Cluster-Grouping: From Subgroup Discovery to Clustering

Albrecht Zimmermann and Luc De Raedt

Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, P.O. Box 2402, Belgium
{Albrecht.Zimmermann,Luc.DeRaedt}@cs.kuleuven.be

Abstract. We introduce the problem of **cluster-grouping** and show that it can be considered a subtask in several important data mining tasks, such as *subgroup discovery*, mining correlated patterns, *clustering* and *classification*. The algorithm CG for solving *cluster-grouping* problems is then introduced, and it is incorporated as a component in several existing and novel algorithms for tackling *subgroup discovery*, *clustering* and *classification*. The resulting systems are empirically compared to state-of-the-art systems such as CN2, CBA, Ripper, Autoclass and CobWeb. The results indicate that the CG algorithm can be useful as a generic local pattern mining component in a wide variety of data mining and machine learning algorithms.
Keywords: correlated pattern mining, subgroup discovery, associative classification, clustering.

1 Introduction

The representation of conjunctive rules occupies a central position in the field of symbolic machine learning and data mining. It is used to represent local patterns in the data and sets of such rules form the output of a wide variety of systems, tackling diverse tasks ranging from association rule mining, correlated pattern mining, subgroup discovery, rule learning to conceptual clustering. [1, 34, 28, 12, 38] These systems all possess a local pattern mining or rule learning component, which raises the question as to whether there exists a unified or universal local pattern mining approach that can be used across these systems. The key contribution of the present paper is that we answer this question positively, by first, introducing the task of *cluster-grouping*, based on an extension of the work of [34], and second, proposing an algorithm, CG, employing upper-bound pruning techniques for exhaustively solving this task. The algorithm guarantees to find the best pattern (or rule) with regard to a correlation measure (e.g. χ^2), and forms an alternative to the often heuristic (beam-search) methods employed in machine learning.

As evidence for the wide applicability of the CG algorithm, we use it together with different wrappers to tackle the tasks of correlated pattern mining, subgroup

This paper integrates and extends an abstract published at ECML 2004 [47] as well as related work, published at DS 2004 [48], and as a book contribution [49].

discovery, classification and conceptual clustering. The resulting systems are then empirically evaluated on a large number of UCI data sets [4] and compared to state-of-the-art machine learning and data mining systems such as CN2-SD, RIPPER, CBA, and COBWEB [28, 12, 30, 16]. This also results in number of novel systems. Especially interesting are the novel CBC system, for realizing associative classification using correlated patterns rather than pure association rules as CBA [30] and CMAR [29] do, and the novel CG-CLUS system, based on a divisive decision-tree like algorithm, for realizing conceptual clustering. The tests in the CG-CLUS trees are based on conjunctive descriptions. The results of the experiments provide evidence for the key claims of this work – that the *cluster-grouping* task and CG algorithm can be useful as a component across a wide variety of data mining and machine learning algorithms. An additional and new finding is that using the CG algorithm instead of the common beam-search not only leads to good performance, but that the exhaustive method is *at least as* efficient as heuristic ones.

We proceed as follows. In the next section we introduce the concept of local pattern mining. In Section 3, we present the underlying principles of CG, the algorithm we propose for addressing the local pattern mining task. In Section 4, we introduce the general mechanism for combining local patterns into a global model. Additionally, we show how different data mining tasks can be cast in this description, describe influential systems based on existing paradigms and experimentally compare CG-based systems to existing solutions. In Section 5 we refer to related work before we conclude in the last section.

2 Local Pattern Mining

Throughout the paper, we use attribute-value representations, and hence, employ conjunctions of attribute-value pairs to describe patterns. More formally, let $\mathcal{A} = \{A_1, ..., A_d\}$ be a tuple of attributes and $\mathcal{V}[A] = \{v_1, ..., v_p\}$ the domain of A. A tuple $\langle v^1, ..., v^d \rangle$ with $v^i \in \mathcal{V}[A_i]$ is called an *instance*. A multiset $\mathcal{E} = \{e_1, ..., e_n\}$ of instances is called a *data set*.

Definition 1 ***(Condition)*** *A* ***condition*** *l is an attribute-value-pair $A = v$ with $v \in \mathcal{V}[A]$. An instance $\langle v_1, ..., v_d \rangle$ is* ***covered*** *by a condition l of the form $A_i = v$ iff $v_i = v$.*

Definition 2 ***(Pattern)*** *A* ***pattern*** *p is a conjunction of conditions, $l_1 \wedge ... \wedge l_i$. An instance e is covered by p iff it is covered by all its conditions.*

Such patterns are called *local* patterns if they describe instances that show an unexpectedly high density of certain attribute values compared to a background model. We define certain attributes $A_i^T \in \mathcal{A}$ as *target attributes* and define target conditions. In the case of *cluster-grouping*, the background model is supplied by the observed distributions of these target conditions.

We consider patterns to be *interesting* if the distribution of the target conditions deviates unexpectedly from the background distribution in the subset

Table 1. Contingency table for p w.r.t. $\{A^T\}$

	$A^T = v_1$	$A^T = v_2$	
p	$sup(p \wedge A^T = v_1) = y_1^+$	$sup(p \wedge A^T = v_2) = y_1^-$	$sup(p) = y_1^+ + y_1^-$
$\neg p$	$sup(\neg p \wedge A^T = v_1) = m_1 - y_1^+$	$sup(\neg p \wedge A^T = v_2) = n - m_1 - y_1^-$	$sup(\neg p) = n - (y_1^+ + y_1^-)$
	$sup(A^T = v_1) = m_1$	$sup(A^T = v_2) = n - m_1$	n

specified by the pattern. The unexpectedness is quantified by setting a threshold on the values of certain measures with regard to the pattern considered. To quantify the quality of a given pattern, different *interestingness* measures can be used, such as *accuracy* or *confidence*, or correlation measures, such as χ^2, *Information Gain*, and *Category Utility*. Accuracy measures the purity of the described population w.r.t. a given target condition, while correlation measures quantify the deviation between assumed distributions of target conditions and the actual distribution in the subset of instances defined by the patterns.

Definition 3 ***(Support)*** *For a pattern p, we define*

$$sup(p) = |\{e \in \mathcal{E} \mid e \text{ is covered by } p\}|$$

the support of p. The support of a pattern w.r.t. a single target $A^T = v_1$ is

$$sup(p \wedge A^T = v_1) = |\{e \in \mathcal{E} \mid e \text{ is covered by } p \text{ and } A^T = v_1\}|$$

To facilitate the use of correlation measures, occurrence counts are often organized in contingency tables. A contingency table for a pattern and a single binary-valued target attribute is shown in Table 1.

We use the following notation to refer to occurrence counts of patterns:

Definition 4 ***(Occurrence Counts)*** *For a given pattern p, target attributes $A_1^T, \ldots, A_d^T$ and a given data set $\mathcal{E}$ we define:*

$$\boldsymbol{n} = |\mathcal{E}|,\ \boldsymbol{m}_i = sup(A_i^T = v_1),\ \boldsymbol{y}_i^+ = sup[p \wedge (A_i^T = v_1)],\ \boldsymbol{y}_i^- = sup[p \wedge \neg(A_i^T = v_1)]$$

Note that the sum of the cells in a row (column) is equal to the margins of the table, that is the rightmost (down-most) entry in a row (column). Correlation measures compare for a given cell the product of the corresponding margins to the cell count, thus comparing *expected* (under an independence assumption between patterns and target attribute values) to *observed* frequency, and score the difference. Consider for instance the upper left cell of Table 1: the value of the cell itself, the *observed* value, is y^+. The coverage of the pattern on the entire data is $y^+ + y^-$, as seen on the upper right margin, and the size of $A^T = v_1$ is m_1, as seen in the lower left margin. This leads to a straight-forward *expected* value for the upper left cell: $m_1 \cdot (y^+ + y^-)/n$. To compare these two values, one can for instance subtract them from each other, squared so that both higher and lower than expected behavior is treated symmetrically: $(y_1^+ - m_1(y_1^+ + y_1^-)/n)^2$.

In the χ^2 measure this term would then be discounted with the expected value, giving a complete term of:

$$\frac{(y_1^+ - m_1(y_1^+ + y_1^-)/n)^2}{m_1(y_1^+ + y_1^-)/n}$$

Table 2. Pseudo-Contingency table for p w.r.t $\{A_1^T, A_2^T\}$

	$A_1^T = v_1$	$A_1^T = v_2$	$A_2^T = v_1$	$A_2^T = v_2$	
p	$y_1^+ =$ $sup[p \wedge (A_1^T = v_1)]$	$y_1^- =$ $sup[p \wedge (A_1^T = v_2)]$	$y_2^+ =$ $sup[p \wedge (A_2^T = v_1)]$	$y_2^- =$ $sup[p \wedge (A_2^T = v_2)]$	$sup(p) = y_1^+ + y_1^-$ $= y_2^+ + y_2^-$
$\neg p$	$m_1 - y_1^+ =$ $sup(\neg p \wedge A_1^T = v_1)$	$n - m_1 - y_1^- =$ $sup(\neg p \wedge A_1^T = v_2)$	$m_2 - y_2^+ =$ $sup(\neg p \wedge A_2^T = v_1)$	$n - m_2 - y_-^2 =$ $sup(\neg p \wedge A_2^T = v_2)$	$sup(\neg p)$ $n - (y_1^+ + y_1^-)$
	$m_1 =$ $sup(A_1^T = v_1)$	$n - m_1 =$ $sup(A_1^T = v_2)$	$m_2 =$ $sup(A_2^T = v_1)$	$n - m_2 =$ $sup(A_2^T = v_2)$	n $= \lvert\mathcal{E}\rvert$

Increasing the number of involved target attributes usually leads to an increase of dimension of the contingency table to capture all dependencies among the conditions. Our focus is on the effect that pattern presence has on the A_i^T, the target attributes defining the background model. This means that we can disregard dependencies between those target attributes, decreasing the computational complexity of mining processes by instead using *pseudo-contingency* tables such as the one in Table 2. The main difference with regard to a regular high-dimensional contingency table, a so-called multi-way table, is that the margin of a row is not equal to the sum of row-cells anymore. A correlation measure still compares the product of the margins to the cell count.

Definition 5 ***(Stamp Point [34])*** *The* **stamp point** *of a pattern p w.r.t. a data set $\mathcal{E}$, and a set of target attributes $\{A_1^T, \ldots, A_d^T\}$, is the tuple of occurrence counts $\langle y_1^+, y_1^-, ..., y_d^+, y_d^- \rangle$.*

Consider an interestingness measure such as *accuracy*, χ^2, *Category Utility*, *Information Gain*, or *Weighted Relative Accuracy* defined on a pseudo-contingency table. Since $\boldsymbol{n}$ and the $\boldsymbol{m}_i$ are constant for a given data set, a given interestingness measure $\sigma(p)$ is a function of $2d$ variables

$$\sigma : \mathbb{N}^{2d} \mapsto \mathbb{R},$$

mapping the stamp point *sp(p)* to a real number.

We can now introduce the *cluster-grouping* problem, which – as we shall argue – can be used in a wide variety of data mining and machine learning problems.

Definition 6 ***(Cluster-Grouping Problem)***
Given:

- *a pattern language $\mathcal{L}$, defining the attribute-value pairs to be used in patterns,*
- *a data set $\mathcal{E}$,*
- *an interestingness measure σ,*
- *an interestingness threshold τ and/or maximum number of patterns k, and*
- *a set of target attributes $\{A_1^T, \ldots, A_d^T\}$*

Find:
A k-theory

$$Th_k(\mathcal{L}, \sigma, \mathcal{E}, \tau, \{A_1^T, \ldots, A_d^T\}) = \arg_k \max_{p \in \mathcal{L}} \{\sigma(sp(p)) \geq \tau\}$$

The k-theory consists of the k best patterns expressible in $\mathcal{L}$ according to σ w.r.t. the background model induced by the A_i^T on data set $\mathcal{E}$.

An example for such a task would be the inner loop of a rule learner performing sequential covering: For $k = 1$, and *accuracy* as the interestingness measure σ, the most accurate rule on $\mathcal{E}$ that can be formulated in $\mathcal{L}$ will be mined. The target attribute in this case is the class label. After removing covered instances, another 1-theory $Th_1(\mathcal{L}, acc, \mathcal{E}, \tau, \{C\})$ is mined, and this continues until all instances are covered.

In the next section, we propose a branch-and-bound algorithm in the mold of the family of optimization algorithms from [43] for solving *cluster-grouping* problems that is guaranteed to find optimal solutions. This contrasts with some heuristic approaches to solving instances of the *cluster-grouping* problem (such as beam-search) that are sometimes encountered in machine learning, cf. [10, 16].

3 Upper Bound on Convex Correlation Measures

Based on the convexity of correlation measures it is possible to calculate an upper bound on the future value of σ for specializations of a given pattern. This upper bound can be used to prune away parts of the search space known not to produce interesting solutions, and to focus the search on promising parts of the search space. The main insight underlying this technique is that convex functions attain their maximal values at the extreme points of their domain. To our knowledge, this idea was introduced by Morishita *et al.* [34].

