Cluster Description Formats, Problems and Algorithms

Byron J. Gao Martin Ester
School of Computing Science, Simon Fraser University, Canada, V5A 1S6
bgao@cs.sfu.ca ester@cs.sfu.ca

Abstract

Clustering is one of the major data mining tasks. So far, the database and data mining literature lacks systematic study of cluster descriptions, which are essential to provide the user with understandable knowledge of the clusters and support further interactive exploration. In this paper, we introduce novel description formats leading to more descriptive power. We define two alternative problems of generating cluster descriptions, Minimum Description Length and Maximum Description Accuracy, providing different trade-offs between interpretability and accuracy. We also present heuristic algorithms for both problems, together with their empirical evaluation and comparison to state-of-the-art algorithms.

1 Introduction

The general objective of data mining is finding useful patterns and knowledge 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 existing clustering algorithms represent clusters as sets of points. They do not generate patterns, do not summarize the dataset so as to provide insights into the data.

Cluster descriptions should generalize cluster contents in a human-understandable manner. For numerical data, hyper-rectangles generalize multi-dimensional points, and a standard approach is to summarize a set of points with a set of isothetic hyper-rectangles [1, 5, 6]. 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* ≤ 0.5 and $0 \leq$ *minutes in gym per week* ≤ 30) intuitively describes a group of "nerds". Such expressions can also be used as search conditions in SQL query statements to retrieve cluster contents, supporting query-based iterative mining and interactive exploration of the clusters.

To be understandable, cluster descriptions should appear short in length and simple in format. Sum of Rectangles (SOR) has been the canonical format for cluster descriptions in the database literature. SOR is simple and generally considered understandable. However, this relatively restricted format may lead to unnecessarily lengthy descriptions. We introduce two novel description formats, leading to more descriptive power yet still simple 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 bounding box. The $kSOR^{\pm}$ format allows describing different parts of a cluster separately, using either SOR or SOR^- descriptions.

Meanwhile, cluster descriptions should be accurate, which is a trade-off for interpretability. Previous research considered the Minimum Description Length (MDL) problem, generating perfectly accurate descriptions that cover a cluster completely and exclusively. Such descriptions can become very lengthy and hard to interpret for arbitrary shape or high-dimensional clusters. In order to specify alternative trade-offs between interpretability and accuracy, we introduce the novel Maximum Description Accuracy (MDA) problem with the objective to maximize the description accuracy at a given length. We also develop effective heuristic algorithms for solving the MDL and MDA problems with respect to different description formats.

Related research exists. Greedy Growth [1] constructs an exact covering of a cluster, a set of connected dense cells, with maximal isothetic rectangles. 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. Algorithm BP [5] further allows to cover some "don't care" cells to reduce the cardinality of the cover. Starting from the set of rectangles returned by Greedy Growth, BP greedily considers possible pair-wise merges of rectangles without covering undesired cells. Also motivated by database applications, [6] formally defines the MDL problem and theoretically studies concise descriptions. Some classification tools such as axis-parallel decision trees (e.g., [2]) can also be used for description purposes. Their preference for shorter trees coincides with the preference for shorter description length in the MDL problem; their pruning phase allows trading accuracy for shorter trees, which is similar to the motivation of the MDA problem.

We discuss description formats in section 2, formalize the MDL and MDA problems in section 3, present algorithms for both with respect to different formats in section 4, and conclude the paper in section 5.

2 Description Formats

In this section we discuss cluster description formats SOR, SOR^-, and $kSOR^\pm$.

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 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.

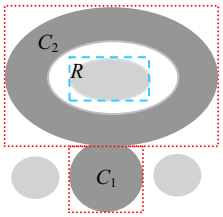
Two descriptions E_1 and E_2 are equivalent, denoted by $E_1 = E_2$, if they cover the same set of points. Logical equivalence implies equivalence but not vice versa. The length of a description E, $||E||$, is a major indication for its interpretability and defined as the cardinality of V_E. Consider C as a cluster, $E_F(C)$ denotes a description for C in format F. In the following, we also use B_C to denote the bounding box for C and $C^- = B_C - C$.

A description problem, viewed as searching, is to search good descriptions that optimize some objective function in a given description language. A more powerful language certainly contains better results at the cost of larger search space.

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 as seen in [1, 5]. 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 length. Simple formats such as SOR may restrict the search space too much leading to languages with low descriptive power. On the other hand, if a description format allows arbitrary Boolean operations, we certainly have the most general and expressive language containing the shortest descriptions, but such descriptions are likely hard to comprehend due to their complexity in format despite their succinctness in length. Moreover, complex formats bring difficulties in manipulation of symbols and design of efficient and effective searching algorithms.

