Entropy-based Subspace Clustering for Mining Numerical Data

By

CHENG, Chun-hung

Supervised By :

Prof. Ada Wai-chee Fu

Submitted to the Division of Department of Computer Science & Engineering

in partial fulfillment of the requirements for the

degree of Master of Philosophy

at the

Chinese University of Hong Kong

1999

Entropy-based Subspace Clustering for Mining Numerical Data

submitted by

CHENG, Chun-hung

for the degree of Master of Philosophy

at the Chinese University of Hong Kong

Abstract

Mining numerical data is a relatively difficult problem in data mining. Clustering is one of the techniques. We consider a database with numerical attributes, in which each transaction is viewed as a multi-dimensional vector. By studying the clusters formed by these vectors, we can discover certain behaviours hidden in the data. Traditional clustering algorithms find clusters in the full space of the data sets. This results in high dimensional clusters, which are poorly comprehensible to human. One important task in this setting is the ability to discover clusters embedded in the subspaces of a high-dimensional data set. This problem is known as subspace clustering. We follow the basic assumptions of previous work CLIQUE. It is found that the number of subspaces with clustering is very large, and a criterion called the coverage is proposed in CLIQUE for the pruning. In addition to coverage, we identify new useful criteria for this problem and propose an entropy-based method called ENCLUS to handle the criteria. Our major contributions are: (1) identify new meaningful criteria of high density and correlation of dimensions for goodness of clustering in subspaces, (2) introduce the use of entropy and provide evidence to support its use, (3) make use of two closure properties based on entropy to prune away insignificant subspaces efficiently (ENCLUS_SIG), (4) propose a mechanism to mine non-

minimal correlated subspaces which are of interest because of strong clustering (ENCLUS_INT), (5) experiments are carried out to show the effectiveness of the proposed method.

Acknowledgments

First of all, I would like to thank Prof. Ada W. C. Fu, my supervisor, for her guidance and patience. My research could not have been done reasonably without the insightful advice from her, for I know little about research at the beginning. In the past three years under her guidance, which covered an undergraduate final year project in addition to this dissertation, she improved my research and writing skills significantly, which I firmly believe is invaluable to the rest of my life.

I would like to thank Prof. M. H. Wong and Prof. Raymond W. H. Yeung, who have marked my term papers and taught me some very useful courses. I also thank Prof. Yi Zhang for helping me in working out the mathematical proofs of my algorithms.

My gratitude goes to the members of my research groups, Chun-hing Cai, Kin-pong Chan, Wang-wai Kwong, Po-shan Kam, Wai-ching Wong and Wai-chiu Wong. They have shared with me their experiences on doing research and given me a lot of inspiration. My research will not be nearly as successful without their help.

Finally, I wish to express my thanks to my colleagues, Kam-wing Chu, Yuk-chung Wong, Chi-wing Fu, Yuen Tsui, Hong-ki Chu and Yiu-fai Fung. They made a lot of useful suggestions on the technical problems I encountered during these years of research. They are all very intelligent and capable people.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Modern technology provides efficient and low-cost methods for data collection. However, raw data are rarely of direct benefit for higher level management, decision making or more intelligent analysis. Data mining, or *knowledge discovery in databases*, is the exploration and analysis of large data sets to discover meaningful patterns and rules. It aims at the construction of automatic or semi-automatic tools for the analysis of such data sets.

Contrary to top-down processes like *hypothesis testing* where the past behaviour is used to verify or disprove preconceived ideas, data mining is a bottom-up process in which priori assumptions on the data are not made. [17] describes knowledge discovery as a non-trivial process of extraction of implicit, previously unknown and potentially useful information from databases.

1.1 Six Tasks of Data Mining

According to [6], most tasks of data mining can be phrased in terms of six tasks, namely, classification, estimation, prediction, market basket analysis, clustering and description. No single algorithm is equally applicable to all these tasks. We need different tools and techniques to deal with different tasks. These tasks are

briefly described in this section.

1.1.1 Classification

Classification is a task of examining features of each object and assigning it to one of a predefined set of classes, e.g., classifying a group of animals into fishes, birds and mammals. The major characteristics of classification is that there is a well-defined definition of the classes. A training set, which consists of preclassified examples, is given. That is why classification is sometimes referred to as *supervised learning*.

1.1.2 Estimation

Estimation is similar to classification. Classification deals with discrete outcomes, but estimation deals with continuously valued outcomes. The estimation approach has the advantage that the individual record is rank ordered. For instance, a company with limited advertising budget can target at the customers who are most likely to use its services. Neural networks are well-suited to the task of estimation.

1.1.3 Prediction

Prediction is different from classification and estimation in that the objects are classified according to some predicted future behaviours or estimated future values. Historical data are used to build a model that predicts the future behaviours. We can only wait to see the accuracy of the created model.

1.1.4 Market Basket Analysis

One classical problem in data mining is the mining of binary association rule. Large amount of research on data mining went into this problem. It is originated from the analysis of buying patterns in supermarkets. That is why it is called market basket analysis. The following is an example of the binary association rule:

$$BuyApple \Rightarrow BuyOrange\ [0.9, 0.2]$$

This rule says that with high probability people buying apples will also buy oranges. A rule is associated with a pair of values, namely the confidence factor and support, which are 0.9 and 0.2 respectively in the above example. The confidence factor is the probability that the rule is true and the support is the fraction of the transactions in the database that contains all the items in the rule.

1.1.5 Clustering

Clustering is a task of segmenting a heterogeneous population into a number of more homogeneous groups of objects. Clustering is different from classification in that it does not depend on a predefined set of classes. There is no training set so it is sometimes called *unsupervised learning*.

1.1.6 Description

Description is a task of describing what happens in a complicated database. The description should help to understand the people, processes or products of

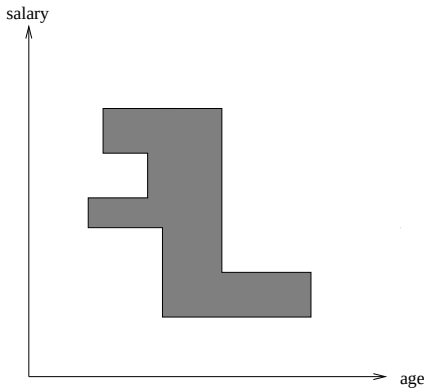


Figure 1.1: Example of a cluster.

the database. It hopefully gives an explanation for the behaviours of them. A good description can often convey important insights and directions to look for explanations.

1.2 Problem Description

In this thesis, we attempt to mine numerical data using clustering techniques. In particular, we work on the subspace clustering problem. We consider a database consisting of numerical attributes. Each transaction of this database is viewed as a multi-dimensional vector. Clustering is to discover homogeneous groups of objects based on the values of these vectors. Hence, we can study the behaviour of the objects by looking at the shapes and number of clusters. See Figure 1.1 for an example. The cluster in this figure describes the relationship between age and salary.

Not all clustering algorithms are suitable for our problem. They must satisfy some special requirements in order to be useful to us. One important requirement is the ability to discover clusters embedded in subspaces of high dimensional data. Given a space X with dimensions formed from a set of attributes S, a space Y with dimensions formed from a subset of S is called a **subspace** of

X. Conversely, X will be called a **superspace** of Y. For instance, suppose there are three attributes A, B and C. Clusters may exist inside the subspace formed by A and B, while C is independent of A and B. In such case, C is a noise variable. Since high dimensional information is hard to interpret, it is more desirable if the clustering algorithm can present the cluster in the subspace AB rather than the full space ABC. Real-life databases usually contain many attributes so that either there is no proper cluster in the full space, or knowing the existence of a cluster in the full space is of little use to the user. Therefore, the ability to discover embedded clusters is important. This problem is called **subspace clustering** in [2].

1.3 Motivation

The mining of binary association rule has been extensively studied in recent years, but databases in the real world usually have numerical attributes in addition to binary attributes. Unfortunately, mining numerical data is a more difficult problem and relatively little work has been done on this topic. Some previous work includes [20, 18, 19].

The mining of clusters is preferable to that of multi-dimensional quantitative association rules, because association rules consist of antecedent and consequent parts. We learn from statistics that it is possible to find correlation among different factors from raw data, but we cannot find the direction of implication and it can be risky to conclude any causal relationship from raw data [21]. Clustering is a method that finds correlations while not inferring any causal relationship.

Our most important requirement is, as mentioned in the previous section, the ability to discover embedded cluster. Also, data mining by definition deals with huge amount of data, which are often measured in gigabytes or even terabytes.

Although some traditional clustering algorithms are elegant and accurate, they involve too many complicated mathematical computations. These methods are shown to handle problem sizes of several hundreds to several thousands transactions, which is far from sufficient for data mining applications ([11] and [28]). Some algorithms, such as K-means [23, 8], assume that the whole data sets can be placed in main memory. These algorithms would require tremendous amount of disk accesses when the assumption does not hold. We need an algorithm that gives reasonable performance even on high dimensionality and large data sets.

We prefer clustering algorithms that do not assume some restrictive shapes for the clusters. Some clustering algorithms (e.g. CLARANS [29], BIRCH [37] and ScaleKM [7]) assume that the clusters are convex in shape. We would adopt a definition of cluster that does not have the above limitation. A good algorithm should also not make assumptions about the distribution of the data and not be sensitive to the existence of outliers. It should not require the user to specify some parameters on which the user would have difficulty to decide. For instance, the K-means algorithm requires the user to specify the number of clusters, which is often not known to the user. So in practice, we need to repeat the algorithm with different guesses to obtain the best result. Finally there should be a meaningful and effective way to convey the resulting clusters to the user for the purpose of data mining.

A solution to the above problem would consist of the following steps: (1) Find the subspaces with good clustering. (2) Identify the clusters in the selected subspaces. (3) Present the result to the user. We shall focus on Step (1). We propose an entropy-based approach to tackle this problem.

1.4 Terminology

Despite a lot of efforts to keep the terminology consistent in this thesis, there are cases that we have to resort to using different terms for the same meaning. Throughout the whole thesis, we use the terms *attribute*, *variable* and *dimension* interchangeably. These three terms sound more natural in the context of database, information theory and clustering respectively.

1.5 Outline of the Thesis

The rest of this thesis is organized as follows. In Chapter 2, we discuss some related work on similar problems. Chapter 3 points out some new criteria for good clustering and explains why they are needed. We also define the measure entropy for a numerical database, and discuss why it is a suitable measure for the listed criteria. Chapter 4 describes the proposed method ENCLUS (ENtropy-based CLUStering) in details. There are two variations of the algorithm, namely ENCLUS_SIG and ENCLUS_INT. Their common framework is discussed and the complexity is derived. In Chapter 5, we would look at some experimental results. Both synthetic and real-life data are used for experiments. A comparison to the previous work CLIQUE is also given. Chapter 6 discusses some miscellaneous enhancements applicable on ENCLUS. Chapter 7 gives a conclusion.

Part of the result of this thesis was published on *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 1999 (KDD-99)* [25].

Chapter 2

Survey on Previous Work

In this chapter, we introduce some previous work related to our problem. In Section 2.1, we go through recent research on data mining. We cover the recent research on clustering in Section 2.2. Although clustering is one of the techniques in data mining, this problem has been studied by people from different disciplines. We focus on methods proposed by the database research community.

2.1 Data Mining

We give a brief review on the research of data mining. Mining association rules is one of the hottest topics in the field of data mining. Here we explain the meaning of the association rules and its variations, as well as some rules containing numerical attributes. We introduce the famous Apriori [1] algorithm for mining binary association rules, and a variation of association rule that incorporates a statistical measure correlation. After that, we move to study implication rules. Implication rule changes the definition of association rules to give more "meaningful" rules.

2.1.1 Association Rules and its Variations

Most of the recent research on association rules have extended or modified the definition of association rules to introduce new types of rules. Apriori [1] is the classical algorithm for mining association rules.

The Apriori Algorithm

The problem of mining association rules is first introduced in [3]. The original form of association rules consists of binary attributes only. Newer studies often extend the association rules to contain non-binary attributes, so the original association rule is also known as binary association rule. An example of such rule is given at Section 1.1.4. Before the introduction of the Apriori algorithm, two other algorithms AIS [3] and SETM [24] are introduced for this problem, but they are not as efficient as Apriori.

The formal statement of the problem in [3] is as follows. The Apriori algorithm shares the same setting. Let $\mathcal{I}$ be a set of literals, called items. Let $\mathcal{D}$ be a set of transactions, where each transaction T is a set of items such that $T \subseteq \mathcal{I}$. Associated with each transaction is a unique identifier, called its TID. We say that a transaction T contains X, a set of some items in $\mathcal{I}$, if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ has support s in the transaction set $\mathcal{D}$ if $s\%$ of transactions in $\mathcal{D}$ contain $X \cup Y$. Its confidence is $support(X \cup Y)/support(X)$. The task is to discover all rules with support and confidence exceeding the predefined thresholds.

The problem of mining association rules can be divided into two subproblems.

1. Find all sets of item (itemsets) that have transaction support above a predefined threshold called minimum support. These items are known as

Algorithm 2.1 Apriori

```
1  L_1 = {large 1-itemsets};
2  for (k=2; L_{k-1} ≠ ∅; k++) do begin
3      C_k = apriori-gen(L_{k-1}); // New candidates
4      forall transaction t ∈ D do begin
5          C_t = subset(C_k, t); // Candidate contained in t
6          forall candidates c ∈ C_t do
7              c.count++;
8      end
9      L_k = {c ∈ C_k | c.count ≤ minsup }
10 end
11 Answer = ⋃_k L_k;
```

Figure 2.1: The Apriori Algorithm.

large itemsets.

2. Use the large item sets to generate the association rules. The rules must have a confidence level above another predefined threshold called minimum conference.

Step 2 is straightforward and trivial in terms of computational time. We focus on Step 1. The Apriori algorithm for solving Step 1 is given in Figure 2.1. Apriori is an iterative algorithm. The first pass simply scans the database to find the large 1-itemsets (L_1). In any subsequent pass k, the apriori-gen function is involved to generate the candidate itemsets C_k using the large itemsets of the previous pass L_{k-1}. The apriori-gen function has a join and prune step. In the join step, L_{k-1} self-joins to form C_k:

insert **into** C_k
select $p.item_1, p.item_2, \ldots, p.item_k, q.item_k$
from L_{k-1} p, L_{k-1} q
where $p.item_1 = q.item_1, \ldots, p.item_{k-2} = q.item_{k-2}$,
$p.item_{k-1} < q.item_{k-1}$

In the prune step, all k-itemsets having a $(k-1)$-subset not in L_{k-1} are deleted.

The function subset(C_k, t) returns all the candidate itemsets contained in the transaction t. The subset function can be implemented efficiently by storing the candidate itemsets C_k in a hash-tree and traversing the hash-tree when the subset function is involved.

The Apriori algorithm is shown to outperform AIS and SETM. Two similar algorithm AprioriTid and AprioriHybrid are also proposed in [1]. AprioriTid has better performance at higher passes. AprioriHybrid combines Apriori and AprioriTid. It uses Apriori at earlier passes and switches to AprioriTid at later passes.

