

The Advantages of Seed Examples in First-Order Multi-class Subgroup Discovery

Tarek Abudawood¹ and Peter Flach²

Abstract. Subgroup discovery is concerned with finding subsets of a population whose class distribution is significantly different from the overall distribution. Previously subgroup discovery has been predominantly investigated under the propositional logic framework. This paper investigates multi-class subgroup discovery in an inductive logic programming setting, where subgroups are defined by conjunctions in first-order logic. We present a new weighted covering algorithm, inspired by the Aleph first-order rule learner, that uses seed examples in order to learn diverse, representative and highly predictive subgroups that capture the distribution of all classes. Experimental results show that the new subgroup discovery algorithm significantly reduces the number of hypotheses considered during the induction process, leading to faster theory construction. The use of seed examples results in considerable and statistically significant improvement of predictive power, both in terms of accuracy and AUC.

Keywords: Subgroup Discovery, Supervised Learning, Inductive Logic Programming

1 Introduction

Rule induction is a common form of machine learning and data mining often used in classification and association rule learning. It can either be done in a propositional framework, where examples are represented by attribute-value pairs, or in the more expressive framework of first-order logic, where examples, rules and background knowledge are described by logical formulae, allowing the rule learner to deal with complexly structured data.

Classification rule learning is a predictive task aimed at constructing a set of rules that predict the class of future examples. On the other hand, association rule learning, e.g. Apriori [2], is a form of descriptive induction aimed at the discovery of individual rules that express interesting patterns in data. In classification rule learning a target concept is pre-defined and so the search heuristic is usually some form of accuracy. On the other hand, no target concept is given in descriptive rule learning and the search heuristic evaluates measures of interestingness and unusualness in the data, e.g., support and confidence. Subgroup discovery can be seen as being halfway between predictive and descriptive rule learning, as there is a target concept but the goal of subgroup discovery is not necessarily to achieve high accuracy, but rather to find subsets of a population whose class distribution is significantly different from the overall distribution. In this way, the target concept helps us to achieve a trade-off between accuracy and interestingness.

In [12] this trade-off was achieved using weighted relative accuracy, but their propositional CN2-SD algorithm is restricted to

two classes. The work of [1] extended the approach to more than two classes by introducing different multi-class search heuristics that take into account the distribution of all the involved classes. The induced multi-class subgroups were also shown to be highly predictive when used as features in a meta-level learner, resulting in considerably smaller models with comparative performance in terms of accuracy and area under ROC curve (AUC). Their work motivated us to investigate multi-class subgroup discovery in a first-order logic framework. Our main contribution is a new weighted covering algorithm that takes advantage of learning from seed examples inspired by Aleph first-order rule learner [19]. In a rigorous experimental evaluation we show that the use of seed examples leads to considerable and statistically significant improvement of predictive power, both in terms of accuracy (3 percent points on average over the data sets) and AUC (7 percent points).

The paper is structured as follows. Section 2 review some relevant work in subgroup discovery, propositional and first-order logic frameworks. In Section 3 we discuss the benefits of first-order multi-class subgroup discovery. Section 4 describes the ingredients of our Aleph-MSD++ algorithm. An empirical evaluation of multi-class subgroup discovery for feature construction over 11 data sets is presented in Section 5. Finally, we discuss possible future work and conclude the paper in Section 6.

2 Previous Work on Subgroup Discovery

Early work on propositional subgroup discovery included Klösgen’s generic pattern discovery system Explora [10, 11], incorporating various search strategies (exhaustive or heuristic), refinement methods and evaluation measures for predictive as well as descriptive learning. PRIM [7] is another system that searches for subgroups by top-down specialisation followed by a bottom-up generalisation on the induced subgroups. PRIM exhaustively examines all possible solutions which makes it unsuitable for large data sets. More recent work includes the CN2-SD rule learner which learns two-class subgroups based on the CN2 inductive rule learner [4]. Where CN2 employed a covering algorithm, removing training examples as soon as they were covered by a rule, CN2-SD demonstrated the advantages of using a weighted covering algorithm, in which example weights were decreased according to the number of rules that covered them. The evaluation measure used was weighted relative accuracy (WRAcc), which compares the precision of a rule with the overall class distribution, weighted by the coverage. CN2-MSD [1] generalised the approach to multi-class subgroup discovery, theoretically and experimentally comparing a range of different subgroup evaluation measures.

