Turning Clusters into Patterns: Rectangle-based Discriminative Data Description

Byron J. Gao Martin Ester
School of Computing Science, Simon Fraser University, Canada
bgao@cs.sfu.ca ester@cs.sfu.ca

Abstract

The ultimate goal of data mining is to extract knowledge from massive data. Knowledge is ideally represented as human-comprehensible patterns from which end-users can gain intuitions and insights. Yet not all data mining methods produce such readily understandable knowledge, e.g., most clustering algorithms output sets of points as clusters. In this paper, we perform a systematic study of cluster description that generates interpretable patterns from clusters. We introduce and analyze novel description formats leading to more expressive power, motivate and define novel description problems specifying different trade-offs between interpretability and accuracy. We also present effective heuristic algorithms together with their empirical evaluations.

1. Introduction

The ultimate goal of data mining is to discover useful knowledge, ideally represented as human-comprehensible patterns, in large databases. Clustering is one of the major data mining tasks, grouping objects together into clusters that exhibit internal cohesion and external isolation. Unfortunately, most clustering methods simply represent clusters as sets of points and do not generalize them into patterns that provide interpretability, intuitions, and insights.

So far, the database and data mining literature lacks systematic study of cluster description that transforms clusters into human-understandable patterns. For numerical data, hyper-rectangles generalize multi-dimensional points, and a standard approach in database systems is to describe a set of points with a set of isothetic hyper-rectangles [1, 16, 18]. Due to the property of being axis-parallel, such rectangles can be specified in an intuitive manner; e.g., "3.80 $\leq$ *GPA* $\leq$ 4.33 and 0.1 $\leq$ *visual acuity* $\leq$ 0.5 and 0 $\leq$ *minutes in gym per week* $\leq$ 30" intuitively describes a group of "nerds".

Patterns are models with generalization capacity, as well as templates that can be used to make or to generate things. The rectangle-based expressions are interpretable models; as another practical application, they can also be used as search conditions in SELECT query statements to retrieve (generate) cluster contents, supporting query-based iterative mining [13] and interactive exploration of clusters.

To be understandable, cluster descriptions should appear short in length and simple in format. Sum of Rectangles (SOR), simply taking the union of a set of rectangles, has been the canonical format for cluster descriptions in the database literature. However, this relatively restricted format may produce unnecessarily lengthy descriptions. We introduce two novel description formats, leading to more expressive power yet still simple enough to be intuitively understandable. The SOR^- format describes a cluster as the difference of its bounding box and a SOR description of the non-cluster points within the box. The $kSOR^\pm$ format allows describing different parts of a cluster separately, using either SOR or SOR^- descriptions. We prove that the $kSOR^\pm$-based description language is equivalently expressive to the (most general) propositional language [18].

Meanwhile, cluster descriptions should cover cluster contents accurately, which conflicts with the goal of minimizing description length. The Pareto front for the bicriteria problem of optimizing description accuracy and length, as illustrated in Figure 3, offers the best trade-offs between accuracy and interpretability for a given format. To solve the bicriteria problem, we introduce the novel Maximum Description Accuracy (MDA) problem with the objective of maximizing description accuracy at a given description length. The optimal solutions to the MDA problems with different length specifications up to a maximal length constitute the Pareto front. The maximal length to specify (20 in Figure 3) is determined by the optimal solution to the Minimum Description Length (MDL) problem, which aims at finding some shortest perfectly accurate description that covers a cluster completely and exclusively. Previous research only considered the MDL problem; however, perfectly accurate descriptions can become very lengthy and hard to interpret for arbitrary shape clusters. The MDA problem allows trading accuracy for interpretability so that users can zoom in and out to view the clusters.

The description problems are NP-hard. We present heuristic algorithms Learn2Cover for the MDL problem to approximate the maximal length, and starting from which DesTree for the MDA problems to iteratively build the so-called description trees approximating the Pareto front. The resulting descriptions, in the format of SOR or SOR^-, can be transformed into shorter $kSOR^\pm$ descriptions with at least the same accuracy by FindClans, taking advantage of the exceeding expressive power of $kSOR^\pm$ descriptions.

Contributions. (1) Introduction and analysis of novel description formats, SOR^- and $kSOR^\pm$, providing enhanced expressive power. (2) Definition and investigation of a novel description problem, MDA, allowing trading accuracy for interpretability. (3) Presentation and evaluation of effective description heuristics, Learn2Cover, DesTree and FindClans, approximating the Pareto front.

Related work. [1] studies grid data and defines a cluster as a set of connected dense cells. Their proposed Greedy Growth heuristic constructs an exact covering of a cluster with maximal isothetic rectangles. In the heuristic, a yet-uncovered dense cell is arbitrarily chosen to grow as much as possible along an arbitrarily chosen dimension and continue with other dimensions until a hyper-rectangle is obtained. A greedy approach is then used to remove redundancy from the set of obtained rectangles. This special case of cluster description is related to the problem of covering rectilinear polygons with axis-parallel rectangles [11], which is NP-complete [17], no polynomial time approximation scheme [3], and usually studied in 2-dimensional space in the computational geometry community (e.g., [15]).

[16] also studies grid data but generalizes the description problem studied in [1] by allowing covering some "don't care" cells to reduce the cardinality of the set of rectangles. Their proposed Algorithm BP bases on Greedy Growth to generate the initial set of rectangles, then performs greedy pairwise merges of rectangles without covering undesired cells and with limited "don't care" cells for use.

Greedy Growth and BP explicitly work on cluster description. However, despite the grid data limitation, they address the MDL problem solely while our focus is on the more useful and practical MDA problem. In addition, they only use the SOR format while we study and apply novel formats with more expressive power.

Similar to [1] and [16], [18] is motivated by database applications too but with a focus on the theoretical formulation and analysis of concise descriptions. [18] also formally defines the general MDL problem for given language (L-MDL) and proves its NP-completeness. As the initial work of this study, [9] discusses cluster description formats, problems and algorithms at the introductory level. [10] extends the description problem to the classification problem.

Axis-parallel decision trees [5] can be related to cluster description technically as they provide feasible solutions to the MDL and MDA problems even if with different objectives. Consider a closed rectangular instance space, the leaf nodes of a decision tree correspond to a set of isothetic rectangles forming a partition of the training data (and the instance space). Stipulating rectangles to be disjoint, decision tree methods can be considered addressing a partitioning problem (with the additional constraint of partitioning the instance space) while cluster description is essentially a covering problem allowing overlapping. Partitioning problems are "easier" than covering problems in the sense that they have a smaller search space. Algorithms for a partitioning problem usually work too for the associated covering problem but typically generate larger covers. In decision tree induction, the preference for shorter trees coincides with the preference for shorter description length in the MDL problem. As in the MDA problem, decision tree pruning allows trading accuracy (on training data) for shorter trees, and the technique can be applied to generate feasible solutions for the MDA problems.

Indirectly related work in the theory community exists. [6] studies the red blue set cover problem. Given a set of red and blue elements and a family which is a subset of the power set of the element set, find a subfamily of given cardinality that covers all the blue elements and the minimum number of red elements. [7] studies the maximum box problem. Given two finite sets of points, find a hyper-rectangle that covers the maximum number of points from one designated set and none from the other. Both problems are NP-hard and related to the MDA problem (with *precision at fixed recall of 1* and *recall at fixed precision of 1* as the accuracy measures respectively, see §3.1), except that the former is given an alphabet (family) and restricted to the SOR format, and the latter is limited to use one rectangle.

The above discussed research more or less roots in the classical minimum set cover and maximum coverage problems. The former attempts to select as few as possible subsets from a given family such that each element in any subset of the family is covered; the latter, a close relative, attempts to select k subsets from the family such that their union has the maximum cardinality. The simple greedy algorithm, iteratively picking the subset that covers the maximum number of uncovered elements, approximates the two NP-hard problems within $(1+ln\,n)$ [14] and $(1-\frac{1}{e})$ [12] respectively. The ratios are optimal unless NP is constrained in quasi-polynomial time [8]. The minimum set cover and maximum coverage problems are related to the MDL and MDA (with *recall at fixed precision of 1* as the accuracy measure) problems respectively except that they are given an alphabet (family) and restricted to the SOR format.

