Data Min Knowl Disc (2009) 19:210–226
DOI 10.1007/s10618-009-0136-3

On subgroup discovery in numerical domains

Henrik Grosskreutz · Stefan Rüping

Received: 10 June 2009 / Accepted: 20 June 2009 / Published online: 22 July 2009
Springer Science+Business Media, LLC 2009

Abstract Subgroup discovery is a Knowledge Discovery task that aims at finding subgroups of a population with high generality and distributional unusualness. While several subgroup discovery algorithms have been presented in the past, they focus on databases with nominal attributes or make use of discretization to get rid of the numerical attributes. In this paper, we illustrate why the replacement of numerical attributes by nominal attributes can result in suboptimal results. Thereafter, we present a new subgroup discovery algorithm that prunes large parts of the search space by exploiting bounds between related numerical subgroup descriptions. The same algorithm can also be applied to ordinal attributes. In an experimental section, we show that the use of our new pruning scheme results in a huge performance gain when more that just a few split-points are considered for the numerical attributes.

Keywords Pattern mining · Subgroup discovery · Performance · Pruning

1 Introduction

Subgroup discovery (Klösgen 1996; Wrobel 1997) is a Knowledge Discovery task that aims at finding descriptions of subgroups of a population with high generality and distributional unusualness with respect to the target attribute. It belongs to the

Responsible editors: Aleksander Kołcz, Wray Buntine, Marko Grobelnik, Dunja Mladenic, and John Shawe-Taylor.

H. Grosskreutz (✉) · S. Rüping
Fraunhofer IAIS, Schloss Birlinghoven, Sankt Augustin, Germany
e-mail: henrik.grosskreutz@iais.fraunhofer.de

S. Rüping
e-mail: stefan.rueping@iais.fraunhofer.de

group of local pattern mining tasks, like frequent item mining and association mining, but differs from these tasks, among others, by its different quality function. Subgroup discovery is a general approach that has shown to be useful in a variety of application scenarios: Recently, it has been applied in an outstanding Data Mining case study in Dementia Research (Hapfelmeier et al. 2008); other applications scenarios include spatial analysis (Klösgen and May 2002) and marketing campaign planning (Lavrac and Gamberger 2004).

Most subgroup discovery algorithms presented so far (Atzmueller and Puppe 2005; Demsar et al. 2004; Atzmüller and Puppe 2006; Grosskreutz et al. 2008; Klösgen 1996) primarily consider the case where the dataset only involves attributes with nominal (non-target) attributes. In this setting, a subgroup is typically described by a conjunction of attribute-value pairs. Those pairs are sometimes also called *features* (Lavrac and Gamberger 2004). For example, in a dataset describing the characteristics and the price of cars, the search for subgroups with a significantly higher price than average would come up with descriptions involving features like *fuel-type* = *gas* or *aspiration* = *turbo*.

In many real-world domains, however, the data involves attributes that are not nominal but numeric. For example, the gas consumption of a car is a numeric attribute, and so are (in a medical setting) the age of a patient or her blood pressure. Here, the features in the subgroup descriptions should involve either inequalities or intervals, like for example $blood\_pressure \in]80, 120]$ or $age \in]18, 23]$. The standard approach to deal with numerical attributes is to make use of a prior discretization step, to replace every numerical by a single nominal attribute. The effect of this approach is, however, that the subsequent subgroup discovery will typically only find suboptimal subgroup descriptions as only a subset of all valid features are preserved. While some authors (Lavrac et al. 2004; Kralj et al. 2005) consider the transformation of every numerical attribute into a *set* of features to avoid this limitation, to the best of our knowledge no specialized algorithm has been proposed that takes advantage of the constraints among features created from numerical attributes in such a way.

In this paper, we describe a new pruning scheme which exploits the constraints among the quality of subgroups ranging over overlapping intervals and present a new subgroup discovery algorithm, MergeSD. We show empirically that on several real-world datasets MergeSD achieves a huge performance gain compared to previous algorithms when the number of candidate interval endpoints is large. While in this paper we focus on numerical attributes, our approach is also applicable to ordinal attributes.

The remainder of this paper is organized as follows: In Sect. 2, we define the subgroup discovery task. In Sect. 3, we consider previous approaches and their limitations. Thereafter, we present our new pruning scheme in Sect. 4. We present the experiments in Sect. 5 before we conclude in Sect. 6.

2 Preliminaries

A *database* or *dataset* DB is a set of rows $\{R_1, \ldots, R_N\}$, each built up from of $l + 1$ values. We distinguish one attribute C, called the *class attribute*, from the l ordinary attributes $\{A_1, A_2, \ldots, A_l\}$. Throughout this paper, we assume that the class attribute

C is two-valued, with possible values *pos* and *neg*. The other attributes can either be nominal, with domains $D(A_i) = \{v_{i,1}, \ldots, v_{i,m_i}\}$, or numerical; in the latter case, we assume $D(A_i) = \mathcal{R}$. For a database row R_j we use the expression $class(R_j)$ to refer to its class. Rows with class *pos* will be called *positive examples*, the others will be called *negative examples*.

A *subgroup description* sd is a set of *features* (Lavrac and Gamberger 2004) $\{f_1, \ldots, f_k\}$ where every feature f_i is a constraint on an attribute. That is, for nominal attributes a feature f_i has the form $(A_i = v_i)$, $v_i \in D(A_i)$ and for numerical features it has the form $A_i \in]\, t_l, t_r]$, where $t_l \in \mathcal{R} \cup \{-\infty\}$ and $t_r \in \mathcal{R} \cup \{\infty\}$. We use the convention that every (finite) interval endpoint t_i has to occur in the dataset (to avoid running into an infinite number of equivalent split points). The *length* of the subgroup description, $length(sd)$, is the number of features it is built of. We call a subgroup description sd' a *refinement* of a subgroup description sd, denoted by $sd' \succ sd$, if sd is a subset of sd'. In the following, we will sometimes use the term subgroup as an abbreviation for subgroup description.