Previous work on relational or first-order subgroup discovery in-

¹ University of Bristol, United Kingdom, email: Dawood@cs.bris.ac.uk

² University of Bristol, United Kingdom, email: Peter.Flach@bristol.ac.uk

cludes MIDOS [20], which adapts an evaluation measure from Explora and uses it together with a minimal support criterion when evaluating subgroups. MIDOS uses sampling to estimate these evaluation measures in order to reduce the computational cost of querying multiple tables. Another subgroup discovery study in a relational setting has been conducted in [13]. Their RSD system extracts propositionalised first-order features from a given domain and discovers subgroups amongst them where the binary WRAcc search heuristic and the weighted covering algorithm were used as in CN2-SD. In such a two-stage approach, the quality of the subgroups depends on the quality of the features constructed at the first stage, but RSD constructs these in a non-heuristic manner. In this paper we induce subgroups directly from multi-class relational data and we show that the induced subgroups have higher predictive performance when compared to classification approaches.

3 Benefits of First-Order Multi-class Subgroup Discovery

We argue that the key difference between the induction process in classification systems and subgroup discovery systems lies in the search heuristics. In classification tasks, accuracy-based heuristics are typically employed because the ultimate goal is to have models that achieve high predictive accuracy. However, in subgroup discovery a trade-off between accuracy and interestingness is to be achieved. A single subgroup can cover examples from several classes whereas the conventional classification rules try to avoid covering examples from several classes which could lead to overfitting the training model.

Most ILP induction systems use binary evaluation measures that evaluates the class of the rule being learned against one or more other classes. If the problem consists of two classes only, then it is sufficient to induce rules for the positive class only while a default rule is assigned for the negative class. However, there is a difficulty in dealing with multi-class problems and previous research [6, 8, 9, 17] has investigated methods of multi-class classification that usually decompose a multi-class problem into several binary problems either by considering all pairwise combinations of classes (one-vs-one) or considering each class against the union of the other classes (one-vs-rest). The same approach is typically applied in ILP where the rules of the final model are, in practice, a combination of multiple binary models that sometimes can be obtained by applying the induction process on each one of those binarised problems individually.

The search heuristics of the Multi-class subgroup discovery, Definition 3.1, take the distribution of all classes into account when evaluating a subgroup clause during the induction process and does not evaluate a discovered subgroup in a multi-class problem as in a two-class problem. Unlike classification models, a subgroup discovery model comprises headless rules which can be regarded as boolean features representing interesting and statistically unusual patterns. In propositional subgroup discovery such features can be seen as a transformation of the original propositional feature space into a new propositional complex feature space. However, in first-order subgroup discovery the transformation is done from a complex relational feature space into a new propositional complex feature space.

Definition 3.1 (Multi-class Subgroup Discovery). *A multi-class subgroup discovery task can be defined as a process of finding interesting patterns, called subgroups, describing subsets of the population such that they are sufficiently large and statistically unusual w.r.t. the distribution of all classes.*

The advantages of multi-class subgroup discovery in first-order logic may outperform its advantages in propositional logic and could be much more desirable. This is because the multi-class relational data can be very large and complex, and the needs for techniques to extract interesting and significant features across all relations and over all the classes is indeed crucial. The results of such approach can also produce simple symbolic representation of a complex relational problem and that would make it easier to understand and interpret.

We argue that subgroups can be used to form a classification model when used as binary features for a meta-level learner providing that the subgroups cover sufficient amount of the population and identify a variety of significant features. As a result of using subgroup discovery heuristics and the weighted covering algorithm, the resulting subgroups tend to overlap considerably. If a classification model is desired, we propose the use of decision tree as a meta-level learner as it has the ability, following divide-and-conquer paradigm, to make the best use of the information hidden between the subgroup features and construct a predictive decision tree with leaves representing all possible overlapping and non-overlapping regions across all the binary subgroup features as will be seen in Section 5.

One would expect the presence of overlaps is limited when using classification heuristics together with the covering algorithm. This is because a set of independent rules are derived for each class and also training examples covered by newly induced rules are removed immediately before the induction progresses. Still a subsequent rule may cover some of the removed examples in practice. In theory, if classification rules are used as features for a decision tree meta-level learner, this would result in a smaller tree than a tree of subgroups features applied to the same problem.

4 First-Order Multi-class Subgroup Discovery with Aleph-MSD++

In this section we describe our main contribution, the Aleph-MSD++ first-order multi-class subgroup discovery algorithm. Just as CN2-SD and CN2-MSD were derived from an existing propositional rule learner, we started from an existing inductive logic programming (ILP) learning system. There are many ILP systems, including FOIL [18], PROGOL [14] and GOLEM [15]; they are mostly restricted to binary classification. The starting point for our study was Aleph, a Prolog-based inductive logic programming rule learner that induces first-order classification rule models. Aleph is similar to PROGOL but incorporates additional search strategies and some extra options.