Generalizing Association Rules to Correlation

[9] identifies a problem on the original definition of association rules and proposes the use of correlation in rule mining. They also generalize the association rules to consider both the presence and absence of items. They measure the significance of association via the χ^2-test for correlation from classical statistics. Correlation is upward closed in the itemset lattice, which gives us a useful pruning criteria.

Here is an example illustrating the problem of the original definition of association rules from [9]. In the following table, rows t and $\bar{t}$ represent buying and not buying tea respectively. Similarly, columns c and $\bar{c}$ represent buying and not buying coffee respectively.

	c	$\bar{c}$	$\sum row$
t	20	5	25
$\bar{t}$	70	5	75
$\sum col$	90	10	100

The support of the rule $t \Rightarrow c$ is 0.2, which is fairly high. The confidence, $P[c|t] = 0.8$, is quite high too. Therefore, we may conclude that this rule is valid.

However, since the a priori probability that a customer buys coffee is 0.9, a customer who is known to buy tea is actually less likely to buy coffee than the general population. By calculating the correlation $P[t \wedge c]/(P[t] \times P[c]) = 0.2/(0.25 \times 0.9) = 0.89 < 1$, we know there is actually a negative correlation between t and c. The rule $t \Rightarrow c$ is misleading.

The solution to this problem is to employ the χ^2-test for correlation in classical statistics. The test is capable for testing both positive and negative correlations. Before the introduction of the chi^2-test, we look at the proof on the closure property of correlated items.

Let $P(A)$ be the probability that event A occurs and $P(\bar{A}) = 1 - P(A)$ be the probability that event A does not occur. We want to show that if any two items are correlated, the superset of the items must also be correlated. Hence, correlation is *upward closed* in the itemset lattice. The proof is by contradiction.

Proof Suppose A and B are correlated but A, B and C are not.

$$\begin{aligned} P(AB) &= P(ABC) + P(AB\bar{C}) \\ &= P(A)P(B)P(C) + P(A)P(B)P(\bar{C}) \\ &= P(A)P(B) \end{aligned}$$

We can derive similar formulae for $P(A\bar{B})$, $P(\bar{A}B)$ and $P(\bar{A}\bar{B})$. They imply A and B are independent, which is a contradiction. □

Because of the closure property, the itemsets of interest form a *border* in the itemset lattice. This border encodes all the useful information about the interesting itemsets. The algorithm on [9] uses this closure property to prune away the itemsets above the border in the itemset lattice as they do not provide extra information. The algorithm is based on Apriori but replaces the test for

confidence with a χ^2-test for correlation and adds the pruning criteria from the border property.

Here we introduce the χ^2-test for independence. Let $I = \{i_1, \ldots, i_k\}$ be a set of k items. If we have a series of n trials, we denote the number of times item i_j occurs as $O_n(i_j)$. Let R be $\{i_1, \bar{i_1}\} \times \ldots \times \{i_k, \bar{i_k}\}$ and $r = r_1 \ldots r_k \in R$. R is the set of all possible transaction values, which forms a k-dimensional table called a *contingency table*. The value r denotes a *cell* in R. Let $O(r)$ denote the number of transactions falling into cell r. The expectation $E[i_j]$ is calculated under the assumption of independence. Thus,

$$\begin{aligned} E[i_j] &= O_n(i_j) \text{ for a single item} \\ E[\bar{i_j}] &= n - O_n(i_j) \\ E[r] &= n \times E[r_1]/n \times \ldots \times E[r_k]/n \end{aligned}$$

Then we can calculate the χ^2 statistic:

$$\chi^2 = \sum_{r \in R} \frac{(O(r) - E[r])^2}{E[r]}$$

If all variables were really independent, the χ^2 value would be 0. If it is higher than a cutoff value, which can be obtained from a χ^2 distribution table given the required significance, then we would reject the independence assumption.

That paper also proposes the measure of interest. The interest of A and B is defined as

$$\frac{P[A \wedge B]}{P[A]P[B]}$$

Interest allows the detection of negative correlation. To illustrate this, con-

sider an example with the following interest values.

	i_2	$\bar{i_2}$
i_1	1.07	0.44
$\bar{i_1}$	0.89	1.91

The cell $\bar{i_1}\bar{i_2}$ has the most extreme interest, indicating not having the property of i_2 is correlated with not having the property of i_1. This kind of negative association is not usually mined in the classical framework.

The experiment in that paper shows the use of correlation gives more meaningful rules and the new pruning criteria causes an improvement in performance by reducing the number of candidate itemsets.

Implication Rules

The insufficiency of the measures used in association rules is identified by [10]. In that paper, a new measure, *conviction* is proposed as a replacement of confidence. The rules measured in conviction is called *implication rules*.

The definition of conviction for $A \Rightarrow B$ is

$$P(A)P(\neg B)/P(A, \neg B).$$

The measure has the following advantages:

- It does not have the flaw of confidence as mentioned in [9].
- *Interest* is only a measure of departure from independence. It does not measure implication since its definition, $P(A, B)/P(A)P(B)$, is completely symmetrical. Conviction does not have this problem.

- For rules which hold 100% of time, the rules have the highest possible conviction value of ∞. Such rules may have an interest value only slight larger than 1.

2.1.2 Rules Containing Numerical Attributes

The studies on association rules focus on a database of binary or categorical attributes. [20, 18, 19] are among the earliest work on extending association rules to contain numerical attributes. In [20], we consider association rules of the form

$$(Balance \in I) \Rightarrow (CardLoan = yes)$$

which implies that bank customers whose balances fall in a range I tend to use card loan with a high probability.

This kind of rule is one-dimensional because there can be only one numerical attribute in the rule. A follow-up work [18] goes further by extending the rule to the following form

$$((Age, Balance) \in P) \Rightarrow (CardLoan = Yes)$$

which is two-dimensional since two numerical attributes are involved. It implies that bank customers whose ages and balances fall in a planar region P tend to use card loan with a high probability.

One application of the two-dimensional association rule is the construction of a decision tree [19]. In each node of the decision tree, we consider a family $\mathcal{R}$ of grid-regions in the plane associated with a pair of attributes. For $R \in \mathcal{R}$, the

data can be split into two classes: data inside R and data outside R.

The detailed algorithms for the generation for these rules and decision trees are based on geometry. We would give only the basic idea.

To mine one-dimensional association rule, we first divide the data in equi-depth buckets B_1, ..., B_N according to the numerical attribute (e.g. $0 \leq Balance < 1000$ in the first bucket, $1000 \leq Balance < 2500$ in the second bucket... etc.) Now we only consider rules whose ranges are combinations of consecutive buckets. Denote the size of B_i by u_i and the number of tuples in B_i satisfying the objective condition C by v_i. Consider a sequence of points $Q_k = (\sum_{i=1}^{k} u_i, \sum_{i=1}^{k} v_i)$. In the rule,

$$A \in (\text{range of } B_{m+1} \ldots B_n) \Rightarrow C$$

The slope of $Q_m Q_n$ gives the confidence of the rule and x-coordinate of Q_m minus x-coordinate of Q_n gives the support of the rule. Hence, we can apply methods in geometry to find out the rules with sufficient support and confidence.

Now we move to two-dimensional association rule. To simplify the problem, only two classes of regions, *rectangle* and *admissible region*, are considered. Admissible regions are connected *x-monotone regions*, whose intersection with any vertical line is undivided. Like one-dimensional rule, the data are divided into equi-depth buckets. Suppose there are two numeric attributes A and B. We would distribute the values of A into N equal-depth buckets and do the same for the values of B. Hence we can image there are $N \times N$ cells in a two-dimensional plane. Now only regions which are union of cells are considered. Again, we employ methods in geometry to find out the regions with sufficient support and confidence.

In [19], the two-dimensional association rule is applied to the construction

of the decision tree to reduce the size of the tree. In each node of the decision tree, the data are split into two sets, namely inside a region R and outside R. The choice of region R is different from that in a sole two-dimension rule because the goal of optimization is no longer confidence or support, but the entropy of splitting. The algorithm applies the method in geometry to perform the optimization.

2.2 Clustering

In this section, we describe some previous work done on the clustering problem. We focus on the work by the database research community since clustering has been extensively studied by people from different disciplines. To have a general overview on the clustering problem, please refer to the books on clustering [23, 26, 5, 28].

CLARANS [29] is based on randomized search to reduce the search space in the K-means approach. DBSCAN [16] relies on a density-based notion of clusters which is designed to discover clusters of arbitrary shape. It makes use of some spatial data structure for efficient retrieval of the data sets. DBCLASD [35] is based on the assumption that the points inside a cluster are uniformly distributed. The algorithm employs the χ^2-test from statistics to verify the distribution of the clusters. CLARANS, DBSCAN and DBCLASD are all targeted on spatial data.

Incremental DBSCAN [15] improves the DBSCAN algorithm to handle the update of the database efficiently. It takes advantage of the density-based nature of DBSCAN where insertion and deletion of an object only affects the cluster membership of the neighborhood of this object. This algorithm is considerably faster than DBSCAN when the database is updated frequently. GRIDCLUS [33]

uses a multidimensional grid structure, which is a variation of Grid File [30], to organize the value space surrounding the pattern values. The patterns are grouped into blocks and clustered with respect to the blocks by a topological neighbour search algorithm.

Traditional methods like K-means or K-medoid use one point (the mean or medoid) to represent the cluster when calculating the distance between a point and the cluster. CURE [22] extends them by representing each cluster by a certain fixed number of points. A parameter can be set to adjust the representative points so that K-means and the graph theory algorithm based on minimum spanning tree (MST) [23] become two special cases of CURE. The result is an algorithm that recognizes non-spherical clusters while not particularly sensitive to outliers.

BIRCH [37] is a dynamical and incremental method to cluster the incoming points. An important idea of BIRCH is to summarize a cluster of points into a clustering feature vector. This summary uses much less storage than storing all data points in the cluster. A CF-tree is built which splits dynamically. Clusters are stored in the leaf nodes. ScaleKM [7] makes use of a scalable clustering framework and applies it to the K-means algorithm. The clusters found are compressed using sufficient statistics, which is identical to the clustering feature vector in BIRCH. This resolves huge memory requirement of K-means so ScaleKM is suitable for large data sets.

None of the above algorithms satisfies our most important requirement — the ability to identify clusters embedded in subspaces of high-dimensional data. CLIQUE [2] is the only published algorithm we are aware of that satisfies this requirement. Since we follow closely the problem setting of CLIQUE, we shall describe it in more details. Before introducing CLIQUE, we give a comparison of the features of the clustering algorithms at Table 2.2.

Name of algorithm	Spherical clusters only	Sensitive to outliers	Affected by input order	Discover embedded clusters
MST	N	Y	N	N
K-means	Y	Y	N	N
CLARANS	Y	Y	Y	N
DBSCAN	N	N	N	N
DBCLASD	N	N	N	N
Incremental DBSCAN	N	N	Y	N
GRIDCLUS	N	Y	Y	N
BIRCH	Y	N	N	N
ScaleKM	Y	Y	N	N
CURE	N	N	N	N
CLIQUE	N	N	N	Y

Table 2.1: Comparison of clustering algorithms.

2.2.1 The CLIQUE Algorithm

First we introduce the target problem and assumptions of CLIQUE [2]. A set of data points and two parameters, ξ and τ, are given. We discretize the data space $\mathcal{S}$ into non-overlapping rectangular *units*, which is obtained by partitioning every dimension into ξ intervals of equal length. A unit is **dense** if the fraction of total data points contained in the unit is greater than the threshold τ. **Clusters** are unions of connected dense units within a subspace. We need to identify the dense units in different subspaces. The CLIQUE algorithm can be divided into the following three steps: (1) Find dense units and identify subspaces containing clusters. (2) Identify clusters in the selected subspace. (3) Generate minimal description for the clusters in disjunctive normal form.

Although it is theoretically possible to create a histogram in all spaces to identify the dense units. This method would be computationally infeasible when the number of dimensions is large. To reduce the search space, a bottom-up algorithm is used that exploits the monotonicity of the clustering criterion with respect to dimensionality: if a collection of points S is a cluster in a k-dimensional space, then S is also part of a cluster in any $(k-1)$-dimensional projections of the space. The algorithm is iterative: First find 1-dimensional dense units by making a pass over the data. Having determined $(k-1)$-dimensional dense units, D_{k-1}, the candidate k-dimensional units, C_k, are determined using the candidate generation procedures. A pass is made over the data to determine those candidate units that are dense, D_k. The algorithm iterates the above with increasing dimensionality, and terminates if no new candidates are found.

The candidate generation procedure is similar to the one adopted in the well-known Apriori algorithm [1] for mining association rules. It self-joins D_k to form C_k. The join condition is that the units share the first $k-2$ dimensions. Let $u.a_i$ represents an identifier for the ith dimension of the unit u and $u.[l_i, h_i)$ represents

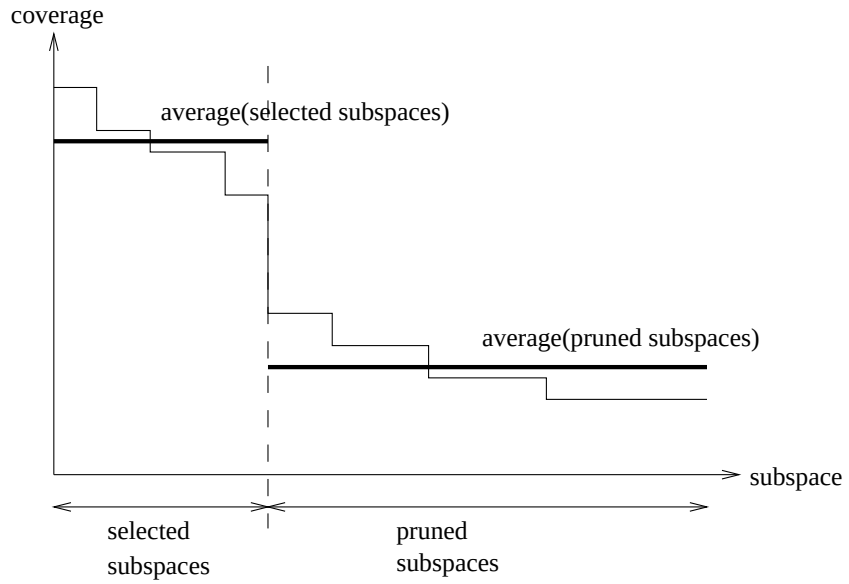


Figure 2.2: MDL-based pruning.

its interval in the ith dimension.

insert into C_k
select $u_1.[l_1, h_1)$, $u_1.[l_2, h_2)$, ..., $u_1.[l_{k-1}, h_{k-1})$, $u_2.[l_{k-1}, h_{k-1})$
from D_{k-1} u_1, D_{k-1} u_2
where $u_1.a_1 = u_2.a_1$, $u_1.l_1 = u_2.l_1$, $u_1.h_1 = u_2.h_1$,
$u_1.a_2 = u_2.a_2$, $u_1.l_2 = u_2.l_2$, $u_1.h_2 = u_2.h_2$, ...,
$u_1.a_{k-2} = u_2.a_{k-2}$, $u_1.l_{k-2} = u_2.l_{k-2}$, $u_1.h_{k-2} = u_2.h_{k-2}$,
$u_1.a_{k-1} < u_2.a_{k-1}$

We then discard those dense units from C_k which have a projection in $(k-1)$-dimensions that is not included in C_{k-1}.