Given a database DB and a subgroup description sd, $DB[sd]$ denotes the set of rows $R_j \in DB$ that satisfy *all* feature $f_i \in sd$. For a nominal attribute A_i, the row $R_j = (v_{j,1}, \ldots, v_{j,l}, c_j)$ satisfies $A_i = v_i$ iff. $v_{j,i} = v_i$; for a numerical attribute A_i, R_j satisfies $A_i \in]\, t_l, t_r]$ iff. $v_{j,i} \in]\, t_l, t_r]$.

The interestingness of a subgroup description sd (on a dataset DB) is measured by a so-called *quality function*. A quality function q is a mapping from a database and a subgroup description to the reals; the higher the quality, the more interesting the subgroup description is. In this paper, we consider quality functions of the form:

$$g^a \cdot (p - p_0) \tag{1}$$

where a is a constant s.t. $0 \leq a \leq 1$, g denotes the generality of the subgroup and p resp. p_0 the fraction of rows of positive class in the subgroup respectively in DB. Formally, $g := |DB[sd]|/|DB|$, while $p := \frac{|\{r \in DB[sd] \mid class(r)=pos\}|}{|DB[sd]|}$, and $p_0 := \frac{|\{r \in DB \mid class(r)=pos\}|}{|DB|}$. In the experimental section we focus on the case where $a = 1$, which is known as the Piatetsky-Shapiro quality function (Klösgen 1996), is order equivalent to the weighted relative accuracy WRACC (Lavrac et al. 2004), and is probably the most popular subgroup quality function. However, our pruning approach works for all a s.t. $0 \leq a \leq 1$.

We are now ready to formulate the task of *subgroup discovery*. Given a database DB, a maximum length d and a number k, find a set S of subgroup descriptions such that every $sd \in S$ has length $\leq d$, the cardinality of S is k and the quality of the solution S is maximal. Thereby, the quality of a set of subgroups, S, is defined as the average quality of the subgroups in S.

An *optimistic estimate* (Wrobel 1997) is a function that, given a subgroup description sd, provides a bound on the quality of all refinements of sd. Formally, the function oe is an optimistic estimate for the quality function q if

$$\forall \text{ subgroups } sd, sd'. sd' \succ sd \implies oe(sd) \geq q(sd').$$

Optimistic estimates allow to significantly speedup the task of (exhaustive) subgroup discovery: if one has already found k subgroups and the least quality of those subgroups is *minQ*, then it is safe to prune the sub-space of refinements of a subgroup with optimistic estimate below *minQ*. For the Piatetsky-Shapiro quality function, the following is a tight optimistic estimate (Grosskreutz et al. 2008): $g \cdot p \cdot (1 - p_0)$.

3 Prior work

In this section, we review how existing subgroup discovery systems deal with numerical attributes and discuss the limitations of these approaches.

3.1 Subgroup discovery implementations

Most subgroup discovery algorithms, like CN2-SD (Lavrac et al. 2002), SD-Map (Atzmüller and Puppe 2006) and DpSubgroup (Grosskreutz et al. 2008), only consider subgroup descriptions involving nominal attributes. Thus, the standard approach of data mining systems based on these algorithms, like Vikamine (Atzmueller and Puppe 2005) or Orange (Demsar et al. 2004), is to discretize all numerical attributes beforehand. Typically, this is done by means of entropy discretization (Fayyad and Irani 1993).

The result is that, for every numerical attribute, an ordered set of *split points* $t_1, \ldots, t_n$ is calculated and that the numerical attribute is replaced by a new nominal attribute with domain $\{]-\infty, t_1],]t_1, t_2],]t_2, t_3], \ldots,]t_n, \infty]\}$. The result of this procedure is, roughly speaking, that only subgroup descriptions ranging over non-overlapping *base intervals* are considered. Intervals like $]t_1, t_3]$ that overlap with $]t_1, t_2]$ and $]t_2, t_3]$ are ignored.

To overcome this restriction, some authors do not simply replace a numerical attribute by a single nominal attribute but instead use a binarization scheme where every possible attribute-value inequality or every possible attribute-interval pair is turned into a boolean feature (Kralj et al. 2005; Lavrac and Gamberger 2004). While in principle this allows to find the optimal solution to the subgroup discovery task, in practice it results in a major increase in the search space which can be problematic. As a result, these approaches have to make use of some incomplete search heuristic like beam search. Kralj et al. (2005) gives an interesting comparison of the effect of different discretization methods on the performance of subgroup discovery algorithms. In particular, they consider both entropy discretization and exhaustive feature construction. However, as they consider a beam search approach the results are more about the interplay of different discretization strategies and beam search and not so much about the restrictions imposed by a strategy like entropy discretization on the remaining space of subgroup descriptions. Moreover, neither they nor other authors propose a specialized algorithm which takes advantage of the characteristics of numerical attributes to speedup the subgroup discovery. However, there is potential for doing so, as we will describe in Sect. 4.

3.2 Problems with entropy discretization

We will now present some examples that illustrate the impact of the different discretization strategies.