3.1 Pattern behavior in coverage space

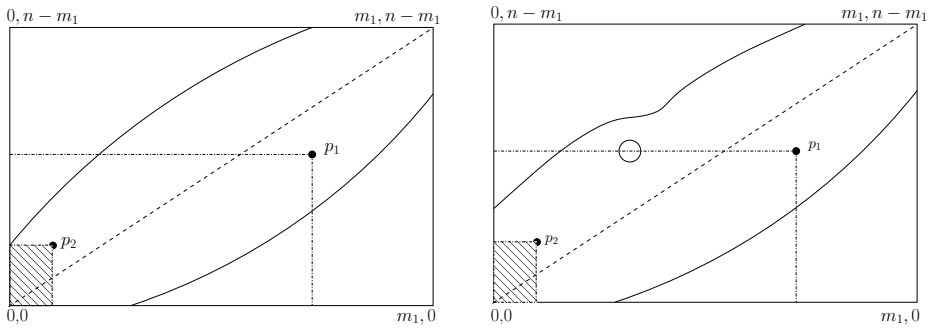
Coverage spaces, introduced in [22], can be used to visualize a pattern's or (collection of patterns') coverage behavior. To this end, a pattern is represented by the number of positives $P = y^+$ and negatives $N = y^-$ it covers (its stamp point w.r.t. a single target attribute), as shown in Figure 1(a). The most general

pattern is situated at the upper right corner $(m_1, n - m_1)$, since all instances are covered. When a pattern is specialized (i.e. extended with additional conditions), its stamp point $sp(p) = \langle y^+, y^- \rangle$ moves to the left and/or downwards, as its coverage decreases.

The diagonal in the diagram corresponds to a proportion of covered positives and covered negatives that is equal to that of the entire data set. At this diagonal, correlation measures evaluate to 0. The farther away from the diagonal a stamp point lies, the more significant it is w.r.t. the background distribution. For a given pattern p with stamp point $sp(p) = \langle y^+, y^- \rangle$, the point in coverage space that can be reached by any specialization p' and is farthest away from the diagonal, and therefore most significant, is either $(y^+, 0)$ or $(0, y^-)$, the extreme points.

Theorem 1. *The upper bound of a specialization of pattern p w.r.t. a convex correlation measure σ is*

$$ub_\sigma(p) = \max\{\sigma(y^+, 0), \sigma(0, y^-)\}$$



(a) Corresponding to value of convex function (b) Corresponding to non-convex case

Fig. 1. Coverage space with isometric lines

A pattern evaluation measure induces so-called isometrics in coverage space – curves that connect all coverage points having the same value for the measure. Consider the coverage space shown in Figure 1(a). The two elliptic lines correspond to a χ^2 threshold. A point between one of the isometrics and the diagonal refers to a pattern that does not pass the threshold, such as the patterns shown in the figure. The right pattern, p_1, has two upper bounds that lie above the threshold, which implies that it is worthy of specialization. The left pattern's (p_2) upper bounds both lie inside the isometrics and therefore no specialization of this pattern can be better than the threshold value, and thus the pattern (and all its specializations) should be pruned away.

3.2 Convexity

Upper bound pruning only works correctly for *convex* functions.

Definition 7 ***(Convexity)*** *A function $f : D \mapsto \mathbb{R}$ is convex iff $D \subseteq \mathbb{R}^d$ is a convex set and $\forall x_1, x_2 \in D, \lambda \in [0,1] : f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda) f(x_2)$.*

This means that, given two points x_1, x_2, all points x that lie on the line connecting x_1 and x_2 must have a value $f(x)$ that lies on or below the line connecting $f(x_1)$ and $f(x_2)$. Isometrics are just projections of a three-dimensional graph's area onto the two-dimensional plane denoting its domain. If "islands" and "dents" exist, such as the ones shown in Figure 1(b), the upper bound technique cannot be utilized since a future point might lie in one of the "islands", thus attaining a higher value than the threshold without lying outside the curves. At the same time, the existence of "islands" and "dents" corresponds to a violation of the convexity criterion.

Functions such as χ^2, *WRAcc*, *Information Gain*, and *Category Utility* are convex. For the proofs of the convexity of χ^2 and *Information Gain* we refer the reader to [34], while the proofs for *WRAcc* and *Category Utility* can be found in Appendix A.

3.3 Extension to Arbitrary Dimensions

If the mining process considers two or more independent target attributes, as we do, the interestingness measure is additive, meaning that the correlation measure can be evaluated separately for each of the independent target attributes, those finally summed up and averaged and/or normalized.

Definition 8 *A function σ over patterns p with $sp(p) = \langle y_1,^+, y_1^-, \cdots, y_d^+, y_d^- \rangle$ over attributes $\mathcal{A}_t$ (the stamp point) is additive if*

$$\sigma(p, \mathcal{A}_t) = \sigma(y_1^+, y_1^-, \ldots, y_d^+, y_d^-) = c \sum_{i=1}^{d} \sigma(y_i^+, y_i^-)$$

for some constant c.

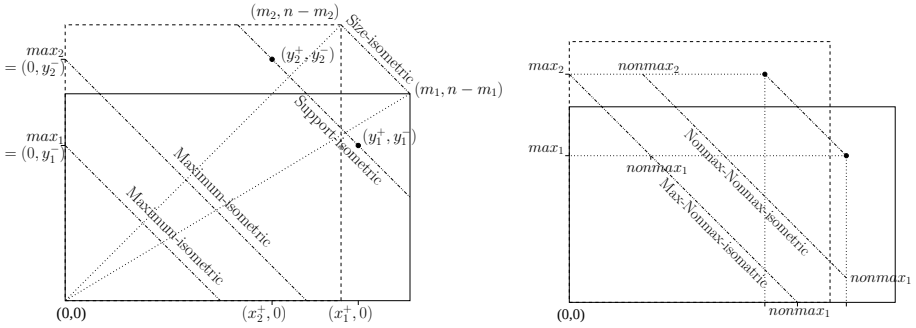
Since a sum of convex functions is a convex function itself, and a possible averaging factor c has no effect on convexity, the upper bound technique can be used on the entire sum. However, computing an upper bound it not so easy. There is a naïve upper bound that simply maximizes each summand:

$$ub_\sigma(p) = c \sum_{i=1}^{d} \max\{\sigma(y_i^+, 0), \sigma(0, y_i^-)\}$$

As shown in Table 2, however, $\forall i : y_i^+ + y_i^- = sup(p)$, which in turn leads to

$$x = sup(p) = y_1^+ + y_1^- = y_2^+ + y_2^- = \ldots = y_d^+ + y_d^-$$

This constraint is potentially violated when each summand is maximized independently of the others.



(a) Inconsistency of naïve maxima (b) Iterative maximization

Fig. 2. Coverage spaces for two target attributes

To illustrate this effect, consider the left-hand side of Figure 2. Shown are overlaid coverage spaces for two different target attributes, with a pattern's coverage denoted by a dark circle. Note that both the upper right corners of the coverage spaces and the dark circles lie on an isometric with a 135 degree slope – all points lying on this line have the same sum $m_i + (n - m_i)$ (the size-isometric), $y_i^+ + y_i^-$ (the coverage-isometric), respectively. The maximal values, visualized by being farthest from the background-distribution-diagonal, that can be reached by specializations of the pattern w.r.t. the two target attributes are denoted by max_1 and max_2 (denoted by maximum-isometric), respectively. The equal-sum-isometrics passing through those two values are not the same however, meaning that the respective maximum values would be reached by specializations with different coverage.

Given that the calculation of a non-naïve upper bound for an arbitrary number of target attributes is crucial to the success of our technique, we will give in the next paragraphs an algorithmic description of how to calculate this upper bound, use this algorithm and the isometrics to give an intuition as to why the upper bound is correct and tighter and finally prove its tighter evaluation.

As mentioned above, the main problem with the naïve technique lies in the fact that conflicting support isometrics could be induced by extreme points maximizing σ independently for each target attribute.

The upper bound calculation we use, shown as Algorithm 1, instead calculates an upper bound for every target attribute separately *under a support constraint*

$$ub_i = \max_{y^+_{i,max} + y^-_{i,min} = y^+_{i,min} + y^-_{i,max} = x} \{\sigma(y^+_{i,max}, y^-_{i,min}), \sigma(y^+_{i,min}, y^-_{i,max})\}$$

and then maximizes the sum of these upper bounds over a range of possible supports of specializations of the current pattern

$$ub_\sigma(p) = \max_{1 \leq x \leq sup(p)-1} \sum_{i=1}^{d} ub_i$$

Algorithm 1 Multi-dimensional upper bound calculation

Given: current pattern p, corresponding stamp point $\langle y_1^+, y_1^-, \ldots, y_d^+, y_d^- \rangle$
Return: upper bound on $\phi(p')$, with p' a specialization of p

$ub = 0$
for $1 \leq x \leq sup(p) - 1$ **do**
  for $1 \leq i \leq d$ **do**
    $y^+_{max} = \min\{x, y_i^+\}, y^-_{min} = x - y^+_{max}$
    $y^-_{max} = \min\{x, y_i^-\}, y^+_{min} = x - y^-_{max}$
    $ub_i = \max\{\sigma(y^+_{max}, y^-_{min}), \sigma(y^+_{min}, y^-_{max})\}$
  end for
  $ub = \max\{ub, \sum_{i=1}^{d} ub_i\}$
end for
return ub

To do this, the algorithm iterates over all possible supports of specializations, which lie between 1 (0 would correspond to a pattern covering nothing) and the current pattern support-1 (since identical support corresponds to a more specific pattern with the same informative value) - the outermost for-loop in Algorithm 1.

As can be seen in the right-hand side of Figure 2, such an isometric can correspond to a maximal value of σ for one of the attributes while corresponding to a non-maximal value for another one. The isometric can also correspond to non-maximal values for **both** attributes (the nonmax-nonmax-isometric). What still holds for the purpose of maximizing σ for a single attribute is that those points should be extreme points (furthest away from the background-distribution diagonal). To achieve this, two points $\langle y^+_{max}, y^-_{min} \rangle$, $\langle y^+_{min}, y^-_{max} \rangle$ are created. Since the support of a pattern, and the current value of y^+ both impose upper bounds on the maximal value of y^+_{max}, the smaller of the two is chosen. Additionally, since the isometric is specified, $y^+_{max} + y^-_{min}$ have to equal x, the specified support. Therefore, y^-_{min} is set to $x - y^+_{max}$. Analogous reasoning holds for $\langle y^+_{min}, y^-_{max} \rangle$.

σ is evaluated on both of these extreme points for an attribute, and the larger value chosen. Finally, the values for all attributes are added up and in this way an upper bound for the score of a hypothetical specialization of support x calculated. By maximizing over all possible future supports, an upper bound for *any* possible specialization of the current pattern is derived.

The preceding discussion explains why this is a *correct* upper bound: maximizing the contribution to σ for each target attribute under a certain support constraint, and doing this for all possible future supports ensures that no future score can exceed this bound. What is left to show is that this bound is *tighter* than the naïve one.

As mentioned in Section 3, convex functions attain their maximal values at the extreme points of their domain. Given the domain induced on a coverage space by y_i^+, y_i^-, it must therefore hold that for all $\langle y^+_{i,max}, y^-_{i,min} \rangle$, $\langle y^+_{i,min}, y^-_{i,max} \rangle$

defined according to Algorithm 1

$$ub_i = \max\{\sigma(y^+_{i,max}, y^-_{i,min}), \sigma(y^+_{i,min}, y^-_{i,max})\} \leq \max\{\sigma(y^+_i, 0), \sigma(0, y^-_i)\}$$

This in turn implies

$$\max_{1 \leq x \leq sup(p)-1} \sum_{i=1}^{d} ub_i \leq \sum_{i=1}^{d} \max\{\sigma(y^+_i, 0), \sigma(0, y^-_i)\}$$

3.4 The CG-Algorithm

In this section we present an algorithm for solving the *cluster-grouping* problem, called CG. For reasons of readability we show a version for finding patterns having a single highest score value (k = 1).

Algorithm 2 The CG algorithm that computes $Th_1(\mathcal{L}, \sigma, \mathcal{E}, \tau, \{A^T_1, \ldots, A^T_d\}) = \arg_1 \max_{p \in \mathcal{L}}\{\sigma(sp(p)) \geq \tau\}$.

