI*.
- Bui, H.; Huynh, T.; and Riedel, S. 2013. Automorphism groups of graphical models and lifted variational inference. In *UAI*, 132–141. AUAI Press.
- de Salvo Braz, R. 2007. *Lifted First-Order Probabilistic Inference*. Ph.D. Dissertation, University of Illinois, Urbana-Champaign, IL.
- Domingos, P., and Lowd, D. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. San Rafael, CA: Morgan & Claypool.
- Geman, S., and Geman, D. 1984. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6:721–741.
- Gogate, V., and Domingos, P. 2011. Probabilistic Theorem Proving. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, 256–265. AUAI Press.
- Gogate, V.; Jha, A.; and Venugopal, D. 2012. Advances in Lifted Importance Sampling. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.
- Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The weka data mining software: An update. *SIGKDD Explor. Newsl.* 11(1):10–18.
- Jha, A.; Gogate, V.; Meliou, A.; and Suciu, D. 2010. Lifted Inference from the Other Side: The tractable Features. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NIPS)*, 973–981.
- Kok, S.; Sumner, M.; Richardson, M.; Singla, P.; Poon, H.; Lowd, D.; Wang, J.; and Domingos, P. 2008. The Alchemy System for Statistical Relational AI. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA. <http://alchemy.cs.washington.edu>.
- Niepert, M. 2012. Markov chains on orbits of permutation groups. In *UAI*, 624–633. AUAI Press.
- Poole, D. 2003. First-Order Probabilistic Inference. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 985–991. Acapulco, Mexico: Morgan Kaufmann.
- Poon, H., and Domingos, P. 2006. Sound and Efficient Inference with Probabilistic and Deterministic Dependencies. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Boston, MA: AAAI Press. This volume.
- Poon, H., and Domingos, P. 2008. Joint Unsupervised Coreference Resolution with Markov Logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 649–658. Honolulu, HI: ACL.
- Richardson, M., and Domingos, P. 2006. Markov Logic Networks. *Machine Learning* 62:107–136.
- Singla, P., and Domingos, P. 2008. Lifted First-Order Belief Propagation. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 1094–1099. Chicago, IL: AAAI Press.
- Singla, P., and Mooney, R. J. 2011. Abductive Markov Logic for Plan Recognition. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 1069–1075.
- Tran, S. D., and Davis, L. S. 2008. Event modeling and recognition using markov logic networks. In *ECCV*.
- van den Broeck, G., and Darwiche, A. 2013. On the complexity and approximation of binary evidence in lifted inference. In *Advances in Neural Information Processing Systems* 26, 2868–2876.
- van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted Probabilistic Inference by First-Order Knowledge Compilation. In *Proceedings of the Twenty Second International Joint Conference on Artificial Intelligence*, 2178–2185.
- van den Broeck, G. 2011. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *NIPS*, 1386–1394.
- Venugopal, D., and Gogate, V. 2012. On lifting the gibbs sampling algorithm. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS)*, 1664–1672.
- Wainwright, M. J.; Jaakkola, T. S.; and Willsky, A. S. 2003. Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In *Ninth Workshop on Artificial Intelligence and Statistics*.
- Yedidia, J. S.; Freeman, W. T.; and Weiss, Y. 2001. Generalized Belief Propagation. In Leen, T.; Dietterich, T.; and Tresp, V., eds., *Advances in Neural Information Processing Systems 13*. Cambridge, MA: MIT Press. 689–695.

Solving Distributed Constraint Optimization Problems Using Ranks

Mihaela Verman and Philip Stutz and Abraham Bernstein

Dynamic and Distributed Information Systems Group

Department of Informatics, University of Zürich

Binzmühlestrasse 14

Zürich, Switzerland

Abstract

We present a variation of the classical Distributed Stochastic Algorithm (DSA), a local iterative best-response algorithm for Distributed Constraint Optimization Problems (DCOPs). We introduce weights for the agents, which influence their behaviour. We model DCOPs as graph processing problems, where the variables are represented as vertices and the constraints as edges. This enables us to create the Ranked DSA (RDSA), where the choice of the new state is influenced by the vertex rank as computed by a modified Page Rank algorithm. We experimentally show that this leads to a better speed of convergence to Nash Equilibria. Furthermore, we explore the trade-off space between average utility and convergence to Nash Equilibria, by using algorithms that switch between the DSA and RDSA strategies and by using heterogeneous graphs, with vertices using strategies in different proportions.

Introduction

Many real-world tasks can be modeled as constraint optimization problems (COPs). To ensure scalability and avoid other major drawbacks of centralized solutions, such as the need to fully disclose all constraints and variable assignments, distributed constraint optimization (DCOP) approaches have been proposed, where the information about and control over each variable is local. As complete DCOP approaches are exponential, most practical applications prefer approximate local iterative algorithms to reach a locally optimal and sufficiently good solution: approximate best-response algorithms are one such class, the other one being message-passing algorithms (Chapman et al. 2011). Since these algorithms are focused on finding good solutions they forgo other goals such as the speed of convergence, which is especially relevant in time-critical applications.

In this paper we present a variation of the classical Distributed Stochastic Algorithm (DSA), which significantly improves the rate of finding a Nash equilibrium. Our method is based on using computed weights for the agents, which give the agent a “strength” dependent on the size of the area that includes it and in which there are no conflicts (computed via a modified version of PageRank (Page et al. 1999)). An agent’s strength is then used as a measure of its influence

on its neighbours’ decision. In addition, we propose to use hybrid approaches that mix weighted and traditional variables/agents allowing us to explore a frontier of solutions that trade off accuracy in a final state with a high rate of finding a Nash equilibrium early during the computation. We believe that this approach can be seen as a first step of adding a measure of probability of convergence for each variable in local-iterative-algorithms allowing them to explore frontiers of solutions based on the trade-offs in the application area.

We evaluate the algorithms and we empirically show that there is a trade-off between the speed of convergence and the solution quality and we explore its frontier by using algorithms that switch between the classical and the ranked strategy, and weighted combinations of two algorithms.

Related Work

Distributed Constraint Optimization Problems

Adapting Chapman et al.’s (Chapman et al. 2011) notation, a *Constraint Satisfaction Problem* (CSP) consists of: a set of variables $\{v_1, \dots, v_n\}$, a set of domains $\{D_1, \dots, D_n\}$ from which each variable v_i is allowed to take a value $s_i \in D_i$, and a set of constraints $c_1, \dots, c_m$ over subsets of the variables, which all need to be satisfied by an assignment of values to the set of variables.