Clearly, we require description languages with high expressive power yet in intuitively understandable formats. For this purpose, we introduce SOR^- descriptions and $kSOR^\pm$ descriptions.



SOR: 5
SOR^-: 4
$kSOR^\pm$: 3

$E_{kSOR^\pm}(C)$
$= E_{SOR}(C_1) + E_{SOR^-}(C_2)$
$= B_{C_1} + (B_{C_2} - R)$

Figure 1: Description formats.

Definition 2.1. (SOR^- *description*) *Given a cluster $C \subseteq U$, 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 $E_{SOR^-}(C)$.

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 1, consider C_2 as a single cluster, certainly SOR^- descriptions are favored. The shortest $E_{SOR}(C_2)$ has length 4 whereas the shortest $E_{SOR^-}(C2)$ 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.

$SOR^\pm$ generally serves well in describing compact and distinctive clusters. However, for the application of arbitrary-shaped and/or high-dimensional clusters, we may want to further increase the description power by allowing less restrictive formats.

Definition 2.2. ($kSOR^\pm$ *description*) *Given a cluster $C \subseteq U$ and a k-partition of C, $< C_1, C_2 ... C_k >$, a $kSOR^\pm$ description for C, $E_{kSOR^\pm}(C)$, is a Boolean expression in the form of $E_{SOR^\pm}(C_1) + E_{SOR^\pm}(C_2) + ... + E_{SOR^\pm}(C_k)$.*

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 1, 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.

The full version of the paper [3] formally proves the language corresponding to $kSOR^{\pm}$ is as powerful as *propositional* language, the most general description language considered in [6] that allows usual set operations of union, intersection and difference. Despite its exceptional descriptive power, the $kSOR^{\pm}$ format is very simple and conceptually clear. It is also well-structured to ease the design of searching strategies.

3 Description Problems

A description problem is to find a description for a cluster in a given format that optimizes some objectives.

Besides the goal of minimizing description length, we also want to maximize description accuracy. An accurate description should cover many points in the cluster and few points not in the cluster. For a description E of cluster C, we have $recall = |E \cdot C| \ / \ |C|$ and $precision = |E \cdot C| \ / \ |E|$. 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 f considers both and is the harmonic mean of the two.

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

A description with $f = 1$ has $recall = 1$ and $precision = 1$, which we call a perfect description. In a perfect description, all rectangles are pure in the sense that they contain same-class points only.

Two additional accuracy measures help to specify constraints on either *recall* or *precision*, "*recall* at fixed *precision*" and "*precision* at fixed *recall*". Often, we want to fix *precision* or *recall* at 1. The measures can be found useful in many situations; say, in a case we want to trade description accuracy for shorter length. It may well happen that we can afford to lose points in C much more than to include points in C^-, then we can choose the "*recall* at fixed *precision* of 1" measure to sacrifice *recall* and protect *precision*.

Description length and accuracy are two conflicting objective measures that cannot be optimized simultaneously. By constraining one and optimizing the other, we define the following two description problems.

DEFINITION 3.1. (*Minimum Description Length (MDL) problem*) *Given a cluster C and a description format F, find a Boolean expression E in format F with minimum length such that $(E \cdot C = C) \wedge (E \cdot C^- = \emptyset)$.*

DEFINITION 3.2. (*Maximum Description Accuracy (MDA) 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 MDL problem is a perfect description problem. It aims at finding a description of minimum length with the constraint of $f = 1$. The MDA problem appreciates imperfect descriptions as well and allows trading description accuracy for shorter length. The MDL problem can be considered as a special case of the MDA problem with no constraint l on the description length, in which case $f = 1$ can always be achieved.

Previous research in the database literature only considered the MDL problem with SOR as the description format. However, in practice, the MDA problem has many important applications. Perfect descriptions for arbitrary shape and/or high-dimensional clusters can easily contain hundreds of rectangles, which devastate interpretability. In such cases, solutions to the MDA problem can provide "zooming out" views of the cluster at different resolutions.

It is also interesting to motivate the description problems from a "data compression" point of view. There are many applications where we need to retrieve the original points in clusters. For example, in a query-based iterative mining environment [4], partial results may need to be stored and then retrieved when the mining process is resumed. In DBMS systems, an isothetic rectangle can be specified by a Boolean search expression. 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 entirely.

In this scenario, the cluster description process resembles encoding or compression, and the cluster retrieval process resembles decoding or decompression. 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 shorter description length. Meanwhile, shorter description length also speeds up the retrieval process by saving condition checking time [6]. While a perfect description (MDL) resembles lossless compression, an imperfect description (MDA) resembles lossy compression. The two are not interchangeable.