Organization of the paper. In Section 2 we introduce and analyze the description formats. In Section 3 we formalize the description problems. We present heuristic algorithms in Section 4, report empirical evaluations in Section 5, and conclude the paper in Section 6.

2. Alphabets, formats, and languages

In this section, we study alphabets, formats, and languages for cluster descriptions in depth so as to gain insights into the description problems with different given formats.

2.1. Preliminaries

Given a finite set of multi-dimensional points U as the universe, and a set of isothetic hyper-rectangles Σ as the alphabet, each of which is a symbol and subset of U containing points covered by the corresponding rectangle. A description format F allows certain Boolean set expressions over the alphabet. All such expressions constitute the description language L with each expression $E \in L$ being a possible description for a given subset $C \subseteq U$. The vocabulary for E, V_E, is the set of symbols in E. We summarize the notations used and to be used for easy lookup.

D: data space; $D = D_1 \times D_2 \times ... \times D_d$
U: data set; $U \subseteq D$
R: rectangle or the set of points it covers; $R \subseteq U$
Σ: alphabet; a set of symbols with each as a rectangle
V_E: vocabulary of expression E; set of symbols used in E
$||E||$: length of expression E; $||E|| = |V_E|$
B_u: bounding box for u; u is a set of points or rectangles
C: set of points in a given cluster; $C \subseteq U$
C^-: set of points in B_C but not in C; $C^- = B_C - C$
$E_{F,\Sigma}$: expression in format F with vocabulary in Σ
$L_{F,\Sigma}$: language comprising all $E_{F,\Sigma}$ expressions

Note that R (E) is overloaded to denote a rectangle (expression) or the set of points it covers (describes). The distinction should be made clear by the context. Σ is often left unspecified in $E_{F,\Sigma}$ and $L_{F,\Sigma}$ when assuming some default alphabet (to be discussed shortly). "$+$", "$\cdot$", "$-$" and "$\neg$" are used to denote Boolean set operators union, intersection, difference and complement.

Two descriptions E_1 and E_2 are equivalent, denoted by $E_1 = E_2$, if they cover the same set of points. Logical equivalence implies equivalence but not vice versa.

$||E||$ indicates the interpretability of E for a given format. There are two simple ways of defining $||E||$, absolute length and relative length. Absolute length is the total number of occurrences of symbols in E; relative length is the cardinality of V_E. Neither alone captures the interpretability of E perfectly. The former overestimates the repeated symbols; the latter underestimates the repeated symbols. The two converge if the repeated symbols are few, which we expect to be the case for cluster descriptions. We define $||E||$ to be the relative length of E for the ease of analysis.

Description accuracy is another important measure for the goodness of E, which we will discuss in more details in section 3. For the use of this section, we conservatively define the "more accurate than" relationship for descriptions. A description for C is more accurate than the other if it covers more points from C and less points from C^-.

Definition 2.1 (*more accurate than*) *Given E_1 and E_2 as descriptions for a cluster C, we say E_1 is more accurate than E_2, denoted by $E_1 \geq_{accu} E_2$, if $|E_1 \cdot C| \geq |E_2 \cdot C|$ and $|E_1 \cdot C^-| \leq |E_2 \cdot C^-|$.*

$(E_1 \cdot C \supseteq E_2 \cdot C) \wedge (E_1 \cdot C^- \subseteq E_2 \cdot C^-) \Rightarrow E_1 \geq_{accu} E_2$.

A description problem, viewed as searching, is to search good descriptions that optimize some objective function in a given description language. A more general description language implies more expressive power. Language L_1 is more general than language L_2 if $L_1 \supseteq L_2$. To characterize expressive power more precisely, we define the "more expressive than" relationship for languages. L_1 is more expressive than L_2 if for any description of cluster C in L_2, there is a shorter and more accurate description in L_1.

Definition 2.2 (*more expressive than*) *Given two description languages L_1, L_2 and a cluster C, we say L_1 is more expressive than L_2, denoted by $L_1 \geq_{exp} L_2$, if for any description $E_2 \in L_2$, there exists some description $E_1 \in L_1$ with $||E_1|| \leq ||E_2||$ and $E_1 \geq_{accu} E_2$ with respect to C.*

Also, $L_1 =_{exp} L_2$ if $(L_1 \geq_{exp} L_2) \wedge (L_2 \geq_{exp} L_1)$.

A more expressive language is guaranteed to contain "better" expressions with respect to length and accuracy. Certainly, $L_1 \supseteq L_2 \Rightarrow L_1 \geq_{exp} L_2$. Yet to restrict the search space, we do not want languages to be unnecessarily general. This concern carries on through the following discussions on alphabets and formats, by which languages are specified.

2.2. Description alphabets

Unlike the set cover problem, alphabet Σ is not explicitly given in cluster description. Assuming given Σ, a description problem, MDA or MDL, can be considered searching through a given language L for the optimal expression. We call such problems L-problems; in particular, L-MDA or L-MDL. The L-problems, to be detailed shortly, are variants of the set cover problem. We brief alphabets mainly for the purpose of gaining insights into the description problems by relating their L-problems to the classical set cover problem.

Σ is potentially an infinite set since for a set of points, there are an infinite number of covering rectangles with each being a candidate symbol. A simple finite alphabet can be defined as the set of bounding boxes for the subsets of B_C, i.e., $\Sigma_{most} = \{B_S \mid S \subseteq B_C\}$. The alphabet is finite since there is a unique bounding box for a set of points. Σ_{most} is the most general alphabet relevant to the task of describing C; however, it can be unnecessarily general for some L-problem with a mismatched format. It is desirable to have some *most specific sufficiently general* alphabets.

Let $f(L\text{-}p)$ denote the feasible region of an L-problem L-p; certainly, $f(L\text{-}p) \subseteq L$. We say Σ is *sufficiently general* for $L_{F,\Sigma}\text{-}p$ if $L_{F,\Sigma} \geq_{exp} f(L_{F,\Sigma_{most}}\text{-}p)$, which roughly means Σ does not lose to Σ_{most} if used in the L-p case. On top of being sufficiently general, Σ is *most specific* if removing any element from it, Σ would not be sufficiently general anymore. In the following, we define some alphabets with this desireable property.

$$
\begin{aligned}
\Sigma_{pure} &= \{B_S \mid B_S \cdot C^- = \emptyset \wedge S \subseteq C \wedge S \neq \emptyset\} \\
\Sigma^-_{pure} &= \{B_S \mid B_S \cdot C = \emptyset \wedge S \subseteq C^- \wedge S \neq \emptyset\} \\
\Sigma_{mix} &= \{B_S \mid S \subseteq C \wedge S \neq \emptyset\} \\
\Sigma^-_{mix} &= \{B_S \mid S \subseteq C^- \wedge S \neq \emptyset\}
\end{aligned}
$$

Σ_{pure} (Σ^-_{pure}) contains pure rectangles covering points from C (C^-) only. Symbols in Σ_{mix} (Σ^-_{mix}), however, can be mixed allowing points from C and C^- to co-exist. Apparently, $\Sigma_{pure} \subseteq \Sigma_{mix} \subseteq \Sigma_{most}$ and $\Sigma^-_{pure} \subseteq \Sigma^-_{mix} \subseteq \Sigma_{most}$. Multiple subsets may match to the same bounding box, e.g., $B_{13} = B_{123}$ in Figure 1, where B_{13} is short for $B_{\{1,3\}}$ and so on. The figure illustrates Σ_{pure} and Σ_{mix}.

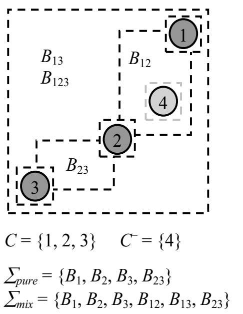
Examples of some L-problems with matching alphabet and format are $L_{SOR,\Sigma_{pure}}$-MDL, $L_{SOR^-,\Sigma^-_{pure}}$-MDL, $L_{SOR,\Sigma_{mix}}$-MDA and $L_{SOR^-,\Sigma^-_{mix}}$-MDA. In addition, we define $\Sigma_k = \Sigma_{mix} + \Sigma^-_{pure}$, then $L_{kSOR^\pm,\Sigma_k}$-MDL and $L_{kSOR^\pm,\Sigma_k}$-MDA are also such examples. Due to the page limit, we omit further explanations.

