JMLR: Workshop and Conference Proceedings 13: 177-192
2nd Asian Conference on Machine Learning (ACML2010), Tokyo, Japan, Nov. 8–10, 2010.

Exploiting the High Predictive Power of Multi-class Subgroups

Tarek Abudawood DAWOOD@CS.BRIS.AC.UK
Computer Science Department
University of Bristol

Peter Flach PETER.FLACH@BRISTOL.AC.UK
Computer Science Department
University of Bristol

Editor: Masashi Sugiyama and Qiang Yang

Abstract

Subgroup discovery aims at finding subsets of a population whose class distribution is significantly different from the overall distribution. A number of multi-class subgroup discovery methods has been previously investigated, proposed and implemented in the CN2-MSD system. When a decision tree learner was applied using the induced subgroups as features, it led to the construction of accurate and compact predictive models, demonstrating the usefulness of the subgroups. In this paper we show that, given a significant, sufficient and diverse set of subgroups, no further learning phase is required to build a good predictive model. Our systematic study bridges the gap between rule learning and decision tree modelling by proposing a method which uses the training information associated with the subgroups to form a simple tree-based probability estimator and ranker, `RankFree-MSD`, without the need for an additional learning phase. Furthermore, we propose an efficient subgroup pruning algorithm, `RankFree-Pruning`, that prunes unimportant subgroups from the subgroup tree in order to reduce the number of subgroups and the size of the tree without decreasing predictive performance. Despite the simplicity of our approach we experimentally show that its predictive performance in general is comparable to other decision tree and rule learners over 10 multi-class UCI data sets.

Keywords: Subgroup Discovery, Classification, Probability Estimation, Ranking, Supervised Learning

1. Introduction

Subgroup discovery is an induction task that can be categorised under the umbrella of rule induction, which is a common form of machine learning and data mining often used in classification and association rule learning. Classification rule learning is a predictive task aimed at constructing a set of highly accurate rules, based on pre-labeled training instances and their observed attributes, to predict a target concept (class label) of unseen future instances. Association rule learning, on the other hand, is a form of descriptive induction aimed at the discovery of individual rules that express interesting patterns in data where a target concept is undefined. Subgroup discovery can be seen as being halfway between predictive and descriptive rule learning, as they are applied to labeled data, but the goal of subgroup discovery is not necessarily to achieve highly accurate rules. Rather, the target concept helps us to achieve a trade-off between accuracy and interestingness.

In recent years subgroup discovery has gained increased attention in machine learning community. While most of the research focused on the use of subgroup discovery to improve classification

©2010 Tarek Abudawood and Peter Flach.

accuracy, in this paper we investigate the ability to extend subgroup discovery methods to perform probability estimation and ranking in multi-class domains. This is because in many applications it is hard to provide a crisp (discrete) prediction whether each instance belongs to a particular class or not. Nevertheless, it might be more sensible to provide a probability membership score to show the expected likelihood of an instance to be of a particular class. Moreover, it could be desirable to rank the total instances globally according to their estimated membership probabilities on a particular class. A probability estimator or a ranker perform more general tasks than classification and can be adapted easily to perform classification tasks by setting a threshold over the probabilities or ranking scores.

The outline of the paper is as follows. We start by presenting our motivation and discussing related work in Section 2, followed by some formal definitions in Section 3. We then propose our tree-based probability estimator and ranker in Section 4 and a suitable pruning algorithm in Section 5, where the former uses the induced subgroups to construct a ranking tree while the latter prunes the tree for the purpose of eliminating the redundant subgroups and reducing the complexity of the tree. We support our argument by conducting an extensive experimental evaluation in Section 6, and show that we can achieve high predictive performance by just using the training information obtained when inducing subgroups without any further learning. We conclude and discuss some possible future extensions in Section 7.

2. Motivation and Related Work

Previous research (Boström, 2007; Fawcett, 2001; Sulzmann and Fürnkranz, 2009) has investigated probability estimation and ranking methods for multi-class classification rules in general but not specifically targeted towards subgroup discovery. If the subgroups are evaluated and induced in a binary fashion (e.g. evaluated on a positive class against a negative class), it is possible to apply those probability estimation and ranking techniques. However, this one-versus-rest technique ignores interaction between classes and in general results in less compact models. `CN2` (Clark and Niblett, 1989) and IREP (Fürnkranz and Widmer, 1994) are examples of such classification rule learners while `CN2-SD` (Lavrač et al., 2004) is subgroup discovery learner that can be seen as an upgrade of the former.

Classification rule learners typically learn ordered rule lists or unordered rule sets. In the former case, the rules are listed in the final model such that the most predictive rule will come first and least predictive rule comes last. The latter model can be seen as a collection of multiple sets of rules where each set is induced for one particular class independently of the others with no particular order. Apart from the fact that rules that are induced later have access to fewer training instances due to the use of the covering algorithm, the search heuristics typically maximise two-class accuracy-related measures.

If the rules were mutually exclusive (i.e. cover disjoint subsets of instances) then it is straightforward to calculate the probabilities and rank the examples in a similar way as with decision trees. Unfortunately this is not generally the case as rules that come later in the covering approach do not have access to all the training instances, rather they have access only to the instances not covered by any rule induced earlier. As a consequence, multiple rules predicting different classes may overlap. Rule list models have the advantage of avoiding rule overlaps imposing a simple conflict resolution model (the first rule that fires decides the class of the instance) and therefore it is straightforward to obtain a ranking. In contrast, rule set models require resolving such conflicts during the classi-

fication phase (Fawcett, 2001). We argue that the advantage of using rule sets lies in the fact that they carry more important information regarding the overlaps which makes them more suitable for probability estimation and ranking tasks. It is possible to come up with a criterion to impose an ordering on rule sets in a post-learning process and deal with them similar to rule lists. However; the work of Wilkins and Ma (1994) has proved that optimal selection of rules is an NP-hard problem due to *sociopathic* interactions between rules and therefore it is not feasible to optimally prioritise rule sets and order them except heuristically (Fawcett, 2001).