$\mathcal{E}$ - data set, σ - correlation measure, τ_{user} - user-defined minimum threshold on σ

1: $P := \{\top\}, \tau := \tau_{user}, S := \emptyset$
2: **while** $P \neq \emptyset$ **do**
3: $p_{mp} := \arg\max_{p \in P}\{ub(p)\}$
4: $C := \rho(p_{mp})$
5: **for all** $c_i \in C$ **do**
6: compute $sp(c_i)$, calculate $\sigma(c_i)$
7: $ub_\sigma(c_i) :=$ *UpperBound*$(sp(c_i))$
8: $\tau := \max\{\tau, \sigma(c_i)\}$
9: **end for**
10: $S := \{s \in S \mid \sigma(s) = \tau\} \cup \{c \in C \mid \sigma(c) = \tau\}$
11: $S := S \setminus \{s \in S \mid \exists s' \in S : s' \prec s \wedge sp(s') = sp(s)\}$
12: $P := \{p \in P \mid ub_\sigma(p) \geq \tau\} \cup \{c \in C \mid ub_\sigma(c) \geq \tau\}$
13: **end while**
14: **return** S

The *cluster-grouping* algorithm CG (listed as Algorithm 2) is essentially a branch-and-bound algorithm along the lines of the family of optimization algorithms proposed in [43]. Starting from the most general pattern (denoted by $\top$), in each iteration the pattern $p_{mp} \in P$ (the set of *potential* solutions) with the highest upper bound is specialized (line 3). We use an optimal refinement operator ρ:

Definition 9 ***(Optimal Refinement Operator)*** *Let $\mathcal{L}$ be a set of conditions, $\prec$ a total order on the literals in $\mathcal{L}$, $\tau \in \mathbb{R}$.*

$$\rho(p) = \{p \wedge l_i \mid l_i \in \mathcal{L}, ub_\sigma(l_i) \geq \tau, \forall l \in p : l \prec l_i\}$$

is an ***optimal refinement operator****.*

The optimality of the refinement operator ρ ensures that each pattern will be created and evaluated only once during a run of the algorithm. The pattern p to be refined has an upper bound above the threshold since it would have been pruned otherwise. Since only conditions are added whose upper bound exceeds the threshold, the resulting specializations may have a score that exceeds or matches the current threshold.

The created specializations are then evaluated on the data set and the σ-scores and upper bounds are calculated (lines 6 and 7). If possible, the threshold is raised (line 8). The solution set S is composed of all patterns which have a score matching the threshold τ (line 10). In Algorithm 2, this threshold is either the best score seen so far or a user-defined threshold, whichever is larger. Specializations whose scores match the current threshold are added to the set of solutions S only if the solution set does not already include a generalization having the same stamp point. The rationale behind this is that literals not included in the more general pattern do not change the coverage and therefore do not add information. The algorithm can be easily modified so that k best patterns are found, by using the kth-best score as threshold, in which case solutions have to exceed, not match, the threshold. Finally, the set of promising patterns P is pruned using the threshold and all specializations whose upper bound exceeds τ are added (lines 11 and 12).

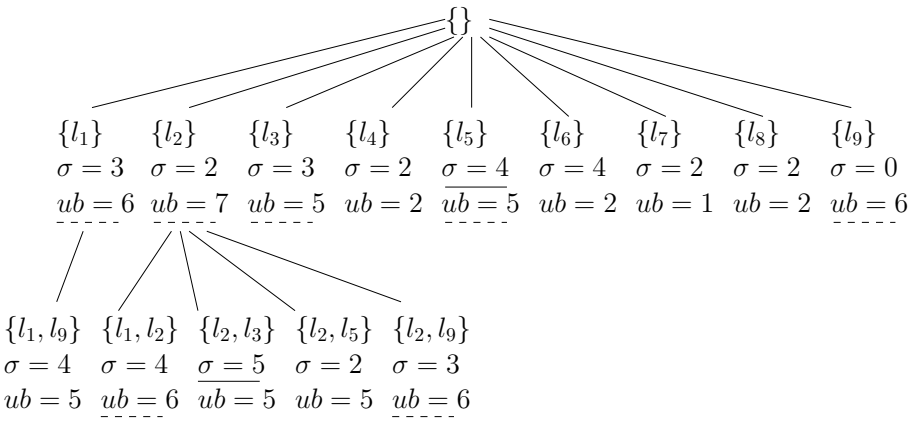


Fig. 3. Enumeration tree induced by CG

As an illustration of CG, consider Figure 3. After refining the empty set, all single-literal patterns are evaluated and their score and upper bound calculated. The highest σ is underlined using a solid line, while upper bounds that exceed it are underlined using a dashed line. The highest score encountered after the first refinement is 4 and the corresponding best solution so far $\{l_5\}$ (ties are broken lexicographically). The only literals that can be part of a solution exceeding this threshold are l_1, l_2, l_3, l_5 and l_9. Since l_2 has the highest upper bound of these, it

is selected for refinement. One of the new solutions, $\{l_2, l_3\}$, *does* have a better score than the current threshold and is selected as new best pattern. Increasing the threshold further reduces the literals that can be used for refinement to l_1, l_2 and l_9. Since we aim for general patterns, l_1 is refined next, using l_9 since l_2 has already been refined. The upper bound of $\{l_1, l_9\}$ which is 5, invalidates the more optimistic upper bounds of l_1, l_2 and l_9 and the algorithm terminates.

4 CG as component of several data mining systems

In the previous sections we have introduced the *cluster-grouping* task and outlined the CG algorithm for solving it. *Cluster-grouping* is typically not a goal in itself but rather – as we shall argue – an important step for building global models. A key contribution of our work is that we show that CG can be a useful component for machine learning and data mining systems tackling a wide variety of tasks such as correlated pattern mining, subgroup discovery, classification, and clustering. Many such systems can be decomposed into **two** main components:

- A *local* pattern mining algorithm to find patterns describing/ predicting the behavior of a subset of the data
- A *control* structure, or "wrapper", that, depending on the local miner's result, manipulates the data and/or restarts the local mining process, possibly with a different parameter setting

In this section, we will show that the *cluster-grouping* task and CG algorithm can be used as the local pattern mining component together with a wrapper for correlated pattern mining, subgroup discovery, classification and clustering. The main conceptual difference to the task of Sese *et al.* lies in the fact that we view the algorithm as a component in a complete system, unlike their stand-alone formulation. Especially in the case of classification and *subgroup discovery*, we replace heuristic local pattern mining components by our exhaustive alternative, which is a novel approach to the best of our knowledge.

The resulting systems will also be empirically evaluated and compared to state-of-the-art systems. This empirical comparison is meant to provide insight into both the effectiveness and the efficiency of the CG algorithm for the above mentioned tasks. Whereas the criterion for effectiveness depends on the specific task considered, the efficiency of the algorithms will be measured by the number of patterns evaluated during the search, rather than cpu-time or used memory, because these values are implementation-dependent. We now turn our attention to the different subtasks: correlated pattern mining, subgroup discovery, classification and clustering.

4.1 Correlated Pattern Mining

Problem Description *Correlated pattern mining* [6, 34] is motivated by the observation that association rules with very high confidence may still carry only

little information. If every single person shopping in a grocery store bought bread and every second person bought milk then an association rule $milk \Rightarrow bread$ would have a support of 0.5 and a confidence of 1.0 but still be useless. Therefore, using a correlation measure, grounded in statistical principles, rather than frequency will typically result in more interesting relationships. Reformulated, while classical association rule mining assumes frequent patterns to be interesting, correlated pattern mining looks for local patterns for which the distribution of the target item significantly deviates from the distribution in the entire data set.

Correlated Pattern Mining Using Cluster-Grouping Morishita and Sese [34] model *correlated pattern mining* in the following way – the attribute of interest is restricted to a single, fixed item and the quality of patterns is quantified using the χ^2-statistic to compare expected and observed occurrence counts. *Correlated pattern mining* can be modeled as a *cluster-grouping* problem with:

- $\mathcal{L} = \{I = true \mid I \in \mathcal{I}\}$, with $\mathcal{I} = \{I_1, ..., I_z\}$, the set of items, and $\forall I \in \mathcal{I}$: $\mathcal{V}[I] = \{false, true\}$,
- $\mathcal{E}$ a transaction database,
- σ a (convex) correlation measure such as χ^2,
- τ_{user} the user threshold, and
- $\mathcal{A}_t = \{I_0\}$

Inclusion in the actual solution set is based either on whether a pattern belongs to the k best patterns according to the correlation measure used or on a p-value for the measure. This gives it a sounder statistical interpretation than the setting of a support threshold.

Due to the large number of patterns returned, some can be expected to be considered significant by the measure, however. Without a correction, for instance Bonferroni correction, the result set will therefore include false positives. This problem exists also for a low minimum support threshold, however. In addition, minimum support can be expected to wrongly reject patterns that will be accepted by an uncorrected significance test. Furthermore, techniques exist for efficiently performing such corrections [44], and could be incorporated into CG. The "wrapper" for this approach actually only performs a single call to CG with certain parameters.

Algorithm 3 The correlated pattern mining algorithm

$S = Th_\infty(\mathcal{L}, \chi^2, \mathcal{E}, \tau_{user}, I_0)$

Denoting the local pattern mining step in Algorithm 3 for *correlated pattern mining* the generic notation $Th_k(\mathcal{L}, \sigma, \mathcal{E}, \tau_{user}, \{A_1^T, \ldots, A_d^T\})$ from Definition 6 is used, where the target attribute has been restricted to the item I_0. Furthermore, S denotes the set of computed solutions.

Experimental Evaluation Since the patterns mined by Morishita and Sese are special cases of *cluster-grouping* patterns and the pruning technique is based on the same principles, it follows that the CG algorithm is applicable to *correlated pattern mining* and also that it will produce exactly the same solutions as Morishita and Sese's approach. We therefore include no experiments on this task.

4.2 Subgroup Discovery

Problem Description In *subgroup discovery*, the goal is to find groups of instances in the data that show unexpected behavior with regard to a target attribute. For instance, a higher than expected frequency of lung cancer, as compared to the overall population, in people living in areas with high air pollution or a lower than expected number of cardiac arrests in persons whose diet is rich in olive oil. Again, *subgroup discovery* can be viewed as an attempt to find patterns for which the distribution of values for a given target attribute deviates from their distribution in a different context (e.g. the entire data set or a particular subset [26]).

Subgroup Discovery using Cluster-Grouping Lavrač *et al.* [28] show how a rule learning algorithm such as CN2 [10], used together with a function measuring positive correlation such as *Weighted Relative Accuracy* (*WRAcc*)[27], can be employed to find subgroups. The resulting system is CN2-SD, which we will use here as a representative subgroup discovery system. The local pattern mining component of CN2-SD can be modeled as a *cluster-grouping* problem with:

- $\mathcal{L} = \{A = v \mid A \in \mathcal{A} \setminus \{A_t\}, v \in \mathcal{V}[A]\}$
- $\mathcal{E}$ a data set
- σ is *Weighted Relative Accuracy*
- $k = 1$
- $\mathcal{A}_t = \{A_t = v_i\}$

Given that WRAcc is an asymmetrical measure that rewards higher-than-expected occurrence of a value, and penalizes lower-than-expected occurrence, the target attributes for *subgroup discovery* are derived by turning the actual target attribute into a binary one denoting presence of a value.

The System CN2-SD performs beam search within a "wrapper" that reweights instances that have already been covered to reduce their importance. The complete mining system is shown in Algorithm 4, where $\tau = -\infty$ implies that the k best patterns, regardless of their score, are included.

The local pattern mining step in CG-SD that computes Th_1 is based on an incomplete search strategy, beam search, which does not guarantee that the k best patterns are found. However, this step can also be performed by CG, we call the resulting algorithm CG-SD.

Algorithm 4 The general weighted covering algorithm

$S = \emptyset$
$\forall e_i \in \mathcal{E}$: weight$(e_i) = 1$
repeat
  Find $Th_1(\mathcal{L}, WRAcc, \mathcal{E}, -\infty, A_t = v_i)$
  for all $e_i \in E$ **do**
    if e_i is covered by Th_1 **then**
      $weight(e_i) = \left(\frac{1}{weight(e_i)} + 1\right)^{-1}$
    end if
  end for
  if $Th_1 \notin S$ **then**
    $S = S \cup Th_1$
  end if
until $\forall e_i \in \mathcal{E} : weight(e_i) < 1$

Table 3. Comparison for induction of a single subgroup per class value, setting A. The first column lists the data set, the last columns the number of candidate pattern evaluated by CG-SD, corresponding to 100%, columns 2–4 the corresponding percentage-values for different settings of CN2-SD.

Dataset	CN2-SD$_{20}$	CN2-SD$_{10}$	CN2-SD$_5$	CG-SD
Balance-2-Class	644.00%	436.00%	278.00%	50 (100%)
Breast-W	3443.04%	1791.14%	948.10%	79 (100%)
Breast-W-equal	3061.36%	1609.09%	856.82%	88 (100%)
Car	1722.22%	898.08%	481.61%	261 (100%)
Colic	10569.26%	5336.49%	2723.65%	296 (100%)
Colic-equal	10699.64%	5394.31%	2772.95%	281 (100%)
Credit-G	2106.84%	1062.73%	541.76%	1492 (100%)
Credit-G-equal	2036.56%	1028.06%	523.89%	1436 (100%)
Diabetes	2445.24%	1329.76%	705.95%	84 (100%)
Diabetes-equal	1014.78%	550.25%	291.63%	203 (100%)
Heart-H	3682.01%	1876.19%	976.72%	189 (100%)
Heart-Statlog	2639.30%	1342.36%	696.07%	229 (100%)
Heart-Statlog-equal	2416.27%	1227.38%	637.70%	252 (100%)
Krkopt	1463.92%	765.52%	413.20%	2697 (100%)
Mfeat-Morpho	2090.53%	1244.44%	672.43%	243 (100%)
Mfeat-Morpho-equal	2086.42%	1249.38%	676.95%	243 (100%)
Nursery	3283.92%	1692.60%	888.75%	311 (100%)
Segment	7784.20%	3949.24%	2015.46%	595 (100%)
Tic-Tac-Toe	1717.58%	879.69%	461.33%	256 (100%)
Voting Record	7201.55%	3655.04%	1883.72%	129 (100%)
Zoo	13206.91%	6714.63%	3400.96%	1982 (100%)
Pendigits	5523.76%	2800.83%	1313.24%	846 (100%)
Mushroom	11928.74%	5997.13%	3074.71%	522 (100%)
Average	4155.07%	2119.10%	1090.17%	

Experimental Evaluation To compare the complete search method of CG-SD with the heuristic approach of CN2-SD, we set up experiments to answer the following questions:

Q1 Does CN2-SD find all subgroups found by CG-SD?
Q2 Is CN2-SD more efficient than CG-SD?