Such a desirable Σ is assumed given by default if it is left unspecified in $E_{F,\Sigma}$ or $L_{F,\Sigma}$. Although these default alphabets are made specific, they can still be prohibitively large (e.g., $O(|2^C|)$ and not scalable to real problems. Therefore, it is practically infeasible to generate Σ and apply existing set cover approximations on description problems.

2.3. Description formats and languages

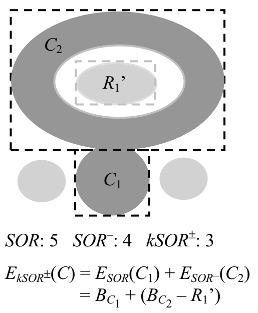
We require descriptions to be interpretable. For descriptions to be interpretable, the description format has to have a simple and clean structure. Sum of Rectangles (SOR), denoting the union of a set of rectangles, serves this purpose well and has been the canonical format for cluster descriptions in the literature (e.g., [1, 16]). For better interpretability, we also want descriptions to be as short as possible. To minimize the description length of SOR descriptions has been the common description problem.

Nevertheless, there is a trade-off between our preferences for simpler formats and shorter description length. Simple formats such as SOR may restrict the search space too much leading to languages with low expressive power. On the other hand, if a description format allows arbitrary Boolean operations over a given alphabet, we certainly have the most general and expressive language containing the shortest descriptions, but such descriptions are likely hard to



1
4
2
3
B_{13}
B_{123}
B_{12}
B_{23}
$C = \{1, 2, 3\}$ $C^- = \{4\}$
$\Sigma_{pure} = \{B_1, B_2, B_3, B_{23}\}$
$\Sigma_{mix} = \{B_1, B_2, B_3, B_{12}, B_{13}, B_{23}\}$

Figure 1. Alphabets.



C_2
R_1'
C_1
SOR: 5 SOR^-: 4 $kSOR^\pm$: 3
$E_{kSOR^\pm}(C) = E_{SOR}(C_1) + E_{SOR^-}(C_2)$
$= B_{C_1} + (B_{C_2} - R_1')$

Figure 2. Formats.

comprehend due to their complexity in format despite their succinctness in length. Moreover, not well-structured complex formats bring difficulties in manipulation of symbols and design of efficient and effective searching strategies.

Clearly, we require description languages with high expressive power yet in intuitively understandable formats. In the following, we explore several alternative description formats beyond SOR, in particular, SOR^- and $kSOR^\pm$.

While SOR takes the form of $R_1 + R_2 + ... + R_l$, a SOR^- description, in describing C, takes the set difference between B_C and a SOR description for C^-.

Definition 2.3 (SOR^- *description*) *Given a cluster* C*, a* SOR^- *description for* C*,* $E_{SOR^-}(C)$*, is a Boolean expression in the form of* $B_C - E_{SOR}(C^-)$*, where* $E_{SOR}(C^-)$ *is a* SOR *description for* C^-*.*

In addition, a $SOR^\pm$ description for C, $E_{SOR^\pm}(C)$, is an expression in the form of either $E_{SOR}(C)$ or $E_{SOR^-}(C)$. Clearly, $L_{SOR^\pm} \supseteq L_{SOR}$ and $L_{SOR^\pm} \geq_{exp} L_{SOR}$. In describing a cluster C, SOR and SOR^- descriptions together nicely cover two situations where C is easier to describe or C^- is easier to describe. Different data distributions, which are usually not known in advance, favor different formats. In Figure 2, consider C_2 as a single cluster to be described with perfect accuracy, certainly SOR^- descriptions are favored. The shortest $E_{SOR}(C_2)$ has length 4 whereas the shortest $E_{SOR^-}(C_2)$ has length 2.

SOR^- descriptions have a structure as simple and clean as SOR descriptions. In addition, the added B_C draws a big picture of cluster C and contributes to interpretability in a positive way. The two formats together also allow us to view C from two different angles. Note that the special rectangle B_C is required for the format, it is not included in the default alphabets either counted in $||E||$ for simplicity.

$SOR^\pm$ descriptions generally serve well for the purpose of describing compact and distinctive clusters. Nevertheless, arbitrary shape clusters are not uncommon and for such applications, we may want to further increase the expressive power of languages by allowing less restrictive formats. For example, if some parts of cluster C favor SOR and some other parts favor SOR^-, then $SOR^\pm$ is too restrictive to consider different parts separately. Instead, it can only provide a global treatment for C. To overcome this disadvantage, we introduce $kSOR^\pm$ descriptions.

Definition 2.4 ($kSOR^\pm$ *description*) *Given a cluster C, a $kSOR^\pm$ description for C, $E_{kSOR^\pm}(C)$, is a Boolean expression in the form of $E_{SOR^\pm}(C_1) + E_{SOR^\pm}(C_2) + ... + E_{SOR^\pm}(C_k)$, where $\bigcup_{i=1}^{k} C_i = C$.*

Clearly, $kSOR^\pm$ descriptions generalize $SOR^\pm$ descriptions by allowing different parts of C to be described separately; and the latter one is a special case of the former one with $k = 1$. In Figure 2, C_1 favors SOR whereas C_2 favors SOR^-. The shortest $E_{SOR}(C)$ and $E_{SOR^-}(C)$ have length 5 and 4 respectively. $kSOR^\pm$ is able to provide local treatments for C_1 and C_2 separately and the shortest $E_{kSOR^\pm}(C)$ has length 3. In many situations $kSOR^\pm$ can be found much more effective than other simpler formats; but how expressive is $L_{kSOR^\pm}$ precisely? In the following, we compare it with the *propositional language*, the most general description language we consider (as in [18]).

Definition 2.5 (*propositional language*) *Given Σ as the alphabet, $L_{P,\Sigma}$ is the propositional language comprising expressions allowing usual set operations of union, intersection and difference over Σ.*

Theorem 2.6 $L_{kSOR^\pm,\Sigma_k} =_{exp} L_{P,\Sigma_k}$

Proof $L_{P,\Sigma_k} \supseteq L_{kSOR^\pm,\Sigma_k} \Rightarrow L_{P,\Sigma_k} \geq_{exp} L_{kSOR^\pm,\Sigma_k}$; we only need to prove $L_{kSOR^\pm,\Sigma_k} \geq_{exp} L_{P,\Sigma_k}$.

Consider any $E \in L_{P,\Sigma_k}$. Since E can be re-written as E_{DNF} in disjunctive normal form with the same set of vocabulary, thus $||E|| = ||E_{DNF}||$ and $E = E_{DNF}$. Each disjunct in E_{DNF} is a conjunction of literals and each literal takes the form of R or $\neg R$ where $R \in \Sigma_k$. Consider any disjunct E_j in E_{DNF}, since axis-parallel rectangles are intersection closed, E_j can be re-written as E'_j, which takes one of the following three forms: (1) $E'_j = R_0$; (2) $E'_j = R_0 \cdot \neg R_x \cdot \neg R_y \cdot ... \cdot \neg R_z$; (3) $E'_j = \neg R_x \cdot \neg R_y \cdot ... \cdot \neg R_z$. In all the three cases $||E'_j|| \leq ||E_j||$ and $E'_j = E_j$.

For case 3, due to the generalized De Morgan's law, $E'_j = \neg R_x \cdot \neg R_y \cdot ... \cdot \neg R_z = \neg(R_x + R_y + ... + R_z) = B_C - (R_x + R_y + ... + R_z)$, which is a SOR^- description.

For case 1 and 2, we suppose $R_0 \in \Sigma_k$. Then for case 1, E'_j is a SOR description. For case 2, due to the generalized De Morgan's law, $E'_j = R_0 \cdot \neg(R_x + R_y + ... + R_z) = R_0 - (R_x + R_y + ... + R_z)$, which is a SOR^- description.