As the number of dimensions increases, the above method may still produce a large amount of dense units in the subspace and the pruning above may not be effective enough. CLIQUE uses a new criteria for the pruning of subspace which is based on the **coverage**. The coverage x_{S_j} of a subspace S_j: $x_{S_j} = \sum_{u_i \in S_j} count(u_i)$ is the fraction of the database that is covered by the dense units, where $count(u_i)$ is the number of points that fall inside u_i. Subspaces with high coverages are selected and those with low coverages are pruned away. A minimal

code length method chooses subspaces which is likely to contain clusters. The subspaces $S_1, \ldots, S_n$ are sorted descendingly according to their coverage. We want to divide the subspaces into the selected set I and the prune set P so that the subspaces with high coverages are selected and those with low coverages are pruned away (see Figure 2.2). The code length is calculated as follows:

$$\mu_I(i) = \left\lceil \frac{\sum_{1 \le j \le i} x_{S_j}}{i} \right\rceil \; ; \; \mu_P(i) = \left\lceil \frac{\sum_{i+1 \le j \le n} x_{S_j}}{n-i} \right\rceil$$

$$\begin{aligned} CL(i) = \quad & \log_2(\mu_I(i)) + \textstyle\sum_{1 \le j \le i} \log_2(|x_{S_j} - \mu_I(i)|) + \\ & \log_2(\mu_P(i)) + \textstyle\sum_{i+1 \le j \le n} \log_2(|x_{S_j} - \mu_P(i)|) \end{aligned}$$

We choose the value of i whose code length is minimized as the optimal cut point. Hence, $S_1, \ldots, S_i$ belong to I and $S_{i+1}, \ldots, S_j$ belong to J. I is the set of subspaces likely to contain clusters while the dense units in the set of subspaces J are discarded to save memory. Note that it is possible to miss some legitimate clusters by using the minimal code length method.

When the subspaces containing clusters are identified, the clusters in each subspace are to be determined. Recall that clusters are connected dense units. We can simply use a depth-first search algorithm [4] to find the connected components. The final step is to generate minimal cluster descriptions. The description is given in the form of DNF expression, e.g. $((30 \le \text{age} < 50) \wedge (4 \le \text{salary} < 8)) \vee ((40 \le \text{age} < 60) \wedge (2 \le \text{salary} < 6))$. This is equivalent to a union of some hyper-rectangular regions. The regions can be found by a greedy growth method. We start with any dense unit and greedily grow a maximal region in each dimension. The process is repeated until the union of all regions cover the whole cluster. Then, we need to remove the redundant regions. This is achieved by repeatedly removing the smallest redundant region until no maximal region

can be removed. Break ties arbitrarily in the process of removing redundant region. This would give us the DNF expression describing the clusters.

Chapter 3

Entropy and Subspace Clustering

In this chapter, we discuss the properties of the subspaces of good clustering and define the criteria for subspace clustering. We propose to use *entropy*, which originates from information theory, as a measure of the quality of clustering. We support the use of entropy as the measure by showing it can handle our proposed subspace clustering criteria.

3.1 Criteria of Subspace Clustering

There are many factors to be considered for a clustering algorithm in data mining. We mentioned some of these in the introduction: efficiency, shape of clusters, sensitivity to outliers, and the requirements of parameters. A clustering algorithm assume a certain set of criteria for a cluster, as well as criteria for what is a good clustering given a set of data.

In addition to the clustering problem, we would like to handle the problem of determining subspaces that have "good clustering". We therefore need addition criteria for determining which of two clustering for two different sets of data is better. For CLIQUE there is the definition of the **coverage**, which is used as the measurement of the goodness of a clustering. This is a reasonable criterion since

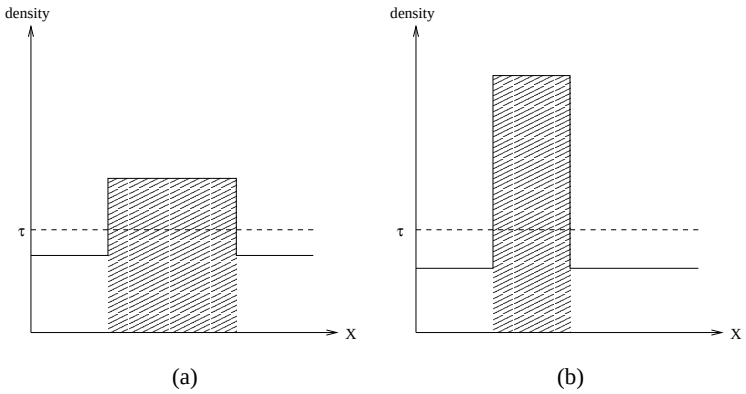


Figure 3.1: Example of two data sets with equal coverage but different densities.

a subspace with more distinguished clusters will have high coverage, whereas a subspace with close to random data distribution will have low coverage. However, we believe that other criteria are also needed. The first criterion that we add is the criterion of high density.

3.1.1 Criterion of High Density

Suppose we use only the coverage for measurement of goodness. A problem case is illustrated in Figure 3.1. It shows the probability density function of a random variable X. The value of coverage can be represented by the area of the shade portion since coverage is the fraction of the database that is covered by the dense units. In this example, both cases (a) and (b) have the same coverage. However, this contradicts with our intuition, because the points in case (b) is more closely packed and more qualified as a cluster.

3.1.2 Correlation of Dimensions

The third criterion that we consider is related to the correlation of dimensions. We note that finding subspaces with good clustering may not be always be helpful, we also want the dimensions of the subspace to be correlated. The reason is

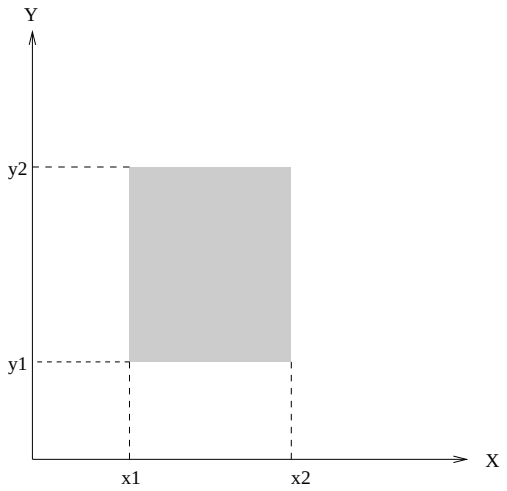


Figure 3.2: Problem with independent dimensions.

that although a subspace may contain clusters, this may not be interesting to us if the dimensions are independent to each other. For example, Figure 3.2 shows such a scenario in 2D. In this example, since all the data points projected on X lies on $[x1, x2)$ and projected on Y lies on $[y1, y2)$, the data objects must be distributed at $[x1, x2) \times [y1, y2)$ in the joint space. If the points are uniformly distributed at $[x1, x2) \times [y1, y2)$, although there is a cluster, looking at the joint space gives us no more knowledge than looking at each of the dimensions independently.

Hence, we also require the dimensions of the subspace to be correlated. Note that when we say correlated here, we mean the dimensions are not completely independent but it need not mean there is a very strong correlation.

Having identified a number of criteria for clustering, we shall find a metric that can measure all the criteria simultaneously. A subspace which has good clustering by the criteria will have high score in this metric. Then we can set a threshold on this measurement and find subspaces which exceed this threshold. The metric that we use is the entropy, which we shall discuss in the next section.

3.2 Entropy in a Numerical Database

We propose to use an entropy-based method. The method is motivated by the fact that a subspace with clusters typically has lower entropy than a subspace without clusters.

Here we introduce the concept of entropy. Entropy is a measure of uncertainty of a random variable. Let X be a discrete random variable, $\mathcal{X}$ be the set of possible outcomes of X and $p(x)$ be the probability mass function of the random variable X. The entropy $H(X)$ is defined by the following expression [13].

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$

If the base of log is 2, the unit for entropy is bit. If the natural log is used, the unit for entropy is nat. Note that 1 nat = 1.44 bits [34]. When there are more than one variable, we can calculate the joint entropy to measure their uncertainty.

$$H(X_1, \ldots, X_n) = -\sum_{x_1 \in \mathcal{X}_1} \cdots \sum_{x_n \in \mathcal{X}_n} p(x_1, \ldots, x_n) \log p(x_1, \ldots, x_n)$$

When the probability is uniformly distributed, we are most uncertain about the outcome. The entropy is the highest in this case. On the other hand, when the data points have a highly skewed probability mass function, we know that the variable is likely to fall within a small set of outcomes so the uncertainty and the entropy are low.

3.2.1 Calculation of Entropy

Similar to CLIQUE, we divide each dimension into intervals of equal length Δ, so the high-dimensional space is partitioned to form a grid. Suppose the data set is scanned once to count the number of points contained in each cell of the

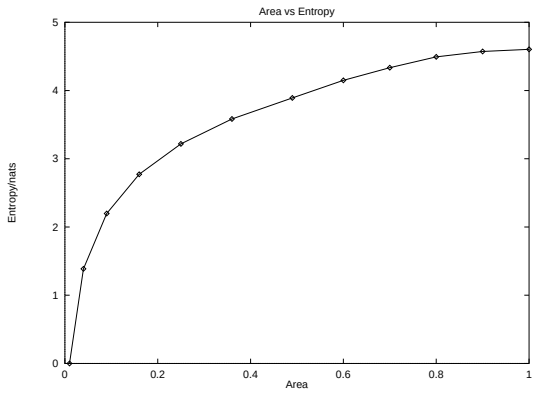


Figure 3.3: Area of Cluster vs Entropy.

grid. The density of each cell can thus be found. Let $\mathcal{X}$ be the set of all cells, and $d(x)$ be the density of a cell x in terms of the percentage of data contained in x. We define the entropy of the data set to be:

$$H(X) = -\sum_{x \in \mathcal{X}} d(x) \log d(x)$$

When the data points are uniformly distributed, we are most uncertain where a particular point would lie on. The entropy is the highest. When the data points are closely packed in a small cluster, we know that a particular point is likely to fall within the small area of the cluster, and so the uncertainty and entropy will be low.

Figure 3.3 shows the result of an experiment studying the relationship between the area of cluster in a two dimensional space [0,1) $\times$ [0,1). The smaller the area of the cluster, the more closely packed the points and the lower the entropy.

The size of interval Δ must be carefully selected. If the interval size is too small, there will be many cells so that the average number of points in each cell can be too small. On the other hand, if the interval size is too large, we may not be able to capture the differences in density in different regions of the space. Unfortunately, without knowing the distribution of the data sets, it is difficult

to estimate the minimal average number of points required in each cell to have the correct result. It is inappropriate to assume any distribution because that is exactly what we are studying. We suggest that there should be at least 35 points in each cell on the average since 35 is often considered as the minimum sample size for large sample procedures [14]. The size of interval Δ should be set accordingly.

3.3 Entropy and the Clustering Criteria

In Section 3.1, we propose the use of 3 criteria for the goodness of clustering: high coverage[1], high density and dimensional correlation. In this section, we discuss how the use of entropy can relate to the criteria we have chosen for the selection of subspaces. First we list the symbols used in the discussion in Table 3.1.

n	the total number of units
k	the total number of dense units
τ	the threshold on the density of a dense unit (in percentage of the data)
c	coverage (percentage of data covered by all dense units)
$p_1, \ldots, p_k$	the densities of the dense units
$p_{k+1}, \ldots, p_n$	the densities of the non-dense units

Table 3.1: Notations used in the discussion of entropy and clustering criteria.

3.3.1 Entropy and the Coverage Criterion

To investigate the relationship between entropy and the coverage, we consider the following case. By assuming that the ratio of the densities of any two units $p_i : p_j$, where $1 \leq i, j \leq n$, are constant, the densities of the units $p_1, \ldots, p_n$

[1]Coverage is the percentage of data covered by all dense units in a particular subspace. The original authors of CLIQUE define it as the number of objects covered by all dense units. Our definition is slightly different here.

become differientiable by c. It follows that

$$\begin{aligned} p_1 + \ldots + p_k &= c \\ p_{k+1} + \ldots + p_n &= 1 - c \\ \frac{dp_1}{dc} + \ldots + \frac{dp_k}{dc} &= 1 \\ \frac{dp_{k+1}}{dc} + \ldots + \frac{dp_n}{dc} &= -1 \end{aligned}$$

We want to establish the relationship that, under certain conditions, the entropy decreases as the coverage increases, i.e. $\frac{dH(X)}{dc} \leq 0$.

Theorem 1 $\frac{dH(X)}{dc} \leq 0$ if and only if $p_1^{\frac{dp_1}{dc}} \ldots p_n^{\frac{dp_n}{dc}} \geq 1$.

Proof

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n} p_i \log p_i \\ &= -\sum_{i=1}^{k} p_i \log p_i - \sum_{j=k+1}^{n} p_j \log p_j \end{aligned}$$

Let us differentiate the entropy with respect to the coverage.

$$\begin{aligned} \frac{dH(X)}{dc} &= -\sum_{i=1}^{k} \left[\frac{dp_i}{dc} \log p_i + \frac{dp_i}{dc} \right] - \sum_{j=k+1}^{n} \left[\frac{dp_j}{dc} \log p_j + \frac{dp_j}{dc} \right] \\ &= -\sum_{i=1}^{k} \left[\frac{dp_i}{dc} \log p_i \right] - 1 - \sum_{j=k+1}^{n} \left[\frac{dp_j}{dc} \log p_j \right] + 1 \\ &= -\sum_{i=1}^{k} \left[\frac{dp_i}{dc} \log p_i \right] - \sum_{j=k+1}^{n} \left[\frac{dp_j}{dc} \log p_j \right] \\ &= -\log \left[p_1^{\frac{dp_1}{dc}} \ldots p_n^{\frac{dp_n}{dc}} \right] \end{aligned}$$

The result follows and the proof is completed. □

Now we have the necessary and sufficient condition for our desirable property to hold. However, the condition is complicated and difficult to understand. Further investigation is needed to make it more comprehensive.

Theorem 2 Suppose that $\frac{dp_i}{dc} \geq 0$ for $i = 1, \ldots, k$ and $\frac{dp_j}{dc} \leq 0$ for $j = k{+}1, \ldots, n$ and $\min_{1 \leq i \leq k}(p_i) \geq \max_{k+1 \leq j \leq n}(p_j)$. Then we have

$$\frac{dH(X)}{dc} \leq 0$$

Proof

$$\begin{aligned} & p_1^{\frac{dp_1}{dc}} \ldots p_n^{\frac{dp_n}{dc}} \\ \geq & \left[\min_{1 \leq i \leq k}(p_i)\right]^{\frac{dp_1}{dc}+\ldots+\frac{dp_k}{dc}} \cdot \left[\max_{k+1 \leq j \leq k}(p_j)\right]^{\frac{dp_{k+1}}{dc}+\ldots+\frac{dp_n}{dc}} \\ = & \frac{\min_{1 \leq i \leq k}(p_i)}{\max_{k+1 \leq j \leq n}(p_j)} \\ \geq & 1 \end{aligned}$$

Then, Theorem 1 applies and the proof is completed. □

The conditions of Theorem 2 hold when the coverage is increased by increasing the densities of some denser units and decreasing the densities of some non-dense units. Although it is not true for all conditions, this is a supportive evidence of the use of entropy to reflect the coverage of clustering for a subspace.