A *Constraint Optimization Problem* (COP) (Schiex et al. 1995) assigns a utility function for the satisfaction or violation of each constraint $u_{c_k}(s_{c_k})$ with the given values s_{c_k} of the variables v_{c_k} over which the constraint was defined. The goal is to find the variable assignment that will maximize the global utility function, typically defined as the sum of utilities over all constraints $u_g(s) = \sum_{k \in 1..m} u_{c_k}(s)$.

A *Distributed Constraint Satisfaction Problem* (DCSP) (Yokoo et al. 1992) is produced when each variable is controlled by an agent. Analogously, a *Distributed Constraint Optimization Problem* (DCOP) shares the same characteristic as a DCSP with the added utility functions defined for COPs. As (Chapman et al. 2011) underline, we can thus define a private utility function u_i for each agent i , which will depend on the satisfaction of all constraints in which the variable v_i (controlled by the agent i) is involved, typically the sum of utilities over all these mentioned constraints. Thus, the private utility of the agent will depend on its own state (the value of its variable) and on the states of all neigh-

boring agents. The neighborhood relation is defined between two agents for which there exists a constraint over a subset that contains both their variables.

One of the key results in (Chapman et al. 2011) is the proof that DCOPs can be formulated as *team games*, a subset of potential games.

Iterative Approximate Best-Response Algorithms for DCOPs

(Chapman et al. 2011) categorize algorithms for DCOPs into three main classes. The first one consists of distributed complete algorithms, where the global optimum is obtained for the price of exponential complexity (either in the time of processing for each agent or in the number or size of messages passed). The second and third ones are two types of local iterative algorithms. The local-iterative message-passing algorithms have the characteristic that agents pass on information about their neighbors' states.

The last class, the one which they approach in their article and that we will focus on, are approximate best-response algorithms, where agents can only be aware of their immediate neighbors' states.

The authors identify algorithms that fit into the approximate best-response category, which come from both game theory and computer science backgrounds. Based on the demonstration that DCOPs can be formulated as potential games, they prove that the algorithms from both literatures are equivalent in functionality.

Furthermore, the same proof leads to the result that in DCOPs, when using approximate best-response algorithms, Nash equilibria will characterize the set of local optima and that the global optimum will necessarily be a pure-strategy Nash equilibrium. Thus, the authors conclude that this class of algorithms can be used for solving DCOPs and, more generally, for finding Nash equilibria in potential games.

In (Chapman et al. 2011) and (Chapman, Rogers, and Jennings 2011), a unifying framework is proposed for iterative approximate best-response algorithms used in DCOPs, regardless of the literature the algorithms were introduced in. The framework decomposes the algorithms into three components, making them easily comparable and more plugable, in order to create new hybrid algorithms. It contains:

the state evaluation, which consists of updating an algorithm-specific **target function** to evaluate prospective states /todoi think the paragraph above is not clear enough. updating a function?

the decision rule, which represents how the agent decides which action to take next by using the already computed target function from the state evaluation step; most algorithms use either stochastic functions, the *argmax-A* (the agent necessarily keeps the previous state if it is one of the maximizing states), or *argmax-B* (the agent randomly selects one of the maximizing states) functions;

the adjustment schedule, which refers to the order in which the agents execute their processes and is mostly left unspecified, but it can either be parallel (purely parallel: also called flood, or parallel random: also called "with Inertia"), or preferential.

Based on this framework, (Chapman et al. 2011) provides an overview of the most important approximate best-response algorithms. From this category, we will focus on the Distributed Stochastic Algorithm.

The Distributed Stochastic Algorithm

The Distributed Stochastic Algorithm (DSA) (Tel 2000) originates from the computer science literature and is one of the baseline algorithms usually used for evaluation. Our implementation (Algorithm 1) comprises the modified B variant presented by (Arshad and Silaghi 2003), *DSA-B*. Compared to the original algorithms presented by (Zhang, Wang, and Wittenburg 2002) these modifications randomly choose a maximizing state when multiple ones exist.

The state evaluation target function first finds a candidate state *candState* from the possible values $\bar{s}$ the agent *i* can choose at time *t* that maximize the function U_i , given the fixed states of the neighbors s_{-i} observed at time $t - 1$. The decision rule then uses the difference between the utilities of the candidate state and the past state.

DSA-B and *DSA-B* choose the candidate state over the past state, with a given probability *p*, only when the candidate is strictly better than the past state or when it is as good as the past state but there are still conflicts.

For shortness, we will from now on refer to *DSA-B* as *DSA*.

Algorithm 1 DSA (agent *i*, step *t*, degree of parallelism *p*)

```

candState :=  $\operatorname{argmax}_{\bar{s} \neq s_i(t-1)} U_i(\bar{s}, s_{-i}(t-1))$ 
currUtil :=  $U_i[s_i(t-1), s_{-i}(t-1)]$ 
 $\Delta := U_i[\text{candState}, s_{-i}(t-1)] - \text{currUtil}$ 
if  $\text{random}(0, 1) < p$  then
    if  $(\Delta > 0 \vee (\Delta = 0 \wedge \text{allConstraintsSatisfied}))$  then
        return candState
    end if
end if
return  $s_i(t-1)$ 

```

DCOPs as graph processing problems

DCOPs can be modeled as graph-processing, vertex-centric problems as follows: *First*, variables are represented as vertices; the neighborhood relationship is described by edges. Thus, two vertices should have an edge between them if there is a constraint that contains both their variables. *Second*, given that DCOP algorithms often employ an agent-based view where every variable is "guarded" by an agent that ensures that its constraints are kept, it makes sense to define the behavior of every agent within the function that modifies the vertex state. The utility function of this "agent" is dependent on the state of the adjacent vertices. *Third*, agents signal their state to their neighbours via the edges. *Last*, every vertex holds the list of all constraints it is part of, and uses it to calculate values of the utility function specific to each algorithm.

The adjustment schedule is modeled mainly by the choice of a synchronous (for the flood schedule) execution mode. In the case of the parallel random adjustment schedule,

the inertia mechanism can be introduced as part of the decision rule. For preferential adjustment schedules such as those used in the Maximum Gain Messaging algorithm (Yokoo and Hirayama 1996) (Maheswaran, Pearce, and Tambe 2006), the function would be implemented for the two stages (when the vertex computes the gain it has from changing strategy and when it changes its state if it has the maximum gain out of all its neighbors). Messaging would be done accordingly.

This mapping enables us to use additional graph algorithms to enhance the performance of local iterative best-response algorithms.

