VECTOR-ITEM PATTERN MINING ALGORITHMS AND THEIR APPLICATIONS

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Jianfei Wu

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Department:
Computer Science

November 2011

Fargo, North Dakota

North Dakota State University

Graduate School

Title

VECTOR - ITEM MINING ALGORITHMS AND THEIR APPLICATIONS

By

JIANFEI WU

The Supervisory Committee certifies that this *disquisition* complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

ABSTRACT

Wu, Jianfei, Ph.D., Department of Computer Science, College of Science and Mathematics, North Dakota State University, November 2011. Vector-Item Pattern Mining Algorithms and Their Applications. Major Professor: Dr. Anne M. Denton.

Advances in storage technology have long been driving the need for new data mining techniques. Not only are typical data sets becoming larger, but the diversity of available attributes is increasing in many problem domains. In biological applications for example, a single protein may have associated sequence-, text-, graph-, continuous and item data. Correspondingly, there is growing need for techniques to find patterns in such complex data. Many techniques exist for mapping specific types of data to vector space representations, such as the bag-of-words model for text [58] or embedding in vector spaces of graphs [94, 91]. However, there are few techniques that recognize the resulting vector space representations as units that may be combined and further processed.

This research aims to mine important vector-item patterns hidden across multiple and diverse data sources. We consider sets of related continuous attributes as vector data and search for patterns that relate a vector attribute to one or more items. The presence of an item set defines a subset of vectors that may or may not show unexpected density fluctuations. Two types of vector-item pattern mining algorithms have been developed, namely histogram-based vector-item pattern mining algorithms and point distribution vector-item pattern mining algorithms. In histogram-based vector-item pattern mining algorithms, a vector-item pattern is significant or important if its density histogram significantly differs from what is expected for a random subset of transactions, using χ^2 goodness-of-fit test or effect size analysis. For point distribution vector-item pattern

mining algorithms, a vector-item pattern is significant if its probability density function (PDF) has a big KullbackLeibler divergence from random subsamples. We have applied the vector-item pattern mining algorithms to several application areas, and by comparing with other state-of-art algorithms we justify the effectiveness and efficiency of the algorithms.

ACKNOWLEDGMENTS

First of all, I would like to take this opportunity to express my deepest appreciation to my advisor, Dr. Anne M. Denton, for her knowledgeable and patient guidance, for her encouragement, and for providing research directions in my journey of pursuing my Ph.D degree. Her tremendous support for my research makes this dissertation possible.

Secondly, I would also like to thank all my other committee members, Dr. Kendall E. Nygard, Dr. Birgirt M. Prüss, and Dr. Jun Kong, for their contributions to my academic growth and for their valuable suggestions on my dissertation. Furthermore, I would like to thank all faculty members in the Department of Computer Science at NDSU for their kind help during my course studies and course projects.

Thirdly, I would like express my gratitude to my parents, Hougao Wu and Huijuan Jin, for their unconditional support and encouragement.

Last but not least, special thanks to my dearest wife, Meng Luo, for her encouragement, love, and understanding that enables this dissertation to be completed. I am also very proud of my daughters, Lucy Wu and Rachel Wu, for their loveliness and intelligence. Your supports make everything possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

With new data being accumulated fast in addition to already large amounts of existing data from both industry and science domains, and the urgent need for finding useful information and knowledge among the data, data mining has attracted great attention among scientists and researchers. In the past several decades, many efficient and effective data mining algorithms have been developed, covering areas from market analysis, sales prediction, production control, social network analysis, to fraud detection, intrusion detection, customer prediction, search engine and so on.

Data mining can be defined as the process of extracting interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from large information repositories such as relational databases, data warehouses, XML repositories, etc [35]. Many people treat data mining as a synonym for "Knowledge Discovery from Data" (KDD). Alternatively, others view data mining as simply an essential step in the process of knowledge discovery [68]. Knowledge discovery mainly consists of three processes:

1. **Preprocessing**

   Preprocessing includes the following steps:

   a. Data cleaning, which is to remove inconsistent records from the data sets.

   b. Data integration, which is to integrate or combine multiple data sources.

   c. Data selection, where data relevant to the analysis task are retrieved from the database.

   d. Data transformation, where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations.

2. **Data mining**

   This is the step where various data mining algorithms are actually performed.