3.3.2 Entropy and the Density Criterion

In the example shown in Figure 3.1, the entropy of case (b) is lower than that of case (a), which suggests case (b) is a better cluster. We see that entropy can better capture our intuition of good clustering as compared to the mere use of metric of coverage. To examine the relationship between entropy and density, we

consider the following case again. Assume that the density of dense units are all equal to α, the density of non-dense units are all equal to β. The total number of dense units is k and thus the total number of non-dense units is $n-k$. Then we have

$$\begin{aligned} H(X) &= -\sum_{i=1}^{k} p_i \log p_i \\ &= -\left(\sum_{i=1}^{k} p_i \log p_i + \sum_{j=k+1}^{n} p_j \log p_j \right) \\ &= -[k\alpha \log \alpha + (n-k)\beta \log \beta] \end{aligned}$$

By assuming that α and β change continuously, the entropy becomes a differentiable function of density.

Theorem 3 $\frac{dH(X)}{d\alpha} \leq 0$ if and only if $\alpha \geq \beta$.

Proof Note that

$$k\alpha + (n-k)\beta = 1$$

So

$$k + (n-k)\frac{d\beta}{d\alpha} = 0$$

Differential the entropy with respect to the density α, then we have

$$\begin{aligned} \frac{dH(X)}{d\alpha} &= -\left[k \log \alpha + k + (n-k)\frac{d\beta}{d\alpha}(\log \beta + 1) \right] \\ &= -k[\log \alpha - \log \beta] \\ &= k \log \frac{\beta}{\alpha} \end{aligned}$$

This shows that $\frac{dH(X)}{d\alpha} \leq 0$ if and only if $\alpha \geq \beta$. The proof is completed. □

Since the above value is also negative, the entropy decreases as α increases.

Hence entropy can relate to the measurement of density in the clustering of a subspace.

3.3.3 Entropy and Dimensional Correlation

The problem of correlated dimensions can be easily handled by entropy because the independence and dependence of the dimensions can be detected using the following relationships in entropy [12].

$$H(X_1, \ldots, X_n) = H(X_1) + \ldots + H(X_n) \text{ iff } X_1, \ldots X_n \text{ are independent} \quad (3.1)$$

$$H(X_1, \ldots, X_n, Y) = H(X_1, \ldots, X_n) \text{ iff } Y \text{ is a function of } X_1, \ldots, X_n \quad (3.2)$$

Traditionally, the correlation between two numerical variables can be measured using the correlation coefficient, but we can also detect correlation by entropy. Since we are already using entropy in the algorithm, using entropy to detect correlation introduces a negligible computational overhead. A set of variables $X_1, \ldots, X_n$ are correlated if Equation 3.1 is not satisfied. To express it more precisely, we define the term *interest*[2] as below.

$$interest(\{X_1, \ldots, X_n\}) = \sum_{i=1}^{n} H(X_i) - H(X_1, \ldots, X_n)$$

Equation 3.1 is not satisfied when interest is greater than 0. In this thesis, we define the degree of correlation by interest. The higher the interest, the stronger the correlation. To avoid the correlation occurred by random, we consider the variables to be correlated if and only if the interest exceeds a predefined threshold. So in this thesis, the correlation in a subspace is defined in terms of interest. The interests of one-dimensional subspaces are always 0.

[2]The definition of interest is equivalent to the mutual information between all individual dimensions of a subspace $I(X_1; X_2; \ldots; X_n)$. We use the term interest instead of 'mutual information between all individual dimensions" to simplify our terminology.

This is one of the advantage of using entropy over coverage, because we cannot discover correlation by coverage. Relationships like Equation 3.1 and 3.2 do not exist in coverage. We also propose another measure *interest gain*, which measures the increase in the correlation by adding a new dimension to a subspace. It is further discussed in Section 4.5.

Chapter 4

The ENCLUS Algorithms

In this chapter, we introduce the proposed method ENCLUS in more details. There are two variations of ENCLUS, namely ENCLUS_SIG and ENCLUS_INT, which are discussed in Section 4.4 and 4.5 respectively. The overall strategy for solving subspace clustering consists of three main steps:

1. Find out the subspaces with good clustering by an entropy-based method.
2. Identify the clusters in the subspace found.
3. Present the result to the users.

In Step 2 and Step 3, we can adopt the method in CLIQUE or some of the existing clustering algorithms. We examine Step 1 since there has been less research on it.

4.1 Framework of the Algorithms

The proposed algorithms have a framework similar to [9] which introduces an algorithm for mining correlation rules. It is based on the Apriori algorithm [1]

for mining association rule. In Apriori, we start with finding large 1-itemsets. Then, we use the results to generate the candidate 2-itemsets, which are checked against the database to determine large 2-itemsets. The process is repeated with increasing itemset sizes until no more large itemset is found.

The algorithm for mining correlation rules [9] extends the framework of Apriori by using a pair of downward and upward closure properties. In contrast, only the downward closure property is adopted in Apriori. For a downward closure property, if a subspace S satisfies the property, all the subspaces of S also do. For an upward closure property, if a subspace S satisfies the property, all the superspaces of S also do.

Downward closure property is a pruning property. If a subspace does not satisfy this property, we can cross out all its superspaces because we know they cannot satisfy the property either. Upward closure property, by contrast, is a constructive property. If a subspace satisfies the property, all its superspaces also satisfy this property. However, upward closure property is also useful for pruning. The trick is that we only find *minimal* correlated subspaces. If we know a subspace is correlated, all its superspaces must not be minimal correlated. Therefore, upward closure becomes a pruning property.

Suppose we have a downward closure property $\mathcal{D}$ and an upward closure property $\mathcal{U}$. The outline of our algorithm is as follows.

1. We start with finding all one-dimensional subspaces satisfying $\mathcal{D}$. They enter the one-dimensional candidate set.

2. Then for each subsequent pass k, we form a candidate set of k-dimensional subspaces. This set contains any subspace with all its $(k-1)$-dimensional projections satisfying $\mathcal{D}$ but not $\mathcal{U}$.

3. Each candidate is examined. Those satisfying $\mathcal{D}$ and $\mathcal{U}$ go into the result

set.

4. Go back to Step 2 unless we have an empty candidate sets.

Our method has two variations. The algorithm ENCLUS_SIG follows the above framework. In one of the variation ENCLUS_INT, only the downward closure is utilized. We do not consider the upward closure property $\mathcal{U}$, so everything about the upward closure property $\mathcal{U}$ can be removed from the above outline of the algorithm.

4.2 Closure Properties

We propose the closure properties in this section. Previously we use the term good clustering to indicate that a subspace contains a good set of clusters in an intuitive sense. Here we shall give the term a more concrete definition by means of entropy. We need to set a threshold ω. A subspace whose entropy is below ω is considered to have **good clustering**. Similarly, we define a subspace whose interest (defined in Section 3.3.3) is above another threshold ϵ to be **correlated**.

We note a downward closure property for entropy. This is given by the non-negativity of Shannon's information measures[1] [12]. The correctness of the bottom-up approach is based on this property.

Lemma 1 (Downward closure) If a k-dimensional subspace $X_1, \ldots, X_k$ has good clustering, so do all $(k-1)$-dimensional projections of this space.

Proof Since the subspace $X_1, \ldots, X_k$ has good clustering, $H(X_1, \ldots, X_k) < \omega$.

[1]The values of entropy, conditional entropy, mutual information and conditional mutual information are always non-negative. This is not true to differential entropy because the value of differential entropy may be either positive or negative.

$$
\begin{aligned}
& H(X_1, \ldots, X_{k-1}) \\
\leq\ & H(X_1, \ldots, X_{k-1}) + H(X_k|X_1, \ldots, X_{k-1}) \text{ (non-negativity)} \\
=\ & H(X_1, \ldots, X_k) \\
<\ & \omega
\end{aligned}
$$

Hence, the $(k-1)$-dimensional projection $X_1, \ldots, X_k$ also has good clustering. The above proof can be repeated for other $(k-1)$-dimensional projections. □

In Section 3.1.2 we discuss the criterion of dimensional correlation. In Section 3.3.3 we examine how entropy can be used to measure dimensional correlation. Here we show the upward closure property of this criterion.

Lemma 2 (Upward closure) If a set of dimensions S is correlated, so is every superspace of S.

Proof Suppose X_1, ..., X_k are correlated, and X_1, ..., X_k, ..., X_n is a superset of it where $n > k$. Recall we define correlation of a subspace by interest so

$$interest(X_1, \ldots, X_k) > \epsilon$$

$$
\begin{aligned}
& interest(X_1, \ldots, X_n) \\
=\ & \left(\sum_{i=1}^{n} H(X_i)\right) - H(X_1, \ldots, X_n) \\
=\ & \left(\sum_{i=1}^{n} H(X_i)\right) - H(X_{k+1}, \ldots, X_n|X_1, \ldots, X_k) - H(X_1, \ldots, X_k) \\
=\ & \left(\sum_{i=1}^{k} H(X_i)\right) - H(X_1, \ldots, X_k) + \left(\sum_{i=k+1}^{n} H(X_i)\right) - H(X_{k+1}, \ldots, X_n|X_1, \ldots, X_k) \\
\geq\ & \left(\sum_{i=1}^{k} H(X_i)\right) - H(X_1, \ldots, X_k) \\
=\ & interest(X_1, \ldots, X_k) \\
>\ & \epsilon
\end{aligned}
$$

Hence, the subspace $X_1, \ldots, X_n$ is also correlated. □

4.3 Complexity Analysis

In this section, we examine the worst case complexity of our algorithm. Note that the target problem of Apriori is NP-hard [27], so an algorithm based on the same framework is unlikely to run in polynomial time theoretically. However, we hope the algorithm would run faster than its theoretical bound in practice. The performance evaluation of the algorithm by experiment is presented in Chapter 5.

First, we present the notation used for this analysis in Table 4.1.

N	Number of transactions in the database
D	Total dimensionality of the database
m	Number of intervals each dimension divided into

Table 4.1: Notation for the complexity analysis.

In each pass, the database is scanned for the calculation of entropy. This requires $\mathrm{O}(N)$. According to Section 3.2.1, each calculation of entropy requires summing up m^k terms for the pass k. In the worst case, there can be ${}_DC_k$ candidate subspaces. Therefore, pass k requires $\mathrm{O}(N + {}_DC_k \cdot m^k)$. There can be totally D passes. So,

$$\begin{aligned}\sum_{k=1}^{D}\left(N + {}_DC_k \cdot m^k\right) &= ND + \sum_{k=1}^{D} {}_DC_k \cdot m^k \\ &= ND + (m+1)^D - 1\end{aligned}$$

Hence, the overall worst-case complexity is $\mathrm{O}(ND + m^D)$. Practically, the number of passes and candidate subspaces generated by our algorithms are often

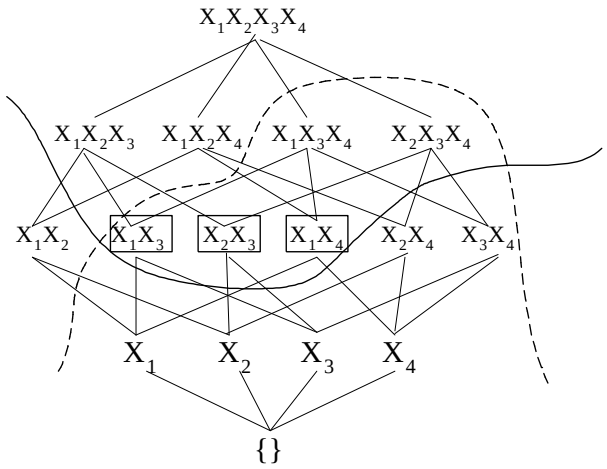


Figure 4.1: A lattice for 4 variables.

much smaller than the values used above. The complexity derived here only represents a worst-case scenario.

4.4 Mining Significant Subspaces

k	Current number of iterations
C_k	Set of k-dimensional candidate subspaces
S_k	Set of k-dimensional significant subspaces
NS_k	Set of k-dimensional subspaces with good clustering but not minimal correlated

Table 4.2: Notations used in the algorithm.

We call the subspaces with good clustering and minimal correlated to be *significant subspaces*. Due to the upward closure property, the subspaces we are interested in form a *border*. The border stores all the necessary information. Refer to Figure 4.1 for an example. In this figure, the subspaces below the dotted lines all have good clustering (downward closed). Subspaces above the solid line are all correlated (upward closed). The border $\{X_1X_3, X_2X_3, X_1X_4\}$ stores all the significant subspaces, i.e. minimal correlated subspaces with good clustering.

Algorithm 4.1 ENCLUS_SIG(ω, ϵ)

1 $k = 1$
2 Let C_k be all one-dimensional subspaces.
3 For each subspace $c \in C_k$ do
4 $f_c(\cdot)$ = cal_density(c)
5 $H(c)$ = cal_entropy($f_c(\cdot)$)
6 If $H(c) < \omega$ then
7 If $interest(c) > \epsilon$ then
8 $S_k = S_k \cup c$.
9 else
10 $NS_k = NS_k \cup c$.
11 End For
12 C_{k+1} = candidate_gen(NS_k)
13 If $C_{k+1} = \emptyset$, go to step 16.
14 $k = k + 1$
15 Go to step 3.
16 $Result = \bigcup_{\forall k} S_k$

Figure 4.2: Algorithm for mining significant subspaces.

The details of the algorithm, called ENCLUS_SIG, are given in Figure 4.2. Table 4.2 lists the notations used. The description of the procedures used in the algorithm is given as follows.

cal_density(c) Build a grid to count number of points that fall in each cell of the grid as described in section 3.2.1. The density of each cell can thus be estimated.

cal_entropy($f_c(\cdot)$) Calculate the entropy using the density information obtained from scanning the data set.

candidate_gen(NS_k) Generate the candidate subspaces for $(k+1)$ dimensions using NS_k. There is a join step and a prune step in the candidate generate function. The join step can be expressed by the following pseudo-code. It joins two subspaces having common first $(k-1)$ dimensions.

```
insert   into C_{k+1}
select   p.dim_1, p.dim_2, ..., p.dim_k, q.dim_k
from     NS_k p, NS_k q
where    p.dim_1 = q.dim_1, ..., p.dim_{k-1} = q.dim_{k-1},
         p.dim_k < q.dim_k
```

In the prune step, any subspace having k-dimensional projection outside NS_k is removed.