Recently multi-class heuristic measures have been studied and proposed by Abudawood and Flach (2009) to guide the search for finding interesting subgroups in the existence of multiple classes. Each candidate subgroup considered during the learning is evaluated over all the classes and assigned a single score reflecting the degree of significance and divergence of instances it covers. Unlike other work (Boström, 2007; Fawcett, 2001; Sulzmann and Fürnkranz, 2009), those scores are proper multi-class scores and result in increased overlap between the subgroups, especially in the presence of the weighted covering algorithm which reduces initial weights of instances instead of removing them (Lavrač et al., 2004). Therefore Abudawood and Flach (2009) used the subgroups as binary features for a decision tree meta-level learner (double induction). The decision tree learner was able to handle the rule overlaps efficiently and use them to build highly accurate predictive decision tree models.

Although decision tree learning is more conservative than rule learning by searching a smaller hypothesis space in a divide-and-conquer approach as indicated by Bostrom (1995), it has the interesting property of being able to handle multi-class distributions by applying discriminative heuristic measures, e.g. information gain and entropy. Moreover, it does not run into overlapping problems due to the nature of the trees as the set of leaves forms disjoint partitions over the instances. Despite that, decision trees have been criticised for yielding poor probability estimates (Provost and Domingos, 2003; Glöckner, 2009), and therefore poor ranking performance. Several research suggested various method of improving their probability estimates, e.g. Laplace or m-estimate smoothing and avoiding pruning. As pointed out by Glöckner (2009), if decision tree learners have access to prior information expressing importance of attributes, that would lead to a construction of more efficient decision trees and therefore better ranking performance, but this is not the case in common decision tree learners. LexRank (Flach and Matsubara, 2007) tries to learn such information on the attributes such that it can rank positive and negative instances lexicographically accordingly. As it stands, several multi-class heuristics investigated by Abudawood and Flach (2009), e.g. *Chi-squared* and *Gini-Split*, provide such prior information exemplified by the subgroup scores which can be ordered monotonically expressing the importance of the subgroups such that the subgroup with the highest score comes first and the subgroup with the lowest score comes last.

The above discussion has motivated our research of conducting a further investigation to make the most of the multi-class subgroup discovery heuristic scores as well as the coverage information obtained during the learning process to construct a tree-based probability estimator and ranker. We emphasise that the use of a tree representation has several advantages for multi-class subgroup discovery. First, there is no need to learn the class of a subgroup as a leaf of the tree would handle the instance predictions probabilistically. Secondly, a tree is naturally a collision-free model where the leaves hold disjoint set of instances and in each leaf all its instances have the same probability and consequently the same rank. Thirdly, all information required to build a tree is obtained during the training phase when subgroups are induced. Fourthly, the probabilities obtained in our context are in fact empirical probabilities, derived from training instances, which are calibrated by nature

when it comes to compute membership probabilities or rank the instances adding extra reliability to our approach.

3. Multi-class Subgroup Discovery

Subgroup discovery is similar to classification rule learning as it is applied to labeled data and employs the covering algorithm in a general-to-specific fashion over the hypothesis space. The fundamental difference between the two is the use of different search heuristics that reflect their different goals. In classification rule learning, the search heuristic is usually some form of accuracy or precision. In contrast, subgroup discovery heuristics typically apply a trade-off between accuracy and interestingness. The latter can be exemplified by the *weighted relative accuracy* heuristic defined as $WRAcc_k = \frac{e_k}{E} - \frac{e}{E}\frac{E_k}{E}$ where E is the total number of training instances, e is the number of instances covered by a hypothesis subgroup, E_k is the total number of instances of the kth class and e_k is the number of instances belonging to class k and covered by the subgroup.

Conventional rule learners and subgroup discovery search heuristics can handle two classes only and thus an induced rule maximises the class designated as the positive class. However, multi-class subgroup discovery learners use multi-class search heuristics which enable it to discover interesting subgroups that maximises divergence and interestingness over n multiple classes. *Chi-squared* and *multi-class weighted relative accuracy*

$$Chi^2 = \sum_{i=1}^{n} \frac{[e_i E - eE_i]^2}{eE_i(E-e)} \qquad MWRAcc = \frac{1}{nE^2}\sum_{i=1}^{n} |e_i E - eE_i|$$

are two examples of such multi-class heuristics (Abudawood and Flach, 2009).

We can define the multi-class subgroup discovery task in propositional logic informally as the process of finding interesting conjunctions of attribute-values, called *subgroups*, describing subsets, possibly with overlaps, of the population such that they are sufficiently large and statistically unusual w.r.t. the distribution of all classes. The formal definition of the multi-class subgroup discovery task is given in Definition 1.

Definition 1 (Multi-class Subgroup Discovery) *Let $\mathbb{X}$ be an instance space over $f+1$ dimensional attributes $a_1,\ldots,a_{f+1}$ such that each instance $x_i \in \mathbb{X}$ is described by a vector of attribute-values ($a_1 = val \in A_1,\ldots,a_f = val \in A_f, a_{f+1} = val \in C = \{c_1,\ldots,c_n\}$) where the first f dimensions characterise the instances by binary, nominal and/or numerical attributes, the last attribute categorises each instance into one of a given $n \geq 2$ classes, A_z is the set of all possible values accepted by the z-th attribute and $1 \leq z \leq f$. Let Ψ be a multi-class subgroup heuristic function, e.g. Chi^2, that measures the interestingness of a subgroup over the n classes. Then a multi-class subgroup discovery task is a mapping $S : \mathbb{X} \xrightarrow{\Psi} SG$ such that instances are characterised by a set of subgroup (headless)[1] rules where each rule $h \in SG$ of the form ($\square \leftarrow literal_1 \wedge \cdots \wedge literal_u$) and each literal $literal_w, 1 \leq w \leq u$ is formed by $a_z \ op \ val \in A_z$ where $op \in \{>,=,<\}$. The selection of the conjunction of the literals in the body of a subgroup rule is done heuristically and iteratively starting from the empty body such that in each iteration a literal that maximises the chosen heuristic function Ψ in conjunction with current selected literals is added to the body.*

1. Since a multi-class subgroup discovery rule does not predict a class, unlike a classification rule, the head is empty.