A classification rule in first-order logic is normally represented by a clause consisting of a head literal assigning the class and a body consisting of a conjunction of literals. A subgroup does not assign a class, and hence can be alternatively seen as a conjunction of literals, a headless rule, or a query.

4.1 From Aleph to Aleph-MSD++

Aleph conducts general-to-specific search in the θ -subsumption lattice of a single clause hypothesis. It restricts the search by first computing the Most-Specific-Clause (MSC) or bottom-clause from a single seed example. This technique bounds the search space from below. The mode declarations determine which atoms can be used as body literals when constructing the hypothesis clause.

More specifically, learning a single clause starts by picking a single uncovered seed example and generating its MSC. Next, the generalisation step begins from the empty clause guided by a heuristic

function and restricted to literals from the MSC and the mode declarations. A typical search heuristic in Aleph is the *Compression* that maximises the positive coverage and minimises the negative coverage and the number of literals in a hypothesis clause when refining the hypothesis lattice. The search for a clause stops if the heuristic evaluation score could not be improved further. Each time a search for a clause is done, several clauses are considered and the clause with the highest evaluation score is accepted for the final theory. Once a new clause is accepted, all examples it covers are removed from the training examples due to the use of the covering algorithm. The learning proceeds by picking a new seed example from the remaining training examples and the same procedure is applied until no training examples are left or no clauses can be induced for the remaining training examples.

We present a new algorithm which extends Aleph in a number of ways: by incorporating multi-class subgroup discovery search heuristics; by using a novel weighted covering algorithm; by implementing a significance test to be applied on nominated subgroups; as well as some other, smaller modifications. These extensions are discussed in the following sections.

4.2 An Improved Weighted Covering Algorithm

Most ILP induction systems implement the covering algorithm in the induction process where every time a new rule is induced, the training examples covered by this rule are removed from the training set and the induction as well as the statistical evaluation of new rules is restricted to the remaining training examples. This approach clearly biases the rules induced later in the induction process due to the limitation of examples and background information available with the remaining training examples. The weighted covering algorithm comes as a suitable remedy to this problem. In weighted covering, each example is assigned initial weight, typically a value of 1, and that weight gets reduced every time the example is covered by a new induced rule. Weighted covering was introduced to subgroup discovery in [12].

The weight is decreased according to one of the following two schemata: additive or multiplicative. Let $w_t(x)$ denote the weight of example x after being covered by t number of rules: with respect to the additive weight method we have $w_t(x) = \frac{1}{1+t}$ while in the multiplicative method $w_t(x) = \gamma^t, 0 \leq \gamma \leq 1$. Two special cases are worth noting. If $\gamma = 0$ then this is equivalent to the use of covering algorithm where the examples are removed from training once new induced rule covers them. In contrast, if $\gamma = 1$ then no change happens to the weights of the covered examples which means that all the training examples are kept.

Similarly to CN2-SD and CN2-MSD, the weights of all the examples covered by a newly induced subgroup rule in Aleph-MSD is decreased according to the chosen weighting schema when applying the weighted covering algorithm. In Aleph-MSD++, on the other hand, *only the weights of examples matching the class of the selected seed example are decreased*. Restricting the weight reduction in Aleph-MSD++ is a crucial point in our paper. It ensures that we can learn diverse subgroups representing examples from all the different classes. The importance of this issue is magnified when encountering class imbalance since the minority classes may get ignored during the discovery process.

The advantages conveyed by seed examples are not available when using CN2-based classifiers. We will demonstrate these advantages by comparing the new weighted covering algorithm Aleph-MSD++ against a CN2-like version called Aleph-MSD. In the latter algo-

rithm, the weights of *all* training examples covered by a newly induced clause are reduced – not just the ones from the same class as the seed example. The experimental results in Section 5 show that the CN2-like approach leads to subgroups that are no more predictive than the rules learned by Aleph. In contrast, the use of seed examples in Aleph-MSD++ leads to considerable and statistically significant improvement of predictive power, both in terms of accuracy (3 percent points on average over the data sets) and AUC (7 percent points).

4.3 Multi-class Subgroup Discovery Search Heuristics

We employed four of the six evaluation methods for subgroup discovery proposed in [1].³ Let T be a training set of examples labeled by n classes $C_1, \dots, C_n$. We denote the total number of examples in the training set by E and the number of examples belonging to class C_i by E_i . The number of examples in T covered by a subgroup b is denoted by e , and the number of examples belonging to C_i and covered by b is denoted by e_i .