The evaluation is performed in two settings:

1) without the wrapper, where we search for a single top-scoring subgroup, and
2) with the wrapper, where we look for an incrementally constructed set of subgroups.

To answer the questions posed above, we perform experiments on a number of UCI data sets, which were selected such that a large range of data cardinality and dimensionality were covered. The implementation of CG is currently limited to nominal data. Therefore, numerical attributes have been discretized for the experiments, and we only chose data with discrete classes. Two unsupervised discretization approaches were chosen. In the naïve version, the mean value of an attribute is computed and taken as threshold, leading to two nominal values. For some data sets this has the effect that one of the two bins contains *far* more instances than the other one. For these sets, we also chose the threshold in such a way that two roughly equally distributed nominal values result. These data sets are denoted by a trailing "-equal" in the name.

Experimental Setting The experimental settings are the following:

- The attribute of interest is the class label
- Beam sizes for CN2-SD are 5,10,20[1]
- σ is *WRAcc*

Results Tables 3 and 4 report the number of candidate patterns evaluated by CG-SD – which corresponds to 100% – and the corresponding percentage-values for different settings of CN2-SD. Additionally, the tables specify whether CG-SD found a subgroup description, i.e. a pattern, that is better, i.e. has a higher *WRAcc*-score, than the one induced by CN2-SD during one of the iterations.

For setting **1)**, the single subgroup case, CG-SD always evaluates far less candidate patterns than the beam search, for all settings of beam size. For this setting, CN2-SD did find the highest-scoring subgroup for each data set.

For setting **2)**, the result of the single-subgroup run carries over to the suggested setting of beam size 20, even though the difference is not as pronounced as before. While for some data sets CN2-SD needs less candidate patterns than CG-SD for beam sizes 5 and 10, the heuristic technique also fails to find the top-scoring subgroups for these settings. On average, CG-SD needs 5 to 20 times less candidate evaluations, and this factor correlates strongly with the

[1] 20 was suggested by a reviewer, 5 and 10 evaluated as well as to not bias the efficiency estimation against CN2-SD.

Table 4. Comparison of a complete *subgroup discovery* run, setting B. First column lists the data set, last columns the number of candidate pattern evaluated by CG-SD, corresponding to 100%, columns 2–4 the corresponding percentage-values for different settings of CN2-SD.

Dataset	CN2-SD$_{20}$	CN2-SD$_{10}$	CN2-SD$_5$	CG-SD
Balance-2-Class	537.37%	356.48%	214.86%	471 (100%)
Breast-W	1588.41% •	865.23% •	442.74% •	179625 (100%)
Breast-W-equal	1475.91% •	800.42% •	504.28% •	117251 (100%)
Car	689.71%	350.11%	184.15%	61609 (100%)
Colic	1109.36% •	563.09% •	285.68% •	395291 (100%)
Colic-equal	892.98% •	458.43% •	218.20% •	476363 (100%)
Credit-G	231.50% •	113.24% •	55.72% •	543376 (100%)
Credit-G-equal	152.01% •	66.88% •	32.22% •	684859 (100%)
Diabetes	1948.31% •	1061.99% •	486.26% •	8030 (100%)
Diabetes-equal	316.79% •	169.46% •	88.00% •	13836 (100%)
Heart-H	1223.74% •	617.57% •	391.68% •	22415 (100%)
Heart-Statlog	1263.71%	911.69% •	479.43% •	5509 (100%)
Heart-Statlog-equal	1178.30%	595.25%	304.50%	6692 (100%)
Krkopt	655.85% •	337.64% •	182.00% •	394671 (100%)
Mfeat-Morpho	1724.76%	1017.61%	530.07%	11775 (100%)
Mfeat-Morpho-equal	1465.44%	864.24%	450.13%	12052 (100%)
Nursery	2082.05%	1066.15%	552.82%	975 (100%)
Segment	5610.12%	2835.89%	1410.76%	19293 (100%)
Tic-Tac-Toe	771.75%	391.34%	201.31%	2818 (100%)
Voting Record	1500.65% •	2454.68% •	2540.43% •	3643096 (100%)
Zoo	13206.91%	6714.63%	3400.96%	1982 (100%)
Pendigits	2705.73% •	1443.45% •	733.19% •	279217 (100%)
Mushroom	10002.78%	4870.87% •	2377.40% •	8900 (100%)
Average	2210.15%	1150.94%	588.10%	

• denotes that a non-optimal subgroup, that is a subgroup having a lower score than the highest possible, has been found

used beam-size. Note, finally, that even a beam size of 20 does not ensure that all highest-scoring subgroups are found by CN2-SD!

As a consequence, the answers to **Q1** and **Q2** are negative for both settings **1)** and **2)**, and CN2-SD is neither as effective as CG-SD, nor more efficient. Especially the second finding is surprising and significant since it clearly indicates that heuristic search for rules, which is common practice in machine learning, is suboptimal, both in terms of quality of found solutions and in terms of efficiency. It therefore seems more appropriate to employ branch-and-bound algorithms whenever the evaluation measure is upper-boundable or convex.

Related Work In addition to the CN2-SD algorithm, against which we evaluated our approach, there have been other systems for exhaustively searching of subgroup descriptions. The earliest one is the EXPLORA system introduced, cf. [26]. The system supports a wide variety of interestingness measures, both convex and non-convex, and both heuristic and exhaustive search strategies.

Newer work includes the APRIORI-SD [25] and SD-MAP [2] systems which adapt frequent pattern mining techniques to the task of subgroup discovery. Both of these systems show a conceptual difference to SD-CN2 and SD-CG, however: they mine the k most interesting subgroups in relation to the background distribution *in the entire data*. The sequential approaches evaluated in this section, on the other hand, assume that knowledge of discovered subgroups should inform the assessment of future subgroup descriptions. In addition, there is an algorithmic difference to the CG-based approach: instead of using *WRAcc* directly for pruning the space of descriptions, this pruning is based on a minimum frequency threshold. The selection of the actual descriptions is performed as filtering by interestingness in a post-processing step. While this allows for the use of non-convex measures, as Atzmüller *et al.* point out in [2], it relies on the specification of a meaningful frequency threshold by the user.

Finally, there has been work on finding bounds on the quality of subgroup descriptions. A weaker upper bound for the *WRAcc* measure can be found in [46], and Scheffer *et al.* [40] proposed a sequential database sampling scheme for approximating the k-best subgroup descriptions problem. The latter work allows a PAC-like bound on the quality of the final solution but as the authors point out, it is not applicable to measures such as the convex χ^2-statistic.

4.3 Finding Rules for Classification

Problem Description *Classification* is related to *subgroup discovery*, because rules for classification concern groups whose class distribution differs from the default one. Once rules describing such groups are found, they can be used to predict the value of the class attribute. The main difference with *subgroup discovery* is that rule-based *classification* aims at inducing a set of rules that, taken together, correctly predict the *entire* set of training instances.

Classification and Cluster-Grouping: Since the goal of classification can be re-interpreted as finding rules that separate two classes from one another, measures such as χ^2 and *Information Gain* can be used as well as – to a certain degree – accuracy to solve the task in the *cluster-grouping* framework. The problem then has the following characteristics:

- $\mathcal{L} = \{A = v \mid A \in \mathcal{A} \setminus \{A_t\}, v \in \mathcal{V}[A]\}$
- $\mathcal{E}$ a data set
- σ is *accuracy*, χ^2, *Information Gain* or *Category Utility*
- $k = 1$,τ_{user}
- $\mathcal{A}_t = \{C\}$, where C is the class attribute

Note, that when accuracy is used, its asymmetry will require the same binarization of the target attribute as for the *subgroup discovery* setting.

The System Rule-based classifiers often rely on the covering paradigm [22]. Our focus lies essentially on the sequential covering paradigm in which patterns are mined, covered instances removed, and the process iterated on the remainder of the data set. Mining can either be done in a greedy way, e.g. optimizing some measure's score using beam search [10], or exhaustively, by setting thresholds on e.g. the support and confidence of interesting patterns and performing the covering step as post-processing [30, 29].

This gives rise to two possible "wrappers", which are shown in Algorithms 5 and 6: sequential covering and complete mining.

Algorithm 5 The sequential covering algorithm.

$S = \emptyset$
repeat
  $\mathcal{E} = \mathcal{E} \setminus \{e | e \text{ covered by } Th_1(\mathcal{L}, \sigma, \mathcal{E}, \tau_{user}, C = c_1)\}$
  $S = S \cup Th_1$
until $\mathcal{E} = \emptyset \vee Th_1 = \emptyset$
return S

Algorithm 6 The complete mining algorithm.

$S = \text{post-process}\big(Th_k(\mathcal{L}, \sigma, \mathcal{E}, \tau_{user}, C = c_1)\big)$

By instantiating the inner loop of the sequential covering algorithm (Algorithm 5) with CG we derive CN2-CG, by using a beam search maximizing χ^2, CN2$_{\chi^2}$. We empirically compare those two techniques below, also to RIPPER [12], one of the most sophisticated sequential covering algorithms that use beam search.

For the complete mining algorithm, sketched in Algorithm 6, one choice is to use *minimum frequency* to estimate significance; the resulting system has been introduced as CBA [30] – *classification based on association*. Alternatively, by using CG instead for mining the significant patterns, we obtain the novel CBC – *classification based on correlation* algorithm, which we empirically evaluate below.

Experimental Evaluation As demonstrated in the section on *subgroup discovery*, beam size is important for both the quality of found solutions, and the efficiency of the mining technique, when comparing CN2$_{\chi^2}$ to CN2-CG. At the same time, we are also interested in comparing the performance to RIPPER. This yields to the following reformulations of **Q1** and **Q2** for the sequential covering approaches:

Q3 How does the quality of the rules found by CN2$_{\chi^2}$, CN2-CG and RIPPER compare?
Q4 Is CN2$_{\chi^2}$ more efficient than CN2-CG?

However, for measuring the quality of the discovered solutions in the classification setting, we employ classification accuracy rather than the correlation measures used for *subgroup discovery*. We are also interested in a comparison to the state-of-the-art system for classification RIPPER.
For the comparison of the complete mining algorithms, the following questions result:

Q5 How does the quality of CBA's classifiers compare to those of CBC?
Q6 Is CBA more efficient than CBC?

Experimental setup The experimental setting for the sequential covering approach are as follows:

- Beam sizes for CN2$_{\chi^2}$ are 5, 10, 20.
- Minimum significance threshold for CN2$_{\chi^2}$ and CN2-CG is 3.84.
- RIPPER is run as WEKA's [20] JRIP implementation with default parameters and pruned classifiers evaluated.
- CN2$_{\chi^2}$- and CN2-CG-classifiers are unpruned.

For the exhaustive techniques, the following parameter settings are used:

- WEKA's APRIORI implementation is used in CBA, with 1% minimum support and 50% minimum confidence.
- Minimum significance threshold of CBC is 3.84.
- Maximum number of mined rules is $50,000$[2]

[2] An exception is the *Kr-vs-KP* data set where $90,000$ rules are needed for CBA to find rules with confidence of at least 90%

– CBC mines only the 1000 most accurate rules, a restriction motivated by an observation in [36], that the rules chosen for the final classifier fall well within the 1000 highest-ranked rules.

The data sets used were again discretized. The discretization scheme was more sophisticated than in the *subgroup discovery* experiments, using Fayyad and Irani's supervised discretization method [15]. The disretization algorithm was run on the training folds, with the resulting intervals used on test data, such as not to introduce bias into the data.

Table 5. Average accuracy and standard deviation for $CN2_{\chi^2}$**,** CN2-SD**,** RIPPER**,** CBA**, and** CBC**.** The left-most column lists data sets, columns 2-4 accuracy estimates for sequential covering approaches (2 & 3 annotated with statistical t-test comparison to RIPPER), $CN2_{\chi^2}$ results are also annotated with the width of the beam giving rise to the result, columns 5 & 6 list exhaustive techniques (column 6 annotated with t-test comparison to CBA).