A single multi-class subgroup (or subgroup rule) can be interpreted as a headless rule with a conjunction of one or more attribute-values in the body. A subgroup set is a collection of multiple subgroups without any particular order.

It is important to point out that although feature construction methods might appear to behave similarly to subgroup discovery methods, the two address different tasks. This is because the former is done prior to learning or as part of a learning task generating large number of candidate features among which a selection must be made during the learning process. In contrast, the latter integrates feature construction with learning and selection as part of the same process. The subgroups can be thought of as a summary of the interesting and useful information found in the original instance space $\mathbb{X}$.

There is a limited number of systems that can learn multi-class subgroups. Such systems includes `CN2-MSD`, which learns subgroups from propositional machine learning data sets, Explora, (Klösgen, 2004, 2002), and PRIM (Friedman and Fisher, 1999), both of which learn subgroups from a database. `CN2-MSD` is more geared toward machine learning application while Explora and PRIM are generally aimed at data mining and knowledge discovery applications. Therefore `CN2-MSD` is a natural choice in the context of our study. Abudawood and Flach (2009) investigated the ability of generating highly useful and interesting subgroups and suggested six different heuristic measures for this task. Our focus is to uncover the predictive power of the multi-class subgroups by extending it to perform further tasks that include classification, probability estimation and ranking. We do this by just using the useful information acquired during subgroup discovery induction in a simple, systematic and sound approach as well be seen in the next section.

4. Multi-class Subgroup Ranker and Probability Estimator

Let us start by drawing a distinction between classification, probability estimation and ranking tasks. A classification task is a mapping $Q : \mathbb{X} \rightarrow C$ such that a single class $\hat{c}_i \in C$ is predicted for each instance $x_i \in \mathbb{X}$. A probability estimation task assigns a class membership probability distribution $\{\hat{p}(c_k|x_i)|0 \leq \hat{p}(c_k|x_i) \leq 1\}$ to each instance. On the other hand ranking is the mapping $R : \mathbb{X} \times \mathbb{X} \rightarrow \{\succ_{c_k}, =_{c_k}, \prec_{c_k}\}$, forming a total order of instances, possibly with ties, with respect to a particular class c_k. If we have a good probability estimator then it becomes a straightforward task to come up with a probabilistic or scoring ranker. This is because the probabilities can be used to rank the instances with regards to the class of interest.

Given that the multi-class subgroups hold significant, interesting and important information the question is how one could extend the use of multi-class subgroups for classification, ranking or probability estimation tasks without undertaking additional learning. One major issue in this context is dealing with subgroup overlaps. This is because overlaps between the multi-class subgroups are usually much larger than with classification rules due to the use of heuristics promoting overlaps. 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. Therefore, subgroups are usually more generic than classification rules and they tend to cover larger subsets of examples and from several classes. In contrast, the conventional classification rules try to avoid covering examples from several classes which could lead to overfitting the learning model by learning too many specific rules that characterises each class.

Since subgroups can be seen as binary features, a simple intuitive solution to extend the subgroup discovery to perform classification, probability estimation and ranking can be achieved by representing the subgroups in a binary tree. Assume that we have a set of multi-class subgroups (which can be seen as binary features over the instances) and a multi-class scoring function ω then one could consider prioritising the subgroups to reflect their importance according to Definition 2. Consequently, a simple binary tree can be built using the subgroups with respect to their importance as defined in Definition 3.

Definition 2 (Subgroup Importance) *Let $\omega : SG \rightarrow \mathbb{R}$ be a scoring function that assigns numerical scores to subgroups such that higher scores indicate more significant subgroups, then the subgroups can be ordered with respect to their scores*[2]*. A subgroup $h \in SG$ is said to be more important than a subgroup $g \in SG$ iff $\omega(h) > \omega(g)$*[3]*.*

Definition 3 (Subgroup Tree) *Given a list of $|SG|$ subgroups such that subgroups are ordered according to their importance $SG \times SG \rightarrow \{\succ_\omega, =_\omega, \prec_\omega\}$, then a subgroup tree is a binary tree constructed by iteratively splitting nodes at the deepest level (level l) of the tree, starting from the root ($l = 0$) of the tree, based on the applicability of the body condition*[4] *($literal_1 \wedge \cdots \wedge literal_u$) of the l-th subgroup on the instances of its parent node. The instances where the subgroup is applicable for go to a left split and the remaining instances go to a right split of a considered node.*

The idea of representing subgroups in a binary tree allows one to express and use the multi-class subgroup discovery similarly to a decision tree classifier, probability estimator and ranker exploiting the various overlaps in between the subgroups and model these overlaps into a tree. While subgroups are learned using covering approach (separate-and-conquer) which allow searching for larger hypothesis space, they are represented in a tree (divide-and-conquer) were the tree models all possible partitions (overlapping and non-overlapping) amongst subgroups over the instances. Every complete path from the root to a leaf in the tree represents a unique subset of instances not appearing into other paths allowing a simple, yet significant, multi-class classification, probability estimation and ranking to take place at the tips of the subgroup tree.

The main result of this paper is a single method that performs subgroup discovery, classification, probability estimation and ranking with little overhead. Once we learn a multi-class subgroup set, we use them to build a tree without any additional learning. The training information associated with the subgroups illustrated by the coverage over the instances and heuristic score is used to build the tree recursively level by level. To this purpose we extend the `CN2-MSD` algorithm to the `RankFree-MSD` algorithm (Algorithm 1). Since a multi-class heuristic score is assigned to each subgroup of the induced subgroup set, the algorithm prioritises the subgroups based on their importance in a descending order forming an ordered list of subgroups. Having all training instances and their labels, the algorithm starts building the binary-split tree by adding the first subgroup (with the highest score). Such process could split the instances into a left child, instances covered by the subgroup, and a right child, instances not covered by the subgroup. This process continues recursively by adding the next subgroup in the list to each node in the current level. The process of

2. When ordering subgroups the ties are broken randomly.
3. The scoring function ω can be the same as the subgroup heuristic function Ψ as it is the case in the context of this paper but in other circumstances the are not necessarily the same.
4. A body condition of a rule is applicable for an instance if all literals in the body are executed for that instance.

splitting a node is terminated if all subgroups in the list were considered, a node is pure [5] or a node is empty.