Ranked Distributed Stochastic Algorithm

By observing the way the coloring evolves for DSA we found that the algorithms initially converge quickly to large, conflict-less areas, with conflicts at their borders. As a consequence, our first hypothesis is that assigning some weights to the agents that would correlate with the size and connectivity of the “peaceful” area they are part of and using those weights in order to influence their neighbours’ decision making will help larger areas “win” over smaller areas by pulling conflicted vertices into their area and lead to faster convergence.

To compute the size of an area we decided to use a modified PageRank algorithm (Page et al. 1999) to iteratively compute these weights, since it would convey not only the size of the area, but also the connectivity, which in turn, correlates with the number of constraints that area fulfills. Specifically, in our case a vertex only receives votes from the neighbours that are not in conflict with it. These peaceful zones should thus create support for the “border” area. In our implementation, since the future state of the neighbours is unknown, a vertex splits its rank for all its neighbours, and, as it can be observed in Algorithm 2, the message it sends to its neighbours comprises not only of its state, but also the split rank.

An agent uses the ranks that are received from the neighbours for two purposes: the first is to compute the target function in order to make a decision about the next action, the second is recalculating its own rank, which will be split between the neighbours during the next step.

The difference between DSA and RDSA lies mostly inside the target function. We define the allies as the neighbours that are not in conflict with the agent and the opponents as the neighbours that are in conflict with the agent.

The target function for an action is no longer the utility of the action given the neighbours’ actions at the previous step, but the difference between the total ranks received from the allies and the total ranks that could have been received from the opponents: $target(\bar{s}, s_{-i}(t-1)) = allyRankSum(\bar{s}, s_{-i}(t-1)) - opponentRankSum(\bar{s}, s_{-i}(t-1))$.

The ranks are recalculated at every step, but, unlike the original PageRank algorithm, ranks are only received from allied neighbours (i.e. neighbours that are not in conflict with the vertex).

Algorithm 2 RDSA(agent i , step t , degree of parallelism p , $baseRank$, damping factor d)

```

candState :=  $argmax_{\bar{s} \neq s_i(t-1)} target(\bar{s}, s_{-i}(t-1))$ 
currExpectedUtil :=  $target[s_i(t-1), s_{-i}(t-1)]$ 
 $\Delta := target[candState, s_{-i}(t-1)] - currExpectedUtil$ 
if  $random(0, 1) < p$  then
    if  $(\Delta > 0 \vee (\Delta = 0 \wedge allConstraintsSatisfied))$  then
        rank =  $baseRank + d * allyRankSum(candState)$ 
        return candState, rank/neighboursNumber
    end if
end if
rank =  $baseRank + d * allyRankSum(s_i(t-1))$ 
return  $s_i(t-1), rank/neighboursNumber$ 

```

Switching Distributed Stochastic Algorithms

In our first pilot runs we observed that RDSA converged faster but that DSA eventually achieved higher solution quality. These observations prompted us to examine if there is a strategy of switching between the two algorithms such that we obtain both faster convergence and higher solution quality.

Algorithm 3 Switch-RDSA(agent i , step t , degree of parallelism p , $baseRank$, damping factor d , threshold step T , switching probability p_s)

```

if  $(switched = F) \wedge (t > T) \wedge (random(0, 1) < p_s)$  then
    switched := T
end if
if switched = F then
    candState :=  $argmax_{\bar{s} \neq s_i(t-1)} target(\bar{s}, s_{-i}(t-1))$ 
    currExpUtil :=  $target[s_i(t-1), s_{-i}(t-1)]$ 
     $\Delta := target[candState, s_{-i}(t-1)] - currExpUtil$ 
else
    candState :=  $argmax_{\bar{s} \neq s_i(t-1)} U_i(\bar{s}, s_{-i}(t-1))$ 
    currUtil :=  $U_i[s_i(t-1), s_{-i}(t-1)]$ 
     $\Delta := U_i[candState, s_{-i}(t-1)] - currUtil$ 
end if
if  $random(0, 1) < p$  then
    if  $(\Delta > 0 \vee (\Delta = 0 \wedge allConstraintsSatisfied))$  then
        rank =  $baseRank + d * allyRankSum(candState)$ 
        return candState, rank/neighboursNumber
    end if
end if
rank =  $baseRank + d * allyRankSum(s_i(t-1))$ 
return  $s_i(t-1), rank/neighboursNumber$ 

```

Specifically, we developed three types of algorithms that switch their behaviour between DSA and RDSA.

The *Switch algorithms* (Algorithm 3) make the agents behave like RDSA until a given iteration step t , after which, with a probability p_s , the agent will switch to the original DSA algorithm.

The *Inverted Switch algorithms* behave the same, but start with the original DSA and then switch to RDSA.

The ranks are calculated and communicated to the neighbours regardless of the strategy.

To prevent convergence to lower quality Nash equilibria and to obtain a dynamic way of switching (since The Switch and Inverted Switch variants need to be given the threshold iteration step in advance), we also propose the *Dynamic RDSA* (4). An agent behaves as described by DSA when it is in a local minimum of the ranked target function and as described by RDSA when it is not. Dynamic RDSA could help escape local minima if there are several actions for a vertex which lead to the same maximum utility.

Algorithm 4 Dynamic-RDSA(agent i , step t , degree of parallelism p , $baseRank$, damping factor d)

```

candState :=  $\arg\max_{\bar{s} \neq s_i(t-1)} target(\bar{s}, s_{-i}(t-1))$ 
currExpectedUtil :=  $target[s_i(t-1), s_{-i}(t-1)]$ 
 $\Delta := target[candState, s_{-i}(t-1)] - currExpectedUtil$ 
if  $\Delta = 0$  then
    candState :=  $\arg\max_{\bar{s} \neq s_i(t-1)} U_i(\bar{s}, s_{-i}(t-1))$ 
    currUtil :=  $U_i[s_i(t-1), s_{-i}(t-1)]$ 
     $\Delta := U_i[candState, s_{-i}(t-1)] - currUtil$ 
end if
if  $random(0, 1) < p$  then
    if ( $\Delta > 0 \vee (\Delta = 0 \wedge allConstraintsSatisfied)$ ) then
        rank =  $baseRank + d * allyRankSum(candState)$ 
        return candState, rank/neighboursNumber
    end if
end if
rank =  $baseRank + d * allyRankSum(s_i(t-1))$ 
return  $s_i(t-1), rank/neighboursNumber$ 

```

Heterogeneous graphs

To further the investigation of the trade-off between high solution quality and fast convergence to a Nash equilibrium beyond discrete switches, we also propose the usage of heterogeneous graphs that could enable the exploration of a finer range over this trade-off. We can create heterogeneous graphs by combining vertices that use one of two algorithms, in a certain proportion. We choose to create combinations of two between DSA, RDSA, and Dynamic Ranked DSA.