Definition 4.1 (Weighted Multi-class WRAcc). *The weighted multi-class weighted relative accuracy score of a subgroup b is defined as $WMWRAcc(b) = \frac{1}{E^3} \sum_{i=1}^n E_i |e_i E - e E_i|$.*

Definition 4.2 (Mutual Information). *The mutual information score of a subgroup b is defined as*

$$MI(b) = \sum_{i=1}^n \left(\frac{e_i}{E} \log \frac{e_i}{E} + \frac{E_i - e_i}{E} \log \frac{E_i - e_i}{E} \right) - \sum_{i=1}^n \frac{E_i}{E} \log \frac{E_i}{E} - \frac{e}{E} \log \frac{e}{E} - \frac{E - e}{E} \log \frac{E - e}{E}$$

Definition 4.3 (Gini-split). *The Gini-split score of a subgroup b is defined as $GS(b) = \frac{1}{nE^2} \sum_{i=1}^n \frac{[e_i E - e E_i]^2}{e(E - e)}$.*

Definition 4.4 (Chi-Squared). *The Chi-squared score of a subgroup b is defined as $Chi^2(b) = \sum_{i=1}^n \frac{[e_i E - e E_i]^2}{e E_i (E - e)}$.*

These measures are adapted to take advantage of example weights in the obvious way: e.g., e_i is the total weight of examples of class C_i covered by the subgroup.

4.4 Other Improvements

Other improvements to Aleph implemented in Aleph-MSD++ are:

Selecting Seed Examples To emphasise learning subgroups from various examples and classes, the selection of seed examples is done randomly but only on examples that have never been considered as seed examples before, and that have never been covered by any subgroup induced from a seed example of the same class.

Significance Testing In order to reduce the number of subgroups during the learning and maintaining high quality subgroups at the same time, we apply Chi^2 significance testing at significance level $p = 0.95$. This is important especially when using a weighted covering algorithm since the number of subgroups can be large as the examples covered by previously induced subgroups do not get removed. [3] suggested the use of significance testing to reduce the number of induced rules in their CN2 rule learning system, and it was used in CN2-SD for the same reason.

³ The remaining two were variants of WRAcc that performed equally well or worse in our experiments.

Algorithm 1 Aleph-MSD++

Require: X (Examples), BK (Background Knowledge), MD (Mode Declarations) and $WSat$ (Weight Setting);

```

1: let seed examples  $X_{seed} = X$ ;
2: let subgroup rules  $SR = \{\}$ ;
3: let weights  $W$  be initialised to 1;
4: while  $X_{seed} \neq \{\}$  and  $\exists w_i \in W : w_i = 1.0$  do
5:   let  $x_i = \text{select\_example\_randomly}(X_{seed})$ ;
6:   let  $MSC = \text{construct\_MSC}(x_i, BK, MD)$ ;
7:   let rule  $r = \{\}$  and score  $s = -\infty$ ;
8:   while search limit not exceeded do
9:      $r^{new} = \text{construct\_subgroup\_rule}(MSC, MD)$ ;
10:    ensure  $\forall sr \in SR, sr$  does not  $\theta$ -subsume  $r^{new}$ ;
11:     $[X_{covered}, s^{new}] = \text{evaluate}(r^{new}, X, BK, W)$ ;
12:    if  $\text{significant}(r^{new})$  and  $s^{new} > s$  then
13:       $r = r^{new}$  and  $s = s^{new}$ ;
14:    end if
15:  end while
16:  if  $r \neq \{\}$  then
17:     $SR = SR \cup \{r\}$ ;
18:    for all  $x_j \in X_{covered}$  such that  $\text{class}(x_j) = \text{class}(x_i)$  do
19:       $w_j = \text{recompute\_weight}(w_j, WSat)$ ;
20:       $X_{seed} = X_{seed} \setminus \{x_j\}$ ;
21:    end for
22:  else
23:     $X_{seed} = X_{seed} \setminus \{x_i\}$ ;
24:  end if
25: end while
Output Subgroup Rules  $SR$ ;

```

Rule Construction Aleph-MSD++ rules are constructed based on a heuristic search of combination of literals appear in the MSC as in Aleph. Despite the use of new seed examples in every iteration of the induction process, the same rule can be induced several times due to the use of the weighted covering algorithm; therefore, Aleph-MSD++ algorithm checks whether the new rule is not θ -subsumed by a previously induced rules before accepting it.