The *empirical probability* of an instance $x_i \in \mathbb{X}$ of class $c_k \in C$ that falls into the yth leaf $x_i \in L_y \subset \mathbb{X}$ of the tree is the percentage of training instances belonging to c_k in that particular leaf, hence, $p(c_k|x_i \in L_y) = \frac{|L_y \cap \mathbb{X}_k|}{|L_y|}$ where $\mathbb{X}_k$ is the subset of training instances of class c_k. Since these empirical probabilities are obtained from the training data, they are well suited for probability estimation and ranking. We measure the ranking performance by the area under the ROC curve (AUC) using a separate test set. The AUC is a two-class measure the evaluates the ability of a classifier to rank the positive instances before the negatives. A multi-class AUC is obtained by averaging all one-vs-rest AUCs weighted by the class prior: $AUC_{avg}^{OVR} = \sum_{k=1}^{n} auc(c_k) \cdot \frac{E_k}{E}$.[6] Detailed algorithms for calculating the AUC can be found in (Fawcett, 2004, 2006). Pseudo-code for the `RankFree-MSD` algorithm is given in Algorithm 1.

5. `RankFree-Pruning`

Since the subgroups in the subgroup tree are ordered globally with respect to their importance (Definition 2), we assume that a subgroup at level $l-1$ of the tree is more significant or at least equally significant to a subgroup at level l. In order to investigate the feasibility of improving the tree and its ranking performance we developed a pruning algorithm, `RankFree-Pruning`, that aims at reducing the number of subgroups and consequently the size of the subgroup tree without decreasing the ranking performance of the tree. The scientific motivation lies in the fact that the existence of a subgroup at the max level v of the tree may be found redundant with respect to the existence of all its ancestor subgroups at levels 1 to $v-1$. If a subgroup was found redundant, it can then be removed from the tree. We define subgroup redundancy formally as follows.

Definition 4 (Subgroup Redundancy) *Given a subgroup tree of at least v levels, a subgroup g at level v is considered redundant w.r.t. all ancestor subgroups iff the ranking performance at level v, auc_v, is less than or equal to the ranking performance at level $v-1$, auc_{v-1}. (auc_v and auc_{v-1} denote AUC_{avg}^{OVR} of the subgroup tree at depth v with subgroup g and at $v-1$ without the subgroup g respectively.)*

However, if the probabilities are empirical, training set AUC will never decrease by pruning because of the monotonic property of the AUC. Given a tree of v levels, $auc_{v-1} \leq auc_v$ iff the probabilities are empirical. In such a situation, a subgroup g at level v can be redundant only if the probability distribution at each leaf of level v does not differ from its parent node or a leaf is empty. In such cases, the AUC will not change if all leaves at level v are removed.

To illustrate the monotonic increase of AUC from a parent node to its children nodes we consider the following example.

Example 1 (Example of a Monotonic Increasing AUC of Decision Trees) *Let us start having an empty tree (level 0) for a two-class problem, then $auc_0 = 0.5$ represents a random ranking performance where a diagonal line is drawn from the trivial point point $(0,0)$ to the trivial point point $(1,1)$*

5. A pure node is a node that contains instances of a single class.
6. Alternatively, it is possible to compute the AUC in a one-vs-one fashion: $AUC_{avg}^{OVO} = \frac{2}{n(n-1)} \sum_{k_1=1}^{n} \sum_{k_2=1, k_2 \neq k_1}^{n} auc(c_{k_1}, c_{k_2})$.

Algorithm 1 `RankFree-MSD` and `Pruned-RankFree-MSD` algorithms (An extension to `CN2-MSD` algorithm)

Require: instances, subgroups, their heuristic scores, their coverage over the instances and pruneFlag;
let *tree* be an empty binary tree; //a global variable holding the current tree
let $v = 0$ be represent the current tree level; //a global variable holding the current depth of the tree
procedure RANKFREE-MSD
  BuildSubgroupTree(); //RankFree-MSD
  if *pruneFlag* is true **then**
    RankFree-Pruning (); //pruning the tree for Pruned-RankFree-MSD
  end if
Output *tree*;
end procedure
procedure BUILDSUBGROUPTREE
  sort subgroups on scores descending;
  assign $node_{v,1}$ as the root of the tree (level 0, node 1) and let it holds the set of all training instances ;
  for $l = 0$ to max #subgroups **do**
    for $m = 1$ to max #nodes at level l **do**
      if $node_{l,m}$ contains instances of same class or empty **then**
        do not split $node_{l,m}$;
      else
        split $node_{l,m}$ to $node^{true}_{l+1,m}$ and $node^{false}_{l+1,m}$ based on the coverage of subgroup no. $l+1$ over the instances;
        link $node^{true}_{l+1,m}$ and $node^{false}_{l+1,m}$ to $node_{l,m}$;
        if $v \leq l$ **then**
          $v = l + 1$;
        end if
      end if
    end for
    if $l = v$ **then**
      end procedure; //stop if no split has been made
    end if
  end for
end procedure
procedure RANKFREE-PRUNING
  let $tempTree = tree$;
  $tempTree = removeLastSubgroup(tempTree)$; //remove all nodes at the last level (v) of $tempTree$
  let $old\_AUC = auc(tree)$; //AUC at level v
  let $new\_AUC = auc(tempTree)$; //AUC at level $v-1$
  while $new\_AUC \geq old\_AUC$ and $v > 1$ **do**
    $v = v - 1$;
    $tree = tempTree$;
    $tempTree = removeLastSubgroup(tempTree)$;
    let $old\_AUC = new\_AUC$;
    let $new\_AUC = auc(tempTree)$;
  end while
end procedure

forming a single segment in ROC space. Any further split at level 1 *will cause the original segment to be split into two new segments. The probabilities of the new splits will help us to achieve an optimal ordering amongst them such that they end up having a new ROC curve with at least the same ranking performance as the diagonal, and so* $auc_1 \geq 0.5$.