4 Description Algorithms

In this section we first present Learn2Cover for the MDL problem, and then DesTree for the MDA problem with respect to different accuracy measures. After that, we present FindClans, which takes advantage of the exceptional expressive power of $kSOR^{\pm}$ to shorten descriptions generated by either Learn2Cover or DesTree without reducing the accuracy. The experimental results of the three algorithms are also briefly summarized at the end of the section.

4.1 Learn2Cover. Learn2Cover returns a description for cluster C in either SOR or SOR^- format with $f = 1$. For this purpose, it suffices to learn a set of pure rectangles $\mathfrak{R}$ covering C and a set of pure rectangles $\mathfrak{R}^-$ covering C^- completely. Learn2Cover is carefully designed such that $\mathfrak{R}$ and $\mathfrak{R}^-$ are learned simultaneously in a single run; besides, the extra learning of $\mathfrak{R}^-$ does not come as a cost but rather a boost to the running time. Algorithm 1 and 2 give the pseudo codes for the simplified Learn2Cover and its subroutine cover().

Algorithm 1 Learn2Cover

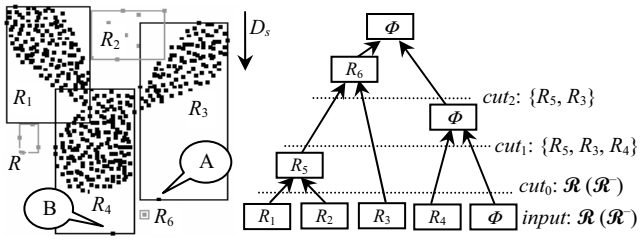
```
1. preprocessing(); //B_{C-sorted} returned, sorted along D_s
2. for each (o_x ∈ B_{C-sorted}) { //fetched in order
3.     if (o_x ∈ C)
4.         cover(ℜ, o_x, ℜ⁻);
5.     else
6.         cover(ℜ⁻, o_x, ℜ); }
```

Algorithm 2 subroutine cover($\mathfrak{R}$, o_x, $\mathfrak{R}^-$)

```
1. for each (R ∈ ℜ⁻)
2.     if (cost(R, o_x) == 0)
3.         close R;
4. for each (R ∈ ℜ && R is not closed) {
5.     if (cost(R, o_x) == 0) {
6.         extend R to cover o_x;
7.         return; }}
8. for each (R ∈ ℜ) { //fetch in ascending order of cost
9.     if (no violation against ℜ⁻) {
10.        expand R to cover o_x;
11.        return; }}
12. insert(ℜ, R_new); //o_x not covered; R_new = o_x
```

In preprocessing(), the points in B_C are sorted along selected dimension D_s to obtain $B_{C-sorted}$. Initially $\mathfrak{R} = \emptyset$ and $\mathfrak{R}^- = \emptyset$. Let o_x be the next point from $B_{C-sorted}$ to be processed. cover($\mathfrak{R}$, o_x, $\mathfrak{R}^-$) or cover($\mathfrak{R}^-$, o_x, $\mathfrak{R}$) is called to process it depending on $o_x \in C$ or C^-. The two situations are symmetric; we suppose $o_x \in C$ and show how cover($\mathfrak{R}$, o_x, $\mathfrak{R}^-$) works.

The subroutine cover($\mathfrak{R}$, o_x, $\mathfrak{R}^-$) chooses a non-closed $R \in \mathfrak{R}$ with minimum covering cost with respect to o_x to expand and cover o_x. This choice is on condition that R does not cover any points covered by rectangles in $\mathfrak{R}^-$. Otherwise, a new rectangle R_{new} minimally covering o_x will be created and added to $\mathfrak{R}$. 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. 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. 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.



(a) Learn2Cover (b) Description tree w.r.t. F-measure

Figure 2: Description algorithms.

The behavior of Learn2Cover largely depends on how $cost(R, o_x)$ is defined, i.e., the cost of R in covering point o_x. Intuitively, rectangles should keep maximal potential for future expansions without incurring covering violations. For a more detailed discussion see [3].

Figure 2(a) 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 on processing point A, a better choice of R_3 (*cost* 0) was made while R_4 was also available. If R_4 had been chosen to cover A, it would have been closed before covering point B.