Experiments

So far we suggested modeling COPs as graph computations and described the implementations of several variations of the DSA algorithm involving the computation of ranks.

In this section we discuss the experiments that we ran in order to compare the variations with the original algorithm and our findings about the performance of the algorithms in terms of solution quality and speed of convergence to a Nash equilibrium.

Experimental Setup

To evaluate our algorithms we chose the *vertex coloring problem*, which is typically used in the literature to benchmark DCOP algorithms. We evaluated on the 40-vertex graphs from the Adopt repository.¹ All 50 graphs have 3 colors, link densities 2 and 3. All constraints had utility 1.

¹<http://teamcore.usc.edu/dcop/>

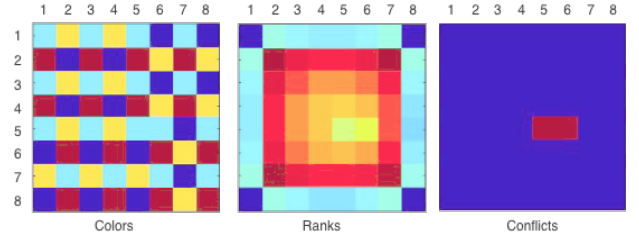


Figure 1: Coloring problem for an 8x8 grid.

The algorithms were all derived from the DSA- $\bar{B}$ and they were implemented inside the graph processing framework SIGNAL/COLLECT (Stutz, Bernstein, and Cohen 2010).

We ran the algorithms synchronously for 100 steps and measured, for each of them: The average global utility ratio over iteration steps, the average end utility ratio, the average number of runs with an optimal solution, the average conflict count at the end, the proportion of runs reaching a Nash equilibrium and the average number of steps to the first Nash equilibrium over the runs where the algorithms did reach one, and the average number of vertices that are in local minima. The utility ratios were computed such that the maximum utility was considered to be when all constraints were satisfied.

We ran each algorithm configuration 10 times on each problem. The algorithms ran with the following parameters:

- The base rank was set to 0.15 and the damping factor to 0.85, for all runs.
- DSA, RDSA, Dynamic RDSA with degrees of parallel executions $p = 0.4, 0.5, 0.6, 0.7, 0.8$.
- Switch and InvertedSwitch: with degrees of parallel executions $p = 0.4, 0.5, 0.6, 0.7, 0.8$ and switch thresholds: 10, 15, 20.
- The hybrid algorithms: the best performing algorithms depending on the degree of parallelism, DSA($p = 0.6$), RDSA($p=0.5$), Dynamic RDSA($p=0.4$). We took combinations of 2 of them in different proportions = 0.1, 0.3, 0.5, 0.7, 0.9.

Results

In our initial evaluation, comparing DSA and RDSA showed that RDSA does converge faster, but the solution quality is lower than the one found by DSA.

The explanation for this phenomenon lies in the fact that after the regions stabilize, a border vertex might receive conflicting “instructions” from vertices of the strongest peaceful area surrounding it. In Figure 1, we see an 8x8 grid with vertices neighbouring on the sides and diagonals. The figure shows a 4-colouring of this grid, and heat maps for the ranks and for the number of conflicts. Vertices in positions (5, 5) and (5, 6) are in conflict, but they are both supported by the same peaceful area.

Figure 2 focuses on the most important areas of the graphs for DSA and RDSA over iteration steps. The first graph illustrates the number of vertices in local minima (where 40

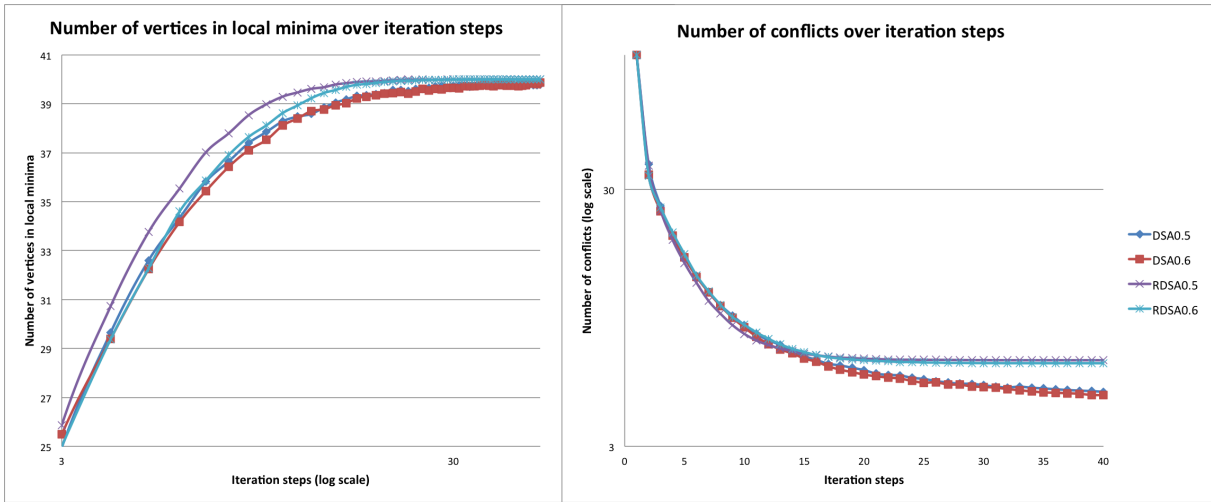


Figure 2: Focused view on the number of vertices in local minima and number of conflicts over iteration steps.

is the best outcome) and the second one the number of conflicts (where 0 is the best outcome). We show the two best-performing degree of parallelism values: 0.5 and 0.6, for both algorithms. The points of the graph are averages over all the runs (50 problems and 10 runs for each problem). We see how the RDSA algorithms converge much faster to a Nash equilibrium, at around iteration 20, while the DSA algorithms converge much slower, but to higher quality solutions. We also see that in the beginning, the number of conflicts drops slightly faster for RDSA with 0.5 degree of parallelism, and a crossover point for the two algorithms happens around the 15th iteration step.

This result led us to evaluate the Switching algorithms for switch thresholds around the value of 15.

Figure 3 shows how the trade-off between utility ratio and number of vertices in local minima evolves over time for DSA, RDSA and two of the Switch algorithms. The several data points for each category come from different degrees of parallelism. We focus on the majority of data points at steps 5, 10, 30 and 100. We observe how RDSA starts with both better solution quality and higher number of vertices in local minima, but, over time, DSA gets better in terms of solution quality. The Switch algorithms end up being quite similar to DSA, but also exhibit a good start.