Nevertheless the monotonic property can be violated, if the sum of the distributions at the children leaves is not equivalent to the distribution of their parent node, consequently the AUC could be increased by pruning. This could be the case if for instance the Laplace smoothing or m-estimate in general is used and thus auc_{v-1} can not be guaranteed to be $\leq auc_v$. The following example illustrates this point.

Example 2 (Example of Violating the Monotonically Increasing AUC of Decision Trees) *Let us consider drawing a ROC curve for a two-class problem of 2 positive instances and 7 negative instances. Before having any split, empty tree,* $auc_0 = 0.5$ *representing a diagonal ROC curve. Now let us have a split of 2 positives and 6 negatives (left child) and the remaining, 0 positives and 1 negatives respectively (right child). Considering that we are interested on the positive class then the probability of the left child is* $\frac{2}{8} = 0.25$ *and the right child is* $\frac{0}{1} = 0.0$*. Therefore we have an* $auc_1 = 0.56$ *and that means* $auc_0 \leq auc_1$*. If, however, the Laplace smoothing is applied then* $\frac{(0)+1}{(1)+2} = 0.33$ *and* $\frac{(2)+1}{(8)+2} = 0.30$ *for the left and right child respectively. Consequently* $auc_1 = 0.44$ *which means* $auc_0 \nleq auc_1$ *violating the monotonic property.*

Interestingly, the Laplace smoothing would help in pruning small leaves[7] by violating this monotonic property and therefore it can be considered as a good and computationally cheap trick for avoiding overfitting. At the same time, the Laplace correction does not have much of an effect on large leaves. Hence, we employ the Laplace correction when calculating the probabilities in our forthcoming experiments. Pseudo-code for the `RankFree-Pruning` algorithm is demonstrated by the third procedure in Algorithm 1. The algorithm will recursively remove a single subgroup, prunes all its leaf nodes, from the bottom of the subgroup tree as long as the AUC does not get decreased.

6. Empirical Evaluation

	Dataset Name	#exs.	# attrs.	# cls.	Dist.
1	Balance-scale	624	5	3	288, 288 and 48
2	Car	1727	7	4	1209, 384, 69 and 65
3	Contraceptive Method Choice	1472	10	3	628, 511 and 333
4	Cover-type	5807	55	7	2833, 2118, 357, 205, 173, 94 and 27
5	Dermatology	365	35	6	112,72, 60, 52, 49 and 20
6	Ecoli	335	9	8	142, 77, 52, 35, 20, 5, 2 and 2
7	Iris	150	5	3	50, 50 and 50
8	Nursery	6314	9	3	2159, 2133 and 2022
9	Waveform	4998	22	3	1695, 1657 and 1646
10	Yeast	1483	10	10	463, 429, 243, 163, 51, 44, 35, 30, 20 and 5

Table 1: UCI data sets used for the experiments.

6.1 Experiments Setup

We performed extensive experiments to test the performance of `RankFree-MSD`, where the multi-class subgroups are used to build a simple and straightforward tree-based ranker and probability estimator with and without applying `RankFree-Pruning` algorithm. We compare our methods against two tree learners and four rule learners in two separate experiments. The tree learners: REPTree (Reduced-Error-Pruning Tree); and J48 (C4.5 Decision Tree). The rule learners: PART (rules of partial trees); RIDOR (RIple DOwn Rules); `CN2` unordered; and `CN2-SD` unordered (with multiplicative weight of $\lambda = 0.5$ and minimum significance level at 0.95). We made sure to disable the pruning for tree learners when it was possible so that the AUCs do not get affected. We used

7. We mean by a small leaf, a leaf with a few number of instances.

the 10 UCI multi-class data sets (Newman et al., 1998) listed in Table 1. Numerical attributes with more than 100 distinct values have been discretised[8].

REPTree, J48, PART and Ridor are implemented under Weka data mining workbench (Witten and Frank, 2005). Our `RankFree-MSD` is an adaptation of `CN2-MSD` which was in turn an upgrade of the Java implementation of `CN2-SD`/`CN2` provided by Lavrač et al. (2004). In our experiment we chose one of the best two combinations of parameter settings suggested by Abudawood and Flach (2009). These parameters are: minimum significance level at 0.95; and *Chi-squared* heuristic evaluation function was chosen in combination with multiplicative weight of $\lambda = 0.5$, see (Abudawood and Flach, 2009) for more details. In `RankFree-MSD` extra parameters such as construction of a global subgroup tree, ordered subgroups, and Laplace corrected probabilities were enabled leading to the default `RankFree-MSD` version. Additionally, `RankFree-Pruning` algorithm was enabled to construct its pruned variant, `Pruned-RankFree-MSD`.

We report 10-fold cross-validated results with respect to AUC, Brier score (average squared deviation between the predicted probabilities and ideal 0-1 probabilities (Brier, 1950)) and its decomposition in terms of refinement losses and calibration losses.[9] Furthermore, we report the average rank (when comparing rules 1 is best and 6 is worst, when comparing trees 1 is best and 4 is worst) of each method across all data sets. This is because averaging performance scores across data sets has limited meaning as these scores are not necessarily commensurate. We use the Friedman test on these average ranks ($p = 0.05$) with Bonferroni-Dunn post-hoc test to check their significance as suggested by Demšar (2006). The post-hoc test results are illustrated by the critical difference diagrams: for example, in Figure 1 the critical difference (CD) is plotted at the top, if the difference between ranks of two methods exceeds the CD, then this difference is significant. The smaller the rank the better the performance. If two methods do not exceed the CD, they are connected by a thick black line.

We also present the critical difference diagrams that correspond to the tree size of tree learners and number of rules of rule learners. The tree size is measured in term of the number of nodes of the subgroup trees or decision trees generated by other tree learners.

6.2 Results and Discussion

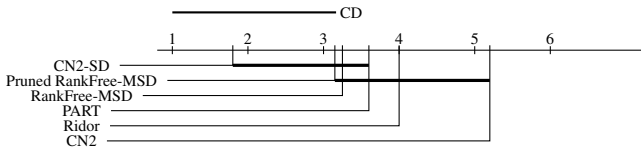

Figure 1: Post-hoc test (significance reported) comparing the AUCs of `RankFree-MSD` and `Pruned-RankFree-MSD` against other rule learners.