3.2.1 Non-overlapping intervals

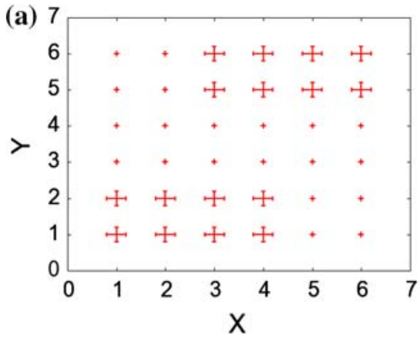
Figure 1a shows a dataset involving two attributes, plotted on the x and y-axis. Positive examples are visualized by cross, while negative examples are visualized by a point. While this example is hand-crafted and completely idealized, it is nevertheless very fitting to understand the problems that occur when dealing with numerical attributes. It should be obvious that there are two subgroups in this dataset: the instances at the lower left and the instances at the upper right. That is, we expect to obtain the following two subgroup descriptions:

$$Y \in]-\infty, 2] \wedge X \in]-\infty, 4] \quad \text{and} \quad Y \in]4, \infty] \wedge X \in]2, \infty]$$

However, entropy discretization and subsequent subgroup discovery on the resulting nominal attributes does not find these subgroups. Instead, the best subgroups found are:

$$\begin{aligned} &X \in]2, 4] \\ &Y \in]-\infty, 2] \\ &Y \in]4, \infty] \\ &Y \in]-\infty, 2] \quad \text{and} \quad X \in]-\infty, 2] \end{aligned}$$

Those subgroups are not the expected result, and indeed they have a strictly lower quality than the optimal subgroups. The reason for the sub-optimal result here is the following: Entropy discretization finds a set of split points for the two numerical attributes and these split points are used to replace the numerical attribute by a nominal attribute with values ranging over non-overlapping *base intervals*. In this example, only the non-overlapping intervals $]-\infty, 2]$, $]2, 4]$, $]4, \infty]$ are considered for attribute



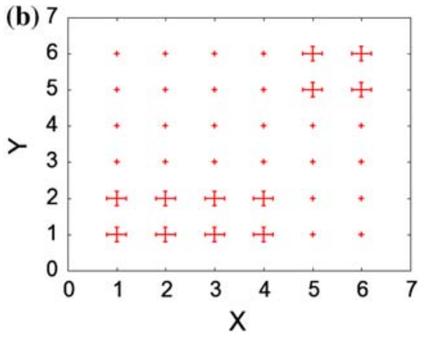


Fig. 1 Examples (**a**) Non-overlapping intervals fail to find the optimal solution (**b**) Entropy discretization does not find any split point for attribute x

x, and similarly for attribute y. Thus, subgroup discovery only considers conditions where the value of x lies within *one* of those base intervals.

3.2.2 Overlapping intervals

The obvious solution to the problem illustrated in the previous section seems to be to consider every ordered pair of the split points $t_1, \ldots, t_n$ as candidate interval, not just the base intervals. This is essentially equivalent to the approach proposed (Kralj et al. 2005), where every numerical attribute is transformed into a *set* of features.

However, there are two caveats. First, this approach results in $O(n^2)$ overlapping candidate intervals, and thus it will become necessary to consider $O(n^2)$ features for the attribute, instead of only $O(n)$ features. Second, it is easy to construct other examples where new problems arise when overlapping intervals based on entropy discretization are used. For example, consider the population illustrated in Fig. 1b.

Obviously, the expected result is the following subgroup description:

$$Y \in]-\infty, 2] \wedge X \in]-\infty, 4]$$

However, entropy discretization fails in partitioning the x dimension. The reason is that entropy discretization performs univariate analysis to determine the splitting points. That is, it will use the projection of the data on the x dimension and on this projection, the proportion of positive and negative examples is uniformly distributed. Hence, entropy discretization will simply consider the attribute x as irrelevant. As a result, the best subgroup would be:

$$Y \in]-\infty, 2]$$

We remark that it is possible to construct high-dimensional datasets where the projection on every dimension results in uniformly distributed positive and negative examples.

3.3 Other related work

One important issue to address is the retrieval of good *sets* of subgroups. In subgroup discovery applications one is often interested in finding a relatively small set of subgroups with a moderate overlap, not in finding all patterns above a certain threshold. In particular, it would not be desirable to obtain a set of subgroup descriptions which only differ minimally in their interval endpoints. A standard approach to deal with this kind of difficulties is the weighted covering approach proposed in Lavrac et al. (2002). Essentially, this is an iterative scheme which searches for the top-1 subgroup, adapts the weight of the positive examples covered by that subgroup, and goes into the next iteration.

Finally, we remark that while it might seem that the task we consider is very similar to impact rule mining (Webb 2001), the difference is that there it is the target attribute which is numeric. Our task is also related to quantitative association rule mining

(Srikant and Agrawal 1996). Like us, they consider both base intervals and combine them to obtain super-intervals. However, their approach is based on itemsets instead of subgroups. Hence, they consider a different quality function and moreover use a maximum support parameter.

4 A pruning scheme for numerical attributes

In the previous section, we have motivated why subgroup discovery algorithms for numerical attributes should consider overlapping intervals. Regardless of the strategy that is used to determine the intervals respectively the split points for the attributes (an issue we will discuss in Sect. 4.4), it is desirable to prune as large a part of the search space as possible. Of course, optimistic estimate pruning can be used for that purpose. However, it is possible to prune a much larger part of the search space by exploiting constraints among the quality of subgroups ranging over overlapping intervals of the same attribute.

In this section, we will describe the algorithm MergeSD that makes use of a new pruning scheme. Basically, MergeSD performs a depth-first-search in the space of subgroup descriptions. In each recursion step, MergeSD checks all combinations of endpoints *unless proof in the form of an upper bound exists that the subgroup refinements will not exceed the quality threshold*. The pruning approach is based on the following ideas:

(1) It is possible to calculate a bound on the quality of all subgroup refinements of $sd' \wedge A \in]t_l, t_r]$ from the maximal quality of all subgroups refinements of $sd' \wedge A \in]t_l, t']$ and $sd' \wedge A \in]t', t_r]$ (for an arbitrary t' s.t. $t_l \leq t' \leq t_r$),

(2) Typically, there are many splitpoints t' between t_l and t_r. Thus, there are multiple possibilities to apply the above and thus one can obtain multiple bounds for the refinements of $sd' \wedge A \in]t_l, t_r]$. To keep track of all those bounds, respectively of their minimum, we make use of a specialized data structure, the *BoundTables*.