The fact that RDSA reaches a lower quality solution motivated our evaluation of the Dynamic RDSA, because the switching to the DSA strategy could help escape the local target function minimum when an agent has a choice between multiple actions leading to the same maximum utility. However, the Dynamic RDSA had both lower solution quality and lower number of vertices in local minima.

For the heterogeneous algorithms evaluation we took the DSA, RDSA and Dynamic RDSA with the degrees of parallelism that enabled the best performance and ran combinations of two in different proportions. In Figure 4 we observe how the heterogeneous versions do indeed vary between the two algorithms that compose them.

Limitations and future work

As we have already seen, RDSA encounters problems when a vertex receives conflicting "instructions" or when two conflicting vertices are supported by the same area.

So far, we made a comparison between the aggregates of 50 different problems, but a method that would permit a more generalized view upon the performance of the algorithms would be to compare the algorithms for each problem and to use statistical tests to determine the significance of potential improvements. In addition, we would like to evaluate on more problems of different sizes and to test whether the algorithms scale to much larger problems and to observe if they change their behaviour.

We observed promising behaviour in the switching strategies we explored, and a next target should be the investigation of other switching strategies. Furthermore, so far, the threshold iteration step was given in advance, but dynamic and automatic method of determining when switching the behaviour would be beneficial for an agent would be a major improvement. It would also be desirable to extend other local iterative approximate best-response algorithms with ranked versions.

Conclusions

DCOPs are ubiquitous in real-world applications that involve managing and optimizing resource allocation, and speed of convergence is of utmost importance in some scenarios. We believe that our findings show the promising effects of graph processing inspired modifications of classical algorithms and the possible benefits that may emerge by combining different algorithms into a heterogeneous system.

Acknowledgments

We would like to thank the Hasler Foundation for supporting the SIGNAL/COLLECT project, which we used to implement our algorithms, and the Dinu Patriciu Foundation for the "Open Horizons" scholarship for Mihaela Verman.

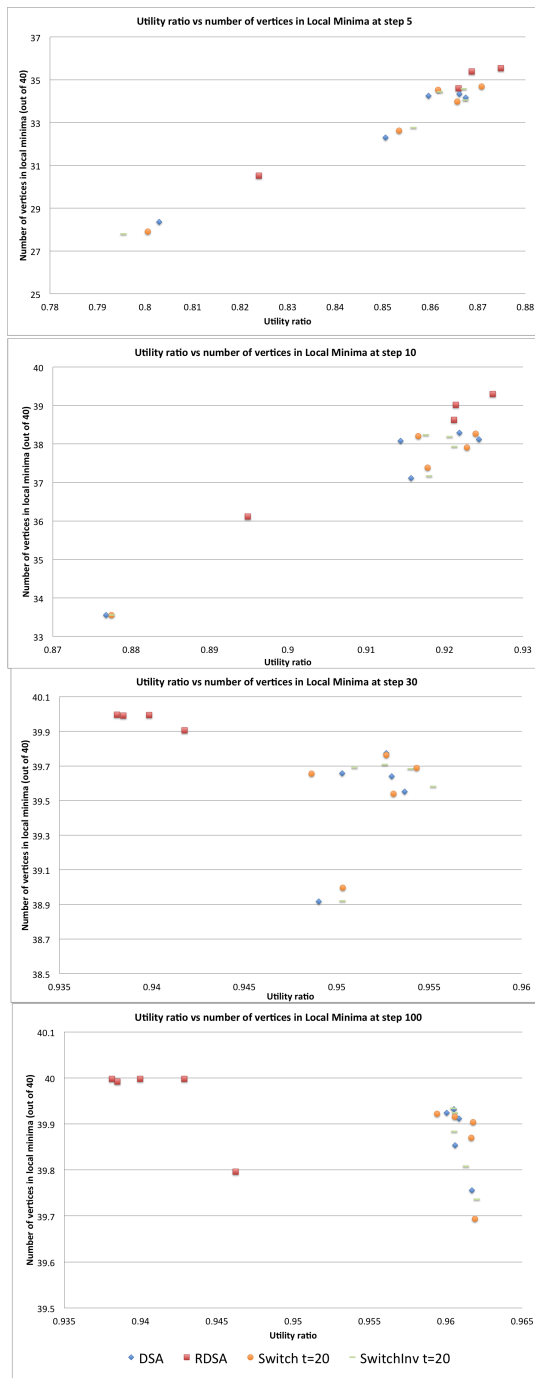


Figure 3: Trade-off between Utility ratio on the X axis and Number of vertices in local minima on the Y axis, for DSA, RDSA, Switch RDSA and Inverted Switch RDSA.

References

- Arshad, M., and Silaghi, M. 2003. Distributed simulated annealing and comparison to ds. In *Proceedings of the fourth international workshop on distributed constraint reasoning*.
Chapman, A.; Rogers, A.; Jennings, N.; and Leslie, D. 2011. A unifying framework for iterative approximate best-

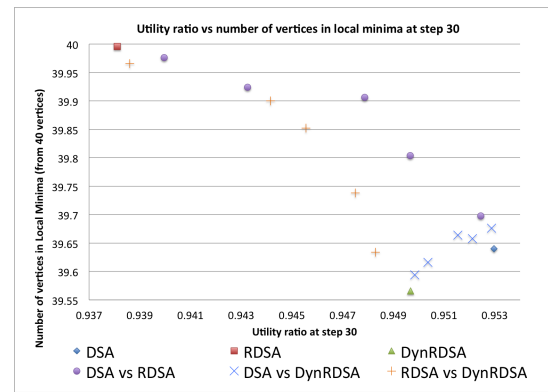


Figure 4: Trade-off between Utility ratio on the X axis and Number of vertices in local minima on the Y axis, for DSA, RDSA, Dynamic RDSA and the Heterogeneous runs.

response algorithms for distributed constraint optimization problems. *Knowledge Engineering Review* 26(4):411–444.

Chapman, A.; Rogers, A.; and Jennings, N. 2011. Benchmarking hybrid algorithms for distributed constraint optimisation games. *Autonomous Agents and Multi-Agent Systems* 22:385–414. 10.1007/s10458-010-9128-3.

Maheswaran, R.; Pearce, J.; and Tambe, M. 2006. A family of graphical-game-based algorithms for distributed constraint optimization problems. *Coordination of large-scale multi-agent systems* 127–146.

Page, L.; Brin, S.; Motwani, R.; and Winograd, T. 1999. The pagerank citation ranking: Bringing order to the web.