Then, E_j can be re-written as an equivalent $SOR^\pm$ description with length $\leq ||E_j||$. We do this for every disjunct of E_{DNF}, then E can be re-written as a $kSOR^\pm$ description $E' \in L_{kSOR^\pm,\Sigma_k}$ with $||E'|| \leq ||E||$ and $E' = E$, which implies $L_{kSOR^\pm,\Sigma_k} \geq_{exp} L_{P,\Sigma_k}$.

Note that for case 1 and 2, we have supposed $R_0 \in \Sigma_k$, which may not hold since Σ_k is not intersection closed. As a simple counter-example, the intersection of two bounding boxes may not be a bounding box. However, we note that for the two cases, the purpose of E'_j is to describe $R_0 \cdot C = C_0 \subseteq C$, thus $R'_0 = B_{C_0} \in \Sigma_k$. As expressions of length 1 describing C_0, $R'_0 \geq_{accu} R_0$ because $(R'_0 \cdot C_0 = R_0 \cdot C_0) \wedge (R'_0 \cdot C_0^- \subseteq R_0 \cdot C_0^-)$. We replace R_0 with R'_0 in E'_j to get E''_j, then $E''_j \geq_{accu} E'_j$ and $||E''_j|| = ||E'_j||$. Then, E_j can be re-written as a more accurate $SOR^\pm$ description with length $\leq ||E_j||$. Wo do this for every disjoint of E_{DNF}, then E can be re-written as a $kSOR^\pm$ description $E'' \in L_{kSOR^\pm,\Sigma_k}$ with $||E''|| \leq ||E||$ and $E'' \geq_{accu} E$, which implies $L_{kSOR^\pm,\Sigma_k} \geq_{exp} L_{P,\Sigma_k}$.

Theorem 2.6 does not hold if description length $||E||$ is defined as the absolute length, in which case $E = (R_1 + R_2) - R_3$ in L_P has length 3 but the equivalent $E' = (R_1 - R_3) + (R_2 - R_3)$ in $L_{kSOR^\pm}$ has length 4. Nevertheless, the general conclusion persists, that is, $L_{kSOR^\pm}$ is a very expressive language close or equal to L_P.

Despite its exceptional expressive power, the $kSOR^\pm$ format is very simple and conceptually clear, allowing only one level of nesting as the SOR^- format. It is also well-structured to ease the design of searching strategies.

Assuming some given default alphabet, previous research studied cluster description as searching the shortest expression in $L_{SOR,\Sigma_{pure}}$, the simplest and least expressive language we discussed in this section. We study the same problem but considering other more expressive languages $L_{SOR^\pm}$ and $L_{kSOR^\pm}$, and our main focus is on the problem of finding the best trade-offs between accuracy and interpretability, as to be introduced in the following section.

3. Cluster description problems

A description problem is to find a description for a cluster in a given format that optimizes some objective. In this section, we introduce cluster description problems with different objective measures.

3.1. Objective measures

We want to describe a given cluster C with good interpretability and accuracy. Simple formats and shorter descriptions lead to improved interpretability. We have studied alternative description formats that are intuitively comprehensible. Within a given description format, description length is the proper objective measure for interpretability.

In addition to interpretability, the objective of minimizing description length can also be motivated from a "data compression" point of view. There are many situations when we need to retrieve the original cluster records; e.g., to send promotion brochures to a targeted class of customers, to perform statistical analysis, or in a query-based iterative mining environment as advocated by [13], to resume the mining process from stored temporary or partial results. Cluster descriptions provide a neat and standalone way of "storing and retrieving" cluster contents.

In DBMS systems, an isothetic rectangle can be specified by a Boolean search condition such as $1 \leq D_1 \leq 10 \wedge ... \wedge 5 \leq D_d \leq 50$. A cluster description is then a compound search condition for the points in the cluster, which can be used in the WHERE clause of a SELECT query statement to retrieve the cluster contents entirely. In this scenario, the cluster description process resembles encoding and the cluster retrieval process resembles decoding. The compression ratio for cluster description E can be roughly defined as $|E| \ / \ (||E|| \times 2)$, as each rectangle takes twice as much space as each point. The goal of large compression ratio leads to the objective of minimizing description length. Meanwhile, shorter length also speeds up the retrieval process by saving condition checking time [18].

Accuracy is another important measure for the goodness of cluster descriptions. An accurate description should cover many points in the cluster and few points not in the cluster. To precisely characterize description accuracy, we borrow some notations from the information retrieval community [2] and define *recall* and *precision* for a description E of cluster C.

$$recall = |E \cdot C| \ / \ |C|$$
$$precision = |E \cdot C| \ / \ |E|$$

If we only consider *recall*, the bounding box B_C could make a perfectly accurate description; if we only consider *precision*, any single point in C would do the same. The F-measure considers both *recall* and *precision* and is the harmonic mean of the two.

$$f = \frac{2 \times recall \times precision}{recall + precision}$$

A perfectly accurate description with $f = 1$ has *recall* = 1 and *precision* = 1. In a perfectly accurate SOR or SOR^- description, all rectangles are pure in the sense that they contain same-class points only.

The F-measure does not fit situations where users want to specify constraints on either *recall* or *precision*. We introduce two additional measures to provide this flexibility. The first is *recall at fixed precision*; often, we want to fix *precision* at 1. The second is *precision at fixed recall*; often, we want to fix *recall* at 1. The measures can be found useful in many situations. If we can afford to lose points in C much more than to include points in C^-, then we can choose *recall at fixed precision of 1* to sacrifice *recall* and protect *precision* as in the maximum box problem [7]; in the opposite situation, we can choose *precision at fixed recall of 1* as in the red blue set cover problem [6].

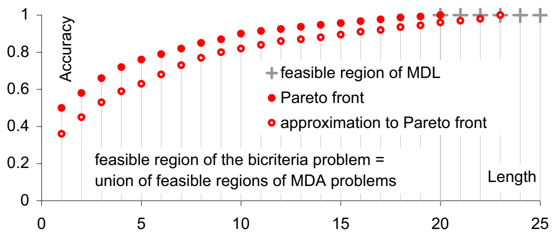


Figure 3. Accuracy vs. length.

3.2. MDA problem and MDL problem

Description length and accuracy are two conflicting objective measures that cannot be optimized simultaneously. The Pareto front for the bicriteria problem, as illustrated in Figure 3, offers the best trade-offs between accuracy and interpretability for a given format. To solve the bicriteria problem and obtain the best trade-offs, we introduce the novel Maximum Description Accuracy (MDA) problem.

Definition 3.1 (*Maximum Description Accuracy problem*) *Given a cluster C, a description format F, an integer l, and an accuracy measure, find a Boolean expression E in format F with $||E|| \leq l$ such that the accuracy measure is maximized.*

The optimal solutions to the MDA problems with different length specifications up to a maximal length constitute the Pareto front. The vertical lines in Figure 3 illustrate the feasible regions of the MDA problems, whose union is the feasible region of the bicriteria problem. The maximal length is the length of some shortest description with perfect accuracy, which is 20 in Figure 3. It is pointless to specify larger lengths as the best accuracy has been achieved. To determine this maximal length, we define the Minimum Description Length (MDL) problem, whose objective is to find some shortest description that covers a given cluster completely and exclusively, i.e., with $f = 1$.

Definition 3.2 (*Minimum Description Length problem*) *Given a cluster C and a description format F, find a Boolean expression E in format F with minimum length such that $(E \cdot C = C) \wedge (E \cdot C^- = \emptyset)$.*

The optimal solution to the MDL problem gives the maximal length to specify in solving the MDA problems. The feasible region of the MDL problem is also illustrated in Figure 3. Previous research only considered the MDL problem with SOR as the description format. However, in practice, perfectly accurate descriptions can become lengthy and hard to interpret for arbitrary shape clusters. The MDA problem allows trading accuracy for interpretability so that users can zoom in and out to view the clusters. From a "data compression" point of view, description requiring perfect accuracy resembles lossless compression while description allowing lower accuracy resembles lossy compression.