Stopping Criteria The learning is terminated according to one of the following conditions: if all examples were covered at least once or if all the remaining uncovered training examples were considered as seed examples even if none of the induced subgroups cover them. This is an advantage over [12] and [1] in the sense that there are no minimum evaluation thresholds imposed in Aleph-MSD++; instead, we ensure that all the examples are at least considered as seed examples in case that they do not get covered by any induced subgroup generated from a seed example of the same class.

Pseudo-code for the Aleph-MSD++ algorithm is given as Algorithm 1. We only show the Pseudo-code for Aleph-MSD++ due to the space limitation but we would like to emphasise that Aleph-MSD is our genuine algorithm as well.

5 Empirical Evaluation

In this section we study the behaviour of the proposed subgroup discovery algorithm and examine its usefulness for feature generation in a classification context. We used the 11 data sets listed in Table 1 with our implementation of Aleph-MSD++ and also its variant Aleph-MSD where the latter uses the standard weighted covering algorithm. For the purpose of evaluating the quality and utility of the induced first-order subgroup rules, we used them as boolean features and applied a decision tree learner (J48, the Weka implementation of C4.5). We also used the classification rules induced by standard Aleph using the Compression search heuristic as features for J48 in order to compare the performance and behaviour of subgroup discovery methods against a typical classification method.

Name	Class distribution
Alzheimer-Amine	1026, 343
Alzheimer-Choline	1026, 343
Alzheimer-Scopolamine	1026, 343
Alzheimer-Toxic	1026, 343
Diterpene	447, 355, 352, 155, 71
English	50, 50, 50
Mutagenesis	125, 63
Protein	116, 115, 77, 73
Car	1210, 384, 69, 65
Ecoli	143, 77, 52
Scale	288, 288, 49

Table 1. Data sets used in the experiments. The first group are first-order data sets (obtained from various sources), the second group are propositional (obtained from UCI [16] repository).

We evaluated the performance of Aleph-MSD++ using the four search heuristics from Section 4.3 and the following six settings of the weighted covering algorithm:

- setting 0: unweighted covering algorithm, $\gamma = 0.00$;
- setting 1: multiplicative weights, $\gamma = 0.25$;
- setting 2: multiplicative weights, $\gamma = 0.50$;
- setting 3: multiplicative weights, $\gamma = 0.75$;
- setting 4: multiplicative weights, $\gamma = 1.00$ (no weight reduction);
- setting 5: additive weights.

For each data set, cross-validated results were recorded for various aspects (accuracy, AUC, size of decision tree, number of leaves, number of induced first-order clauses, number of considered first-order clauses and average size of a induced first-order clauses). Since averaging across data sets has limited meaning as the values may not be commensurate, for each performance aspect we report the average values as well as its average rank (1 is best, 6 is worst). We use the Friedman Test (FT) on these average ranks at $p = 0.10$ with Bonferroni-Dunn post-hoc test to check significance against Compression method as a control learner. The FT records wins and losses in the form of ranks ignoring the magnitude of these wins and losses. FT is considered to be more appropriate when comparing multiple classifiers on multiple data sets. The reader is referred to [5] for more details about the Friedman Test.

Tables 2 and 3 shows the average accuracies and AUCs for Aleph-MSD and Aleph-MSD++ algorithm respectively against the control learner (Compression) when applying J48 on the subgroups and classification rules. Regarding to all the weighted covering algorithm settings (setting 1-5), subgroup discovery methods in Aleph-MSD++ show higher accuracies and significantly better AUCs when compared against Compression (control learner). On the other hand, the same methods reported significantly worse accuracy and comparative AUCs to Compression in Aleph-MSD. It is clear that the use of the new weighted covering (in Aleph-MSD++) outperforms the standard weighted covering (in Aleph-MSD) with regards to both performance aspects.