4.5 Mining Interesting Subspaces

Since correlation can usually be detected at low dimension, the mining of high dimensional clusters is often avoided. This is good because low dimensional clusters are easier to interpret and the time for mining high dimensional clusters can be saved. However, [9] did not consider that sometimes we are interested

Algorithm 4.2 ENCLUS_INT(ω, ϵ')

1 $k = 1$
2 Let C_k be all one-dimensional subspaces.
3 For each subspace $c \in C_k$ do
4 $\quad f_c(\cdot) =$ cal_density(c)
5 $\quad H(c) =$ cal_entropy($f_c(\cdot)$)
6 $\quad$ If $H(c) < \omega$ then
7 $\quad\quad$ If $interest\_gain(c) > \epsilon'$ then
8 $\quad\quad\quad I_k = I_k \cup c.$
9 $\quad\quad$ else
10 $\quad\quad\quad NI_k = NI_k \cup c.$
11 End For
12 $C_{k+1} =$ candidate_gen($I_k \cup NI_k$)
13 If $C_{k+1} = \emptyset$, go to step 16.
14 $k = k + 1$
15 Go to step 3.
16 $Result = \bigcup_{\forall k} I_k$

Figure 4.3: Algorithm for mining interesting subspaces.

in non-minimal correlated subspaces. For instance, A and B are correlated, but we may be interested in the subspace ABC if ABC are more strongly correlated than A and B alone. To measure the increase in correlation, we define the term *interest gain*[2]. The interest gain for subspace $X_1, \ldots, X_n$ is defined as follows.

$$\begin{aligned} interest\_gain(\{X_1, \ldots, X_n\}) &= interest(\{X_1, \ldots, X_n\}) - \\ & \max_i \{interest(\{X_1, \ldots, X_n\} - \{X_i\})\} \end{aligned}$$

The interest gain for one dimensional subspace is defined to be 0. The interest gain of a k-dimensional subspace is the interest of the given subspace minus the maximum interest of its $(k-1)$-dimensional projections. In other words, it is the increase in interest for adding an extra dimension.

Our new goal becomes mining subspaces whose entropy exceeds ω and interest gain exceeds a new threshold ϵ'. We call such subspaces to be *interesting subspaces*. The mining of significant subspace algorithm can be modified slightly to mine interesting subspaces. Figure 4.3 shows the modified algorithm ENCLUS_INT. Since we relax one of the pruning criteria, more candidates and a longer running time are expected. The worst-case complexity of ENCLUS_INT is the same as ENCLUS_SIG, because the computation of interest gain is negligible and we consider the extra-pruning capacity of ENCLUS_SIG in the worst case.

4.6 Example

Here we give an example to illustrate how our algorithm works. In this example, some clusters and noise are generated at predefined positions of the space. The

[2]The definition of interest gain is equivalent to the mutual information between the original subspace $X_{i_1}, \ldots, X_{i_{n-1}}$ and a new dimension X_{i_n}, i.e. $I(X_{i_1}, \ldots, X_{i_{n-1}}; X_{i_n})$. We use the term interest gain instead of "mutual information between the original subspace and the new dimension" to simplify our terminology.

generated data have 4-dimensions X_1, X_2, X_3 and X_4. The points are uniformly distributed inside each cluster. There are four clusters of 500000 points each plus noise data of 300000 points. The values of the dimension X_4 are always uniformly distributed along [0.0,1.0), so it is independent of all other dimensions and can be regarded as a noise attribute. The positions of the clusters are shown in Table 4.3. There are clusters in the subspaces X_1X_2 and $X_1X_2X_3$. The parameters ω, ϵ, ϵ' and Δ are set at 12, 0.01, 0.01 and 0.01 respectively. Since we know where the clusters are located, the result of the algorithms can be compared with the setting.

We mine the significant subspaces first. In the initial iteration, all one-dimensional subspaces are added to the candidate set C_1. By the definition of interest, all one-dimensional subspaces go to NS_1.

$$\begin{aligned} C_1 &= \{X_1, X_2, X_3, X_4\} \\ S_1 &= \emptyset \\ NS_1 &= \{X_1, X_2, X_3, X_4\} \end{aligned}$$

In the second iteration, the candidate set C_2 is generated by the candidate generation procedure, which self-joins NS_1. From Table 4.4, we see that all subspaces in C_2 satisfy the entropy threshold ω but only $\{X_1X_2, X_1X_3, X_2X_3\}$ satisfies the interest threshold ϵ. These subspaces go to S_2 while the rest goes to NS_2.

$$\begin{aligned} C_2 &= \{X_1X_2, X_1X_3, X_1X_4, X_2X_3, X_2X_4, X_3X_4\} \\ S_2 &= \{X_1X_2, X_1X_3, X_2X_3\} \\ NS_2 &= \{X_1X_4, X_2X_4, X_3X_4\} \end{aligned}$$

The candidate generation function gives an empty candidate set C_3, because no two subspaces in NS_2 have common first dimension. The algorithm thus

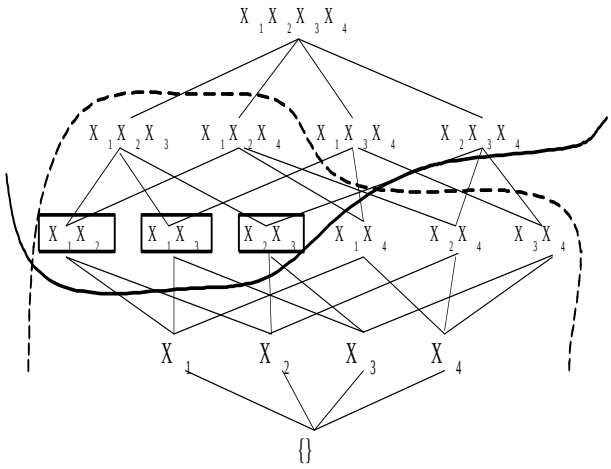


Figure 4.4: The example illustrated in a lattice.

terminates here. The result is all the significant subspaces discovered so far.

$$\begin{aligned} C_3 &= \emptyset \\ result &= \{X_1X_2, X_1X_3, X_2X_3\} \end{aligned}$$

The result set correctly tells us there are clusters at the subspaces X_1X_2, X_1X_3 and X_2X_3. The dimension X_4 is a noise attribute. It does not form clusters with any dimension. This example is illustrated by a lattice in Figure 4.4. All subspaces below the dotted lines have good clustering (downward closure) and subspaces above the solid line are all correlated (upward closure). The subspaces marked by the boxes form the border. They are the significant subspaces. Other subspaces that lie between the two borders have good clustering and are correlated too, but they are not included in the result because they are not minimal correlated.

However, in the setting of this example, clusters are contained in the subspace $X_1X_2X_3$. This is not found by ENCLUS_SIG because this subspace is not minimal correlated. We need to use the ENCLUS_INT algorithm if we wish to

Cluster no.	X_1	X_2	X_3	X_4	Number of points
1	[0.2,0.3)	[0.7,0.8)	[0.9,1.0)	[0.0,1.0)	500000
2	[0.2,0.3)	[0.2,0.4)	[0.0,1.0)	[0.0,1.0)	500000
3	[0.8,0.9)	[0.5,0.6)	[0.0,0.5)	[0.0,1.0)	500000
4	[0.5,0.6)	[0.7,0.9)	[0.0,0.2)	[0.0,1.0)	500000
noise	[0.0,1.0)	[0.0,1.0)	[0.0,1.0)	[0.0,1.0)	300000

Table 4.3: Setting of synthetic data.

find this cluster. We set the parameter ϵ' to be 0.2. The first two iterations are identical to mining sequential subspaces.

$$\begin{aligned} C_1 &= \{X_1, X_2, X_3, X_4\} \\ I_1 &= \emptyset \\ NI_1 &= \{X_1, X_2, X_3, X_4\} \\ C_2 &= \{X_1X_2, X_1X_3, X_1X_4, X_2X_3, X_2X_4, X_3X_4\} \\ I_2 &= \{X_1X_2, X_1X_3, X_2X_3\} \\ NI_2 &= \{X_1X_4, X_2X_4, X_3X_4\} \end{aligned}$$

Among the three dimensional candidate sets, only $X_1X_2X_3$ qualifies as interesting subspace. $X_1X_2X_4$ does not qualify because interest gain does not exceed the threshold ϵ'. $X_1X_3X_4$ and $X_2X_3X_4$ are pruned away because both the entropy and interest gain do not exceed the thresholds. The candidate set C_4 is empty so the algorithm terminates here. The subspaces found in the result set are consistent with our initial setting. Notice that the result still lies within the two borders of the lattice. It means that the interesting spaces also have good clustering and are correlated.

$$\begin{aligned} C_3 &= \{X_1X_2X_3, X_1X_2X_4, X_1X_3X_4, X_2X_3X_4\} \\ I_3 &= \{X_1X_2X_3\} \end{aligned}$$

Subspace	Entropy/nats	Interest/nats	Interest gain/nats
X_1	4.1740	0	0
X_2	4.2712	0	0
X_3	4.4185	0	0
X_4	4.6051	0	0
X_1X_2	7.5466	0.8986	0.8986
X_1X_3	7.9121	0.6804	0.6804
X_1X_4	8.7773	0.0018	0.0018
X_2X_3	8.8062	0.6035	0.6035
X_2X_4	8.8746	0.0017	0.0017
X_3X_4	9.0219	0.0017	0.0017
$X_1X_2X_3$	10.8887	1.9750	1.0781
$X_1X_2X_4$	11.9996	1.0507	0.1521
$X_1X_3X_4$	12.3617	0.8359	0.1555
$X_2X_3X_4$	12.5334	0.7614	0.1579

Table 4.4: The values of entropy, interest and interest gain of the subspaces in the example.

$$\begin{aligned} NI_3 &= \{X_1X_2X_4\} \\ C_4 &= \emptyset \\ result &= \{X_1X_2, X_1X_3, X_2X_3, X_1X_2X_3\} \end{aligned}$$

From this example, we can see that ENCLUS_INT discovers more subspaces, but it generates more candidates so the running time is longer. Which algorithm to use depends on whether we are interested in non-minimal correlated subspaces.

Chapter 5

Experiments

To evaluate the performance and accuracy of the algorithms, we implemented our algorithms on Sun Ultra 5/270 workstation using GNU C++ compiler. Both synthetic data and real-life data are used for experiments. Our goal is to analyse the performance and accuracy of our algorithms under different settings. We also compare our algorithms against CLIQUE.

5.1 Synthetic Data

In this set of experiments, high dimensional synthetic data are generated which contains clusters embedded in the subspaces. We generate two kinds of synthetic data, namely *hyper-rectangular data* and *linearly dependent data*.

5.1.1 Data Generation — Hyper-rectangular Data

Our data generator allows the user to specify the dimensionality of data, the number of subspaces containing clusters, the dimensionality of clusters, the number of clusters in each subspace and the number of transactions supporting each cluster. Table 4.3 is an example of the input to our data generator. This method

of data generation is also used in [2, 36] and the design of our data generator resembles that in these works. We do not use hyper-rectangular data in most of the tests below, because some problems may arise. These problems are discussed in Section 5.4.

For each subspace containing clusters, we insert more than one hyper-rectangular clusters. If there is only one uniform hyper-rectangular cluster inserted in the subspace, it would be pruned away due to independence (see Section 3.1.2).

5.1.2 Data Generation — Linearly Dependent Data

Our experiments focus on linearly dependent data. This kind of data contains linearly dependent variables. Linearly dependent variables are a set of variables, in which at least one of them is the linear combination of the other variables. The following is an example.

Example 1 Let A, B and C be a set of linearly dependent variables. Variables A and B are random variables uniformly distributed along [0,1], and $C = 0.4A + 0.6B$.

Ideally, a subspace clustering algorithm should report the subspace ABC to the user. In our experiments below, we see that not all subspace clustering algorithms do this successfully. Unless otherwise specified, we use data of 10 dimensions and 300,000 transactions in the experiments. Sets of linearly dependent variables contain in five-dimensional subspaces. The default parameters are shown at Table 5.1.2. This set of parameters are obtained from trial and error. They are suitable for the discovery of five-dimensional clusters.

Parameter	Value
Δ	0.1
ω	8.5
ϵ	0.1
ϵ'	0.1

Table 5.1: Default parameters for the experiments.

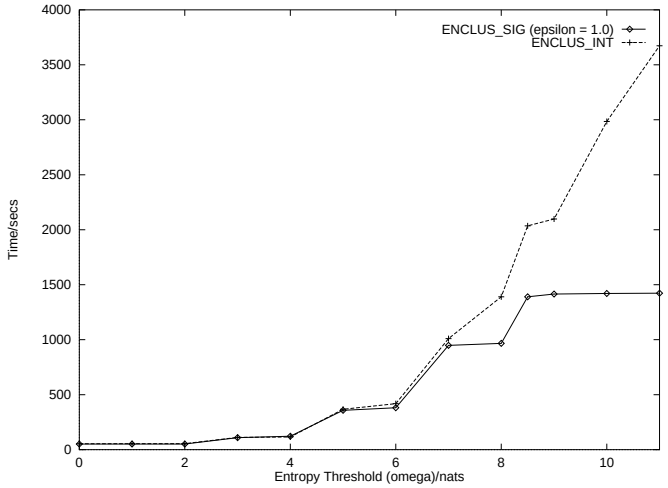


Figure 5.1: Entropy threshold vs running time.

5.1.3 Effect of Changing the Thresholds

Figure 5.1 shows the performance of the algorithms under different values of ω. We do not have a smooth curve here, because when ω increases to a certain value, candidates of a higher dimension are introduced which impose a considerable amount of extra computation. Five dimensional subspaces are discovered when $\omega \geq 8.5$. From the figure, we can see the running time of the algorithm ENCLUS_SIG ceases to increase when ω is high enough, because after that point, the pruning power of entropy is negligible and most pruning is attributed to the upward closure property which is independent of ω. As for the algorithm ENCLUS_INT, the running time keeps on increasing with ω because only the entropy is utilized for pruning. At $\omega = 8.5$, ENCLUS_INT recovers the five-

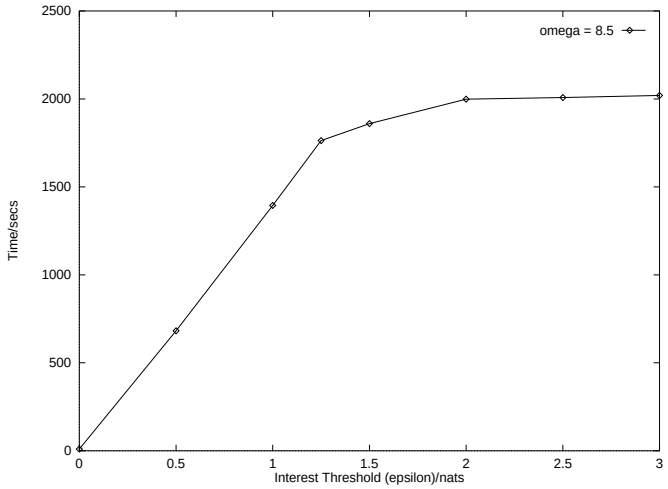


Figure 5.2: Interest threshold vs running time (ENCLUS_SIG).

dimension clusters we have embedded. ENCLUS_SIG represents the correlated variables in a number of two and three dimensional subspaces.

Figure 5.2 shows the performance of the ENCLUS_SIG under different values of ϵ. Again, the running time ceases to increase after a certain point. This is because the pruning power of the upward closure property is negligible and most pruning is done by entropy. As ϵ increases, the clusters are expressed in higher dimensional subspaces, but the performance suffers because less pruning is done. Only five-dimensional subspaces are discovered when ϵ reaches 3. Obviously, when ϵ is set too large, no subspace can be discovered because there does not exist a set of variables having such high correlation. Therefore, the use of high interest threshold ϵ is not recommended.