We will now discuss these steps in more detail in the remainder of this section.

4.1 Quality constraints

Let DB be a database, k a depth level, P a set of split points and sd a subgroup description of length $< k$ involving only intervals with endpoints in P. Then:

$$maxQ(DB, k, sd, P) := \max_{sd^* \in \mathit{refinements}(sd, DB, k, P)} \{q(DB, sd^*)\}$$

Here, $\mathit{refinements}(sd, DB, k, P)$ denotes the set of refinements of sd with length $\leq k$ and interval endpoints in P. Using this definition, we are now ready to formulate the property that will be the basis for our algorithm MergeSD:

Lemma 1 *Let DB be a database, P a set of split points, t_l and t_r two split points from P such that $t_l < t_r$ and A a numerical attribute from DB. Furthermore, let sd' and sd be two subgroup descriptions (of length $\leq k$, involving only interval endpoints from P and attributes from DB) such that sd is a refinement of $sd' \wedge A \in]t_l, t_r]$. Then for every*

t' such that $t_l \leq t' \leq t_r$, the quality of sd on DB is bound by $maxQ(DB, k, sd' \wedge A \in]t_l, t'], P) + maxQ(DB, k, sd' \wedge A \in]t', t_r], P)$.

Proof Let sd^* be a maximum quality subgroup from *refinements*$(sd' \wedge A \in]t_l, t_r], DB, k, P)$. Then, by construction $q(DB, sd) \leq q(DB, sd^*)$. Furthermore, let sd_l^* be the subgroup description obtained from sd^* by replacing the condition $A \in]t_l, t_r]$ by $A \in]t_l, t']$. Similarly, let sd_r^* be the description obtained by replacing the former expression by $A \in]t', t_r]$. If we show that the quality of sd^* is bound by the sum of the qualities of sd_l^* and sd_r^*, then we can deduce

$$\begin{aligned} q(DB, sd^*) &\leq q(DB, sd_l^*) + q(DB, sd_r^*) \\ &\leq maxQ(DB, k, sd' \wedge A \in]t_l, t'], P) \\ &\quad + maxQ(DB, k, sd' \wedge A \in]t', t_r], P) \end{aligned}$$

because sd_l^* and sd_r^* are elements of *refinements*$(sd' \wedge A \in]t_l, t'], DB, k, P)$ resp. *refinements*$(sd' \wedge A \in]t', t_r], DB, k, P)$, and we are done.

We will now complete the proof by showing the inequality $q(DB, sd^*) \leq q(DB, sd_l^*) + q(DB, sd_r^*)$. Let n_l and p_l be negative resp. the positive example in $DB[sd^*]$ which satisfy $A \in]t_l, t']$ and similarly n_r and p_r those that satisfy $A \in]t', t_r]$. Then for $0 \leq a \leq 1$:

$$\begin{aligned} & q(DB, sd^*) \cdot |DB| \\ &= (p_l + p_r + n_l + n_r)^a \left(\frac{p_l + p_r}{p_l + p_r + n_l + n_r} - p_0 \right) \\ &= (p_l + p_r + n_l + n_r)^{a-1} (p_l + p_r - p_0(p_l + p_r + n_l + n_r)) \\ &= (p_l + p_r + n_l + n_r)^{a-1} (p_l - p_0(p_l + n_l)) \\ &\quad + (p_l + p_r + n_l + n_r)^{a-1} (p_r - p_0(p_r + n_r)) \\ &\leq (p_l + n_l)^{a-1} (p_l - p_0(p_l + n_l)) \\ &\quad + (p_r + n_r)^{a-1} (p_r - p_0(p_r + n_r)) \\ &= (q(DB, sd_l^*) + q(DB, sd_r^*)) \cdot |DB| \end{aligned}$$

which completes the proof. □

4.2 Keeping track of all quality bounds

To keep track of all bounds that can be deduced from sub-intervals using Lemma 1, and to make maximum use of them, we define a special data structure, which we call *BoundTables*. A *BoundTables* is a (2-dimensional) table, where *BoundTables*$[i, j]$ contains the bound on the quality of $A \in]t_i, t_j]$. There is one *BoundTables* for every attribute A. Initially, the value of *BoundTables*$[i, j]$ is 0 if $i = j$ and ∞ else. (Only elements where $i \leq j$ are relevant.)

Typically during the exploration of the search space one considers more than one split point for every (numerical) attribute. When a new bound $maxQ$ for the refinements of $A \in]t_i, t_j]$ becomes available, it becomes potentially possible to update the bounds for *all* super-interval of $A \in]t_i, t_j]$, i.e. for all $A \in]t'_i, t'_j]$ such that $i' \leq i$ and $j' \geq j$. Hence, whenever a new bound $max\,Q$ on the quality of subgroups involving $A \in]t_i, t_j]$ becomes available,

- the value *BoundTables*$[i, j]$ is set to $max\,Q$; and
- for all $i' \leq i$ and $j' \geq j$, *BoundTables*$[i', j']$ is updated to *BoundTables*$[i', i] + max\,Q +$ *BoundTables*$[j, j']$.

The correctness of the second update follows directly from the fact that $]t'_i, t'_j]$ is the disjunct union of $]t'_i, t_i]$, $]t_i, t_j]$ and $]t_j, t'_j]$ and Lemma 1.

4.3 Depth-first search in the space of subgroup descriptions

Based on the data structure *BoundTables*, our algorithm recursively explores the space of candidate subgroup descriptions, essentially by performing a depth first search. The use of a breadth-first-search strategy is not possible because our pruning scheme assumes that the complete branch of the search space below an interval is exhaustively explored before the bounds for the super-intervals can be updated. We remark, however, that breadth-first-search is problematic anyway due to its higher memory footprint.