Schiex, T.; Fargier, H.; Verfaillie, G.; et al. 1995. Valued constraint satisfaction problems: Hard and easy problems. In *International Joint Conference on Artificial Intelligence*, volume 14, 631–639. Citeseer.

Stutz, P.; Bernstein, A.; and Cohen, W. 2010. Signal/collect: Graph algorithms for the (semantic) web. *The Semantic Web—ISWC 2010* 764–780.

Tel, G. 2000. *Introduction to distributed algorithms*. Cambridge Univ Pr.

Yokoo, M., and Hirayama, K. 1996. Distributed breakout algorithm for solving algorithm for solving distributed constraint satisfaction and optimization problems. In *Proceedings of the 2nd Intl. Conference on Multiagent Systems*.

Yokoo, M.; Ishida, T.; Durfee, E.; and Kuwabara, K. 1992. Distributed constraint satisfaction for formalizing distributed problem solving. In *Distributed Computing Systems, 1992., Proceedings of the 12th International Conference on*, 614–621. IEEE.

Zhang, W.; Wang, G.; and Wittenburg, L. 2002. Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance. In *Proceedings of AAAI Workshop on Probabilistic Approaches in Search*.

Efficient Probabilistic Inference for Dynamic Relational Models

Jonas Vlasselaer and Wannes Meert and Guy Van den Broeck and Luc De Raedt

Department of Computer Science, KU Leuven, Belgium
 firstname.lastname@cs.kuleuven.be

Abstract

Over the last couple of years, the interest in combining probability and logic has grown strongly. This led to the development of different software packages like PRISM, ProbLog and Alchemy, which offer a variety of exact and approximate algorithms to perform inference and learning. What is lacking, however, are algorithms to perform efficient inference in relational temporal models by systematically exploiting temporal and local structure. Since many real-world problems require temporal models, we argue that more research is necessary to use this structure to obtain more efficient inference and learning. While existing relational representations of dynamic domains focus rather on approximate inference techniques we propose an exact algorithm.

Motivation

A dynamic relational model allows to compactly model a stochastic process by means of a relational or logic representation. Although temporal domains can be modeled in many statistical relational languages, only some representations have been proposed that treat time as a first-class citizen, e.g. CPT-L (Thon, Niels, and De Raedt 2011) and Relational Dynamic Bayesian Networks (RDBNs) (Manfredotti 2009). Hence, only some languages offer specific techniques for inference and learning in temporal models. Existing general-purpose implementations typically scale exponentially with the number of time steps.

There are two distinct properties of dynamic relational representations that can be exploited for efficient inference. First, the logical structure that is abundant in relational models leads to local structure in the probabilistic dependencies and determinism in the distribution. It is well-known that exploiting such properties leads to exponential speed gains (Darwiche 2009). Second, these models contain repetitive structures, both over time, and within one time slice.

To the best of our knowledge, no existing work exploits both of the aforementioned properties for exact inference. Most research on inference for (R)DBNs is centered around approximate techniques like (relational) particle filters and sampling. In this work, however, we focus on exact inference and propose a knowledge compilation approach to

maximally exploit all the available structure: we compile a circuit once, and reuse it multiple times in the course of answering a single query.

Example Below we show a temporal model which handles the task of finding failures in a digital circuit (see Figure 1). The model is expressed in ProbLog¹, but any probabilistic logic language can be used. The goal is to compute the health state of a digital component at time T , i.e. the probability the component is faulty, given a set of electrical inputs and outputs up until time τ . As such, the query is $P(\text{healthy}(G, T) \mid \text{input}(W, 0:\tau), \text{output}(W, 0:\tau))$.

```
wire(1). wire(2). wire(3). wire(4).
in(1). in(2). out(4).
gate(a, not, [1], 3). gate(b, and, [3, 2], 4).

0.990::healthy(G, 0) :- gate(G, _, _, _).
0.990::healthy(G, T) :- T>0, healthy(G, T-1).
0.001::healthy(G, T) :- T>0, \+healthy(G, T-1).

0.5::high(W, T) :- in(W).
0.5::high(W, T) :- gate(G, _, _, W), \+healthy(G, T).
high(W, T) :- gate(G, not, [I], W), healthy(G, T), \+high(I, T).
high(W, T) :- gate(G, and, [I, J], W), healthy(G, T),
    high(I, T), high(J, T).

input(W, T) :- in(W), high(W, T).
output(W, T) :- out(W), high(W, T).
```

Inference in (R)DBNs

None of the deterministic facts in our logic program depend on *time* as the structure of a digital circuit typically does not change. Hence, the model defined above can be seen as a Relational Dynamic Bayesian Network (RDBN) which is fully defined by a pair of networks $(M_0, M_{t \rightarrow t+1})$ with M_0 a Relational Bayesian Network (RBN) that defines the prior state distribution, and $M_{t \rightarrow t+1}$ a two-time-slice RDBN that defines the transition model. In the transition model, every predicate at time t has a set of *parent predicates* at time $t-1$ or t . As the structure does not change over time, grounding out the relational DBN results in a propositional DBN which is completely defined by a pair of static BNs $(B_0, B_{t \rightarrow t+1})$.

One way to do inference in DBNs is to unroll the network and apply any algorithm for Bayesian Networks. This approach, however, is not guaranteed to scale linear with the

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

¹<http://dtai.cs.kuleuven.be/problog>

number of time steps. To obtain an algorithm which does scale linear in time, one has to define a set of variables in the transition model which d-separate the past from the future. Typically, this set is called the *interface* and contains all variables at time t which have an outgoing arc to $t + 1$.

Exploiting Recursive Structure

The recursive structure in (R)DBNs, given by the transition model, in combination with the definition of the interface allows to perform inference by recursively computing the joint probability distribution of the interface, often referred to as the *forward* and *backward message* (Murphy 2002). We propose to use knowledge compilation (Darwiche 2009) to first compile the structure into an Arithmetic Circuit (AC) and, next, evaluate the obtained circuit multiple times to compute the messages. As the transition model remains static over time, it suffices to compile it only once. As such, this one time compilation cost is amortized over all evaluation steps necessary to compute the messages. One evaluation can be done in time polynomial in the size of the AC.

The recursive structure which we need to compile contains all nodes from the second time slice in the transition model together with all their parents in the first time slice. As this does not take into account the dependencies between the variables of the incoming interface, i.e. the variables from the first time-slice, we introduce a second compilation step. This step is necessary and conditions the obtained AC on all combinations of truth values of the incoming interface variables. Once this is done, the circuit is reused $N \cdot M$ times to recursively compute the messages, where $N = 2^I$ (with I the number of variables in the interface) and M is the number of time steps.