We have also performed a similar set of experiments for ENCLUS_INT with $\omega = 8.5$. The performance of the ENCLUS_INT under different values of ϵ' is nearly identical, since only the entropy is used for pruning.

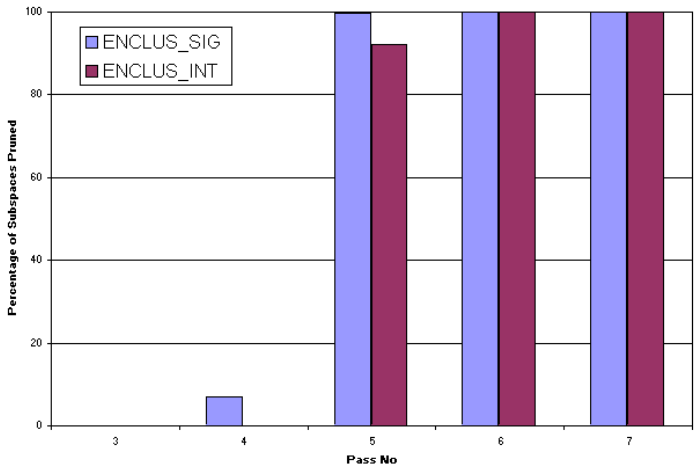


Figure 5.3: Pass no vs percentage of subspaces pruned.

5.1.4 Effectiveness of the Pruning Strategies

The pruning power of the algorithm is illustrated in Figure 5.3. Our methods are compared to the naive algorithm which examines all possible subspaces. From the result, we can see our methods achieve significant reduction on the number of candidates in the later passes. ENCLUS_SIG always prunes more candidates than ENCLUS_INT. This explains why ENCLUS_SIG runs faster than ENCLUS_INT. The experiment is carried out with a 20-dimensional data set.

5.1.5 Scalability Test

The result of the scalability test is shown in Figure 5.4. As expected, ENCLUS_SIG outperforms ENCLUS_INT because ENCLUS_SIG only finds minimal correlated subspaces while ENCLUS_INT has to spend extra time to mine the non-minimal correlated subspaces. The gap between ENCLUS_SIG and ENCLUS_INT increases with the dimensionality, which suggests the pruning power of the upward closure is more significant there. We run the experiment up to 30 dimensions. For higher dimensions, the computation time would increase further. We suggest some extra pruning strategies to improve the performance in

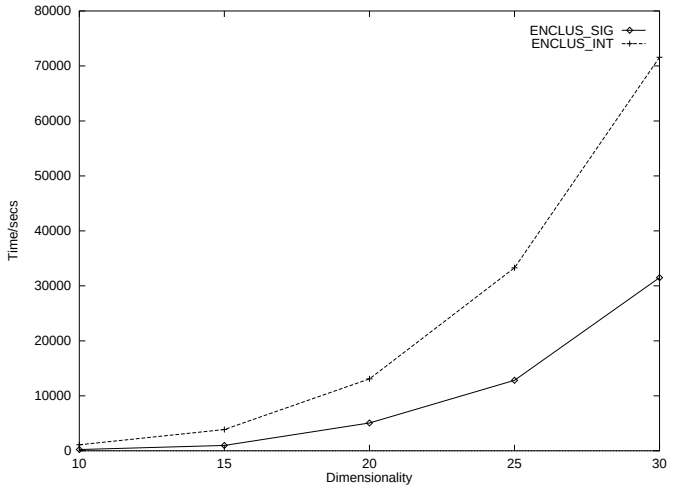


Figure 5.4: Scalability test on dimensionality of data set.

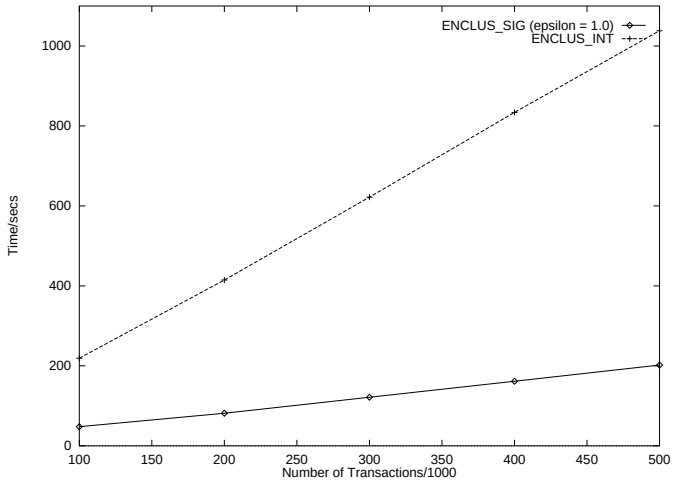


Figure 5.5: Scalability test on the number of transactions of the data set.

Section 6.1.

Figure 5.5 shows the scalability of our algorithms under databases of 100,000 to 500,000 transactions. From the experiment, our algorithm scales linearly with the number of transactions, because the sole effect of changing the number of transactions is on the time reading the database. The number of passes remains constant and the time reading the database increases linearly with the number of transactions.

This result is consistent with the complexity analysis done in Section 4.3,

which points out that the worst case complexity is $O(ND + m^D)$. In both the complexity and experimental analysis, our algorithms scale linearly with N and exponentially with D.

5.1.6 Accuracy

To investigate the accuracy of the algorithms, we performed an experiment using a data set containing some 5-dimensional clusters in five disjoint subspaces. The total dimensionality of the data set is 25. ENCLUS_INT successfully discovers the five 5-dimensional subspaces that contains our embedded clusters without reporting false alarms of other 5-dimensional subspaces. ENCLUS_SIG, again, expresses the correlated variables using a number of two-dimensional subspaces. It does not examine higher dimensional subspaces because they are not minimal correlated.

5.2 Real-life Data

In this section, we apply our algorithms to the real-life data sets in order to verify the validity of the results of the algorithms. Two sets of data are used, namely the US 1990 census data and Hong Kong stock price data during 1993-1996.

5.2.1 Census Data

The US census database is available at the web site IPUMS-98[1]. We choose to work on the person record on the 1990 data. It consists of many kinds of attributes. Since most of them are categorical, we choose only 24 numerical attributes out of the whole data set for our analysis. The algorithm ENCLUS_INT

[1]The URL of IPUMS-98 is http://www.ipums.umn.edu/.

Subspace
NCHILD ELDCH YNGCH
OCCSCORE SEI UHRSWORK
OCCSCORE SEI WKSWORK1
EDUCREC OCCSCORE SEI
ELDCH YNGCH AGE

Table 5.2: Subspaces of highest interest at three dimensions (Census database).

Subspace
NCHLT5 YRSIMMIG INCTOT
CHBORN YRSIMMIG INCTOT
FAMSIZE YRSIMMIG WRKSWORK1
NCHLT5 INCTOT INCBUS
FAMSIZE NCHLT5 SPEAKENG

Table 5.3: Subspaces of lowest interest at three dimensions (Census database).

rather than ENCLUS_SIG is used because it does not stop at low dimension and allows us to see more interesting subspaces. We show some three-dimensional subspaces with highest and lowest interests on Table 5.2 and 5.3 respectively. The mnemonic used is given on Table 5.4. The subspaces of low interest are not usually the target of our algorithms. We only use them to contrast with the subspaces of high interest. From the result, we can see our algorithms discover meaningful subspaces. The subspaces of high interest are much more likely to have good clustering than the subspaces of low interest.

5.2.2 Stock Data

In this section, we use our algorithm to study the stock price of Hong Kong stock market during 1993-1996. Our study is limited to 29 stocks out of 33 Hang Seng Index constituent stocks. The remaining 4 stocks are omitted because of

Mnemonic	Variable Name
FAMSIZE	Number of own family members in household
NCHILD	Number of own children in household
NCHLT5	Number of own children under age 5 in household
ELDCH	Age of eldest own child in household
YNGCH	Age of youngest own child in household
AGE	Age
CHBORN	Number of Children ever born
YRSIMMIG	Year of immigration
SPEAKENG	Speaks English
EDUCREC	Education attainment recode
OCCSCORE	Occupational income score
SEI	Duncan Socioeconomic Index
WRKSWORK1	Weeks worked last year
UHRSWORK	Usual hours worked per week
INCTOT	Total personal income
INCBUS	Non-farm business income

Table 5.4: Mnemonic used in the census data sets.

missing data. A high interest threshold ϵ (1.0) is used because we expect price movements of the stocks are highly correlated.

The subspaces of highest and lowest interests are shown at Table 5.5 and 5.6 respectively. Again, our algorithm produces meaningful results. For instance, the subspace Cheung Kong, Henderson Land and SHK PPT is likely to have good clustering because they are all real estate stocks. The price movement of each of these stocks is likely to affect the others. Similarly, the subspace Cheung Kong, HSBC and Henderson Land contains the leading stocks. In contrast, the subspaces with low interest contain stocks from different industries.

From the experiments on these two sets of real-life data, we can conclude that the results produced by our algorithms are valid and meaningful.

Stock name
Cheung Kong, Henderson Land, SHK PPT
Cheung Kong, HSBC, Henderson Land
Cheung Kong, Hutchison, SHK PPT
HSBC, Hutchison, First Pacific

Table 5.5: Subspaces of highest interest at three dimensions (Stock database).

Stock name
HK Electric, Sino Land, TVB
HK Electric, Bank of East Asia, TVB
HK Telecom, Sino Land, TVB
HK Telecom, Bank of East Asia, Cathay Pacific

Table 5.6: Subspaces of lowest interest at three dimensions (Stock database).

5.3 Comparison with CLIQUE

In this section, our algorithms are compared to the CLIQUE algorithm. The CLIQUE algorithm is implemented on the same platform as ENCLUS. Recall that there are three steps in CLIQUE. For fair comparison, we only use CLIQUE to discover the subspaces with good clustering (Step 1). We do not discover the actual position of the clusters (Step 2 and 3). It is explained in Section 5.1.3 that the performance of our algorithm depends on the values of the thresholds. The result of a similar evaluation on CLIQUE is shown on Figure 5.6, from which we see that the performance of CLIQUE also varies greatly with its threshold τ. Hence, special attention on setting the thresholds has been paid to the comparison experiment. In this experiment, the threshold values are set in a way such that the target subspaces are discovered by the algorithms. Also, since the running time is implementation dependent, it is advised we look at the growth rate of the running time rather than at the absolute value.

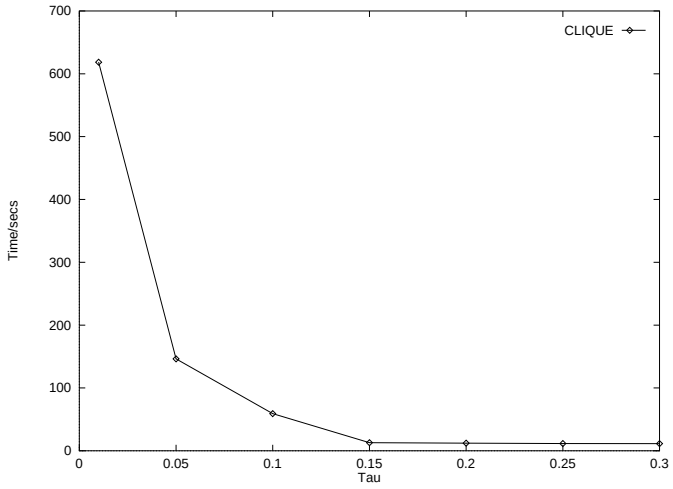


Figure 5.6: Performance of CLIQUE under different thresholds.

Algorithm	Thresholds
ENCLUS_SIG	$\omega = 4.0$, $\epsilon = 0.1$
ENCLUS_INT	$\omega = 4.0$, $\epsilon' = 0.1$
CLIQUE	$\tau = 0.1$, $\xi = 10$

Table 5.7: Parameters used in the comparison experiment.

The threshold values used are listed at Table 5.7. The data set of this experiment is 100,000 transactions of synthetic data at different dimensionality. Because of the reasons described in Section 5.3.1 and Section 5.4, only simple hyper-rectangular data are generated for this experiment. Only one subspace is chosen and three dimensional clusters are embedded in this subspace. All algorithms discover the subspace successfully. Figure 5.7 shows the performance of them. They all scales non-linearly with the dimensionality. The running time of the 50-dimensional case is 25.0, 18.4 and 31.40 times of the running time of the 10-dimensional case for ENCLUS_SIG, ENCLUS_INT and CLIQUE respectively. This suggests our algorithms have better scalability than CLIQUE.

We suspect that the ENCLUS algorithms run faster than CLIQUE because the candidate sets in ENCLUS are based on subspaces rather than on units in CLIQUE. Since the potential number of dense units is much larger than the

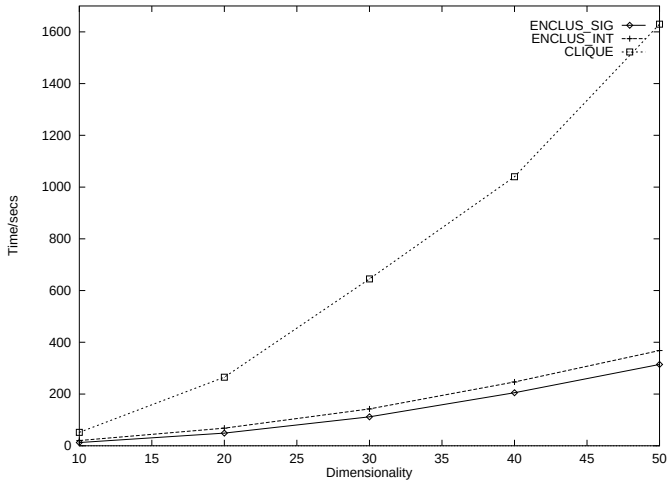


Figure 5.7: Comparison of our algorithms with CLIQUE.

potential number of subspaces, ENCLUS can have much fewer candidates at high dimensions. Also, ENCLUS_SIG uses two closure properties for pruning, which is more effective than one closure property in CLIQUE. A comparsion of the pruning power, rather than just the running time, of ENCLUS and CLIQUE can be a topic of further research. However, special attention must be paid to ensure the fairness of the comparsion. As the number of dense units is much larger than the potential number of subspaces, the base for the calculation of the pruning ratio in CLIQUE is larger. The result can be misleading because CLIQUE may have more candidates despite better pruning ratio. Also, MDL-based pruning in CLIQUE sacrifices the accuracy for better pruning. The ENCLUS algorithms do not sacrifice the accuracy. Therefore, their pruning ratios are not directly comparable.