BP considers all inter-class points in violation checking. In Learn2Cover, since points are processed in sorted order, the only points that could lead to violation in the expansion of $R_i \in \mathfrak{R}$ ($\mathfrak{R}^-$) are currently processed points in $R_j \in \mathfrak{R}^-(\mathfrak{R})$ such that the expanded R_i overlaps with R_j. Therefore, $\mathfrak{R}$ 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 and improve efficiency.

4.2 DesTree. DesTree is based on the novel concept of description tree. A *description tree* is a tree structure with each node representing a rectangle such that a parent rectangle is normally the bounding box for the children rectangles. Each horizontal cut of the tree defines a set of rectangles. For the so-called C-description tree, the set of rectangles constitute the vocabulary for a SOR description of C; for the so-called C^--description tree; the set of rectangles constitute the vocabulary for a SOR description of C^-, which leads to a SOR^- description of C. Each cut of a tree offers an alternative trade-off between description length and description accuracy. The higher in the tree we cut, the shorter the description length and the lower the description accuracy, as illustrated in Figure 2 (b).

DesTree is a greedy approach starting from leaf nodes, the sets of pure rectangles $\mathfrak{R}$ or $\mathfrak{R}^-$ generated by Learn2Cover, building the tree in a bottom-up fashion. Pairwise merges 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.

Merging rectangles in a C-description tree (C^--description tree) may cause *precision* (*recall*) to decrease; removing a rectangle in a C-description tree (C^--description tree) may cause *recall* (*precision*) to decrease. Both operations trade the F-measure for shorter length and we want to consider both. Removals of rectangles do not have resulting parents, thus there are no links as in a usual tree representation indicating the parent-children relationship. Then a cut of a "tree" after some removals may not be able to identify when the removals were performed and decide if the rectangles should be included in the cut set or not. In order to maintain a tree structure and be able to make cuts, we add the symbolic empty set $\emptyset$ as a leaf node and define the *merge* operator as follows.

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*

Figure 2 (b) shows a description tree example with respect to the F-measure. The lowest cut cut_0 is $\mathfrak{R}$ or $\mathfrak{R}^-$; cut_1 and cut_2 can clearly identify when R_4 was removed. Each cut corresponds to a SOR or SOR^- description. Take cut_2 as an example, if the tree is a C-description tree, cut_2 corresponds to $E_{SOR}(C) = R_5 + R_3$; if it is a C^--description tree, cut_2 corresponds to $E_{SOR^-}(C) = B_C - (R_5 + R_3)$.

For other accuracy measures, description trees can be built in a similar fashion [3]. A general property that these description trees share in common is the accuracy measure monotonically decreases along the merging process. Description trees are not necessarily binary. A merge could result in more rectangles fully contained in the parent rectangle. Nevertheless, the merging criterion discourages branchy trees and Figure 2 (b) is a typical example.

4.3 FindClans. FindClans takes a SOR or SOR^- description as input and outputs a $kSOR^{\pm}$ description with shorter length and equal or better accuracy. The input can be specified by a cut from a description tree. The algorithm is based on the concept of *clan*. Intuitively, a *clan* is a rectangular local region in which rectangles of different classes are unevenly distributed, which makes it possible to rewrite a SOR description of the local data into a shorter SOR^- description.

The full version of the paper [3] provides formal definition of *clan* and greedy heuristic of finding clans, together with instructions on rewriting input descriptions into the $kSOR^{\pm}$ format and its correctness proof.

4.4 Experimental results. We compared our methods with BP and CART (Salford 5.0) on UCI and synthetic datasets. For the MDL problem, Learn2Cover gained 50% length reduction over CART and 20% $\sim$ 50% over BP, leading to a substantial improvement in interpretability. FindClans achieved additional reduction of about 20%. For the MDA problem, Destree consistently achieved a significantly higher f than CART for all datasets and all values of length l, providing good trade-offs between description length and accuracy.

5 Conclusions

In this paper, we have investigated description formats, problems and algorithms for clusters of numerical records using sets of isothetic rectangles. We note that, the proposed framework is actually applicable to other scenarios requiring discriminative summarization of arbitrary sets of labeled objects. However, our study is in the context of clustering because the majority of data are unlabeled and clustering is the major unsupervised way of generating data labels. Besides, descriptions are generally only meaningful if the data exhibit clustered pattern allowing short, yet accurate descriptions.

Last but not least, cluster descriptions are patterns that can be stored as tuples in a relational table, so that a clustering and its associated clusters become queriable data mining objects. Thus, this research can serve as a first step for integrating clustering into the framework of inductive databases [4], 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] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[3] B. Gao and M. Ester. Summarizing data clusters: description formats, problems and algorithms. *Technical Report, TR 2006-01, Simon Fraser University*, 2006.

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

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

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