Dataset	$CN2_{\chi^2}$	CN2-CG	RIPPER	CBA	CBC
Balance (2 Class)	86.8 ± 3.90 (5)$\circ$	$86.8 \pm 3.9\circ$	80.0 ± 3.4	79.18 ± 4.59	79.18 ± 4.59
Breast-Cancer	81.5 ± 8.4 (10)	80.4 ± 0.8	71.7 ± 0.7	68.19 ± 8.48	66.77 ± 9.28
Breast-W	96.4 ± 2.40 (5)	96.4 ± 2.4	95.7 ± 2.1	94.71 ± 1.90	95.71 ± 1.34
Colic	82.9 ± 5.50 (5)	88.9 ± 5.9	83.9 ± 7.7	81.27 ± 8.07	76.91 ± 6.51
Credit-A	86.5 ± 2.5 (20)	85.8 ± 2.1	85.4 ± 2.5	85.65 ± 4.35	84.06 ± 4.48
Credit-G	79.4 ± 6.0 (10)$\circ$	$79.4 \pm 6.0\circ$	69.4 ± 5.4	71.40 ± 2.63	69.80 ± 4.89
Diabetes	77.4 ± 5.4 (10)	75.1 ± 6.2	76.0 ± 3.9	75.92 ± 4.14	75.78 ± 4.23
Heart-H	83.3 ± 7.50 (5)	81.6 ± 6.3	79.2 ± 7.4	83.33 ± 6.69	82.66 ± 5.82
Kr-vs-Kp	94.3 ± 1.40 (5)$\bullet$	$94.3 \pm 1.4\bullet$	99.3 ± 0.4	80.72 ± 1.75	$95.63 \pm 1.29\circ$
Mushroom	98.5 ± 0.30 (5)$\bullet$	$98.5 \pm 0.3\bullet$	100.0 ± 0.0	99.53 ± 0.19	$100.00 \pm 0.00\circ$
Spambase	91.4 ± 1.4 (10)	$89.0 \pm 1.4\bullet$	92.7 ± 1.1	86.39 ± 1.62	86.09 ± 1.61
Tic-Tac-Toe	84.6 ± 2.20 (5)$\bullet$	$83.1 \pm 2.2\bullet$	97.1 ± 1.2	100.00 ± 0.00	100.00 ± 0.00
Voting Record	95.3 ± 3.20 (5)	96.2 ± 3.0	95.6 ± 2.8	94.25 ± 3.10	93.10 ± 3.58

$\circ$ denotes statistical wins at the 99% level, base-line being RIPPER, and CBA, respectively
$\bullet$ denotes statistical losses at the 99% level, base-line being RIPPER, and CBA, respectively

Results $CN2_{\chi^2}$, CN2-CG, and RIPPER give rise to solutions of similar quality, cf. Table 5, with RIPPER being significantly better than $CN2_{\chi^2}$ three times (four times *vs* CN2-CG), χ^2-based optimization being significantly better twice. With $CN2_{\chi^2}$ outperforming CN2-CG once, one can conclude w.r.t. **Q3** that the quality of the heuristically derived classifiers is at least equal to the ones found using CN2-CG. The sequential covering approach allows the correction of errors in the induction of local patterns by steering the search process for the overall classifier in the right directions. It should be noted however that selecting the right beam size is non-trivial, mirroring the results of the *subgroup discovery* experiments.

Two of the cases in which RIPPER finds the better solution are large data sets with rules that have high accuracy on only small subsets (*Kr-vs-Kp, Tic-Tac-Toe*). This indicates that significance measures at some point tend to penalize low-frequency rules too much. On the other hand, the χ^2-based approaches significantly outperform RIPPER on the *Balance* and *Breast-Cancer* data sets, both of which are rather small, more likely leading to overfitted rules, which are discounted by the significance estimate of χ^2. In contrast, all data sets on which RIPPER performs well are large - giving a large enough sample to counteract the overfitting stemming from accuracy maximization. Thus there seems to be a slight advantage provided from the sophisticated pruning techniques of RIPPER which has less of an effect on small data sets though.

The comparison of CBC with CBA shows that using χ^2 instead of support to measure significance (and ranking them accordingly) gives better results. CBC never performs significantly worse than CBA and in two cases is significantly better, answering **Q5** positively regarding CBC effectiveness. In the case of the *Mushroom* data set the ordering of the rule set before pruning is decidedly different between the two approaches and thus different rules are selected for the final classifier. In the *Kr-vs-Kp* scenario, limiting the mining process to the $90,000$ most significant rules according to support excludes many high-confidence rules. Even at $200,000$, the highest confidence is at just 0.92, while for CBC rules with confidence 1.0 are found within the $50,000$ most significant rules according to χ^2.

Table 6. Average number of patterns mined by the CN2_{χ^2}**, and** CN2-CG**, number of patterns mined and used by** RIPPER

	CN2_{χ^2}	CN2-CG	RIPPER	
Dataset	# mined	# mined	# mined	# used
Balance (2 Class)	6.0 ± 1.00	5.4 ± 2.12	8.7 ± 2.35	5.2 ± 1.22
Breast-Cancer	27.1 ± 6.30	24.6 ± 6.10	11.2 ± 3.97	3.1 ± 0.74
Breast-W	12.8 ± 1.30	11.4 ± 0.80	19.8 ± 2.04	6.6 ± 0.97
Colic	16.5 ± 2.90	8 ± 0.90	12.9 ± 2.23	3.6 ± 0.70
Credit-A	16.2 ± 2.40	14.6 ± 1.84	25.5 ± 2.42	5.8 ± 1.81
Credit-G	31.1 ± 6.10	30 ± 3.37	15.3 ± 2.79	5.5 ± 2.12
Diabetes	10.0 ± 2.30	11.6 ± 3.37	20.1 ± 3.14	5.2 ± 0.91
Heart-H	9.0 ± 2.00	8.8 ± 1.75	10.8 ± 1.39	3.5 ± 0.85
Kr-vs-Kp	3.0 ± 0.00	2.0 ± 0.00	19.2 ± 1.13	15.4 ± 1.27
Mushroom	4.3 ± 0.50	3 ± 0.00	8.8 ± 0.79	8.7 ± 0.68
Spambase	10.6 ± 1.00	5.8 ± 0.92	53.9 ± 3.28	27.3 ± 3.23
Tic-Tac-Toe	6.3 ± 1.30	8.8 ± 0.40	12.1 ± 2.02	10.6 ± 1.27
Voting Record	4.8 ± 0.6	3.2 ± 0.42	8.8 ± 0.63	2.9 ± 1.20

Since the quality of found rules for the heuristic (CN2_{χ^2}) and complete (CN2-CG) χ^2 maximization is very similar, **Q4** focusses on whether one of the two techniques is more efficient. Note that the introduction of beam search mainly attempts to make the search space manageable by focusing on certain

subspaces. Upper bound pruning on the other hand, uses a different kind of restriction, also with the aim of focussing on the relevant parts of the search space. Table 6 shows that CN2-CG often, though not always, selects fewer rules than $CN2_{\chi^2}$, apparently capturing the underlying regularities better.

Table 7. Number of candidate patterns evaluated by $CN2_{\chi^2}$ (for beam size giving the best solution) and CN2-CG. Column three lists number of pattern evaluated by CN2-CG, equating 100%, column two the corresponding percentage value for $CN2_{\chi^2}$

Dataset	$CN2_{\chi^2}$	CN2-CG
Balance (2 Class)	139.83%	261.60 (100%)
Breast-Cancer	108.00%	46153.30 (100%)
Breast-W	101.69%	6817.00 (100%)
Colic	2277.10%	2288.70 (100%)
Credit-A	10.76%	1003999.50 (100%)
Credit-G	1.35%	17061266.50 (100%)
Diabetes	40.60%	13030.90 (100%)
Heart-H	36.27%	17809.40 (100%)
Kr-vs-Kp	5.46%	240431.00 (100%)
Mushroom	90.66%	28758.60 (100%)
Spambase	16.14%	2828763.30 (100%)
Tic-Tac-Toe	102.56%	2975.40 (100%)
VotingRecord	39.17%	11726.50 (100%)
Average	228.43%	

The number of candidate patterns evaluated, shown in Table 7, does not give a clear answer to question **Q4**. On several occasions CN2-CG is better, most pronounced for the *Colic* data set, while e.g. on *Credit-G* $CN2_{\chi^2}$ finds effective rules far quicker. A possible explanation is that effective rules are found early, the upper bound is not tight enough though, thus exploring large parts of the search space without gaining anything. The main insight is again that using exhaustive search can be as efficient as heuristic search, given the right pruning mechanisms, while in addition it guarantees optimality. Thus, complete branch-and-bound algorithms seem to be preferable to heuristic ones.

The final question to be answered is **Q6**, namely whether CBA is more efficient than CBC. Table 8 shows again no clear-cut advantage for either technique. On average CBA mines slightly fewer patterns than CBC .

Again, large data sets on which accurate rules have small coverage, and data sets with minority classes make upper-bound pruning less effective. More specifically, basing *associative classification* mining on CG compares worst on *Kr-vs-Kp*, *Spambase*, and *Tic-Tac-Toe*. We have seen however that *Kr-vs-Kp* gives also CBA trouble and subsequent experiments in which the number of mined patterns is set to 1.8 million still does not give classifiers comparing well with the CG-solution while exceeding its number of evaluated candidate patterns significantly.

Table 8. Number of candidate pattern evaluated by the complete mining algorithms The last column lists number of patterns for CBC, equating 100%, column two shows the corresponding percentage value for CBA

Dataset	CBA	CBC
Balance (2 Class)	156.01%	99.80 (100%)
Breast-Cancer	127.44%	8179.20 (100%)
Breast-W	52.58%	12913.80 (100%)
Colic	136.29%	73682.90 (100%)
Credit-A	98.49%	65226.50 (100%)
Credit-G	39.14%	155020.90 (100%)
Diabetes	126.52%	3875.80 (100%)
Heart-H	172.57%	14680.00 (100%)
Kr-vs-Kp	15.92%	687715.50 (100%)
Mushroom	95.86%	53615.80 (100%)
Spambase	15.13%	445856.30 (100%)
Tic-Tac-Toe	38.14%	24511.10 (100%)
Voting Record	65.41%	88996.10 (100%)
Average	87.65%	

To summarize, the experiments show that using statistically well-founded measures improves the prediction accuracy of heuristic methods on small data sets and generally improves upon the accuracy of frequency-based *associative classification* methods. While the efficiency of CG-based techniques is on average as good as or better than the alternative approaches', for particular data sets existing methods can outperform CG. These findings suggest 1) that the robustness of sequential covering algorithms such as RIPPER or CN2 that use beam search, a heuristic technique, may be improved improved by using CG 2) that it may be advantageous to replace the use of support and confidence in *associative classification* techniques such as CBA [30] and CMAR [29] by using correlation measures grounded in statistical theory, 3) that also decision tree approaches, who choose an optimal pattern based on a single attribute, may profit from using CG instead, cf. also our approach to clustering below and the Tree^2 approach of [7].

Related Work To the best of our knowledge, all rule-based approaches to classification mine local patterns in some way and build classifiers from them. Decision trees solve the problem of finding the optimal splitting pattern by essentially limiting the number of conditions to one. There has been work on multi-variate splitting criteria [35] - there, similar decisions on the induction mechanism have to be made as in sequential covering. Sequential covering algorithms like RIPPER or CN2 use beam search, a heuristic technique, inside the covering loop and their robustness could be improved by using CG. *Associative classification* techniques such as CBA [30] and CMAR [29] mine patterns based on user-specified values for support and confidence. Both approaches rely on the declaration of parameters by the user. The exhaustive algorithms have parameters which are rather difficult to decide upon but which can have an important

effect on the resulting set [11]. Additionally, the result set often is made up of a very large number of rules, making interpretation by the user difficult. Basing the choice of cut-off value on statistical theory and pushing the significance test inside the mining step should improve both efficiency and effectiveness of such techniques.

4.4 Conceptual Clustering

Problem Description In *clustering*, the goal is to partition the instances of a data set into typically disjoint subsets (clusters) that exhibit high *intra*-cluster *similarity* and high *inter*-cluster *dissimilarity*. For the numerical case clusters can be represented, for instance, by centroids or medoids and the similarity quantified by a vector norm such as the L1 or L2 norm.

In *conceptual clustering*, clusters have to be described in terms of nominal values instead. This usually also means that the instances that are clustered have nominal values (potentially in addition to numerical ones). A similarity measure is then often harder to define. In general, instances are considered similar if they agree on the values of many attributes. One measure for judging the quality of a set of clusters is *Category Utility* [23] though others have been defined in the literature as well.

Clustering and Cluster-Grouping Clusters are often arranged into a hierarchy, with clusters closer to the root of the clustering tree (or *dendrogram*) described by more general concepts. Such a dendrogram can be obtained using either a divisive or an agglomerative approach. Here, we focus on a divisive approach, which bears some similarities to decision tree induction, in which clusters are repeatedly divided into sub-clusters according to some criterion. According to Höppner et al. [24] clusters can be considered as deviations in distribution from a default (or background) distribution w.r.t. certain attributes. Therefore, CG can be used to identify patterns that capture the deviating areas and can be used to split the clusters. Using CG assures the best split without restarts of the clustering algorithm and allows the induction of conjunctive descriptions for clusters. Maximizing e.g. *Category Utility* – with binary attributes only – is a *cluster-grouping* task with the following characteristics:

- $\mathcal{L} = \{A = v \mid A \in \mathcal{A}, v \in \mathcal{V}[A]\}$, where $\mathcal{A} = \{A_1, ..., A_d\}, \forall A \in \mathcal{A} : \mathcal{V}[A] = \{true, false\}$
- $\mathcal{E}$ a data set
- σ is *Category Utility*
- $k = 1$
- $\mathcal{A}_t = \{A_i \mid A_i \in \mathcal{A}\}$

Since the goal, as mentioned above, is similarity in as many attributes as possible, *all* attributes are considered targets, with the symmetric measure *CU* leading to the induction of patterns in whose coverage space either the occurrence of *true* or false values will be higher than expected.