During the depth first search, at every node our algorithm explores the child (i.e. the branch of refinements) with highest bound first. If all intervals that have not yet been explored have a bound below the minimum required quality, *minQ*, the loop ends. Else, the attribute and interval with the highest bound are determined using the *BoundTables*s. As a heuristic, we ensure that the next interval will never range over base intervals that have not been checked.[1] Once the recursive calculation for the branch below the selected attribute/interval ends, the corresponding *BoundTables* is updated. Moreover, if k subgroups have already been collected, the minimum quality *minQ* is updated to the quality of the k-th subgroup. Thereafter, the next run of the loop starts.

As a simple but effective optimization, our algorithm tests all direct refinements (i.e. all refinements resulting from the addition of a single feature) first before switching to recursive exploration. This approach allows to have a better estimate of *minQ* before the expensive recursive explorations start. We use a separate set of *BoundTables*s to prune as large a number of direct refinements as possible. Please note that it is necessary to use separate sets of *BoundTables*s because they store bounds for subgroup descriptions of different length.

Another important optimization is motivated by the observation that it is possible to derive bounds for super-intervals not only using the exact maximum quality for the refinements of sub-intervals, but also using optimistic estimates for sub-intervals. For example, it is possible to establish a bound for $]t'_i, t'_j]$ using the exact maximum quality calculated for the branch below $]t_i, t_j]$ and the optimistic estimates for $]t'_i, t_i]$

[1] The rationale for this heuristic is roughly the following: the information about the quality of subgroup descriptions ranging over a base interval allows to update the bounds for many super-intervals, while the knowledge about the quality of subgroup descriptions ranging over a larger interval does not provide that much potential for updates of super-interval bounds.

Algorithm 1 The algorithm MergeSD

```
MergeSD(Database $DB$, int $k$, depth limit $d$):
1: Initialize priority queue $result$ with maximal length $\leq k$
2: call recurse($DB$, $\emptyset$, $d$, $-\infty$, $result$)
   return $result$
Function recurse($DB$, $sg$, $d$, $minQ$, $result$):
1: Initialize local variable $bestQBelow$ with quality of $sg$
2: for all remaining attributes $a$ do
3:   Initialize $BoundTables$ $B^a_{direct}$ for direct refinements of $sg$ involving constraint on $a$
4:   Initialize $BoundTables$ $B^a_{rec}$ for recursion-depth refinements
5: end for
6: while $\exists$ interval with estimate $\geq minQ$ according to the $B^a_{direct}$ tables do
7:   determine next refinement $sg* := sg \wedge a \in ]t_l, t_r]$
8:   calculate quality of $sg*$ and add $sg*$ to $result$ if its quality is $\geq minQ$
9:   set $bestQBelow$ to maximum of $bestQBelow$ and quality of $sg*$
10:  update $B^a_{direct}$ using quality of $sg*$
11:  update $B^a_{rec}$ using optimistic estimate for $sg*$
12:  increase $minQ$ if $result$ contains $k$ subgroups with quality $> minQ$
13: end while
14: if length of $sg$ is $< (d$ - 1$)$ then
15:   while $\exists$ interval with estimate $\geq minQ$ according to the $B^a_{rec}$ tables do
16:     determine next refinement $sg* := sg \wedge a \in ]t_l, t_r]$
17:     call recurse($DB$, $sg*$, $d$, $minQ$,$result$) and store result in local var $bestQBelowSg*$
18:     set $bestQBelow$ to maximum of $bestQBelow$ and $bestQBelowSg*$
19:     update $B^a_{rec}$ using $bestQBelowSg*$
20:     increase $minQ$ if $result$ contains $k$ subgroups with quality $> minQ$
21:   end while
22: end if
23: return $bestQBelow$
```

and $]t_j, t'_j]$. Thus, whenever we calculate the quality of a direct refinement, we also calculate its optimistic estimate and use it to update the corresponding *BoundTables*.

We give pseudo-code for MergeSD in Algorithm 1. The computation is essentially done by the function recurse. Lines 6–13 of recurse deal with the direct refinements: The next subgroup is determined and considered, and *minQ* and the *BoundTables*s are updated as described earlier. Lines 14–22 perform the recursive calls, updating the *BoundTables*s and *minQ* when possible. Finally, the value of the best subgroup considered below this branch, *bestQBelow*, is returned to allow the caller of recurse to update its own *BoundTables*.

4.3.1 Mixed attributes

One remark on subgroup descriptions involving both numerical and nominal attributes: For the sake of simplicity, here we only considered numerical attributes. However, it is relatively simple to extend the algorithm to deal with both numerical and nominal attributes. One possibility is to explore the search space by first considering nominal attributes (using standard techniques) and then to switch to the numerical attributes.

4.4 Split points and candidate intervals

So far, we did not discuss how the split points for an attribute are calculated. There is a wide range of possible approaches to do so (Dougherty et al. 1995). In this paper, we consider three different approaches: the approach of Fayyad and Irani (1993) based on entropy minimization coupled with the minimal description length principle; equal frequency intervals (Dougherty et al. 1995) and exhaustive search where every value in the data set is considered as a split point.

4.4.1 Optimizations

For the evaluation described in Sect. 5, where we search for the top-1 subgroup, we used two optimizations to somewhat reduce the set of candidate intervals considered by equal frequency discretization and exhaustive search. First, we only consider a value as candidate split point if it separates two adjacent examples having different class. Second, we do not consider an interval if there is a narrower candidate interval which covers the same set of positive examples. We remark that in general, i.e. in top-k discovery with $k > 1$, these optimzations can have the effect that a subgroup description is *not* returned although it has a sufficiently high quality. However, it is easy to verify that the best subgroup found will still have maximal quality, i.e. top-1 subgroup discovery is not affected. This case is of particular importance because, as explained in Sect. 3.3, in practice top-1 subgroup discovery is often combined with an iterative weighted covering approach (Lavrac et al. 2002).