5.3.1 Subspaces with Uniform Projections

In [2], CLIQUE is studied through hyper-rectangular synthetic data. From our experiments, we observe that CLIQUE is not very capable of dealing with some data sets in which some subspaces have good clustering but their projections

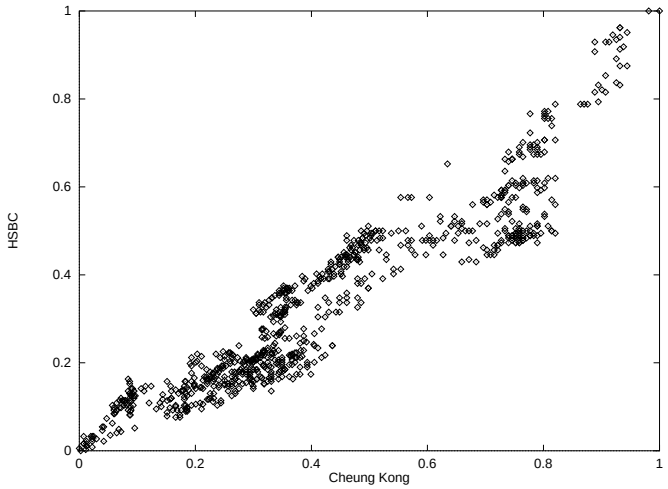


Figure 5.8: The stock price of Cheung Kong and HSBC (normalized to [0,1]).

look uniform. These subspaces would be missed by CLIQUE. One kind of such data sets is linearly dependent data.

In Example 1 on Section 5.1.2, the variables A and B as well as the subspace AB are uniform. They are likely to be pruned away by CLIQUE since they have low coverages due to their uniform distribution. Unfortunately, this would inhibit the discovery of the subspace ABC.

We test CLIQUE on linearly dependent data. We generate a data set containing two sets of linearly dependent variables. CLIQUE is unable to discover them even though different values of the threshold τ are tried. On the other hand, ENCLUS_SIG and ENCLUS_INT can handle them successfully, because a subspace containing linearly dependent variables would give high interest. Although the uniform distribution in the lower subspaces give high entropy, these subspaces would not be pruned away by our algorithms, because the downward closure property keeps all the potential subspaces until their entropy exceeds the threshold ω. When the entropy of a subspace exceeds the threshold ω, their superspaces are impossible to have good clustering.

We also try CLIQUE on the stock data set, because this data set most closely

Cluster no.	X_1	X_2	X_3	X_4	Number of points
1	[0.2,0.3)	[0.2,0.3)	[0.0,1.0)	[0.0,1.0)	500000
2	[0.0,1.0)	[0.0,1.0)	[0.8,0.9)	[0.8,0.9)	500000

Table 5.8: Example illustrating the problem of hyper-rectangular data.

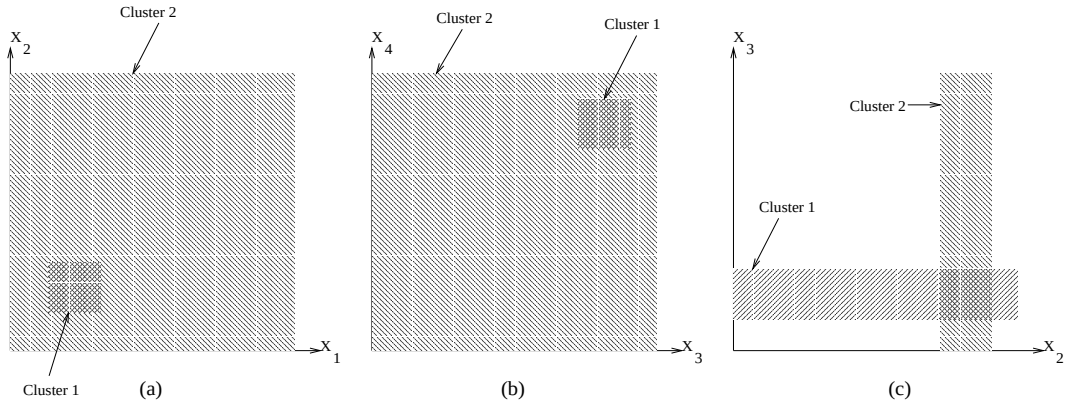


Figure 5.9: Example illustrating the problem of hyper-rectangular data.

resembles the linearly dependent data. For example, Figure 5.8 shows the stock price of Cheung Kong against that of HSBC. It shows a close relationship between their stock prices. Their one-dimensional projection, though not strictly uniform, does not show particular clusters. We run CLIQUE with the stock data set. The three-dimensional subspace with the highest coverage is HSBC, HK Electric and Hang Seng. Their interest (1.69) only ranks 1763th out of the 3654 candidate three-dimensional subspaces. The highest interest is 2.54. Many high-interest subspaces according to our algorithm are missed by CLIQUE.

5.4 Problems with Hyper-rectangular Data

Although the method of generating hyper-rectangular data is used in previous work such as [2, 36], we do not use it for most of our experiments. This manipulation of such data is awkward because one cluster can cross many subspaces.

Let us look at the setting of Table 5.8 for an example. In this example, we intend to embed clusters at the subspaces X_1X_2 and X_3X_4 only. However, clusters may also arise at other subspaces such as X_1X_3, X_1X_4, X_2X_3 and X_2X_4. Figure 5.9 gives an graphical illustration. Cluster 1 is a cluster at subspace X_1X_2, but it is uniform in subspace X_3X_4. Similar argument holds for Cluster 2. The entropy of subspaces X_1X_2 and X_3X_4 would not be particular low because one of the clusters looks uniform. On the other hand, in other subspaces, say X_2X_3, each cluster looks like a bar. As a result, it seems to contain clusters although that is not what we intended. Because of this phenomenon, a subspace clustering algorithm often discovers more subspaces than we expected.

We tested ENCLUS and CLIQUE in a set of hyper-rectangular data. A set of 10-dimensional data with 300,000 transactions are generated. We embed some clusters at three 5-dimensional subspaces. Both ENCLUS and CLIQUE do not give satisfactory accuracy in hyper-rectangular data. ENCLUS_INT recovers the 3 target subspaces at five dimensions but also introduces 9 other subspaces. The result of CLIQUE deviates too much from our expected result for a detailed analysis. It cannot recover our target subspaces but introduces a lot of other subspaces. It also stops before any 5-dimensional subspace is discovered.

This result is confusing because CLIQUE is tested by hyper-rectangular data in [2] and is reported to have good accuracy. However, we cannot replicate such result unless some very simple hyper-rectangular data sets (having low number of subspaces containing clusters), like those used in the experiment of previous section, are used.

Chapter 6

Miscellaneous Enhancements

In the previous chapter, we discuss the details of the algorithm ENCLUS. In this chapter, we go further to look at some miscellaneous enhancements applicable to ENCLUS. These include some extra pruning strategies and adjustment to the clustering criteria. Our target is to provide better performance or more meaningful and comprehensible results.

6.1 Extra Pruning

Here we attempt to propose a new pruning method for improving the performance. In the experiments, we notice that many useless subspaces are not pruned away until the late passes. This suggests there are rooms for further reduction of the number of candidate subspaces. One way of doing this is by making the following assumption.

Assumption 1 If a subspace $\mathcal{S}$ is significant or interesting, any projection of $\mathcal{S}$ with two or more dimensions has at least interest $\bar{\epsilon}$, which is less than ϵ.

In the above assumption, we set a new threshold $\bar{\epsilon}$. The rationale for the above assumption is based on the observation that in many data sets, when a set

of variables is correlated, its subset also shows some degree of correlation. We have verified this with the two real-life data sets used in our experiments. When a subspace is correlated, its projections are never independent. This assumption is, however, not always true. For instance, it does not hold on linearly dependent data.

With this assumption, any subspace with interest less than $\bar{\epsilon}$ can be pruned away. We make simple modification to our algorithm to incorporate this pruning techniques. The line 6 of Figure 4.2 (ENCLUS_SIG) or Figure 4.3 (ENCLUS_INT) should be replaced as follows.

$$\text{If } H(c) < \omega \text{ and } (k = 1 \text{ or } interest(c) \geq \bar{\epsilon}) \text{ then}$$

We examine the performance of our algorithm with this extra pruning technique using hyper-rectangular synthetic data. Owing to the problems described in Section 5.4, the clusters are only embedded in two five-dimensional subspaces. We generate 300,000 transactions at different dimensionality. The parameters $\bar{\epsilon}$ is set at 0.05. The result of this experiment is shown at Figure 6.1. We can observe that the algorithms with extra pruning outperforms those without by a large margin.

For this data set, the target subspaces are successfully recovered by the algorithms. This accuracy cannot be achieved when Assumption 1 does not hold on the data set. Therefore, the extra pruning cannot be applied on linearly dependent data, where the assumption does not hold.

6.2 Multi-resolution Approach

Another possible improvement on ENCLUS is the consideration of the number of clusters. Recall that the K-means method has a problem in determining the

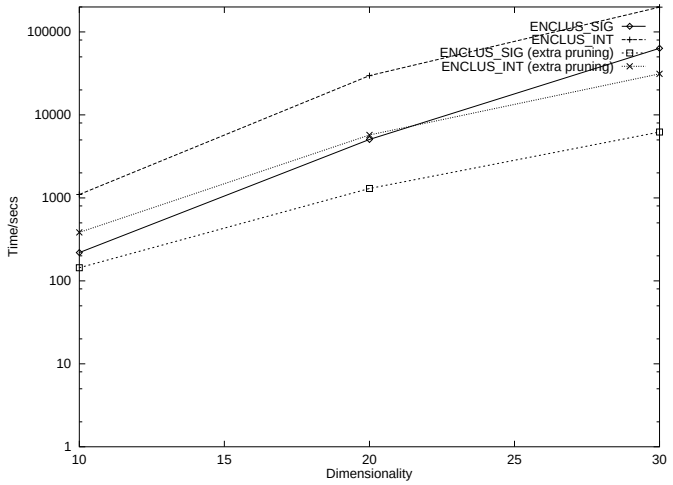


Figure 6.1: The performance of the algorithms with and without extra pruning.

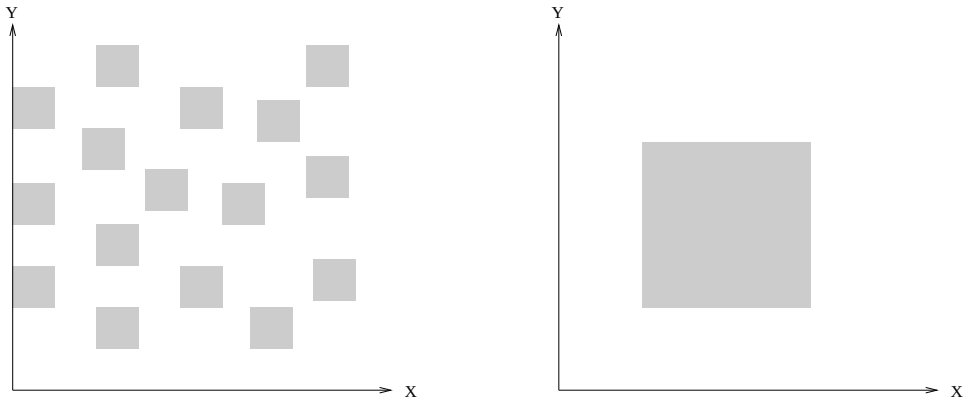
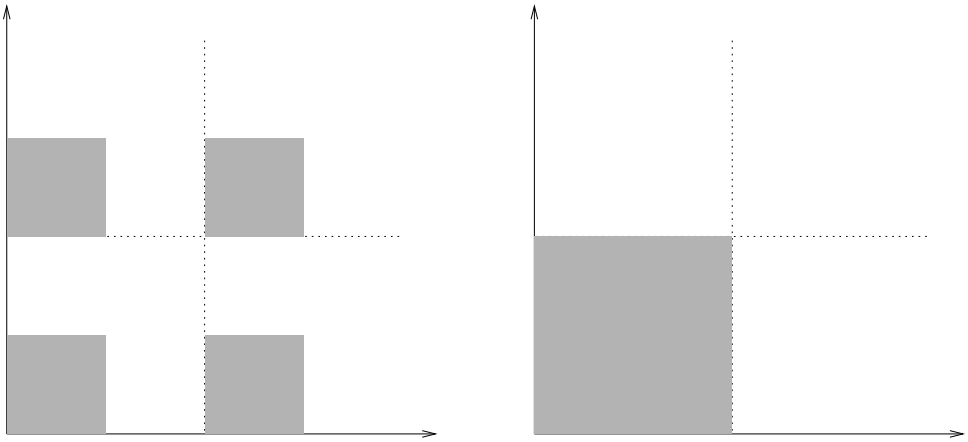


Figure 6.2: Two data sets with equal coverage and density but different number of clusters.

number of clusters: generally a large number of clusters will lower the average distance of points from their cluster centroids, giving the dilemma that forming one cluster for each point will give an optimal distance measurement. We can see that the number of clusters is a valid consideration in the determination of goodness of clustering.

For two data sets, it is possible that the coverage and density of the two sets are the same, but one set contains a large number of clusters while the other set contains a small number of clusters. Figure 6.2 shows such an example. It is intuitive that the set with a smaller number of clusters should be considered the



The clusters fall on different grids. The entropy is higher.

Entropy = 1.38 nats

The clusters fall on the same grid. The entropy is lower.

Entropy = 0

Figure 6.3: An example explaining how multi-resolution method works.

set with better clustering. Unfortunately, our algorithm does not take this into account.

We can handle this criterion by using a multi-resolution approach in calculating the entropy. We repeat calculating the entropy again using different values of the size of interval Δ. In a coarse resolution, the entropy value favours small number of clusters. The data points fall on the same cells in a subspace with small number of clusters but they fall on different cells in a subspace with large number of clusters. Therefore, the entropy value is lower for smaller number of clusters. See Figure 6.3 for an illustration.

We cannot just use a coarse resolution, however, because it cannot capture the difference in the densities of different regions. Therefore, we use a multi-resolution approach. We extend our algorithm by having various entropy thresholds for the entropy in different resolutions. In other words, we have multiple downward closure properties.

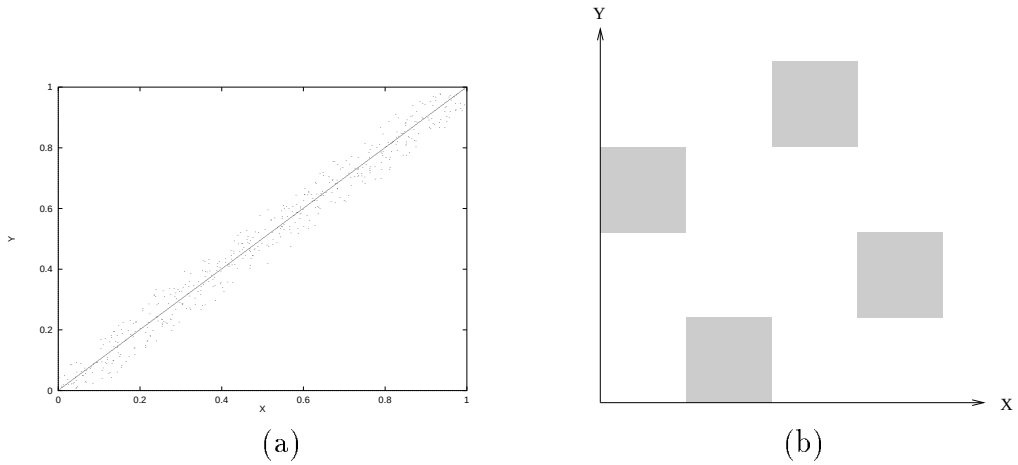


Figure 6.4: Two correlated variables X and Y.