The System A dendrogram is a decision tree-like structure, with cluster membership decided by the concepts in the nodes. Therefore the wrapper for a CG-based clustering algorithm is a bit more involved, compared to the other wrappers, mirroring recursive decision tree algorithms. Algorithm 8, which we term CG-Clus, thus calls a function Split that recursively constructs a tree, whose inner nodes denote patterns (or their negations). This is to the best of our knowledge the first time that correlation based patterns are used in divisive clustering approach.

Algorithm 7 Split

Input: Data set $\mathcal{E}$, Leaf node t
if $Th_1(\mathcal{L}, \mathrm{CU}, \mathcal{E}, -\infty) \neq \emptyset$ **then**
  $\mathcal{E}_1 = \mathcal{E} \setminus \{e | e \text{ covered by } Th_1(\mathcal{L}, \mathrm{CU}, \mathcal{E}, -\infty)\}$
  $\mathcal{E}_2 = \mathcal{E} \setminus \mathcal{E}_1$
  Create left child of t, t_l, containing $Th_1(\mathcal{L}, \mathrm{CU}, \mathcal{E}, -\infty)$
  Create right child of t, t_r, containing $\neg Th_1(\mathcal{L}, \mathrm{CU}, \mathcal{E}, -\infty)$
  Split$(\mathcal{E}_1, t_l)$
  Split$(\mathcal{E}_2, t_r)$
end if

Algorithm 8 CG-Clus

$T = \emptyset$
Split$(\mathcal{E}, T)$
return T

A second approach is *cluster mining* [38], where a clustering algorithm is used to find a clustering, each cluster treated as a single class, and conjunctive concepts learned on them. Afterwards, all instances matching a concept are considered to be in one cluster, possibly producing overlapping clusters.

To evaluate CG-Clus, we shall compare it to Cobweb [16], the arguably best-known conceptual clustering technique, and a cluster mining technique using Autoclass [9] and Ripper. Cobweb iteratively processes instances, using four operators: assigning an instance to an existing dendrogram node, creating a new node, splitting an existing node, or merging two existing nodes.

Experimental Evaluation Cobweb's direct assignment and iterative processing gives it great flexibility in assembling clusters but also makes it vulnerable to ordering effects in data. In addition, by using conditional probability vectors instead of conjunctions to describe the clusters, it has fewer restrictions which instances to cluster together. Thus, a question pertaining to Cobweb is:

Q7 Do Cobweb's clusterings have higher CU than the ones of CG-Clus?

This question is meant to provide an insight into the effectiveness of CG-Clus.

Autoclass is based on Bayesian principles and thus not directly optimizing *CU*. On the other hand, it has greater flexibility than CG-Clus in assigning instances *directly* to clusters – not indirectly via the found description. In addition, decoupling the processes of forming clusters and finding a description gives the actual concept formation greater flexibility than CG-Clus possesses. Since both Cobweb and a *cluster-mining* approach have greater flexibility (and make conjunctive concept formation a non-integral part of the mining process), two further questions are:

Q8 How similar are CG-Clus' and Cobweb's/Autoclass' clusterings?
Q9 How complex are conjunctive descriptions of Cobweb's/Autoclass' clusters, compared to CG-Clus' ones, and how much information about the underlying instances is recovered?

Experimental setup To compare the agreement of two clusterings, we use the *Rand* index, which is the fraction of pairwise grouping decisions on which the two clusterings agree. Let $\mathcal{E} = \{e_1, ..., e_n\}$ be a data set and $\mathcal{C}_1, \mathcal{C}_2$ two clusterings of $\mathcal{E}$. For each pair of instances e_i, e_j, $\mathcal{C}_l$ either assigns them to the same cluster or to different clusters. Let *same* be the number of decision where e_i, e_j are in the same cluster in both clusterings and *diff* the number of decisions where they belong to different clusters in both $\mathcal{C}_l$. The *Rand Index* is defined as:

$$Rand(\mathcal{C}_1, \mathcal{C}_2) = \frac{same + diff}{n * (n-1)/2}$$

If the number of clusters in a clustering is different, the *Rand* index will obviously show a dissimilarity. Therefore, we attempted to form a number of clusters corresponding to the number of class values on each data set in our experiments.

To obtain a given number of clusters from a Cobweb dendrogram, there are two possibilities – the user selects certain nodes in the tree, disregarding the structure underneath them, or the growth of the dendrogram is limited.

In the first case, the fact that Cobweb often constructs dendrograms in which every instance is sorted into its own cluster, makes this a non-trivial procedure. Also, selecting all nodes from the same level of the tree does not guarantee a good solution *CU*-wise.

Instead, in the WEKA implementation, a minimum *CU*-gain can be set which determines whether new nodes in the dendrogram are introduced, or existing nodes split. By starting out with a lenient threshold, systematically tightening it when more than the desired number of clusters is formed and relaxing it when the tightening proved to be too strict, it is possible to approximate the desired number of clusters.

Unfortunately, this method does not always guarantee obtaining the target number of clusters, since Cobweb sometimes forms just one cluster or tens/hundreds depending on a 0.0001 difference in the threshold value. Instead of arbitrarily merging clusters, we used the Cobweb-solution whose number of

clusters is closest to the actual number of classes, unless this number is 1, i.e. all instances were sorted into the same cluster. After determining the COBWEB-clustering, we attempted to construct the same number of clusters using CG-CLUS.

AUTOCLASS can be supplied with the number of clusters it should create. For each data set, AUTOCLASS performed 250 restarts with 200 iterations each. The assumed model was single multinomial for all attributes. We used the best clustering found for comparison with the CG-based approach.

As for our technique, CG-CLUS, to obtain the desired number k of clusters, the $k-1$ best splits are used. Since a good *CU* score on a small subset is easier to achieve than on a larger one, patterns' scores are weighted with the proportion of instances of the complete set that they were derived on. The resulting dendrogram is decision tree-like in the way data is split on patterns and their impact discounted on the population size.

Owing to the need for binary attributes, discretization was performed as in the *subgroup discovery* experiments, and nominal attributes binarized.

Results To answer **Q7** and **Q8**, we report *CU*-values and the Rand-index for CG-CLUS- and COBWEB-clusterings for a variety of data sets in Table 9. For the data sets for which hundreds or even thousands of clusters were formed by COBWEB we did **not** attempt to form the same number of clusters using CG. Instead we report on the *Category Utility* of the "correct" solution of CG-CLUS (i.e. the clustering having as clusters the classes in the data), the average *CU* for COBWEB and no value (N/A) for the *Rand*-index. In the cases where we match COBWEB's number of clusters, these can vary for different runs, leading to variations of the *CU* which we report. If we do not match the number of clusters COBWEB produces, we report only a single *CU*-value since CG-CLUS forms only one partition with as many clusters as classes in the data.

The resulting *Category Utilities* show that far from always giving rise to superior scores by using the more flexible clustering scheme, the quality of COBWEB's solution is clearly affected by ordering effects in the data. If the right ordering of instances exists, COBWEB constructs very good solutions, if not, *CU* values are rather low or the dendrogram is huge. When threshold differences of 0.0001 make the difference between a single cluster and a dendrogram having hundreds of leaves it is difficult for the user to make an informed decision on which clusters to merge. For the data sets where a reasonable number of clusters was constructed, COBWEB's average *CU* is larger than that of CG-CLUS four times, less six times, while at the same time exhibiting similarities of the clusterings in excess of 0.7. This means that **Q7** has to be answered negatively, COBWEB does not always translate the greater flexibility of its assignment mechanism into better *CU* values. Also, the solutions are rather similar, giving the answer to **Q8**.

Table 10 is used to give insight into question **Q9**. It lists the number of classes in the data, the average number of clusters in COBWEB's clusterings over ten runs, the number of rules learned by RIPPER (unpruned) on these clusters and their accuracy. It should be noted that CG-CLUS builds a tree of conjunctive

Table 9. *CU* of the CG-Clus clusterings, and *CU* of Cobweb's solution, averaged over 10 runs, Rand-index of the two clusterings

Dataset	CU_{CG}	CU_{CW}	Rand
Credit-G	0.4408	0.0239 ± 0.002	N/A
Credit-G-Equal	0.4753	0.1161 ± 0.0293	N/A
Kr-vs-Kp	0.5343 ± 0.0040	0.5782 ± 0.0012	0.7817 ± 0.0029
KrkOpt	0.1536 ± 0.0072	0.1369 ± 0.002	0.7396 ± 0.028
Letter	0.1742 ± 0.0075	0.1342 ± 0.0151	0.7629 ± 0.0275
Letter-Equal	0.1677 ± 0.0053	0.1439 ± 0.0063	0.8759 ± 0.0001
Mfeat-Fourier	0.4743	0.4487 ± 0.1334	N/A
Mfeat-Fourier-Equal	0.7183	0.1855 ± 0.0289	N/A
Mfeat-Karhunen	0.457	0.0203	N/A
Nursery	0.3555	0.0846 ± 0.0273	N/A
Optdigits	0.4609 ± 0.0029	0.5234 ± 0.0229	0.7865 ± 0.0148
Optdigits-Equal	0.6865 ± 0.0203	0.7936 ± 0.0565	0.8509 ± 0.0069
Pendigits	0.4336 ± 0.0091	0.4015 ± 0.0074	0.8519 ± 0.0004
Segment	0.5878 ± 0.0122	0.5438 ± 0.0505	0.7994 ± 0.1029
Segment-Equal	0.7925 ± 0.0083	0.7916 ± 0.0063	0.8984 ± 0.0039
Waveform	0.9791	1.1624 ± 0.0229	0.7822 ± 0.0192

descriptions (and their negations) for clusters – which have 100% accuracy – and can easily be constrained to form as many clusters as classes exist in the data.

The experiments show that Cobweb hardly forms a number of clusters that corresponds to the number of underlying classes. In addition, most of the time far more rules than classes will be learned on the data, which do not always capture the clusters very well. So the conjunctive descriptions found using Cobweb's clusterings are at the same time rather complex and not always reliable.

Regarding the cluster mining solution using Autoclass and Ripper, Table 11 lists the number of classes per data set (both CG-Clus and Autoclass form the same number of clusters), the number of rules learned by Ripper and their accuracy on Autoclass' solution, as well as the *Rand*-index of the two clusterings.

The similarity of the clusterings produced by the two methods is generally very high, with three exceptions, thus answering **Q8**. While Autoclass' solutions give rise to smaller descriptions than Cobweb's do, and Autoclass' rules achieve a far higher accuracy on the underlying clusters, they still exceed the number of clusters (and thus conjunctive descriptions in CG-Clus' tree) by far. This means that while being rather close in actual composition, the description of Autoclass' solution is far more complex.

To summarize, while being less flexible in forming clusters, the novel system CG-Clus finds clusterings that are highly similar to the solutions of two more flexible schemes that are well-established in the literature. The mining process itself guarantees high intra-cluster similarity in a single run of the algorithm and in addition, the formulation of conjunctive cluster descriptions of low complexity.