Assuming some given default alphabets as previously discussed, the L-problems, easier than their counterparts, can be related to some variants of the set cover problem. In particular, $L_{SOR,\Sigma_{pure}}$-MDL corresponds to the minimum set cover problem, which is known to be NP-hard. Other L-MDL problems are harder in the sense that they have larger search spaces with more general languages.

Given the *recall at fixed precision of 1* accuracy measure, the $L_{SOR,\Sigma_{pure}}$-MDA problem corresponds to the maximum coverage problem, which is known to be NP-hard. Given the *precision at fixed recall of 1* accuracy measure, the $L_{SOR,\Sigma_{pure}}$-MDA problem corresponds to the red blue set cover problem, which is also NP-hard [6]. Given the F-measure, the decision version of the $L_{SOR,\Sigma_{pure}}$-MDA problem is reducible to the decision problem of either of the two. Other L-MDA problems are harder in the sense that they have larger search spaces.

As we have seen, the cluster description problems, with different format and accuracy measure specifications as discussed, are all NP-hard. We present efficient and effective heuristic algorithms for these problems in the next section.

4. Description algorithms

In this section, we present three heuristic algorithms. Learn2Cover solves the MDL problem approximating the maximal length. Starting with the output of Learn2Cover, DesTree iteratively builds the so-called description tree for the MDA problems approximating the Pareto front. FindClans transforms the output descriptions from DesTree into shorter $kSOR^{\pm}$ descriptions without reducing the accuracy.

4.1. Learn2Cover

Given a cluster C, Learn2Cover returns a description E for C in either SOR or SOR^{-} format with $f = 1$. For this purpose, it suffices to learn a set of pure rectangles $\Re$ covering C and a set of pure rectangles $\Re^{-}$ covering C^{-} completely. Learn2Cover is carefully designed such that $\Re$ and $\Re^{-}$ are learned simultaneously in a single run; besides, the extra learning of $\Re^{-}$ does not come as a cost but rather a boost to the running time.

Sketch of Learn2Cover. To better explain the main ideas, we give the pseudocode for the simplified Learn2Cover and its major procedure cover() in the following.

In preprocessing(), the bounding box B_C is determined; the points in B_C are normalized against B_C and sorted along a selected dimension D_s. Ties are broken arbitrarily.

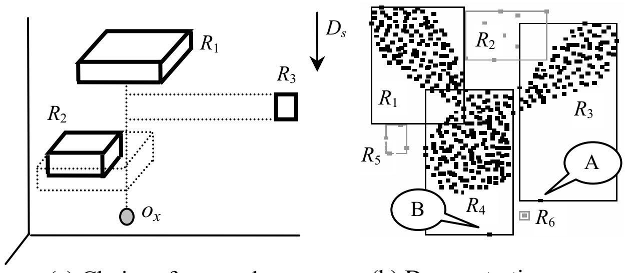
```
Algorithm: Learn2Cover
1  preprocessing(); // sort $B_C$ along $D_s$
2  foreach $(o_x \in B_C)$ { // processed in sorted order
3      if $(o_x \in C)$
4          cover($\Re$, $o_x$, $\Re^-$);
5      else
6          cover($\Re^-$, $o_x$, $\Re$); }
```

```
Procedure: cover($\Re$, $o_x$, $\Re^-$)
1  foreach $(R \in \Re^-)$ {
2      if $(cost(R, o_x) == 0)$
3          close $R$; }
4  foreach $(R \in \Re$ && $R$ is not closed$)$ {
5      if $(cost(R, o_x) == 0)$ {
6          extend $R$ to cover $o_x$;
7          return; }}
8  foreach $(R \in \Re)$ { // processed in ascending order of cost
9      if (no violation against $\Re^-$) {
10         expand $R$ to cover $o_x$;
11         return; }}
12 insert($\Re$, $R_{new}$); // $o_x$ was not covered. $R_{new} = o_x$
```

At the moment we suppose there are no mixed ties involving points from different classes. In general, the choice of D_s does not have a significant impact if the data is not abnormally sparse; thus D_s can be arbitrarily chosen. Nevertheless, Learn2Cover offers an option to choose D_s with the maximum variance, which makes the algorithm more robust against some rare, malicious cases such as large mixed tie groups. Learn2Cover is deterministic once D_s is chosen.

Initially $\Re = \emptyset$ and $\Re^{-} = \emptyset$. Let o_x be the next point from B_C in the sorted order to be processed. cover($\Re$, o_x, $\Re^{-}$) or cover($\Re^{-}$, o_x, $\Re$) is called upon depending on $o_x \in C$ or C^{-}. The two situations are symmetric.

Suppose $o_x \in C$, procedure cover($\Re$, o_x, $\Re^{-}$) chooses a non-closed $R \in \Re$ covering no points covered by rectangles in $\Re^{-}$ and with the minimum covering cost with respect to o_x to expand and cover o_x. Otherwise, a new rectangle R_{new} minimally covering o_x will be created and added to $\Re$ (line 12). A rectangle is closed if it cannot be expanded to cover any further point without causing a *covering violation*, i.e., covering points from the other class (lines 1, 2, 3). Violation checking can be expensive; therefore, we always calculate $cost(R, o_x)$ first. If there is a non-closed R with $cost(R, o_x) = 0$, we need to extend R only along D_s to cover o_x, in which case violation checking is unnecessary (lines 4, 5, 6, 7). Otherwise rectangles are considered in ascending order of $cost(R, o_x)$ for violation checking. The first qualified rectangle will be used to cover o_x (lines 8, 9, 10, 11). In the following we discuss covering cost and covering violation in more details.



(a) Choice of rectangles (b) Demonstration run

Figure 4. Learn2Cover.

Covering cost and choice of rectangles. The behavior of Learn2Cover largely depends on how the covering cost $cost(R, o_x)$ is defined, i.e., the cost of R to cover point o_x. This cost should estimate the reduction of the potential of R to cover further points after o_x. Intuitively, we want to choose R with the minimum increased volume, so that rectangles can keep maximal potential for future expansions without incurring covering violations.

Yet there are more issues to be concerned beyond this basic principle. First, when calculating the increased volume for R, we should not consider D_s. Since points are sorted on D_s and processed in the sorted order, the extension of R along D_s is the distance it has to travel to cover further points after o_x. To keep R short on D_s does not help to keep its potential for future expansions. In Figure 4(a), if we considered D_s, R_1 would have the biggest increased volume and not be chosen to cover o_x. But this saved space would be part of the expanded R_1 in covering any point after o_x, and whether R_1 had been expanded already to cover o_x or not would not make a difference. This suggests that the increased volume of R_1 with respect to o_x should be 0, ignoring D_s in the calculation.

Second, if the expanded R has a length of 0 or close to 0 in any dimension, its volume and increased volume will be 0 or close to 0, which makes R a favorable choice. But R may have traveled far along some other dimensions to cover o_x; its expansion potential would thus be limited. In Figure 4(a), both R_1 and R_3 require the same increased volume of 0 to cover o_x since R_1 has to be extended only along D_s and R_3 has a length of 0 in one of its dimensions. However, R_3 has to travel far along some dimensions other than D_s whereas R_1 does not. Moreover, R_2 does not have the increased volume of 0, but it seems not to be a worse choice than R_3 because o_x is more local to R_2 than R_3. Therefore, $cost$ needs to fix the illusion of 0 increased volume and take into account the locality of rectangles.

In the following, we propose a definition for $cost(R, o_x)$ with these issues in consideration. Let $l_j(R)$ denote the length of rectangle R along dimension D_j, and R' denote the expanded R in covering o_x.

$$vol(R) = \prod_{j=1..d, j\neq s} l_j(R)$$
$$aveIncVol(R, o_x) = (vol(R') - vol(R))^{1/(d-1)}$$
$$dist(R, o_x) = (\sum_{j=1..d, j\neq s} |l_j(R') - l_j(R)|^2)^{1/2}$$
$$cost(R, o_x) = aveIncVol(R, o_x) + dist(R, o_x)$$

D_s is ignored if we project R and o_x onto the subspace $D \backslash D_s$. $vol(R)$ is the volume of the projected R; $aveIncVol$ can be viewed as the increased volume averaged on each dimension; $dist$ is precisely the Euclidean distance from the projected o_x to the projected R. $cost$ is the sum of $aveIncVol$ and $dist$, i.e., we assign equal weights to both components of $cost$. According to this definition of $cost(R, o_x)$, the choices in Figure 4(a) would be R_1, R_2, and R_3 in priority order.