8. The discretisation was done with the help of the unsupervised attribute filtering method in Weka (weka.filters.unsupervised.attribute.Discretize).
9. Brier score = calibration loss + refinement loss. Refinement loss decreases if the probabilities are closer to 0 or 1. Calibration loss decreases if the probabilities are closer to empirical probabilities. See (Flach and Matsubara, 2007) for more details.

	RankFree-MSD	Pruned RankFree-MSD	CN2	CN2-SD	PART	Ridor
1	**0.91(1.50)**	**0.91(1.50)**	0.69(6.00)	0.88(3.00)	0.87(4.00)	0.82(5.00)
2	0.95(3.50)	0.95(3.50)	0.92(6.00)	0.93(5.00)	**0.98(1.00)**	0.97(2.00)
3	0.70(3.00)	0.70(2.00)	0.58(6.00)	**0.71(1.00)**	0.65(5.00)	0.65(4.00)
4	0.77(2.00)	0.77(3.00)	0.56(6.00)	**0.80(1.00)**	0.75(4.00)	0.73(5.00)
5	0.96(5.50)	0.96(5.50)	0.96(4.00)	**0.99(1.00)**	0.97(3.00)	0.97(2.00)
6	0.93(3.00)	0.93(2.00)	0.82(6.00)	**0.94(1.00)**	0.91(4.00)	0.88(5.00)
7	0.98(3.50)	0.98(3.50)	0.98(2.00)	**0.98(1.00)**	0.96(6.00)	0.97(5.00)
8	0.92(5.50)	0.92(5.50)	0.94(4.00)	0.94(3.00)	**0.99(1.00)**	0.97(2.00)
9	0.87(2.50)	0.87(2.50)	0.54(6.00)	**0.89(1.00)**	0.80(4.00)	0.76(5.00)
10	0.77(2.50)	0.77(2.50)	0.57(6.00)	**0.82(1.00)**	0.72(4.00)	0.70(5.00)
Average	0.88(3.25)	0.88(3.15)	0.76(5.20)	**0.89(1.80)**	0.86(3.60)	0.84(4.00)

Table 2: Comparing AUCs of RankFree-MSD and Pruned-RankFree-MSD against other rule learners (ranks in brackets) over 10 UCI data sets.

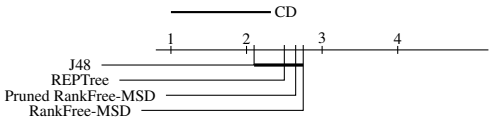

Figure 2: Post-hoc test (no significance reported) comparing the AUCs of `RankFree-MSD` and `Pruned-RankFree-MSD` against other tree learners.

	RankFree-MSD	Pruned RankFree-MSD	REPTree	J48
1	**0.91(1.50)**	**0.91(1.50)**	0.85(4.00)	0.91(3.00)
2	0.95(3.50)	0.95(3.50)	0.98(2.00)	**0.99(1.00)**
3	0.70(2.00)	**0.70(1.00)**	0.70(3.00)	0.69(4.00)
4	0.77(3.00)	0.77(4.00)	0.81(2.00)	**0.82(1.00)**
5	0.96(3.50)	0.96(3.50)	**0.98(1.00)**	0.98(2.00)
6	0.93(3.00)	0.93(2.00)	0.50(4.00)	**0.95(1.00)**
7	0.98(3.50)	0.98(3.50)	0.98(2.00)	**0.98(1.00)**
8	0.92(3.50)	0.92(3.50)	0.99(2.00)	**1.00(1.00)**
9	0.87(2.50)	0.87(2.50)	**0.88(1.00)**	0.86(4.00)
10	**0.77(1.50)**	**0.77(1.50)**	0.52(4.00)	0.74(3.00)
Average	0.88(2.75)	0.88(2.65)	0.82(2.50)	**0.89(2.10)**

Table 3: Comparing AUCs of `RankFree-MSD` and `Pruned-RankFree-MSD` against other tree learners (ranks in brackets) over 10 UCI data sets.

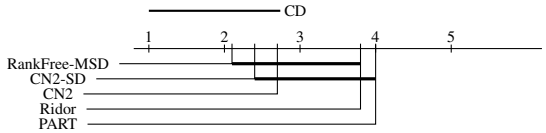

Figure 3: Post-hoc test (significance reported) comparing the average rank of `RankFree-MSD` and other rule learners with respect to the number of rules.

Ranking By looking at Table 2 and 3 the reader can notice that our methods (`RankFree-MSD` and `Pruned-RankFree-MSD`) are highly competitive with other tree and rule learnering

	RankFree-MSD	Pruned RankFree-MSD	CN2	CN2-SD	PART	Ridor
1	**0.10(1.50)**	**0.10(1.50)**	0.16(6.00)	0.11(3.00)	0.11(4.00)	0.13(5.00)
2	0.05(3.50)	0.05(3.50)	0.09(6.00)	0.08(5.00)	**0.02(1.00)**	0.02(2.00)
3	0.19(3.00)	0.19(2.00)	0.20(4.00)	**0.19(1.00)**	0.27(5.00)	0.31(6.00)
4	0.07(2.00)	0.07(3.00)	0.09(4.00)	**0.07(1.00)**	0.09(5.00)	0.10(6.00)
5	0.04(4.50)	0.04(4.50)	0.05(6.00)	**0.02(1.00)**	0.02(3.00)	0.02(2.00)
6	0.05(5.00)	0.05(4.00)	0.07(6.00)	**0.04(1.00)**	0.04(2.00)	0.05(3.00)
7	0.03(4.00)	0.03(3.00)	0.04(6.00)	0.03(2.00)	0.04(5.00)	**0.02(1.00)**
8	0.08(3.50)	0.08(3.50)	0.09(5.00)	0.10(6.00)	**0.01(1.00)**	0.03(2.00)
9	**0.13(1.50)**	**0.13(1.50)**	0.22(6.00)	0.15(3.00)	0.18(4.00)	0.21(5.00)
10	0.07(2.50)	0.07(2.50)	0.07(4.00)	**0.06(1.00)**	0.08(5.00)	0.09(6.00)
Average	0.08(3.10)	0.08(2.90)	0.11(5.30)	**0.08(2.40)**	0.08(3.50)	0.10(3.80)

Table 4: Comparing Brier scores of RankFree-MSD and Pruned-RankFree-MSD against other learners (ranks in brackets) over 10 UCI data sets.