Table 10. Classes per data set, average number of clusters formed by COBWEB, number of conjunctive rules learned on the clustering using RIPPER, recovery rate (that is training set accuracy of learned rules)

Data sets	# of Classes	# of Clusters	# of Rules	Recovery rate
Credit-G	2	349.5 ± 180.96	38.9 ± 8.86	$65.44\% \pm 17.61$
Credit-G-Equal	2	202.5 ± 191.02	42.7 ± 8.30	$78.34\% \pm 18.80$
Kr-vs-Kp	2	2.5 ± 0.70	15.7 ± 14.47	$99.74\% \pm 0.30$
KrkOpt	18	13.8 ± 5.20	14.5 ± 5.98	$99.99\% \pm 0.003$
Letter	26	18.7 ± 13.01	139.6 ± 40.65	$98.71\% \pm 0.64$
Letter-Equal	26	24.5 ± 5.23	211.2 ± 24.04	$97.41\% \pm 0.61$
Mfeat-Fourier	10	54.9 ± 60.58	51.9 ± 24.82	$95.13\% \pm 3.84$
Mfeat-Fourier-Equal	10	58 ± 24.35	24.6 ± 2.87	$96.99\% \pm 1.20$
Mfeat-Karhunen	10	663.7 ± 173.25	86.3 ± 19.59	$64.28\% \pm 8.75$
Nursery	5	1480.8 ± 958.07	1102.3 ± 710.23	$97.22\% \pm 0.92$
Opdigits	10	8.5 ± 1.65	103.4 ± 16.59	$93.33\% \pm 1.81$
Optdigits-Equal	10	8.6 ± 1.17	111.2 ± 17.21	$93.47\% \pm 1.00$
Pendigits	10	7.2 ± 0.63	60.5 ± 7.01	$99.73\% \pm 0.08$
Segment	7	4.5 ± 0.7	6.6 ± 1.17	$99.99\% \pm 0.01$
Segment-Equal	7	6.2 ± 0.42	38.9 ± 3.24	$99.42\% \pm 0.13$
Waveform	3	3.0 ± 0.00	52.4 ± 2.75	$97.31\% \pm 1.04$

Table 11. Classes per data set, number of descriptive rules found by RIPPER on the AUTOCLASS-solution and their accuracy, and similarity of found clusterings

Data sets	# of Classes	# of Rules	Recovery rate	Rand
Credit-G	2	7	100%	0.5012
Credit-G-Equal	2	25	99.1%	0.5580
Kr-vs-Kp	2	2	100%	0.9185
KrkOpt	18	31	100%	0.9663
Letter	26	268	98.86%	0.9016
Letter-Equal	26	277	97.16%	0.9402
Mfeat-Fourier	10	92	99.15%	0.8673
Mfeat-Fourier-Equal	10	64	99.4%	0.8481
Mfeat-Karhunen	10	89	99%	0.8729
Nursery	5	5	100%	0.6543
Optdigits	10	117	95.53%	0.8656
Optdigits-Equal	10	115	96.89%	0.8887
Pendigits	10	73	99.66%	0.9019
Segment	7	9	99.91%	0.8345
Segment-Equal	7	10	99.65%	0.9002
Waveform	3	50	98.2%	0.7877

Related work The field of *conceptual clustering* is too vast to exhaustively discuss everything relating to our work so we will restrict our discussion mainly to the papers mentioned in Section 4.4. One of the earliest approaches to inducing conjunctive descriptions of conceptual clusters is the CLUSTER/2 system [32]. While CLUSTER/2 works bottom-up in a heuristic manner, generalizing pairs of seed instances, our technique induces concepts top-down and gives guarantees w.r.t. the quality of found solutions. The *conceptual cluster mining* task, introduced by Perkowitz *et al.* [38], is similar to *cluster-grouping* w.r.t. to clustering. Their goal is to induction clusters that are cohesive but also describable by a simple concept. To this end, they use their PAGEGATHER system for clustering webpages and RIPPER to learn the concept separating each cluster from all others. The final solution consists of all subsets of instances that correspond to the learned concepts. Since both the clustering and the rule learning algorithm could be instantiated differently, the *conceptual cluster mining* framework is rather general. The approach does have potential drawbacks as discussed in Section 4.4. In [37], an incremental branch-and-bound clusterer for the formation of hierarchies was introduced. Since addition of new observations can have a severe effect on the existing hierarchy, re-insertion of instances and clusters is performed during the formation process. To restrict the number of evaluation steps needed, the set of nodes that could act as parents in the hierarchy to the instance or cluster to be inserted is limited. To this end, an upper bound on the best value a node can give is calculated based on the evaluation of picking this node's parents as parents of the new instance or cluster. The measure we used for the quality of cluster descriptions in this work is *Category Utility*, with the probably best-known clusterer using this measure being COBWEB. Due to its incremental instance processing, the ordering of instances has an effect on the solution. To address this effect, Fisher [17] explores several re-distribution and re-clustering techniques for greedily improving an existing clustering. We find the optimally discriminating patterns in the first run instead. A second issue addressed in Fisher's work is related to the effect we observed in Section 4.4, namely that COBWEB on certain data sets tends to create a large amount of clusters, gaining only a small increase in *Category Utility*. The solution discussed in [17] is similar to post-pruning in decision tree learning in that certain branches of the clustering hierarchy are removed during validation on a separate data set. Third, Fisher discusses possible shortcomings of *Category Utility* as a quality measure for clusterings. Assuming that a clustering is used for classification afterwards, he suggests properties such as number of leaves, maximum path length, branching factor and classification cost, e.g. number of attributes to be evaluated, for measuring the quality of a clustering tree. All these parameters could be affected by a user, given a suitable wrapper around CG. Finally, possible alternatives to *Category Utility* mentioned in this work could be used in CG if they are convex.

An additional work that has to be mentioned is that of Blockeel *et al.* [5]. In their approach, a decision tree is constructed, with tests in first order logic in the splitting nodes. While the measure used is intra-class variance, to induce

similarity of numerical features, *CU* could be used instead. The main difference lies in the fact that *TIC* describes clusters by conjunctions formed by along branches of the tree but not in each splitting node.

5 Related Work

Work that is related to the overall *cluster-grouping*-framework can be roughly grouped into two categories: work exploring the relation between local pattern mining and diverse data mining or machine learning tasks, and algorithmically related techniques.

5.1 Local pattern mining for machine learning

The task of correlating pattern mining has been introduced by Brin *et al.* [6]. Their solution to the problem is somewhat different than Morishita and Sese's one in that they mine for all pattern for whom *all* pairs of items correlate. By restricting their definition in that way, they can derive an anti-monotone criterion that can be used for pruning. It also means that they have more flexibility w.r.t. possible rules. The caveat is, however, that there might be correlations which will only emerge if a combination of items is related to a target item. Those will not be found by their technique.

The relation between local pattern mining and classical machine learning tasks has been explored in recent years, notably at the 2004 Dagstuhlseminar: Detecting Local Patterns [33]. Höppner [24] discusses the relation between local pattern mining and clustering and arrives at an algorithm finding clusters characterized by local patterns whose interestingness is measured w.r.t. a background distribution. Found partitions are then successively refined further. The relationship between local and global models w.r.t. classification rule learning is the topic of [21]. This work is more concerned with filtering and combining mined patterns to build a classifier, though.

A short discussion of the unification of supervised (*subgroup discovery*, *classification*) and unsupervised learning (*conceptual clustering*) can be found in [18]. The authors mention that supervised learning aims at predicting a single attribute, unsupervised learning all attributes, and that tasks between those two goals could be imagined but do not seem to pursue this further, as we do.

5.2 Related algorithms

Similar ideas to the ones discussed w.r.t. the CG algorithm have been explored in the Brute system [39]. It performs a bounded, exhaustive search to find the k best so-called *nuggets*. These nuggets are high-accuracy rules, essentially local patterns that have a high predictive power for a potentially small set of instances. In this regard they are similar to rules describing subgroups. Instead of a minimum interestingness threshold, Brute asks the user to specify a minimum

search depth and possibly also minimum number of positives covered and a beam size.

Generally speaking, CG, as well as any other exhaustive pattern miner, can be interpreted as an instantiation of the OPUS system introduced by Webb [43]. Similarly to EXPLORA, referred to in Section 4.2, OPUS is a general system for exhaustive mining that allows the application of supplied mining rules. The optimization version of OPUS, OPUS^O, depends on the use of an optimistic value for pruning, i.e. an upper bound.

Webb *et al.* [45] also presented an algorithm for mining the best k frequent patterns according to an additional interestingness measure, specifically *leverage* which is calculated in the same way as *WRacc*. The approach is similar to CG, employing a dynamic threshold for pruning. The pruning rules are specifically tailored to leverage, in contrast to the technique used here. The author identifies finding additional constraints and according pruning rules as a future research direction. The addition of a frequency constraint seems to conflict with our intuition that a pattern should be interesting w.r.t. statistical considerations.

The CG algorithm is a substantial extension of the Morishita and Sese algorithm for *correlated pattern mining*, APRIORISMP [34]. Morishita and Sese have also adapted the basic APRIORISMP [41] to cope with multiple numerical attributes in the consequent part of rules. By performing clustering of numerical target values using the convex interclass variance criterion, they *are* defining a further *cluster-grouping* task. The most important difference with the work of Morishita and Sese [41] is thus that they only look for the k best rules achieving this clustering effect and did not study the application of these rules to hierarchical *conceptual clustering*, *subgroup discovery*, or *classification*, which is the most important contribution of the present work.

A similar technique has been developed independently by Bay and Pazzani [3]. Their name for patterns that discriminate strongly between several values of a designated attribute is *contrast sets*. Bay and Pazzani also use the convexity of χ^2 to derive an upper bound for patterns regarding a multi-valued target attribute. The main difference with Morishita and Sese's work lies in the fact that the latter derive a general upper-bound framework applicable to all convex correlation measures is developed, which we further generalized into the *cluster-grouping* framework.

The *cluster-grouping* problem is also related to feature selection in *conceptual clustering* and to semi-flexible prediction [42, 8]. Talavera's [42] motivation for feature selection in *conceptual clustering* is somewhat related to our motivation insofar as he is aiming for better comprehensibility, exclusion of irrelevant features and more efficient *clustering* processes (both when creating and using the clusters). There is also some similarity in where in the algorithm the feature are selected, since it is recomputed for each node in the hierarchical *clustering* tree. This is called *local* or *dynamic* selection. The main differences with our work are two-fold: Firstly, Talavera's work still retains COBWEB's representation and only achieves better comprehensibility by reducing the number of considered attributes. Secondly, in his approach each attribute is scored *before* the actual

clustering step, whereas CG performs feature selection as part of the *clustering* process itself.

Cardie [8] defines semi-flexible prediction as learning to predict a *set* of features known *a priori* as opposed to inflexible prediction (classification) and flexible prediction (clustering). Her approach involves automated feature selection for each attribute to be predicted separately. These features are then used in subsequent independent prediction of the attributes. In contrast, we attempt to predict a disjunction of attributes from a shared set of antecedents instead.

Finally, *cluster-grouping* is in many aspects related to the confirmatory induction setting in the *Tertius* system by Flach *et al.* [19]. As in CG, several target attributes are considered. It is interesting to note in this context that the rule head is treated as a single target while CG treats each condition separately. Flach's work diverges from the general correlation setting in which correlation is symmetric and instead focuses on the number of counter-instances to a given rule, thus considering only directed associations. Using an *optimistic* estimate (an upper bound) they prune non-promising candidates and find and rank optimal rules. Focusing on counter-instances only allows more flexibility regarding the rule head, that is, the set of conditions need not be fixed.

6 Conclusions and Future Work

We have introduced the problem of *cluster-grouping* and argued that it can be considered a subproblem in a wide variety of popular machine learning and data mining tasks, such as *correlated pattern mining*, *subgroup discovery*, *classification*, and *conceptual clustering*.

A key contribution of this paper is the formulation of the CG algorithm for tackling the *cluster-grouping* task. We have also argued that it can be used as a universal local pattern mining component in systems tackling important machine learning and data mining tasks. Furthermore, using the CG algorithm has several advantages that often help to alleviate some of the problems with existing systems:

1. CG *always* outputs the k best solutions according to the interestingness function σ. This contrasts with current approaches to the *subgroup discovery*, *classification*, and *conceptual clustering* settings, where the quality of the discovered solutions depends on parameters at best related implicitly to σ, such as a minimum support threshold. At worst, such parameters are only related to the employed heuristic, such as beam-size.
2. While CG is based on a generic brand-and-bound algorithm that has already been used in several works in data mining, it extends these works in that it allows to consider multiple target attributes.
3. An effective pruning technique uses the best σ values seen so far to dynamically remove those parts of the search space that cannot lead to solutions. This procedure often considers *fewer* candidate rules than heuristic techniques, such as beam search (cf. also the experiments in Section 4.2),

or complete enumeration techniques as *associative classification* or brute-force search, combined with post-processing steps. Therefore, unlike the current practice in machine learning, complete branch-and-bound search using convex interestingness measures is often to be preferred over heuristic approaches like beam search.
4. The optimization with regard to interestingness measures is based on statistical principles. Additionally, setting a parameter k to limit the size of the solution set is – arguably – more intuitive than the specification of a beam size or minimum support threshold.

We have shown that our approach is an extension of Morishita's and Sese's work that allows one to apply the underlying ideas to more flexible target definitions and thus additional problem settings. We have provided experimental evidence that CG is well-suited for rule-based *subgroup discovery* (CG-SD), use in classification (CN2-CG,CBC), and conceptual clustering (CG-CLUS). Different variants of existing and novel algorithms were implemented and experimentally compared to state-of-the-art techniques for solving these tasks. In most of cases the CG based approach improved upon alternative techniques in efficiency or performance. Especially worth mentioning are two novel algorithms, CBC and CG-CLUS, which target associative classification and divisive clustering respectively. CBC is a natural alternative to systems such as CMAR and CBA that derive association rules using support and confidence. The CG-CLUS algorithm is competitive with one of the best-known *conceptual clustering* algorithms, COBWEB, and computes rule-sets that are easier to interpret.

Further research will proceed in several directions. First, as can be seen in the experiments, the effectiveness of the pruning step depends strongly on the tightness of the upper bound calculated. Therefore, it is desirable to tighten future support estimates and therefore attainable values of σ. Second, the technique should in principle be usable in the formation of multi-variate decision trees [35]. For such a setting it would be necessary to extend the upper-bound techniques to multi-valued target attributes. For the classification setting this extension could take the form of learning rules involving error-correcting output codes [14, 31].

Another direction is the application to other learning areas. We have already employed the basic principles of CG in a different domain, tree-structured data [7] and the *cluster-grouping* paradigm could also be extended into the area of *logical and relational learning* [13].

Acknowledgments

We sincerely thank Shinichi Morishita and Jun Sese for useful discussion and feedback. We also thank our fellow researchers Kristian Kersting, Björn Bringmann, Siegfried Nijssen, Ulrich Rückert, and the anonymous reviewers for their thorough reviews, constructive comments and helpful suggestions, and the editor of this paper for his constructive feedback and patience.