Sometimes there are no good rectangles available, in which case forcing greedy expansions may deteriorate the overall performance. Learn2Cover provides an expansion control parameter to limit the maximum distance each dimension can travel at each expansion. Since data is normalized, the default choice of 0.5 means that each expansion cannot exceed half of the span of B_C in each dimension. The parameter is user-specified but not sensitive. Without expansion control, Learn2Cover works generally well; but it may help in cases of extremely sparse or malicious datasets.

Figure 4(b) is a real run of Learn2Cover on a toy dataset. Dark and light points denote points in C and C^- respectively. Rectangles are numbered in ascending order of their creation time. Note that on processing A, a better choice of R_3 was made while R_4 was also available. R_3 had $cost$ of 0 with D_s ignored. If R_4 had been chosen to cover A, it would have been closed before covering B.

Covering violation and correctness. Learn2Cover is required to output pure rectangles, any covering of inter-class points will be considered as a violation.

BP considers all inter-class points in violation checking. In Learn2Cover, since points are processed in the sorted order, the only points that could lead to violations in the expansion of $R_i \in \mathfrak{R}$ ($\mathfrak{R}^-$) are currently processed points in $R_j \in \mathfrak{R}^-$ ($\mathfrak{R}$) that overlaps with the expanded R_i. Thus $\mathfrak{R}$ and $\mathfrak{R}^-$, the sets of rectangles to be learned, also exist to help each other in violation checking to dramatically reduce the number of points in consideration. A simple auxiliary data structure is maintained to avoid the possible performance deterioration in the presence of extremely dense and big rectangles. We omit the details due to the page limit.

Learn2Cover outputs pure rectangle collections $\mathfrak{R}$ and $\mathfrak{R}^-$ covering every point in C and C^- respectively. We examine procedure cover() to argue the correctness. From the definition of $cost$, $cost(R, o_x) = 0$ if and only if the projected o_x is in the projected R. In such case, if $R \in \mathfrak{R}$ ($\mathfrak{R}^-$) and $o_x \in C^-(C)$, R will not be able to cover any further point without covering o_x, which causes a violation and R will thus be closed (line 3). If $R \in \mathfrak{R}$ ($\mathfrak{R}^-$) and $o_x \in C$ (C^-), R can be extended along D_s to cover o_x as (line 6) without causing any violation. If there existed o' causing a violation, o' must have been processed before o_x with $cost(R, o') = 0$, which would have caused R to be closed. In line 10, R is expanded to cover o_x after violation checking. In line 12, R_{new} is a degenerate rectangle covering only one point o_x. Thus, upon completion of Learn2Cover, each $o_x \in B_C$ is covered without violation.

In the sketch of Learn2Cover, we have assumed there were no mixed ties involving points from different classes. This case may happen and even frequently on grid data. Mixed tying points may cause covering violation to one another. Learn2Cover identifies mixed tie groups in the preprocessing() step, and then some extra checking is performed on processing o_x belonging to a mixed tie group.

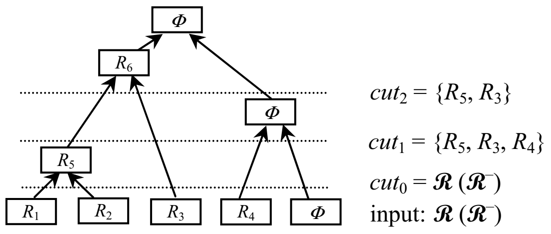
4.2. DesTree

The MDA problem is to find a description E with a user-specified length l for cluster C maximizing a given accuracy measure. Our algorithm DesTree takes the output from Learn2Cover, $\mathfrak{R}$ or $\mathfrak{R}^-$ whose cardinality approximates the maximal length, iteratively builds a so-called *description tree* approximating the Pareto front.

Description trees are tree structures resembling dendrograms to provide overviews on alternative trade-off descriptions of different lengths. Accordingly, DesTree resembles agglomerative hierarchical clustering to iteratively merge child nodes into parent nodes until a single node is left. Each node in a description tree represents a rectangle; and a normal merge operation produces a parent node that is the bounding box of its child nodes. The tree grows bottom-up along a series of merge operations.

Each horizontal cut in the tree defines a set of rectangles. For the so-called C-description tree, a cut set constitutes the vocabulary for a SOR description of C; for the so-called C^--description tree, a cut set constitutes the vocabulary for a SOR description of C^- leading to a SOR^- description of C. The cardinality of a cut set, the description length, equals to the number of links being cut. Each cut offers an alternative trade-off between description length and accuracy. The higher in the tree we cut, the shorter the length and the lower the accuracy.

Consider a SOR (SOR^-) description, merging two rectangles into their bounding box may cause *precision* (*recall*) to decrease; removing a rectangle may cause *recall* (*precision*) to decrease. Both operations trade the accuracy measure, say f, for shorter length and we want to consider both. To integrate the removal operation in building description trees, we add a symbolic node, the empty set $\emptyset$, into the leaf nodes and define the *merge* operator as follows.



Φ
R6
Φ
R5
R1
R2
R3
R4
Φ
$cut_2 = \{R_5, R_3\}$
$cut_1 = \{R_5, R_3, R_4\}$
$cut_0 = \mathfrak{R}\ (\mathfrak{R}^-)$
input: $\mathfrak{R}\ (\mathfrak{R}^-)$

Figure 5. Description tree.

Definition 4.1 (*merge*) R_i *merge* R_j = R_{parent} =
(1) *bounding box for* R_i *and* R_j *if* $R_i \neq \emptyset$ *and* $R_j \neq \emptyset$;
(2) $\emptyset$ *otherwise*

DesTree is a greedy approach starting from the input leaf nodes, a set of pure rectangles $\mathfrak{R}$ or $\mathfrak{R}^-$ generated by Learn2Cover, building the tree in a bottom-up fashion. Pairwise *merge* operations are performed iteratively, and the merging criterion is the biggest resulting accuracy measure. The C-description tree and C^--description tree are built separately in the same fashion.

Figure 5 exemplifies a description tree. $R_1 \sim R_4$ are the input rectangles. R_1 and R_2 are chosen for the first merge to give R_5. The second merge of R_4 and $\emptyset$ results in the removal of R_4. R_6, the parent node of R_5 and R_3, merges with $\emptyset$ to give the symbolic root. The lowest cut cut_0 is $\mathfrak{R}$ or $\mathfrak{R}^-$. Each cut corresponds to a SOR or SOR^- description. Take cut_2 as an example. For a C-description tree, cut_2 corresponds to $E_{SOR}(C) = R_5 + R_3$; for a C^--description tree, it corresponds to $E_{SOR^-}(C) = B_C - (R_5 + R_3)$. Description trees are not necessarily binary. A merge could result in more rectangles fully contained in the parent rectangle. Nonetheless, the merging criterion discourages branchy trees and Figure 5 is a typical example.

The merging process can be simplified for some accuracy measures. Given *recall at fixed precision of 1*, for the C-description tree, only *merge* operation (2) (the removal operation) needs to be considered, and the root is always $\emptyset$; for the C^--description tree, only *merge* operation (1) (the normal merge operation) needs to be considered, and the root is always B_{C^-}. For both cases, *precision* is guaranteed to be 1 and *recall* reduces along the merging process.

Given *precision at fixed recall of 1*, for the C-description tree, only *merge* operation (1) needs to be considered, and the root is always B_C; for the C^--description tree, only *merge* operation (2) needs to be considered, and the root is always $\emptyset$. For both cases, *recall* is guaranteed to be 1 and *precision* reduces along the merging process.

We can easily prove the accuracy measure, *recall at fixed precision of 1* or *precision at fixed recall of 1*, reduces monotonically along the merging process in DesTree. With respect to the F-measure, though evident in experiments, it is non-trivial to construct a proof of the same property.