	RankFree-MSD	Pruned RankFree-MSD	REPTree	J48
1	**0.10(1.50)**	**0.10(1.50)**	0.11(4.00)	0.10(3.00)
2	0.05(3.50)	0.05(3.50)	0.03(2.00)	**0.03(1.00)**
3	0.19(2.00)	**0.19(1.00)**	0.20(3.00)	0.21(4.00)
4	0.07(2.00)	0.07(3.00)	**0.07(1.00)**	0.08(4.00)
5	0.04(3.50)	0.04(3.50)	**0.01(1.00)**	0.02(2.00)
6	0.05(3.00)	0.05(2.00)	0.09(4.00)	**0.04(1.00)**
7	0.03(4.00)	0.03(3.00)	0.03(2.00)	**0.02(1.00)**
8	0.08(3.50)	0.08(3.50)	0.02(2.00)	**0.02(1.00)**
9	**0.13(1.50)**	**0.13(1.50)**	0.13(3.00)	0.14(4.00)
10	0.07(2.50)	0.07(2.50)	0.08(4.00)	**0.06(1.00)**
Average	0.08(2.70)	0.08(2.50)	0.08(2.60)	**0.07(2.20)**

Table 5: Comparing Brier scores of `RankFree-MSD` and `Pruned-RankFree-MSD` against other tree learners (ranks in brackets) over 10 UCI data sets.

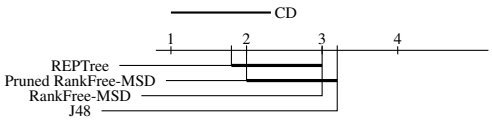

Figure 4: Post-hoc test (significance reported) comparing the average rank of `RankFree-MSD` and `Pruned-RankFree-MSD` and other tree learners with respect to the tree size.

methods in terms of the AUC. The average difference in the AUC between our methods and the best method does not exceed one percent. (Significant differences have been reported only for `CN2-SD` over `CN2`.)

Probability Estimation The Brier score results (Table 4 and table 5), which reflect the probabilistic performance of the methods, are similar to the AUC results discussed above except that the probabilities of tree learners get much more closer to each other. Interestingly, even when the tree was pruned, the `RankFree-Pruning` algorithm did not cause any harm to the probability estimation. This also shows that the pruning algorithm is well suited to the task.

Size of the models Figure 4 illustrates CD with respect to the tree size. REPTree produces the most compact decision trees on average followed by `Pruned-RankFree-MSD`. If we compare the number rules generated by our method against the other rule learners (Figure 3), `RankFree-MSD`[10] is the best with respect to the average rank but not significantly except against PART.

`RankFree-MSD` vs. `Pruned-RankFree-MSD` Despite the reduction in the number of subgroups[11] that has been reported on all the 10 data sets when `RankFree-Pruning` was incorporated (`Pruned-RankFree-MSD`), it generally produced better or no worse results than its unpruned version in all aspects except refinement. It is interesting to see that pruning does not negatively affect the ranking performance unlike the conventional Reduced-Error-Pruning approach which is known for harming the AUCs.

None of the tree and rule learners significantly outperform our `RankFree-MSD` and `Pruned-RankFree-MSD` methods while the opposite is true in some situations. This confirms the significance, suitability and usefulness of our proposed method to extend the subgroup discovery framework to perform predictive tasks such as probability estimation and ranking.

7. Conclusions

In this paper we demonstrated a simple and logical way of using the heuristic and coverage information obtained by the multi-class subgroup discovery system `CN2-MSD` to build a multi-class probability estimator and ranker `RankFree-MSD`. Our approach is significantly different from other approaches as we learn subgroups as well as a predictive model in a single learning task and the subgroups correspond to interesting patterns found with respect to multiple classes. The subgroups are prioritised, with respect to their multi-class heuristic scores, and used to form a binary tree such that the most discriminative and significant subgroups are added first.

We also derived a suitable and yet strong pruning algorithm, `RankFree-Pruning`, and demonstrated experimentally that this method reduces the size of the tree without decreasing the performance when evaluated using the standard 10-fold cross-validation. Despite the simplicity of our approach and the use of heuristic functions that do not maximise the accuracy-related measures put together with the induction of relatively few and overlapping subgroup rules, we showed that our approach produces ranking performance competitive to decision trees and rule learners.

This research emphasises the usefulness and significance of multi-class subgroups as they hold significant information which is sufficient for undertaking accurate predictive tasks on its own. Having said that, dealing with small and imbalanced classes is still a challenge for subgroup discovery.

To conclude, we would like to emphasise that the choice of the heuristic function is an important issue often overlooked when dealing with multi-class domains. This is because most rule learning systems employ binary heuristic measures in one-vs-rest, one-vs-one or all-vs-all paradigms and run into problems of combining them or their predictions in a sub-optimal way. As we have shown in this paper, the tree is a natural choice for at least the prediction, probability estimation, and ranking tasks. There is no need for restricting the learning to strictly predict accurate and non-overlapping hypotheses. We argue that all that is required is to learn representative hypotheses, even if they

10. The number of rules in `RankFree-MSD` are the original number of rules produced by `CN2-MSD` rule learner
11. A removal of a single subgroup causes an entire level in a tree to be removed.

overlap, and a tree-based model is effective in partitioning the instance space and handling the overlaps according to those hypotheses.

In future work we would like to investigate more interesting multi-class heuristic measures and conduct a further and larger investigation on the multi-class subgroups. For instance we would like to apply the methods to more data set and test the performance of the multi-class subgroups when various levels of noise are introduced within data sets and how much effect this will have on the our methods against other learners.

Acknowledgments

We would like to thank the Saudi Arabian Cultural Bureau of the Embassy of Saudi Arabia in London and King Abudulaziz City for Science and Technology Organisation for the scholarship and the continues support of this research.