This work was partly supported by the EU IST project cInQ (consortium on discovering knowledge with Inductive Queries), contract no. IST-2000-26469, and the EU IST project IQ (Inductive Querying), contract no. IST-FET FP6-516169.

References

1. Rakesh Agrawal and Ramakrishan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Databases*, pages 487–499, Santiago de Chile, Chile, September 1994. Morgan Kaufmann.
2. Martin Atzmüller and Frank Puppe. SD-Map - a fast algorithm for exhaustive subgroup discovery. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 6–17. Springer, 2006.
3. Stephen D. Bay and Michael J. Pazzani. Detecting group differences: Mining constrast sets. *Data Mining and Knowledge Discovery*, 5(3):213–246, 2001.
4. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
5. Hendrik Blockeel, Luc De Raedt, and Jan Ramon. Top-down induction of clustering trees. In Jude W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 55–63. Morgan Kaufmann, 1998.
6. Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. In Joan Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data.*, pages 265–276. ACM Press, 1997.
7. Björn Bringmann and Albrecht Zimmermann. Tree^2 - Decision trees for tree structured data. In Alípio Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and João Gama, editors, *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 46–58. Springer, 2005.
8. Claire Cardie. Using decision trees to improve case-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 25–32, Amherst, Massachusetts, USA, June 1993. Morgan Kaufmann.
9. Peter Cheeseman, James Kelly, Matthew Self, John Stutz, Will Taylor, and Don Freeman. Autoclass: A Bayesian classification system. In John E. Laird, editor, *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64, Ann Arbor, Michigan, USA, June 1988. Morgan Kaufmann.
10. Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.
11. Frans Coenen and Paul Leng. Obtaining best parameter values for accurate classification. In Jiawei Han, Benjamin W. Wah, Vijay Raghavan, Xindong Wu, and Rajeev Rastogi, editors, *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 597–600, Houston, Texas, USA, November 2005. IEEE.
12. William W. Cohen. Fast effective rule induction. In Armand Prieditis and Stuart J. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, Tahoe City, California,USA, July 1995. Morgan Kaufmann.
13. Luc De Raedt. *Logical and Relational Learning.* Cognitive Technologies. Springer, 2008.

14. Thomas G. Dietterich and Ghulum Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 572–577, Anaheim, California, USA, July 1991. AAAI Press/The MIT Press.
15. Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1029, Chambéry,France, August 1993. Morgan Kaufmann.
16. Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.
17. Douglas H. Fisher. Iterative optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research (JAIR)*, 4:147–178, 1996.
18. Douglas H. Fisher and Gilford Hapanyengwi. Database management and analysis tools of machine learning. *Journal of Intelligent Information Systems*, 2:5–38, 1993.
19. Peter A. Flach and Nicolas Lachiche. Confirmation-guided discovery of first-order rules with Tertius. *Machine Learning*, 42(1/2):61–95, 2001.
20. Eibe Frank and Ian H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.
21. Johannes Fürnkranz. From local to global patterns: Evaluation issues in rule learning algorithms. In Morik et al. [33], pages 20–38.
22. Johannes Fürnkranz and Peter A. Flach. ROC 'n' rule learning-towards a better understanding of covering algorithms. *Machine Learning*, 58(1):39–77, 2005.
23. Mark A. Gluck and James E. Corter. Information, uncertainty, and the utility of categories. In *Proceedings of the 7th Annual Conference of the Cognitive Science Society*, pages 283–287, Irvine, California, USA, 1985. Lawrence Erlbaum Associate.
24. Frank Höppner. Local pattern detection and clustering. In Morik et al. [33], pages 53–70.
25. Branko Kavsek and Nada Lavrac. Apriori-SD: Adapting association rule learning to subgroup discovery. *Applied Artificial Intelligence*, 20(7):543–583, 2006.
26. Willi Klösgen. Explora: A multipattern and multistrategy discovery assistant. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. The MIT Press, 1996.
27. Nada Lavrač, Peter A. Flach, and Blaz Zupan. Rule evaluation measures: A unifying view. In Sašo Džeroski and Peter A. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, pages 174–185, Bled, Slovenia, June 1999. Springer.
28. Nada Lavrač, Branko Kavsek, Peter A. Flach, and Ljupco Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004.
29. Wenmin Li, Jiawei Han, and Jian Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In Nick Cercone, Tsau Young Lin, and Xindong Wu, editors, *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 369–376, San José, California, USA, November 2001. IEEE Computer Society.
30. Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 80–86, New York City, New York, USA, August 1998. AAAI Press.

31. Francesco Masulli and Giorgio Valentini. Effectiveness of error correcting output codes in multiclass learning problems. In Josef Kittler and Fabio Roli, editors, *Proceedings on the First International Workshop on Multiple Classifier Systems*, pages 107–116, Cagliari, Italy, June 2000. Springer.
32. Ryszard S. Michalski and Robert E. Stepp. Learning from observation: Conceptual clustering. *Machine Learning, An Artificial Intelligence Approach*, 1:331–363, 1983.
33. Katharina Morik, Jean-François Boulicaut, and Arno Siebes, editors. *Local Pattern Detection, International Seminar, Revised Selected Papers*, Dagstuhl Castle, Germany, April 2004. Springer.
34. Shinichi Morishita and Jun Sese. Traversing itemset lattices with statistical metric pruning. In *Proceedings of the Nineteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 226–236, Dallas, Texas, USA, May 2000. ACM.
35. Sreerama K. Murthy. *On Growing Better Decision Trees from Data*. PhD thesis, John Hopkins University, Baltimore,Maryland,USA, 1997.
36. Stefan Mutter, Mark Hall, and Eibe Frank. Using classification to evaluate the output of confidence-based association rule mining. In Geoffrey I. Webb and Xinghuo Yu, editors, *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, pages 538–549, Cairns, Australia, December 2004. Springer.
37. Arthur J. Nevins. A branch and bound incremental conceptual clusterer. *Machine Learning*, 18(1):5–22, 1995.
38. Mike Perkowitz and Oren Etzioni. Adaptive web sites: Conceptual cluster mining. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 264–269, Stockholm, Sweden, July 1999. Morgan Kaufmann.
39. Patricia J. Riddle, Richard Segal, and Oren Etzioni. Representation design and brut-force induction in a Boeing manufacturing domain. *Applied Artificial Intelligence*, 8(1):125–147, 1994.
40. Tobias Scheffer and Stefan Wrobel. Finding the most interesting patterns in a database quickly by using sequential sampling. *Journal of Machine Learning Research*, 3:833–862, 2002.
41. Jun Sese and Shinichi Morishita. Itemset classified clustering. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Proceedings of the 8th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 398–409, Pisa,Italy, September 2004. Springer.
42. Luis Talavera. Dynamic feature selection in incremental hierarchical clustering. In Ramon López de Mántaras and Enric Plaza, editors, *Proceedings of the 11th European Conference on Machine Learning*, pages 392–403, Barcelona, Catalonia, Spain, May 2000. Springer.
43. Geoffrey I. Webb. Opus: An efficient admissible algorithm for unordered search. *J. Artif. Intell. Res. (JAIR)*, 3:431–465, 1995.
44. Geoffrey I. Webb. Discovering significant patterns. *Machine Learning*, 68(1):1–33, 2007.
45. Geoffrey I. Webb and Songmao Zhang. K-optimal rule discovery. *Data Mining and Knowledge Discovery*, 10(1):39–79, 2005.
46. Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In J. Komorowski and J. Zytkow, editors, *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD '97)*, pages 78 – 87, Trondheim, Norway, 1997. Springer-Verlag.

47. Albrecht Zimmermann and Luc De Raedt. Cluster-grouping: From subgroup discovery to clustering. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Proceedings of the 15th European Conference on Machine Learning*, pages 575–577, Pisa, Italy, September 2004. Springer.
48. Albrecht Zimmermann and Luc De Raedt. Corclass: Correlated association rule mining for classification. In Einoshin Suzuki and Setsuo Arikawa, editors, *Proceedings of the 7th International Conference on Discovery Science*, pages 60–72, Padova, Italy, October 2004. Springer.
49. Albrecht Zimmermann and Luc De Raedt. Inductive querying for discovering subgroups and clusters. In Jean-François Boulicaut, Luc De Raedt, and Heikki Mannila, editors, *Constraint-Based Mining and Inductive Databases*, volume 3848 of *Lecture Notes in Computer Science*, pages 380–399. Springer, 2004.

A Convexity Proofs

A.1 Convexity of *Weighted Relative Accuracy*(*WRAcc*)

Let A be an attribute, $v \in \mathcal{V}[A]$ a possible value of A, $\mathcal{E}$ a data set, r a rule of the form $b \rightsquigarrow A = v$, with $x = sup(b), y = sup(x \rightsquigarrow A = v), m = sup(A = v), n = |\mathcal{E}|$.

Then the usual definition of *WRAcc*:

$P(b)(P(A = v|b) - P(A = v))$ can be redefined as: $WRAcc(x, y) = \frac{x}{n}\left(\frac{y}{x} - \frac{m}{n}\right)$

To prove the convexity of *WRAcc*, we directly check the convexity criterion:

$$WRAcc(\lambda(x_1, y_1) + (1-\lambda)(x_2, y_2)) = \frac{\lambda x_1 + (1-\lambda)x_2}{n}\left(\frac{\lambda y_1 + (1-\lambda)y_2}{\lambda x_1 + (1-\lambda)x_2} - \frac{m}{n}\right)$$

$$= \frac{\lambda y_1 + (1-\lambda)y_2}{n} - \frac{m(\lambda x_1 + (1-\lambda)x_2)}{n^2} = \frac{\lambda y_1}{n} - \frac{\lambda m x_1}{n^2} + \frac{(1-\lambda)y_2}{n} - \frac{(1-\lambda)m x_2}{n^2}$$

$$= \frac{\lambda x_1}{n}\left(\frac{y_1}{x_1} - \frac{m}{n}\right) + \frac{(1-\lambda)x_2}{n}\left(\frac{y_2}{x_2} - \frac{m}{n}\right) = \lambda WRAcc(x_1, y_1) + (1-\lambda)WRAcc(x_2, y_2)$$

Since the two terms are equal *WRAcc* is not strictly convex function.

A.2 Convexity of *Category Utility*(*CU*)

As shown in 3.3, *CU* can be decomposed into a sum of partial *CUs*. If reformulated in the stamp point notation, *CU* becomes:

$$\begin{aligned} CU(\langle x, y_1, \ldots, y_d\rangle) &= \sum_{i=1}^{d} CU(x, y_i) \\ &= \sum_{i=1}^{d} \tfrac{1}{2}\tfrac{x}{n}\left(\left(\tfrac{y_i}{x}\right)^2 - \left(\tfrac{m_i}{n}\right)^2 + \left(\tfrac{x-y_i}{x}\right)^2 - \left(\tfrac{n-m_i}{n}\right)^2\right) \\ &\quad + \tfrac{1}{2}\tfrac{n-x}{n}\left(\left(\tfrac{m_i-y_i}{n-x}\right)^2 - \left(\tfrac{m_i}{n}\right)^2 + \left(\tfrac{n-m_i-(x-y_i)}{n-x}\right)^2 - \left(\tfrac{n-m_i}{n}\right)^2\right) \end{aligned}$$

Since a sum of convex functions is itself is again convex it is enough to prove the convexity of partial *CU*. Additionally to directly checking the convexity property there is another way to prove convexity whose presentation takes up less space. For a twice differentiable function to be convex, its Hessian has to be positive semi-definite. The Hessian is the matrix of the function's second partial derivatives and for the two-dimensional case has the form:

$$\begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

A matrix is positive semi-definite if the determinants of all its leading principal minors are ≥ 0. This implies that we have to show that:

$$\frac{\partial^2 f}{\partial x^2} \geq 0, \frac{\partial^2 f}{\partial y^2} \geq 0, \text{ and } \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \frac{\partial^2 f}{\partial x \partial y}\frac{\partial^2 f}{\partial y \partial x} \geq 0$$

A partial CU in stamp point notation is a sum that can be decomposed further into the part corresponding to the instances covered by the rule body and the instances not covered by the rule body. Again it holds that if those two terms are convex, the entire partial CU is convex. The corresponding Hessians are:

$$\begin{pmatrix} \frac{2y^2}{nx^3} & \frac{-2y}{nx^2} \\ \frac{-2y}{nx^2} & \frac{2}{nx} \end{pmatrix} \begin{pmatrix} \frac{2(y-m)^2}{(n-x)^3 n} & \frac{2y-2m}{(n-x)^2 n} \\ \frac{2y-2m}{(n-x)^2 n} & \frac{2}{2(n-x)n} \end{pmatrix}$$

The $\frac{\partial^2 f}{\partial x^2}, \frac{\partial^2}{\partial y^2}$ are obviously greater or than zero, so all that is left is checking the determinants of the whole matrices. For the "positive" part (the part that corresponds to covered instances) this determinant is:

$$\frac{4y^2}{n^2x^4} - \frac{4y^2}{n^2x^4} = 0 \text{ and for the "negative" part: } \frac{4(y-m)^2}{(n-x)^4 n^2} - \frac{4(y-m)^2}{(n-x)^4 n^2} = 0$$

So the Hessian of CU is positive semidefinite and thus CU is a convex function.