5 Evaluation

In this section, we present results concerning the performance of MergeSD, the impact of pruning and the quality of the subgroups found using different discretization strategies. We ran experiments on eight datasets, listed in Fig. 2 along with their most important properties. The dataset ‘mail order fraud’ is a real-world dataset from a project where we were seeking for descriptions of subgroups with a high fraud rate in mail order data. The other datasets are benchmark datasets from the well-known UCI Machine Learning Repository (Asuncion and Newman 2007). All Experiments were run on an Intel Core 2 Duo T7500 with 2 GB of RAM under Windows XP, using the Piatetsky-Shapiro quality function.

5.1 Quality

Figure 3 shows the quality of the best subgroup, i.e. the top-1 subgroup found on two datasets using different strategies. While the x-axis shows the limit on the length of the subgroup description, the y-axis shows the quality of the best subgroup. The figure shows the result for different strategies: non-overlapping frequency discretization with five splits (‘n/o freq.’), non-overlapping entropy discretization (‘n/o entr.’),

Dataset	#rows	#num. attribs	avg #vals per num attr.	max #vals
breast-w	699	9	10	10
diabetes	768	8	157	517
mail order fraud	6714	103	26	450
letter	2000	16	15	16
mammography	961	5	19	73
spambase	4601	57	264	2161
transfusion	748	4	44	78
yeast	1484	8	52	81

Fig. 2 Datasets

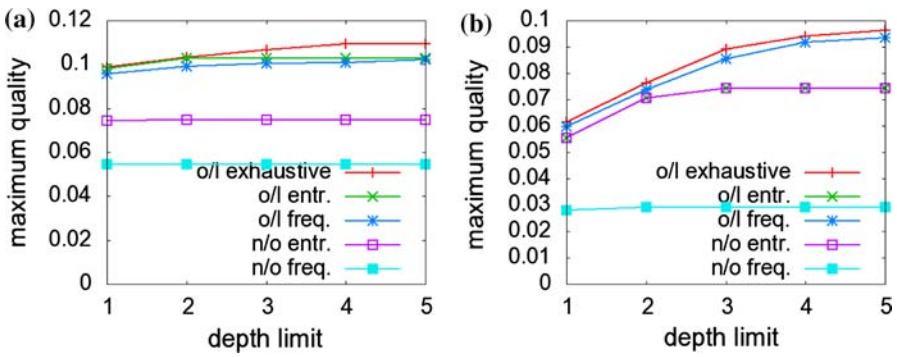


Fig. 3 Quality comparisons (**a**) 'diabetes' (**b**) 'yeast'

overlapping frequency discretization with ten splits ('o/l freq'), overlapping entropy discretization ('o/l entr.') and exhaustive search ('exhaustive').

For the diabetes dataset, the figure shows that the use of non-overlapping intervals result in clearly lower quality subgroups compared to overlapping intervals: Using frequency discretization, the quality is only about 50% of the maximum; using entropy discretization about 68%. Interestingly, for the yeast dataset the result obtained using entropy discretization and overlapping intervals coincides with that using non-overlapping intervals. As expected from the discussion in Sect. 3.2, minimal-entropy discretization is sometimes inferior to frequency discretization: while in 'diabetes' entropy discretization is marginally on the lead, in 'yeast' frequency discretization is clearly better.

In both examples, none of the discretization methods was able to find the optimal solution (i.e. the solution obtained by exhaustive search). We show whether the discretization strategies resulted in the discovery of the optimal solution for the other datasets in Fig. 4. A '+' means that entropy discretization respectively frequency discretization found the optimal solution for a specific depth limit and dataset. The result is that although entropy discretization comes up with the optimal solution in some cases, in most cases it fails to do so. In some of those cases, frequency discretization allows to find the optimal solution, but there are many cases where neither approach

	entr. discr.					freq. discr.				
Dataset	1	2	3	4	5	1	2	3	4	5
breast-w	+	+	-	-	-	+	+	+	+	+
diabetes -	-	-	-	-	-	-	-	-	-	-
m/o fraud	+	-	-			+	-	-		
letter	+	-	-	-		+	+	+	+	
mammo.	+	+	+	+	+	+	+	+	+	+
spambase	+	+				-	-			
transfusion	+	-	-	-	-	+	-	-	-	-
yeast	-	-	-	-	-	-	-	-	-	-

Fig. 4 Optimality of top-1 subgroup for different discretization strategies and depth limits

is able to come up with the optimal solution (Please note that some values are missing because exhaustive search did not complete within 48 h).

5.2 Performance

We compared MergeSD with the Algorithm DpSubgroup (Grosskreutz et al. 2008), a subgroup discovery algorithm that makes use of pruning based on tight optimistic estimates. On datasets involving only nominal attributes DpSubgroup is arguably the best algorithm currently available when it comes to pruning the search space [Presumably, it is also one of the fastest subgroup discovery implementations as it also makes use of the FpTree-based data representations introduced by SD-Map (Atzmüller and Puppe 2006)]. We made some minor modifications to make DpSubgroup applicable to numerical domains: first, the numeric attributes are discretized by calculating a set of split points and using one binary feature for every ordered pair of split points. Second, we ensured that DpSubgroup never considers subgroup descriptions involving more than one feature per numeric attribute (to avoid considering subgroups like $A \in]2, 5] \wedge A \in]2, 4]$).