A graphical illustration of the post-hoc test results on accuracy and AUC of the Aleph-MSD++'s subgroup discovery methods and the control methods is given in Figure 1. For each weight setting the corresponding critical difference diagram is shown vertically (the lower the rank the higher the performance). The figure indicates a superiority of subgroup discovery methods over Compression in almost all weighted covering algorithm settings. This demonstrates the high quality of the subgroups induced in Aleph-MSD++. *GS* achieves the best accuracy in all weighted covering settings except setting 4

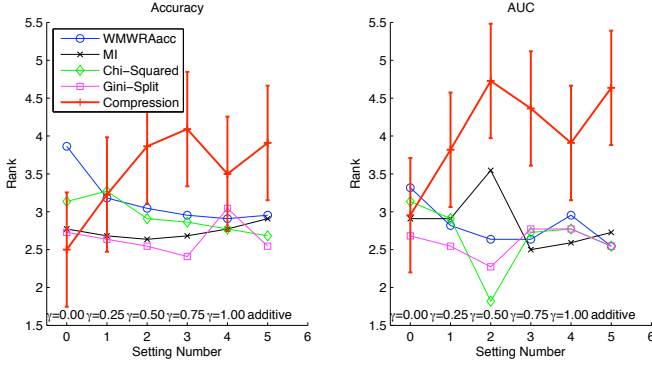


Figure 1. Average ranks achieved by Aleph-MSD++ with regard to accuracy and AUC (lower is better), using different search heuristics over 11 data sets. The left-most setting is the default covering algorithm, while the other settings use different variants of weighted covering. Vertical lines show the critical difference with the control method (Compression): differences outside the critical difference are statistically significant.

($\gamma = 1$) where the weights of covered training examples are not decreased. A striking result on AUC is reported for χ^2 in setting 2 ($\gamma = 0.5$). It deserves to be mentioned that not removing the covered examples nor reducing their weights (setting 4) still achieves better accuracy and significantly better AUC than the control.

Apart from the predictive performance, Aleph-MSD++ in general considers 3 to 4 times fewer hypotheses (Table 5) which contributes to faster building of a clausal theory (with 30 percent fewer number of clauses and 4 times higher coverage on average, results not reported here) when compared against the control method. Among all, the WMWRAcc method was fastest and produced the smallest number of clauses in its models.

The subgroups tend to produce about 25% larger trees, Table 4, when used as features for J48 meta-level learner. This is because the subgroups are highly overlapping due to the use of weighted coverage, and J48 exploits these overlaps to produce superior predictive performance. We should note that the statistical significance of the subgroups, their diversity, high coverage and the overlaps amongst them have all contributed to this significant findings.

6 Conclusion and Future Work

In our previous work [1] we investigated multi-class subgroup discovery in propositional logic framework. In this paper we investigate multi-class subgroup discovery in the inductive logic programming framework, where the induced subgroups are defined by conjunctions in first-order logic. We presented a novel algorithm, Aleph-MSD++, implementing a new weighted covering algorithm based on learning from seed examples and over multi-class domains. We demonstrated the usefulness and the high quality of the subgroups produced by this algorithm and compared it with the standard covering algorithm and also classification rules when used as features for a decision tree learner (J48). At a small increase in the size of the decision tree, we showed that our algorithm is four times faster and produces more accurate predictions in both performance aspects (accuracy and AUC).

If one is to maximise the accuracy, it is recommended to use the combination of setting 2 ($\gamma = 0.5$) and χ^2 heuristic in Aleph-MSD++ while GS is preferred when the AUC is to be max-

imised though it is very close to the other multi-class subgroup discovery heuristics.

In the future we intend to investigate probabilistic multi-class subgroup discovery in both propositional logic and first-order logic. We also would like to exploit more subgroup discovery search heuristics and enhance the discovery of subgroups.