4.3. FindClans

FindClans takes as input a cut (denoted by T in the following) from a description tree representing a SOR or SOR^- description, outputs a $kSOR^\pm$ description with shorter length and equal or better accuracy.

The algorithm is based on the concept of *clan*. Let SOR_V be a SOR description with vocabulary V, e.g., $SOR_V = R_1 + R_2$ for $V = \{R_1, R_2\}$. Intuitively, a clan $N \subseteq T$ is a group of rectangles that dominate (densely populate) a local region, so that by replacing them as a whole, SOR_N can be rewritten as a shorter and more accurate SOR^- description for the targeted points in the region.

Definition 4.2 (*clan*) *Given T as a cut from a C (C^-) -description tree, $N \subseteq T$ is a clan if $|N| - |N'| > 1$ and $B_N - SOR_{N'} \geq_{accu} SOR_N$ in describing $SOR_N \cdot C$ ($SOR_N \cdot C^-$), where B_N is the bounding box of N and N' a set of rectangles associated with N called the replacement of N. We also refer to $|N| - |N'| - 1$ as $N.score$.*

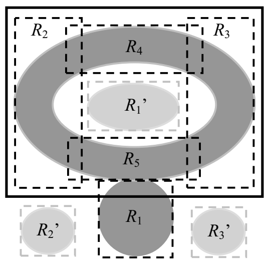
Note that the purpose of SOR_N is to describe $SOR_N \cdot C$ ($SOR_N \cdot C^-$) if T is from a C (C^-) -description tree. $N.score$ is the possible length reduction offered by a single clan N since SOR_N will be replaced by $B_N - SOR_{N'}$. Two clans N_1 and N_2 are disjoint if $N_1 \cap N_2 = \emptyset$. For a set of mutually disjoint clans, $Clans$, the total length reduction is at least $\sum N_i.score$ where $N_i \in Clans$.

Figure 6 uses the example in Figure 2 and illustrates how a clan can help to rewrite a SOR description represented by T into a shorter $kSOR^\pm$ description.

Suppose we have found $Clans$ for T and T is from a C-description tree, it is straightforward to rewrite the input SOR description $E_{SOR}(C) = SOR_T$ into a $kSOR^\pm$ description as illustrated in Figure 6. For each $N \in Clans$, we simply replace SOR_N in SOR_T by the shorter and more accurate $(B_N - SOR_{N'})$.

If T is from a C^--description tree, the input SOR^- description is $E_{SOR^-}(C) = B_C - SOR_T$. For each $N \in Clans$, we replace SOR_N in SOR_T by B_N and add back $SOR_{N'}$. As an example, let $E_{SOR^-}(C) = B_C - SOR_T = B_C - (R_1 + R_2 + R_3 + R_4 + R_5)$. Suppose we have a clan $N = \{R_2, R_3, R_4, R_5\}$ with replacement $N' = \{R'_1\}$, then $E_{kSOR^\pm}(C) = B_C - (R_1 + B_N) + R'_1$. The length reduction $= T.score = 4 - 1 - 1 = ||E_{SOR^-}(C)|| - ||E_{kSOR^\pm}(C)|| = 6 - 4 = 2$. It is easy to verify that after all such replacements, the resulting $E_{kSOR^\pm}(C)$ is shorter and more accurate than $E_{SOR}(C)$ or $E_{SOR^-}(C)$ with respect to any accuracy measure we discussed.

All we need is to find $Clans$. To simplify the task, we define N', the replacement of N, to contain each rectangle $R \in \mathfrak{R}^-$ ($\mathfrak{R}$) overlapping with B_N if T is from a C (C^-) -description tree. Then, it is guaranteed that $B_N - SOR_{N'} \geq_{accu} SOR_N$ in describing $SOR_N \cdot C$ ($SOR_N \cdot C^-$) since N' contains pure rectangles completely covering $B_N \cdot C^-$ ($B_N \cdot C$). Thus given a candidate clan N, N' is uniquely determined and N is a clan if $|N| - |N'| > 1$.



$\mathfrak{R} = \{R_1, R_2, R_3, R_4, R_5\}$
$\mathfrak{R}^- = \{R_1', R_2', R_3'\}$

$T = \mathfrak{R} = \{R_1, R_2, R_3, R_4, R_5\}$
$E_{SOR}(C) = R_1 + R_2 + R_3 + R_4 + R_5$
$N = \{R_2, R_3, R_4, R_5\} \subseteq T$
$N' = \{R_1'\}$
B_N = bounding box for N
$N.score = |N| - |N'| - 1 = 4 - 1 - 1 = 2$

$E_{kSOR^\pm}(C) = R_1 + (B_N - R_1')$
length reduction $= ||E_{kSOR^\pm}(C)|| - ||E_{SOR}(C)|| = 2$

Figure 6. Clan helps in length reduction.

Algorithm Findclans, as presented in the following, continues to call procedure findAClan() to find a clan N in the updated T and insert it into $Clans$. findAClan() first checks each pair of rectangles in T and finds $theN$ with the highest score. $bestN$ is used to keep track of the best stage of $theN$, which continues to grow greedily one more $R \in (T - theN)$ at a time resulting in the largest score increase, until no more rectangles available. $bestN$ is returned if it is a clan; otherwise, $NULL$.

```
Algorithm: FindClans ($T$)
1  $Clans \leftarrow \emptyset$;
2  $N \leftarrow findAClan(T)$;
3  while ($N\ != NULL$) {
4      insert($Clans$, $N$);
5      $T \leftarrow T - N$;
6      $N \leftarrow findAClan(T)$; }
7  return $Clans$;
```

```
Procedure: findAClan($T$)
1  find $theN$; // consider each pair in $T$
2  $bestN \leftarrow theN$;
3  while ($theN \subseteq T$) {
4      grow $theN$; // consider each $R \in (T - theN)$
5      if ($theN.score > bestN.score$)
6          $bestN \leftarrow theN$; }
7  if ($bestN.score \geq 1$)
8      return $bestN$;
9  else
10     return $NULL$;
```

5. Experimental results

We experimentally evaluated and compared our methods against CART (Salford Systems, Version 5.0) and BP [16]. While decision tree classifiers, as argued in related work, can be applied to the MDA and MDL problems, BP addresses the MDL problem only. We implemented Learn2Cover, DesTree, FindClans, and BP. For BP, we also implemented Greedy Growth and a synthetic grid data generator. To make our experiments reproducible, real datasets from the UCI repository [4] with numerical attributes and without missing values were used, where data records with the same class label were treated as a cluster. Note that, in the broad sense, a cluster can be used to represent an arbitrary class of labeled data that require discriminative generalization. The notion of rectangle can be extended (but not implemented in this version) to tolerate categorical attributes; e.g., to use sets instead of intervals. Rectangles do not provide generalization on the categorical attributes.

5.1. Comparisons with CART

To approximate the Pareto front, our approach starts with applying Learn2Cover for the MDL problem, then DesTree for the MDA problems to build description trees. Decision tree classifiers provide feasible solutions to both the MDL and MDA problems. We compared Learn2Cover and DesTree with CART on UCI datasets. In each experiment we described one class of points C, considering all points from other classes within B_C, the bounding box for C, as C^-.

Learn2Cover vs. CART. In each experiment, B_C was fed to Learn2Cover and CART. The CART parameters were set such that a complete tree without misclassifications could be built. The entropy method was used for tree splits.