Figure 5 shows the effect of the new pruning scheme on the number of subgroup descriptions considered when frequency discretization with ten split points is used. On the x axis, we show different depth limits, while the y axis shows the number of

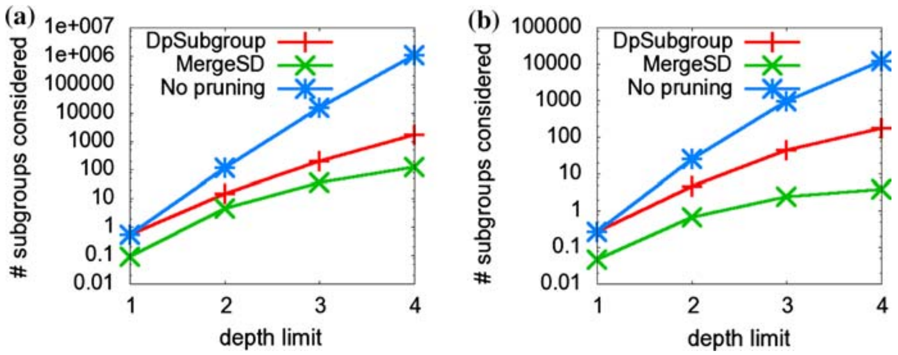


Fig. 5 Number of considered subgroups if frequency discretization is used (**a**) 'diabetes' (**b**) 'transfusion'

nodes explored by the two algorithms, as well as the total number of nodes (which had to be searched if no pruning was applied). The diagram shows that the pruning based on the *BoundTables* allows to prune a much higher fraction of the candidate subgroups (Please note that we used a logarithmic scale in this figure). Moreover, it is interesting to observe that while the optimistic estimate pruning used in DpSubgroup only has an impact on depth level 2 or higher (because optimistic estimate pruning only allows to prune *refinements*), MergeSD also allows to prune a large fraction of the subgroups at depth level 1.

Next, we considered the impact of our new pruning scheme when the subgroups are searched exhaustively. Figure 6 shows the number of candidate subgroup descriptions considered by MergeSD and DpSubgroup on the ’transfusion’ and the ’mammography’ datasets. The figure shows that the use of the new pruning scheme reduces the number of considered subgroup descriptions by orders of magnitude. The impact of the new pruning scheme on the runtime is roughly similar. Figure 7 shows the runtimes of DpSubgroup and MergeSD on four different datasets when frequency discretization is used. The difference is tremendous; please note that again we used a logarithmic scale.

Figure 8 summarizes the speedups achieved by the new algorithm when frequency discretization respectively exhaustive search is used (the figures show the fraction of the runtime of MergeSD compared to DpSubgroup). Thereby, we only considered depth level 2 because DpSubgroup failed to calculate the results on larger depth limits; even with depth limit 2, DpSubgroup failed to complete the exhaustive calculation (for memory or time reasons) on several datasets. On all datasets, the speedup is considerable and often reaches several orders of magnitude.

While MergeSD outperforms DpSubgroup when frequency discretization is used (and even more so if exhaustive search is used), the situation changes when entropy discretization is used. In this case, the performance of MergeSD and the modified DpSubgroup are comparable. The reason is that in most datasets from Fig. 2 entropy discretization produces a very small set of split points (the average number of split points per attribute varied from 1.7 to 3.2 for the different datasets).

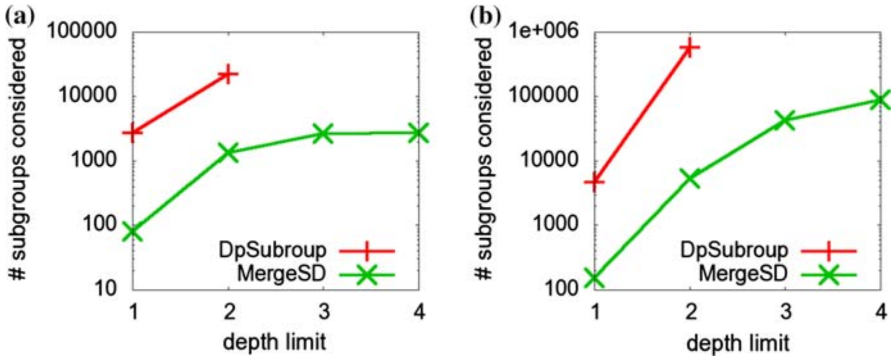


Fig. 6 Number of considered subgroups if an exhaustive set of split points are used (**a**) ‘mammography’ (**b**) ‘transfusion’

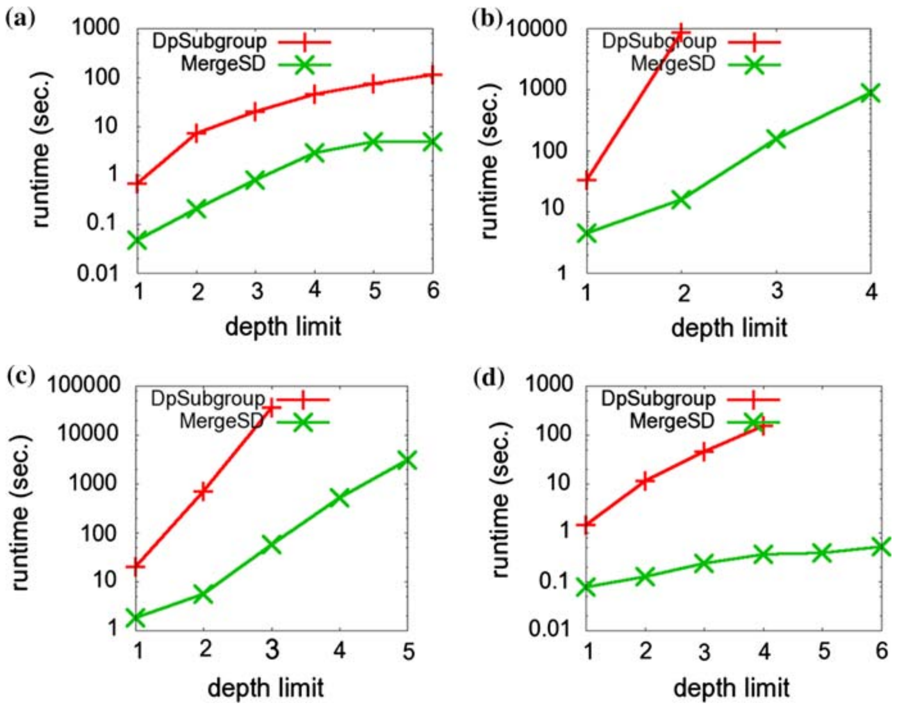


Fig. 7 Frequency discretization: runtimes (**a**) 'diabetes' (**b**) 'mail order fraud' (**c**) 'spambase' (**d**) 'yeast'

Dataset	freq. discr. (10 splits)	exhaustive
breast-w	6	27
diabetes	35	?
fraud	534	?
letter	107	872
mammography	3	139
spambase	125	?
transfusion	14	6312
yeast	92	?