REFERENCES

- [1] Tarek Abudawood and Peter Flach, ‘Evaluation Measures for Multi-class Subgroup Discovery’, in *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD ’09)*, pp. 35–50. Springer-Verlag Berlin Heidelberg, (September 2009).
- [2] Rakesh Agrawal and Ramakrishnan Srikant, ‘Fast Algorithms for Mining Association Rules’, in *Proceedings of the 20th International Conference on Very Large Data Bases*, pp. 487–499, (1994).
- [3] P. Clark and R. Boswell, ‘Rule Induction with CN2: Some Recent Improvements’, in *Proc. 5th European Conference on Machine Learning*, volume 482 of *LNAI*, pp. 151–163. Springer, (1991).
- [4] Peter Clark and Tim Niblett, ‘The CN2 Induction Algorithm’, *Machine Learning*, **3**, 261–283, (1989).
- [5] Janez Demšar, ‘Statistical Comparisons of Classifiers Over Multiple Data Sets’, *Journal of Machine Learning Research*, **7**, 1–30, (2006).
- [6] Jerome H. Friedman, ‘Another approach to polychotomous classification’, Technical report, Stanford University, Department of Statistics, (1996).
- [7] Jerome H. Friedman and Nicholas I. Fisher, ‘Bump Hunting in High-dimensional Data’, *Statistics and Computing*, **9**, 123–143, (1999).
- [8] Chih-Wei Hsu and Chih-Jen Lin, ‘A comparison of methods for multi-class support vector machines’, *Neural Networks*, **13**, 415–425, (March 2002).
- [9] Boonserm Kijsirikul, Nitiwut Ussivakul, and Surapant Meknavin, ‘Adaptive Directed Acyclic Graphs for Multiclass Classification’, in *The 7th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence*, volume LNAI 2417, pp. 158–168. Springer, (2002).
- [10] Willi Klösgen, ‘Subgroup discovery’, in *Handbook of Data Mining and Knowledge Discovery*, eds., Willi Klösgen and Jan M. Zytkow, 354–361, Oxford University Press, (June 2002).
- [11] Willi Klösgen, ‘Explora: A Multipattern and Multistrategy Discovery Assistant’, in *Advances in Knowledge Discovery and Data Mining*, eds., Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, 249–271, MIT Press, (2004).
- [12] Nada Lavrač, Branko Kavšek, Peter Flach, and Ljupco Todorovski, ‘Subgroup Discovery with CN2-SD’, *Journal of Machine Learning Research*, **5**, 153–188, (2004).
- [13] Nada Lavrač, Filip Zelezny, and Peter Flach, ‘RSD: Relational Subgroup Discovery Through First-order Feature Construction’, in *12th International Conference on Inductive Logic Programming*, pp. 149–165. Springer, (2002).
- [14] Stephen Muggleton, ‘Inverse Entailment and Progol’, in *Proceedings of the Sixth International Workshop on Inductive Logic programming*, volume 13, pp. 245–286. Springer, (1995).
- [15] Stephen Muggleton and Cao Feng, ‘Efficient Induction Of Logic Programs’, in *New Generation Computing*. Academic Press, (1990).
- [16] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998.
- [17] John C. Platt and Nello Cristianini, ‘Large margin DAGs for multiclass classification’, in *Advance in Neural Information Processing Systems*, volume 12. MIT Press, (2000).
- [18] J. R. Quinlan and R. M. Cameron-Jones, ‘FOIL: A Midterm Report’, in *Proceedings of the European Conference on Machine Learning*, pp. 3–20. Springer-Verlag, (1993).
- [19] Ashwin Srinivasan, ‘The Aleph Manual’, Technical report, University of Oxford, (2001).
- [20] Stefan Wrobel, ‘An Algorithm for Multi-Relational Discovery of Subgroups’, in *Principles of Data Mining and Knowledge Discovery (PKDD ’97)*, pp. 78 – 87. Springer Verlag Berlin, (1997).

Appendix: Detailed Experimental Results

setting	algorithm	<i>WMWR</i> Acc	<i>MI</i>	<i>Chi</i> ²	<i>GS</i>	<i>Comp</i>
$\gamma = 0.00$	Standard Covering	76.99 (3.86)	77.74 (2.77)	77.38 (3.14)	77.49 (2.73)	77.76 (2.50)
$\gamma = 0.25$	Aleph-MSD++	80.07 (3.18)	79.66 (2.68)	79.77 (3.27)	79.97 (2.64)	77.76 (3.23)
	Aleph-MSD	76.03 (3.45)	74.80 (4.18)	76.46 (2.82)	77.79 (2.27)	77.76 (2.27)
$\gamma = 0.50$	Aleph-MSD++	80.50 (3.05)	80.52 (2.64)	80.48 (2.91)	80.45 (2.55)	77.76 (3.86)
	Aleph-MSD	75.87 (3.00)	75.69 (3.41)	76.69 (2.73)	75.71 (3.18)	77.76 (2.68)
$\gamma = 0.75$	Aleph-MSD++	80.66 (2.95)	80.95 (2.68)	80.85 (2.86)	80.75 (2.41)	77.76 (4.09)
	Aleph-MSD	74.73 (3.68)	75.28 (2.91)	75.43 (3.05)	76.71 (3.00)	77.76 (2.36)
$\gamma = 1.00$	Aleph-MSD++/MSD	79.71 (2.91)	79.73 (2.77)	79.84 (2.77)	79.78 (3.05)	77.76 (3.50)
additive	Aleph-MSD++	80.62 (2.95)	80.32 (2.91)	80.75 (2.68)	80.62 (2.55)	77.76 (3.91)
	Aleph-MSD	74.53 (4.18)	76.90 (2.41)	76.13 (3.00)	76.36 (3.05)	77.76 (2.36)