Table 1. Learn2Cover vs. CART.

dataset	cls	dim	$\|C\|$	$\|C^-\|$	Learn2Cover	CART	FindClans
wine	2	13	71	12	2 / **1**	5 / 5	–0
iono	g	34	225	11	3 / **2**	7 / 6	–0
iris	vir	4	50	18	**3** / 3	5 / 5	–0
blocks	4	10	88	1588	31 / **12**	35 / 20	–6
blocks	2	10	329	4974	26 / **19**	48 / 49	–7
yeast	cyt	8	463	783	**69** / 97	144 / 174	–13
yeast	nuc	8	429	939	**74** / 87	164 / 170	–16
abalone	I	8	1342	2549	**165** / 179	313 / 317	–37
abalone	F	8	1307	2693	**214** / 259	454 / 449	–62
abalone	M	8	1528	2626	**251** / 278	510 / 488	–54

For each dataset, Table 1 presents the class label, the dimensionality, the cardinalities of C and C^-, and the results. For Learn2Cover and CART, a/b denotes the cardinalities of the two sets of rectangles covering C and C^- respectively. For CART, a and b correspond to the numbers of leaf nodes of the two classes. For Learn2Cover, they correspond to $|\mathfrak{R}|$ and $|\mathfrak{R}^-|$. The smallest a or b is highlighted in bold font. We observe that, on average, Learn2Cover needs only half of the description length required by CART. Results from other UCI datasets as well as synthetic datasets are not presented. However, the above observation holds consistently through all the experiments.

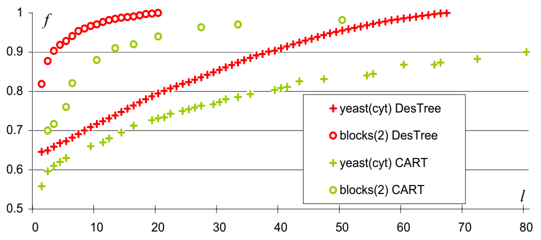


Figure 7. DesTree vs. CART.

The output of Learn2Cover was further fed into FindClans for additional length reduction. In Table 1, $-c$ denotes the additional reduction achieved by FindClans comparing to the shortest length (in bold font). The effectiveness of FindClans depends on the size and distribution of the input data; bigger and more complex datasets are likely to exhibit more clans, leading to more length reduction. We observe that, FindClans further reduces the shortest description length by about 20% on average and up to 50%.

DesTree vs. CART. In each experiment, DesTree took as input $\mathfrak{R}$ or $\mathfrak{R}^-$ returned by Learn2Cover. For CART, the complete tree without misclassifications was pruned step by step. In each step, the misclassifications for both classes were counted to calculate the F-measure. The number of rectangles covering the target class C or C^- was also recorded as the description length.

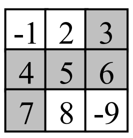
Figure 7 demonstrates the results of DesTree (C-description tree) and CART (target class C), for clarity, on only two of the UCI datasets used in Table 1. f and l denote the F-measure and description length respectively. As expected, for both methods, f decreases monotonically with decreasing l. However, the DesTree results clearly dominate the ones from CART. For each l, DesTree achieves a significantly higher f, and each f a significantly smaller l. Again, this observation holds consistently for all other experiments not presented, including ones on other UCI datasets and synthetic datasets, as well as C^--description tree experiments on both data types.

5.2. Comparisons with BP

While the focus of our study is on the MDA problem, BP addresses the MDL problem only. In this series of experiments, we compared Learn2Cover with BP on synthetic datasets as BP works on grid data. Our data generator follows exactly what [16] does for BP. It takes as input the dimensionality, the number of intervals of each dimension, and the density. Dense cells are randomly generated in a grid with specified density, then one of them is randomly selected to grow a connected dense cell set as cluster C, the rest of the dense cells in B_C constitute C^-.

In our experiments with BP, we did not limit the number of "don't care" cells for use and allowed BP to find the best possible results. Since BP only generates one set of rectangles, we used $\mathfrak{R}$ from Learn2Cover for the comparison. We studied the averaged percentage length reduction compared to BP for varying dataset size and dimensionality. We observed that in all cases Learn2Cover clearly outperformed BP, gaining 20% to 50% length reduction. As a general tendency, the reduction increased with increasing complexity of data. FindClans further improved the results, gaining an additional 25% length reduction on average.

BP starts with the maximal rectangles generated by Greedy Growth in a greedy manner. The "don't care" cells may come too late to be helpful, as illustrated in Figure 8.



C: $\{3, 4, 5, 6, 7\}$; C^-: $\{-1, -9\}$; "don't care" cells: 2, 8
Greedy Growth: $R_{47} + R_{456} + R_{36}$
BP: the same as Greedy Growth since any pairwise merge of rectangles would cause a violation covering -1 or -9
Learn2Cover: $R_{2356} + R_{4578}$

Figure 8. A typical case in BP.

Runtime was not a major concern of this study. We did not integrate the X-tree index, on which BP relies to reduce the violation checking time. Without indexing for both, we observed that Learn2Cover ran faster than BP by one to two orders of magnitude. Recall that Learn2Cover can significantly reduce the number of points in consideration for violation checking. If we assume constant number of rectangles, Learn2Cover has a worst case runtime of $O(|B_C|^2)$.

DesTree and FindClans also have quadratic worst case runtime in the number of input rectangles, $O(|\mathfrak{R}|^2)$ or $O(|\mathfrak{R}^-|^2)$ for DesTree and $O(|T|^2)$ for FindClans respectively. In particular, for DesTree, the accuracy calculation results of all possible pairwise merges of rectangles can be reused in each iteration, recalculation is needed only for the resulting parent rectangle, which takes linear time. For FindClans, in each call of procedure findAClan(), the results from "find $theN$" (line 1) can be reused.

6. Conclusions

In this paper, we systematically studied rectangle-based discriminative data generalization in the context of cluster description, which transforms clusters into patterns and provides the possibility of obtaining human-comprehensible knowledge from clusters. In particular, we introduced and analyzed novel description formats, SOR^- and $kSOR^\pm$, providing enhanced expressive power; we defined the novel Maximum Description Accuracy (MDA) problem, allowing users to specify different trade-offs between interpretability and accuracy; we also presented heuristic algorithms together with their experimental evaluations.

The concept of cluster in our study, in the narrow sense, is the output from clustering algorithms. Our study is motivated from and can find most applications in describing such clusters. In the broad sense, a cluster can be used to represent an arbitrary class of labeled data that require discriminative generalization.

Last but not least, cluster descriptions are patterns which can be stored as tuples in a relational table, so that a clustering and its associated clusters become queriable data mining objects. Therefore, this research can serve as a first step for integrating clustering into the framework of inductive databases [13], a paradigm for query-based "second-generation" database mining systems.

References

[1] R. Agrawal, J. Gehrke, D.Gunopulos, and P.Paghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, 1998.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley-Longman: Harlow, UK, 1999.

[3] P. Berman and B. Dasgupta. Approximating rectilinear polygon cover problems. In *Algorithmica*, 1997.

[4] C. Blake and C. Merz. UCI repository of machine learning databases. 1998.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[6] R. Carr, S. Doddi, G. Konjevod, and M. Marathe. On the red-blue set cover problem. In *SODA*, 2000.

[7] J. Eckstein, P. Hammer, Y. Liu, M. Nediak, and B. Simeone. The maximum box problem and its application to data analysis. In *Comput. Optim. Appl.*, volume 23, pages 285–298, 2002.

[8] U. Feige. A threshold of lnn for approximating set cover. In *Journal of the ACM*, volume 45(4), pages 634–652, 1998.

[9] B. Gao and M. Ester. Cluster description formats, problems, and algorithms. In *SDM*, 2006.

[10] B. Gao and M. Ester. Right of inference: Nearest rectangle learning revisited. In *ECML*, 2006.

[11] M. Garey and D. Johnson. *Computers and Intractability: A guide to the Theory of NP-completeness*. W.H. Freeman: New York, 1979.

[12] D. S. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Company: Boston, 1997.

[13] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. In *Communications of the ACM*, volume 39(11), pages 58–64, 1996.

[14] D. Johnson. Approximation algorithms for combinatorial problems. In *J. Comput. System Sci.*, 1974.

[15] V. A. Kumar and H. Ramesh. Covering rectilinear polygons with axis-parallel rectangles. In *STOC*, 1999.

[16] L. Lakshmanan, R. Ng, C. Wang, X. Zhou, and T. Johnson. The generalized MDL approach for summarization. In *VLDB*, 2002.

[17] W. Masek. Some NP-complete set covering problems. manuscript, MIT, Cambridge, MA, 1979.

[18] A. Mendelzon and K. Pu. Concise descriptions of subsets of structured sets. In *PODS*, 2003.