Exploiting Local Structure

Our proposed approach is closely related to the interface algorithm (Murphy 2002) where one constructs a junction tree for the recursive structure. Murphy’s approach, however, only exploits conditional independences while our compilation approach also exploits local structure. Compiling a network results in an AC which is not necessary exponential in the treewidth. This allows inference in cases where this is impossible with other algorithms which do not exploit this structure. As relational models typically introduce determinism and equal parameters, our approach should outperform the classic implementation of the interface algorithm.

Proof of Concept

As a proof of concept we generated random digital circuits of varying size where we also modeled digital gates with more than two input wires. To vary both the complexity (size) of the digital circuit and the number of variables in the interface, we allowed multiple digital gates to have the same health state. Gates share a health variable, for example, when they share a power line.

We implemented our approach in ProbLog and did some preliminary experiments to compare it with the DBN algorithms implemented in the Bayesian Network Toolbox

(bnt)². The task was to compute the probability of the health predicates at $T = 10$, given evidence on the input and output of the circuit up until $\tau = 9$. The experiments were conducted on a computer with a working space of 4GB RAM.

We can draw two conclusions from the results shown in figure 1. Firstly, the compilation approach is somewhat faster compared to bnt. Secondly, and more importantly, is that our approach allows inference in RDBNs for digital circuits that are almost three times larger compared to bnt as the latter already runs out of memory for models with more than about 30 digital gates. Also important to notice is that, although the total inference time takes approximately 120 seconds for the biggest circuit, the computation of the forward message (the recursive step) only takes approximately 0.5 seconds. As such, most of the time is spent in the (off-line) compilation step which only needs to be conducted once.

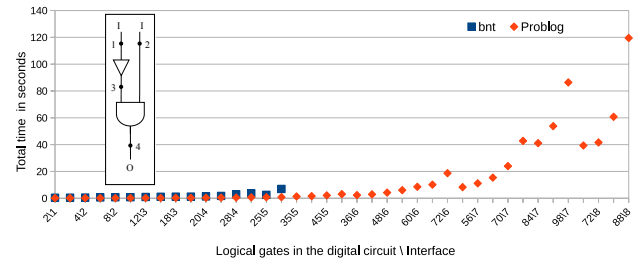


Figure 1: The digital circuit used as running example and the experimental results. The x axis shows the number of gates in the digital circuit and the number of ground atoms h in the interface at each time t . The y axis shows the total inference time in seconds (compilation + evaluation).

Conclusion and Challenges

We presented an algorithm for exact inference in relational DBNs, based on knowledge compilation, to exploit as well the recursive as the local structure in the model. The preliminary results show the promise of our technique for relational DBNs with closed domains. Two remaining challenges are (i) relational DBNs with open domains, i.e. where the number of ground atoms might change over time, and (ii) applying the insights from exploiting local structure to approximate inference techniques.

References

- Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.
- Manfredotti, C. 2009. Modeling and inference with relational dynamic Bayesian networks. In *Advances in artificial intelligence*. Springer. 287–290.
- Murphy, K. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. Dissertation, UC Berkeley.
- Thon, I.; Niels, L.; and De Raedt, L. 2011. Stochastic relational processes: Efficient inference and applications. *Machine Learning* 82(2):239–272.

²<https://github.com/bayesnet/bnt>

ProPPR: Efficient First-Order Probabilistic Logic Programming for Structure Discovery, Parameter Learning, and Scalable Inference

William Yang Wang and **Kathryn Mazaitis** and **William W. Cohen**
 School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA.
 {yww,krivard,wcohen}@cs.cmu.edu

Abstract

A key challenge in statistical relational learning is to develop a semantically rich formalism that supports efficient probabilistic reasoning using large collections of extracted information. This paper presents a new, scalable probabilistic logic called ProPPR, which further extends stochastic logic programs (SLP) to a framework that enables efficient learning and inference on graphs: using an abductive second-order probabilistic logic, we show that first-order theories can be automatically generated via parameter learning; that in parameter learning, weight learning can be performed using parallel stochastic gradient descent with a supervised personalized PageRank algorithm; and that most importantly, queries can be approximately grounded with a small graph, and inference is independent of the size of the database.

Introduction

One key problem for scaling up first-order probabilistic first-order logics is that queries are typically answered by “grounding” the query—i.e., mapping it to a propositional representation, and then performing propositional inference—and for many logics, the size of the “grounding” can be extremely large for large databases.

We propose a first-order probabilistic language which is well-suited to approximate “local” grounding: we present an extension to *stochastic logic programs* (SLP) (Cussens 2001) that is biased towards short derivations, and show that this is related to *personalized PageRank* (PPR) (Page et al. 1998; Chakrabarti 2007) on a linearized version of the proof space. Based on the connection to PPR, we develop a proveably-correct approximate inference scheme, and an associated proveably-correct approximate grounding scheme: specifically, we show that it is possible to prove a query, or to build a graph which contains the information necessary for weight-learning, in time $O(\frac{1}{\alpha\epsilon})$, where α is a reset parameter associated with the bias towards short derivations, and ϵ is the worst-case approximation error across all intermediate stages of the proof. This means that both inference and learning can be approximated in time *independent of the size of the underlying database*—a surprising and important result. The ability to locally ground queries has another important consequence: it is possible to decompose the problem of

Table 1: A simple program in ProPPR. See text for explanation.

<code>about(X,Z) :- handLabeled(X,Z)</code>	# base.
<code>about(X,Z) :- sim(X,Y),about(Y,Z)</code>	# prop.
<code>sim(X,Y) :- links(X,Y)</code>	# sim,link.
<code>sim(X,Y) :-</code> <code>hasWord(X,W),hasWord(Y,W),</code> <code>linkedBy(X,Y,W)</code>	# sim,word.
<code>linkedBy(X,Y,W) :- true</code>	# by(W).

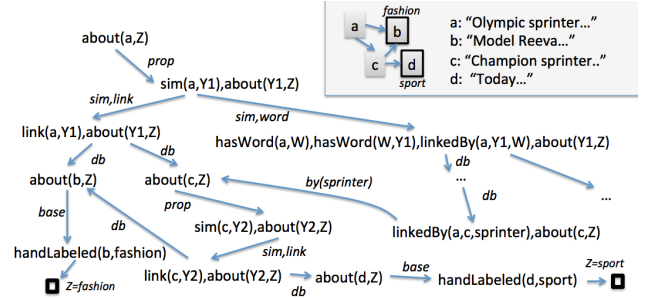


Figure 1: A partial proof graph for the query `about(a,Z)`. The upper right shows the link structure between documents `a`, `b`, `c`, and `d`, and some of the words in the documents.

weight-learning to a number of moderate-size subtasks (in fact, tasks of size $O(\frac{1}{\alpha\epsilon})$ or less) which are weakly coupled. Based on this we outline a parallelization scheme, which in our initial implementation provides an order-of-magnitude speedup in learning time.