6.3 Multi-threshold Approach

In the multi-threshold approach, we use different thresholds at difference passes. The multi-threshold approach is useful in both CLIQUE and ENCLUS. Let us first look at the case in CLIQUE.

Figure 6.4(a) shows a problem case in CLIQUE. The variables X and Y are correlated. The distribution of X and Y is uniform when considered individually, but their joint distribution is not. Depending on the threshold τ, all or no units will be considered dense. For the latter case, the cluster in the two-dimensional space will be missed by CLIQUE. To avoid this situation, the threshold τ must be set low enough. A similar problem occurs when the values of the variables X and Y distribute like the setting in the N-queen problem (see Figure 6.4(b)). Again τ must be set low enough to avoid missing the two-dimensional clusters. However, setting τ too low would create other problems. Firstly, this would generate a tremendous amount of dense units, which introduces a large memory overhead and hampers the performance greatly. For a k-dimensional subspace, the memory requirement for storing the count of dense units is $O(N^k)$. Secondly, this would produce 100% coverages for uniformly distributed subspaces. This is undesirable because we intend to have lower coverage for uniformly distributed

subspaces.

This problem can be solved if multiple thresholds are allowed. Although multiple thresholds are not explicitly set in CLIQUE. the minimal description length (MDL) method in CLIQUE effectively assigns different thresholds for different dimensionalities.

The multi-threshold approach is useful in ENCLUS too. To maintain the downward closure property, the same entropy threshold is used in all pass. Nevertheless, low dimension subspace tends to have low entropy. As a result, a low-dimensional subspace with entropy below the threshold may not be useful to the users. This can be avoided if we give a lower threshold at lower passes. However, this would violate the downward closure property which is essential to ENCLUS. We, therefore, propose to use a post-processing method. Multiple entropy thresholds are set, but we run ENCLUS using the highest entropy threshold. Before we present the result to the users, it is checked against the threshold for each level. Those subspaces that do not satisfy the threshold for their corresponding level are removed.

However, it is clumsy to set multiple thresholds by human. We may also use the MDL-based pruning method in CLIQUE which is based on entropy instead of coverage. The effect of MDL-based pruning on ENCLUS is not studied yet and is left as future work.

Chapter 7

Conclusion

We propose to tackle the problem of mining numerical data using clustering techniques since each transaction with k attributes can be seen as a data point in a k-dimensional space. However, for large databases, there are typically a large number of attributes and the patterns that occur in subsets of these attributes are important. Mining for clusters in subspaces becomes an important problem. The proposed solution consists of three steps, namely the identification of subspaces containing clusters, the discovery of clusters in selected subspaces and presentation to the users. We concentrate on the subproblem of identifying subspaces containing clusters, because few works have been done on it, one better known previous method is CLIQUE [2].

We propose using three criteria for the goodness of clustering in subspaces: coverage, density and correlation. Our proposed method is based on the measure of entropy from information theory, which typically gives a lower value for a subspace with good clustering. Although entropy has been used in decision trees for data mining [31, 32], to our knowledge, no previous work has used it for the problem of subspace clustering. We also justify the approach by establishing some relationship between entropy and the three criteria.

Our algorithm ENCLUS_SIG also incorporates the idea of using a pair of

downward and upward closure properties, which is first used by [9] in the problem of mining correlation rules. This approach was shown effective in the reduction of the search space. In our problem, the downward closure property is given by entropy while the upward closure property is given by the dimensional correlation which is also based on entropy. By the use of the two closure properties, the algorithm has good pruning power. Another algorithm ENCLUS_INT relaxes the upward closure property so that the non-minimal correlated subspaces are also mined. Experiments have been carried out to show the proposed algorithm can successfully identify the significant/interesting subspaces and the pruning is effective and efficient. The algorithms are compared to CLIQUE [2] and are found to have better performance. The accuracy of ENCLUS is also higher in some forms of data. We also propose some miscellaneous enhancements to ENCLUS that can make it more powerful.

Bibliography

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, pages 487–499, 1994.

[2] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD Conference on Management of Data, Montreal, Canada*, 1998.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD, Washington, DC, USA*, pages 207–216, 1993.

[4] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Welsley, 1974.

[5] E. Backer. *Computer-assisted reasoning in cluster analysis.* Prentice Hall, 1995.

[6] Michael J. A. Berry and Gordon Linoff. *Data Mining Techniques for Marketing, Sales and Customer Support.* Wiley, 1997.

[7] P. S. Bradley, Usama Fayyad, and Cory Reina. Scaling clustering algorithms to large databases. In *Proceedings of International Conference on Knowledge Discovery and Data Mining KDD-98, AAAI Press*, 1998.

[8] P. S. Bradley, O. L. Mangasarian, and W. Nick Street. Clustering via concave minimization. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems -9-*, pages 368–374, Cambridge, MA, 1997. MIT Press.

[9] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1997.

[10] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1997.

[11] David K. Y. Chiu and Andrew K. C. Wong. Synthesizing knowledge: A cluster analysis approach using event covering. In *IEEE Transactions on Sytems, Man, and Cybernetics, Vol. SMC-16, No. 2, March/April 1986*, pages 251–259, 1986.

[12] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley Series in Telecommunications, 1991.

[13] I. Csiszár and J. Körner. *Information Theory: Coding Theorems for Discrete Memoryless System.* Academic Press, 1981.

[14] Jay L. Devore. *Probability and Statistics for Engineering and the Sciences.* Duxbury Press, 4th edition, 1995.

[15] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *Proceedings of the 24th VLDB Conference, New York, USA*, 1998.

[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise.

In *Proceedings of International Conference on Knowledge Discovery and Data Mining KDD-98, AAAI Press*, pages 226–231, 1996.

[17] U. Fayyad, G. Piatetsky-Shapiro, and P. Symth. *Advances in Knowledge Discovery and Data Mining.* AAAI/MIT Press, 1996.

[18] Takeshi Fukuda, Yasuhiki Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1996.

[19] Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Constructing efficient decision trees by using optimized numeric association rules. In *Proceedings of the 22nd VLDB Conference, Mumbai(Bombay), India*, 1996.

[20] Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Mining optimized association rules for numeric attributes. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, June 1996.

[21] Clark Glymour, David Madigan, Daryl Pregibon, and Padhraic Smyth. Statistical themes and lessons for data mining. *Data Mining and Knowledge Discovery*, 1:11–28, 1997.

[22] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data, Montreal, Canada*, June 1996.

[23] John A. Hartigan. *Clustering algorithms.* Wiley, 1975.

[24] M. Houtsma and A. Swami. Set-oriented mining of assocation rules. Technical Report RJ 9567, IBM Almaden Research Center, San Joe, California, 1993.

[25] Chun hung Cheng, Ada W. Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99), San Diego*, 1999.

[26] Leonard Kaufman and Peter J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis.* Wiley, 1990.

[27] Heikki Mannila and Hannu Toivonen. On an algorithm for finding all interesting sentences extended abstract. In *Proceedings of the 6th Internation Conference on Database Theory*, pages 215–229, 1996.

[28] Pierre Michaud. Clustering techniques. In *Future Generation Computer Systems 13*, pages 135–147, 1997.

[29] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th VLDB Conference, Santiago, Chile*, 1994.

[30] J. Nievergelt and H. Hinterberger. The grid file: An adaptable, symmetric multikey file structure. In *ACM transactions on Database System*, pages 38–71, 1984.

[31] J.R. Quinlan. Induction of decision trees. In *Machine Learning*, pages 81–106. Kluwer Academic Publishers, 1986.

[32] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[33] Erich Schikuta. Grid-clustering: An efficient hierarchical clustering method for very large data sets. In *Proceedings of Internation Conference on Pattern Recognition (ICPR)*, pages 101–105, 1996.

[34] Jan C A van der Lubbe. *Information Theory.* Cambridge University Press, 1997.

[35] Xiaowei Xu, Martin Ester, Hans-Peter Kriegel, and Jörg Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings of 14th International Conference on Data Engineering (ICDE'98)*, 1998.

[36] Mohamed Zaït and Hammou Messatfa. A comparative study of clustering methods. In *Future Generation Computer Systems 13*, pages 149–159, 1997.

[37] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data, Montreal, Canada*, pages 103–114, June 1996.

Appendix A

Differential Entropy vs Discrete Entropy

The entropy introduced in Chapter 3 is designed for discrete variables. Differential entropy is a continuous version of entropy. Let S be the support set of the random variable and $f(x)$ be the probability density function of the random variable X. The differential entropy $h(X)$ is defined as follows:

$$h(X) = -\int_S f(x)\log(f(x))dx$$

When there are more than one variable, we define the joint differential entropy to measure their uncertainty.

$$h(X_1,\ldots,X_n) = -\int_{S_1}\ldots\int_{S_n} f(x_1,\ldots,x_n)\log(f(x_1,\ldots,x_n))dx_1\ldots x_n$$

In the ideal case, it seems more natural to use differential entropy rather than discrete entropy in our criteria since we focus on mining knowledge from numerical data. However, our decision is not to use differential entropy because:

1. Differential entropy does not have the non-negativity property. The important

downward closure property given by Lemma 1 would not hold if we had chosen to use differential entropy instead.

2. The calculation of differential entropy requires the probability density function $f(x_1, \ldots, x_n)$, which is not available to us. What we have is only the raw data, and to construct the probability density function from high-dimensional data would be computationally expensive.

Although it is undesirable to use differential entropy in our algorithm, we need to justify the use of discrete entropy in place of differential entropy, which we handle in the next section.

A.1 Relation of Differential Entropy to Discrete Entropy

As described in Section 3.2.1, we calculate the entropy of the data by partitioning it into a grid. When we are dealing with only one dimension, this effectively converts the random variable X into its quantized version X^{Δ}. It is proven in [12] the entropy of the X^{Δ} relates to the differential entropy in the following manner.

Theorem If the density $f(x)$ of the random variable X is Riemann integrable, then

$$H(X^{\Delta}) + log\Delta \rightarrow h(f) = h(X), \text{ as } \Delta \rightarrow 0$$

Thus, the entropy of an n-bit quantization of a continuous random variable X is approximately $h(X) + n$. □

Since $H(X^{\Delta})$ and $h(X)$ differ approximately by a constant log Δ, we can compare the values of the entropy of the quantized variables instead of comparing the values of differential entropy. Similar argument applies for higher dimensions. The interval size

Δ must be carefully chosen so that $H(X^{\Delta})$ gives us good approximation of $h(X)$. See Section 3.2.1 for further discussion on this topic.

Appendix B

Mining Quantitative Association Rules

In this chapter, we discuss the mining of quantitative association rules. This is the work we have done before switching to the subspace clustering problem. We would briefly describe the work in this chapter.

In the problem of mining quantitative association rule, we consider a database $\mathcal{D}$ with one or more numerical attributes $A_1, ..., A_n$ and one Boolean attribute C. We want to find the rules in the form of

$$(A_{i_1}, ..., A_{i_k}) \in R \Rightarrow C$$

where $k \leq n$, $A_{i_1}, ..., A_{i_k}$ are k numerical attribute and C is the Boolean attribute. The k numerical attributes span a k-dimensional space. R is a region in this k-dimensional space. Each transaction can be mapped to a point in this space. We call the right side of the arrow to be the presumptive condition and C to be the objective condition. The meaning of the above rule denotes that if $(A_{i_1}, ..., A_{i_k}) \in R$ is satisfied, then it also satisfies the objective condition C with a certain probability. For example,

$$(Age, Education, Length\text{-}of\text{-}service \in R) \Rightarrow Income\text{-}more\text{-}than\text{-}50k$$

As in binary association rules, each rule is associated with a *support* and *confidence* values. Their definitions are similar to their counterpart in binary association rules. The dimensionality of a rule is the number of attributes contained in the presumptive condition. Previous work has proposed algorithms for finding quantitative association rules of one dimension [20] or two dimensions [18]. The region in two-dimension rules are restricted to some special forms, namely *rectangles* and *admissible regions*. We propose algorithms for solving this problem in higher dimensionality. The types of regions are less restrictive as well. Four types of regions are considered.

Unrestrictive	This is the most flexible form of regions and the mining of this kind of region is efficient. However, it is not very meaningful to human.
Path	This is a form of region that can be joined up with a line without branches.
Connected	The region must be connected. It is more flexible than path because a path is a special case of it.
Clusters	Clusters are defined to be one or more connected regions in this work.

B.1 Approaches

We assume we are going to mine a k-dimensional rule and we already have chosen k numerical attributes and an objective condition C. We discretize the space spanned by the k attributes and store the support and confidence of each cell into a multi-dimensional array.

Unrestrictive regions can be mined using greedy approaches. We pick the cells one by one. Each time we pick an unchosen cell with the highest confidence. Since the overall confidence is a weighted average of the confidence of the individual cells, the overall confidence of the region monotonically decreases as we add new cells. Meanwhile, the support increases. We stop adding new cells when we have sufficient support and confidence.

Path can be mined using a depth-first search. We use depth-first search to try all possible paths. Those with sufficient support and confidence go into the result set. Since an exhaustive search is time consuming, we propose two pruning techniques. First, we calculate a confidence bound. When the current confidence drops below the bound, we know a valid path is impossible so we can terminate the tree search. Second, since we may visit the same configuration more than once in the tree search, we record the visited configurations to avoid repeated visits.

Connected regions are mined with a more complicated algorithm. Since there are too many configurations for connected regions, an exhaustive search is out of question. We propose an iterative approximate algorithm. It chooses several seeds from all the cells at the beginning. Then in each subsequent pass, it tries to grow into neighbouring cells. The overall confidence improves in each pass until further improvement on confidence is impossible. If a region have enough support and confidence, it goes into the result set. However, if a region have enough confidence but not support, it tries to gather enough support from the neighbouring cells.

Clusters are mined by combining the method of unrestrictive and connected regions. Since each cluster is a connected region, we use the algorithm for connected regions to discover all possible clusters. Then, a greedy approach is adopted. We add each cluster to the result set in decreasing order of confidence and stop when there is enough support and confidence.

B.2 Performance

The performance of the proposed algorithms are studied through experiments. The greedy algorithm uses trivial computational time so only the algorithms for path and connected regions are worth studying. Both algorithms scale exponentially with the number of cells, but the running time for the algorithm of mining path rises much faster than that of connected regions. This is because the algorithm for mining path is an exhaustive search algorithm if we do not take the two pruning strategies into

account. The algorithm for mining connected regions does not only run faster but also produce regions of higher support and confidence. This can be attributed to the fact that the connected region is a less restrictive form of region than a path. This also suggests the connected region algorithm produces good approximate solutions.

B.3 Final Remarks

The major problem of these algorithms is that we must first have the target k numerical attributes and an objective condition C in mind before applying the algorithms. In a real database, there are many different attributes and we do not know which attributes should go together in a rule. The same problem exists in previous work that studies quantitative association rules [20, 18]. Also, as discussed in the Section 1.3, clusters are a better representation of knowledge than an association rule. Therefore, we switch to study the subspace clustering problem, which helps us to find out the useful subspaces for further processes.