References

Tarek Abudawood and Peter Flach. Evaluation Measures for Multi-class Subgroup Discovery. In *he European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases ECML PKDD'09*, pages 35–50. Springer-Verlag Berlin Heidelberg, September 2009.

Henrik Bostrom. Covering vs. divide-and-conquer for top-down induction of logic programs. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1194–1200. Morgan Kaufmann, 1995.

Henrik Boström. Maximizing the area under the ROC curve with decision lists and rule sets. In *Proceedings of the SIAM international conference on data mining*, pages 27–34, 2007.

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78:1–3, 1950.

P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Tom Fawcett. Using rule sets to maximize ROC performance. In *Proceedings of the IEEE international conference on data mining 2001*, volume 29, pages 131–138, 2001.

Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. Technical report, HP Laboratories, March 16 2004.

Tom Fawcett. An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006.

Peter Flach and Edson Matsubara. A simple lexicographic ranker and probability estimator. In *European Conference on Machine Learning*, volume 4701 of *Lecture Notes in Computer Science*, pages 575–582. Springer, September 2007.

Jerome H. Friedman and Nicholas I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9:123–143, 1999.

J. Fürnkranz and G. Widmer. Incremental reduced error pruning. In *Proc. 11th Int. Conf. on Machine Learning*, pages 70–77. Morgan Kaufmann, 1994.

I. Glöckner. Finding Answer Passages with Rank Optimizing Decision Trees. In *International Conference on Machine Learning and Applications*, pages 208–214. IEEE, 2009.

Willi Klösgen. Subgroup discovery. In Willi Klösgen and Jan M. Zytkow, editors, *Handbook of Data Mining and Knowledge Discovery*, pages 354–361. Oxford University Press, June 2002.

Willi Klösgen. Explora: A multipattern and multistrategy discovery assistant. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 249–271. MIT Press, 2004.

Nada Lavrač, Branko Kavšek, Peter Flach, and Ljupco Todorovski. Subgroup Discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004.

D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases. 1998. URL `http://www.ics.uci.edu/~mlearn/mlrepository.html`.

F. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52 (3):199–215, 2003.

Jan-Nikolas Sulzmann and Johannes Fürnkranz. An empirical comparison of probability estimation techniques for probabilistic rules. In *Discovery Science '09: Proceedings of the 12th International Conference on Discovery Science*, pages 317–331, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-04746-6.

D.C. Wilkins and Y. Ma. The refinement of probabilistic rule sets: sociopathic interactions. *Artificial Intelligence*, 70(1-2):1–32, 1994.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, second edition, 2005.

Appendix: Extra Experimental Results

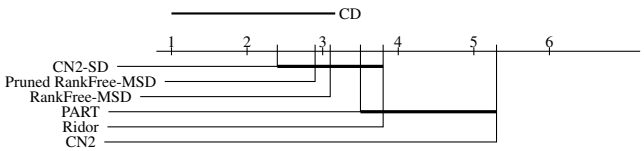

Figure 5: Post-hoc test (significance reported) comparing the Brier scores of `RankFree-MSD` and `Pruned-RankFree-MSD` against other rule learners.

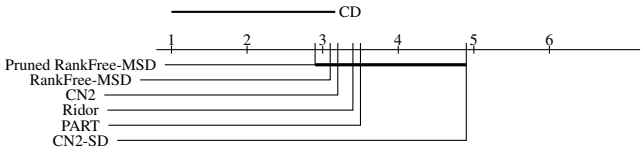

Figure 6: Post-hoc test (no significance reported) comparing the calibration losses of `RankFree-MSD` and `Pruned-RankFree-MSD` against other rule learners.

	RankFree-MSD	CN2	CN2-SD	PART	Ridor
1	18.20(3.00)	**11.00(1.00)**	25.10(4.00)	58.60(5.00)	14.00(2.00)
2	18.10(3.00)	**10.10(1.00)**	11.70(2.00)	82.70(4.00)	112.30(5.00)
3	47.70(4.00)	20.40(2.00)	**13.40(1.00)**	261.00(5.00)	43.00(3.00)
4	32.70(2.00)	41.90(3.00)	**26.00(1.00)**	1187.10(4.00)	1227.00(5.00)
5	**8.30(1.00)**	24.10(5.00)	10.90(2.00)	12.70(3.00)	14.90(4.00)
6	**10.00(1.00)**	22.20(3.00)	29.30(4.00)	19.70(2.00)	38.40(5.00)
7	3.70(2.00)	16.50(5.00)	7.50(4.00)	6.20(3.00)	**3.30(1.00)**
8	**3.00(1.00)**	23.00(3.00)	19.00(2.00)	119.70(5.00)	33.70(4.00)
9	61.90(3.00)	33.90(2.00)	**24.20(1.00)**	760.40(5.00)	161.20(4.00)
10	**18.00(1.00)**	20.30(2.00)	28.50(3.00)	200.40(4.00)	2733.60(5.00)
Average	**22.16(2.10)**	22.34(2.70)	19.56(2.40)	270.85(4.00)	438.14(3.80)

Table 6: Comparing number of rules of `RankFree-MSD` and `Pruned-RankFree-MSD` against other learners (ranks in brackets) over 10 UCI data sets.

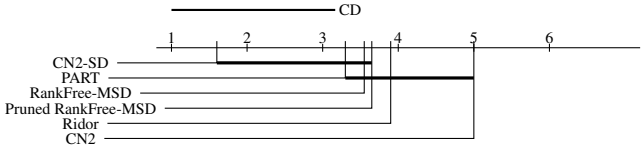

Figure 7: Post-hoc test (significance reported) comparing the refinement losses of `RankFree-MSD` and `Pruned-RankFree-MSD` against other rule learners.

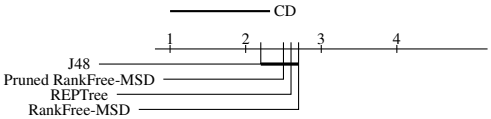

Figure 8: Post-hoc test (no significance reported) comparing the Brier scores of `RankFree-MSD` and `Pruned-RankFree-MSD` against other tree learners.