ProPPR¹

Table 1 and Figure 1 illustrate a ProPPR theory and a corresponding proof graph. We refer the reader to prior papers (Wang, Mazaitis, and Cohen 2013; Wang et al. 2014) for a detailed explanation of ProPPR’s semantics: briefly, a ProPPR theory T is a Horn clause (Prolog) theory, where each clause c is associated with a function ϕ_c which computes a set of features. ProPPR builds a graph rooted at the query goal, where each node corresponds to a conjunction of

¹Programming with Personalized PageRank

goals $A_1, \dots, A_k$ and a substitution θ that imply the query goal (i.e. $T \wedge (A_1, \dots, A_k \theta) \rightarrow Q\theta$), and each edge corresponds to the application of a clause c . The feature-functions ϕ_c are simple: each feature corresponds to a Prolog goal, which may include bound variables from the head of the clause, the bindings of which are supplied when ϕ_c is invoked. A node corresponding to an empty conjunction (denoted $\square$) is a leaf of the graph, and corresponds to a completed proof of the query goal. Every edge created by the clause c is labeled with the features produced by ϕ_c for that application of c . Note that each different $\square$ node corresponds to a different proof for Q , and different proofs may be associated with different substitutions θ and hence different solutions to the query Q .

Finally, we associated a score with each node in the graph by performing a personalized PageRank (PPR) (aka “random walk with restart”) process, where transition probabilities are based on edge weights, which are in turn determined by a weighted function of the features.

ProPPR allows a fast approximate proof procedure, in which only a small subset of the full proof graph is generated. In particular, if α upper-bounds the reset probability, and d upper-bounds the degree of nodes in the graph, then one can efficiently find a subgraph with $O(\frac{1}{\alpha d})$ nodes which approximates the weight for every node within an error of $d\epsilon$ (Wang, Mazaitis, and Cohen 2013), using a variant of the PageRank-Nibble algorithm of Reid *et al* (Andersen, Chung, and Lang 2008).

Experiments

Parameter Learning

ProPPR’s parameter learning framework is implemented using a parallel stochastic gradient descent variant to optimize the log loss using the supervised personalized PageRank algorithm (Backstrom and Leskovec 2011): given the training queries, we perform a random walk with restart process, and upweight the edges that are more likely to end up with a known positive solution. In experiments (Wang, Mazaitis, and Cohen 2013), we show that learning in ProPPR helps improving the AUC of an entity resolution experiment from 0.68 to 0.8 on the CORA dataset; and that significant speedup in parallel learning can be achieved by multithreading. We also demonstrate that, for a bag-of-words classification task on the WebKB dataset, parameter learning for ProPPR is able to improve the AUC from 0.5 to 0.797; and that the average training time for 16 threads is only about 2 minutes.

Scalable Inference

To demonstrate the effectiveness of ProPPR’s inference scheme, we also consider the inference time of ProPPR on the Cora entity resolution dataset, comparing to various MLN inference methods. In our CIKM paper (Wang, Mazaitis, and Cohen 2013), we fix the amount of test queries, and increase the size of the database for entity resolution of the CORA dataset. We empirically show that the inference time in ProPPR is independent of the size of the

database, whereas all the inference methods (Gibbs sampling, Lifted Belief Propagation, and MAP) in MLN depend on the size of the dataset. Similar results are also observed in the WebKB experiments. In recent work, we also study the joint inference task (Wang et al. 2014) on the 1M-facts subsets of the NELL KB, using recursive first-order theories. Comparing ProPPR to MC-SAT (MLN) in our large-scale inference experiments on the NELL datasets (Wang et al. 2014), we also notice the significant improvement in the inference time, when varying the amount the entities in the database.

Structure learning

ProPPR’s structure learning scheme is based on a meta-interpretive program that resembles Metagol (Muggleton and Lin 2013). The idea is that: using an abductive second-order logic, one can construct the hypothesis space for first-order theories, and thus structure learning can be relaxed to parameter learning. In our preliminary experiments, we find that in a KB completion task using Geoff Hinton’s kinship dataset, neither FOIL nor Alchemy’s MLN structure learning method outperform our simple baseline, and our iterated structural gradient approach further improve the baseline. We then further validate the result on additional datasets, including a Alyawarra kinship dataset (Denham 1973), a UMLS dataset (McCray et al. 2001; McCray 2003), and two NELL subsets. These additional experimental results show that our proposed method consistently outperforms Alchemy’s MLN on various problems.

References

- Andersen, R.; Chung, F. R. K.; and Lang, K. J. 2008. Local partitioning for directed graphs using pagerank. *Internet Mathematics* 5(1):3–22.
- Backstrom, L., and Leskovec, J. 2011. Supervised random walks: predicting and recommending links in social networks. In *WSDM*.
- Chakrabarti, S. 2007. Dynamic personalized PageRank in entity-relation graphs. In *WWW*.
- Cussens, J. 2001. Parameter estimation in stochastic logic programs. *Machine Learning* 44(3):245–271.
- Denham, W. 1973. The detection of patterns in alyawarra nonverbal behavior. *PhD thesis, University of Washington*.
- McCray, A. T.; Burgun, A.; Bodenreider, O.; et al. 2001. Aggregating umls semantic types for reducing conceptual complexity. *Studies in health technology and informatics* (1):216–220.
- McCray, A. T. 2003. An upper-level ontology for the biomedical domain. *Comparative and Functional Genomics* 4(1):80–84.
- Muggleton, S., and Lin, D. 2013. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. In *IJ-CAI*.
- Page, L.; Brin, S.; Motwani, R.; and Winograd, T. 1998. The PageRank citation ranking: Bringing order to the web. In *TR, Computer Science department, Stanford*.
- Wang, W. Y.; Mazaitis, K.; Lao, N.; Mitchell, T.; and Cohen, W. W. 2014. Efficient inference and learning in a large knowledge base: Reasoning with extracted information using a locally groundable first-order probabilistic logic. *arXiv preprint arXiv:1404.3301*.
- Wang, W. Y.; Mazaitis, K.; and Cohen, W. W. 2013. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *CIKM*.

AAAI Press

2275 East Bayshore Road, Suite 160
Palo Alto, California 94303

WS-14-13