Table 2. Comparison of average accuracies (ranks in brackets) when using subgroups of Aleph-MSD++ and Aleph-MSD (columns 3-6) against classification rules (column 7) as features for J48.

setting	algorithm	<i>WMWR</i> Acc	<i>MI</i>	<i>Chi</i> ²	<i>GS</i>	<i>Comp</i>
$\gamma = 0.00$	Standard Covering	0.80 (3.32)	0.81 (2.91)	0.81 (3.14)	0.81 (2.68)	0.79 (2.95)
$\gamma = 0.25$	Aleph-MSD++	0.85 (2.82)	0.84 (2.91)	0.84 (2.91)	0.85 (2.55)	0.79 (3.82)
	Aleph-MSD	0.80 (3.55)	0.79 (3.82)	0.81 (2.55)	0.81 (2.55)	0.79 (2.55)
$\gamma = 0.50$	Aleph-MSD++	0.86 (2.64)	0.85 (3.55)	0.86 (1.82)	0.86 (2.27)	0.79 (4.73)
	Aleph-MSD	0.79 (3.41)	0.80 (2.64)	0.80 (2.91)	0.79 (3.32)	0.79 (2.73)
$\gamma = 0.75$	Aleph-MSD++	0.86 (2.64)	0.86 (2.50)	0.86 (2.73)	0.86 (2.77)	0.79 (4.36)
	Aleph-MSD	0.80 (3.23)	0.80 (2.86)	0.80 (3.27)	0.80 (2.82)	0.79 (2.82)
$\gamma = 1.00$	Aleph-MSD++/MSD	0.85 (2.95)	0.85 (2.59)	0.85 (2.77)	0.85 (2.77)	0.79 (3.91)
additive	Aleph-MSD++	0.86 (2.55)	0.86 (2.73)	0.86 (2.55)	0.86 (2.55)	0.79 (4.64)
	Aleph-MSD	0.79 (4.09)	0.81 (2.55)	0.80 (2.82)	0.80 (2.73)	0.79 (2.82)

Table 3. Comparison of average AUCs (ranks in brackets) when using subgroups of Aleph-MSD++ and Aleph-MSD (columns 3-6) against classification rules (column 7) as features for J48.

setting	<i>WMWR</i> Acc	<i>MI</i>	<i>Chi</i> ²	<i>GS</i>	<i>Comp</i>
$\gamma = 0.00$	27.91 (3.05)	21.20 (2.91)	27.58 (2.68)	27.51 (2.59)	31.69 (3.77)
$\gamma = 0.25$	34.60 (3.41)	32.91 (2.36)	33.13 (2.73)	34.84 (3.32)	31.69 (3.18)
$\gamma = 0.50$	39.98 (3.00)	41.44 (2.77)	41.62 (3.45)	41.16 (3.50)	31.69 (2.27)
$\gamma = 0.75$	42.56 (3.05)	45.56 (3.55)	45.07 (3.68)	41.76 (2.64)	31.69 (2.09)
$\gamma = 1.00$	44.91 (3.50)	42.51 (3.50)	43.20 (2.86)	42.95 (2.95)	31.69 (2.18)
additive	40.53 (2.86)	42.22 (3.27)	42.09 (3.32)	40.73 (3.18)	31.69 (2.36)

Table 4. Average tree size (ranks in brackets) when using Aleph-MSD++'s subgroups (columns 2-5) and classification rules (column 6) as features for J48.

setting	<i>WMWR</i> Acc	<i>MI</i>	<i>Chi</i> ²	<i>GS</i>	<i>Comp</i>
$\gamma = 0.00$	42,056 (1.86)	42,538 (2.45)	50,896 (3.32)	50,940 (3.14)	138,949 (4.23)
$\gamma = 0.25$	40,713 (2.55)	49,782 (3.27)	43,048 (2.55)	41,162 (2.64)	138,949 (4.00)
$\gamma = 0.50$	32,661 (1.91)	34,144 (2.64)	35,103 (2.82)	34,299 (3.55)	138,949 (4.09)
$\gamma = 0.75$	30,721 (1.09)	33,009 (3.55)	32,457 (2.91)	33,272 (3.09)	138,949 (4.36)
$\gamma = 1.00$	33,242 (1.77)	31,163 (3.00)	33,460 (2.95)	33,466 (3.05)	138,949 (4.23)
additive	32,385 (1.91)	32,908 (2.82)	33,576 (3.36)	33,387 (2.55)	138,949 (4.36)

Table 5. Average number of considered first-order subgroups in Aleph-MSD++ and classification rules (Comp) in Aleph.