Fig. 8 Speedup of MergeSD vs. DpSubgroup

6 Summary and discussion

In this paper, we have investigated subgroup discovery on datasets with numerical attributes. We have explained why the standard approach to replace every numerical attribute by a nominal attribute (using entropy discretization and non-overlapping intervals) typically results in suboptimal results. We have also motivated why the use of entropy discretization and overlapping intervals is not optimal either.

As main contribution, we have presented a new pruning scheme that makes use of quality constraints among the quality of subgroups which range over overlapping intervals of the same attribute. We have presented a data structure that keeps track of all bounds that can be deduced from the quality of the subgroups explored so far and have described an algorithm that makes use of these bounds to prune a large part of the search space.

We have made use of our new pruning scheme to calculate the relative quality of the results produced by different subgroup discovery strategies. In particular, we have considered frequency and entropy discretization, combined with both overlapping and non-overlapping intervals, as well as exhaustive search. The result is that overlapping intervals derived by means of entropy discretization often produce quite good subgroup descriptions. However, in most experiments the result is not optimal. Our conclusion is that the use of entropy discretization and overlapping intervals is a good starting point, but that it is suggestive to additionally perform a frequency discretization (or an exhaustive search). When frequency discretization or exhaustive search is used, or when entropy discretization comes up with a large number of split points, the new pruning scheme presented in this paper allows to tremendously speedup the computation.

A nearby extension would be to consider more heuristics for the computation of split points. For example, one could make use of a *local* entropy discretization, where the split points are not globally computed before the subgroup discovery starts but locally, that is inside the recursive computation. In fact, we tested this strategy on all datasets in Fig. 2 but it never resulted in an optimal result where the global entropy discretization did not. This observation fits with the investigations in Dougherty et al. (1995) on global versus local entropy discretization.

Acknowledgments This research was supported by the European Commission under the project *RACWeB* (No. 045101).

References

Asuncion A, Newman DJ (2007) UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences

Atzmueller M, Puppe F (2005) Semi-automatic visual subgroup mining using vikamine. J Univers Comp Sci 11(11):1752–1765

Atzmüller M, Puppe F (2006) SD-map—a fast algorithm for exhaustive subgroup discovery. In: Fürnkranz J, Scheffer T, Spiliopoulou M (eds) PKDD, volume 4213 of lecture notes in computer science. Springer, New York, pp 6–17

Demsar J, Zupan B, Leban G (2004) Orange: from experimental machine learning to interactive data mining. Technical report, faculty of computer and information science. University of Ljubljana, Slovenia

Dougherty J, Kohavi R, Sahami M (1995) Supervised and unsupervised discretization of continuous features. Morgan Kaufmann, Los Altos, pp 194–202

Fayyad UM, Irani KB (1993) Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI, pp 1022–1029

Grosskreutz H, Rüping S, Wrobel S (2008) Tight optimistic estimates for fast subgroup discovery. In: Daelemans W, Goethals B, Morik K (eds) ECML/PKDD (1), volume 5211 of lecture notes in computer science. Springer, New York, pp 440–456

Hapfelmeier A, Schmidt J, Mueller M, Perneczky R, Drzezga A, Kurz A, Kramer S (2008) Interpreting pet scans by structured patient data: a data mining case study in dementia research. In: Proceedings of the eighth IEEE international conference on data mining (ICDM-2008)

Klösgen W (1996) Explora: a multipattern and multistrategy discovery assistant. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (eds) Advances in knowledge discovery and data mining. AAAI/MIT Press, pp 249–271. ISBN 0-262-56097-6

Klösgen W, May M (2002) Spatial subgroup mining integrated in an object-relational spatial database. In: PKDD '02. Springer-Verlag, London, pp 275–286

Kralj P, Lavrač N, Zupan B, Gamberger D (2005) Experimental comparison of three subgroup discovery algorithms: Analysing brain ischemia data. In: Proceedings of the 8th International multiconference information society IS 2005, pp 220–223

Lavrac N, Gamberger D (2004) Relevancy in constraint-based subgroup discovery. In: Boulicaut J-F, Raedt LD, Mannila H (eds) Constraint-based mining and inductive databases, volume 3848 of lecture notes in computer science. Springer, New York, pp 243–266

Lavrac N, Flach PA, Kavsek B, Todorovski L (2002) Adapting classification rule induction to subgroup discovery. In: Proceedings of the 2002 IEEE international conference on data mining (ICDM 2002), 9–12 December 2002, Maebashi City, Japan. IEEE Computer Society, pp 266–273

Lavrac N, Cestnik B, Gamberger D, Flach PA (2004) Decision support through subgroup discovery: three case studies and the lessons learned. Mach Learn 57(1–2):115–143

Srikant R, Agrawal R (1996) Mining quantitative association rules in large relational tables. ACM SIGMOD Record 25(2):1–12

Webb GI (2001) Discovering associations with numeric variables. In: KDD '01: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 383–388

Wrobel S (1997) An algorithm for multi-relational discovery of subgroups. In: Komorowski J, Zytkow J (eds) Proceedings of the first European symposion on principles of data mining and knowledge discovery (PKDD-97). Springer, New York, pp 78–87