3. **Postprocessing**

   Postprocessing evaluates the results and presents the knowledge:

   a. Pattern evaluation, which measures the interestingness of the extracted patterns.

   b. Knowledge presentation, where visualization and knowledge representation techniques are used to present the mined knowledge to the users.

1.1. Categories of Data Mining Techniques

Many data mining techniques have been developed in the last several decades. The majority of them belong into one of the following categories.

1. **Classification:** In a classification task, a model or classifier is constructed to predict binary or multi-labeled categorical labels, such as "safe" or "risky" for loan application data; "yes" or "no" for marketing data; or "treatment A," "treatment B," or "treatment C" for medical data. These categories can be represented by discrete values, where the ordering among values has no meaning [68].

   In general, classification has two-steps. In the first step, a model or classifier is built to fit the training data set. In this learning step the label information of the training instances is being used. In the second step, the model or classifier is used to predict the labels of test instances.

   Many state-of-art classification algorithms have been developed. Decision tree techniques such as C4.5 [114], Naive Bayes classification, Support vector machines [42], Neural Networks [26], Random forests [27], Logistic regression [13], and adaboost [61] are a few examples.

2. **Prediction:** Similar to data classification, prediction is to predict the output of unseen instances, based on the models built on the training data set. The difference between prediction and classification is that the output of classification is categorical labels of testing data set, while the output of prediction is continuous values.

The most populous prediction method so far is regression, which was first published by Legendre in 1805 [89]. Regression analysis can be used to model the relationship between one or more independent/predictor variables and a dependent/response variable. The two most widely used regression methods are linear regression [76] and nonlinear regression.

Since classification and regression both need label or response information of training data sets, they are also called supervised learning methods.

3. **Clustering:** Clustering is the process of grouping a set of physical or abstract objects into classes of similar objects [68]. A cluster is a collection of data points that are similar to one another within the same cluster and dissimilar to the instances in other clusters. Unlike classification or prediction, clustering does not need labeling or response information.

   The clustering algorithms can be roughly categorized into the following groups: Hierarchical methods, such as agglomerative hierarchical clustering [43] and divisive hierarchical clustering [67]; Partitioning methods, such as k-means algorithm [70] and k-medoids algorithm [104]; Density-based methods, DBSCAN [56] and OPTICS [9]; Grid-based methods, for example STING [131]; Model-based methods, for instance EM [25] and COBWEB [124].

   Because clustering algorithms do not need labeling information, they are usually called unsupervised learning methods.

4. **Mining frequent patterns:** Frequent patterns are patterns, such as itemsets, subsequences, or substructures that appear in a data set frequently [68]. Frequent pattern mining algorithms have been successfully applied in many applications such as mining market data. Frequent patterns like "People buying milk also tend to buy bread" can be useful in building sales strategies. In the past several decades

since Pakesh Agrawal *et al* published the first frequent pattern mining paper [5], many related studies have been conducted, resulting in many efficient and effective algorithms in various applications. Basically, the frequent patterns mining algorithms can be grouped as follows: frequent itemset mining, frequent sub-sequence mining, and frequent sub-structure mining.

Frequent itemset mining identifies sets of items, such as milk and bread, notebook and pen, that appear frequently together in a transactional data set. Apriori algorithm [6] is very useful in decreasing the search space. Frequent sub-sequence mining identifies frequent sub-sequences, such as customers in Amazon usually buy a computer first, then a printer and then a scanner. Frequent sub-structure mining intends to mine frequent substructures, such as sub-tree, sub-network, sub-graph, sub-lattice.

Another related data mining task is association rule mining, which applies a confidence threshold to rules constructed from frequent itemsets. One example is {bread, butter} => {milk}, which means that people buying bread and butter tend to buy milk.

5. **Some other data mining areas:** Besides classification, prediction, clustering and frequent pattern mining, there are several other growing data mining fields. Text mining, multimedia mining, spatial mining and web mining are a few examples [1].

   Text mining is a data mining process that extracts useful information from different type of texts. Text mining tasks mainly include text categorization [79], text clustering [140], concept/entity extraction [57, 102], sentiment analysis [109], document summarization [97], entity relation modeling [119] and production of granular taxonomies [20].

[1] Classification, prediction, clustering and frequent pattern mining techniques may also be used in these applications.

Multimedia mining is a relatively new field for data mining [139]. With the fast accumulation of multimedia data, such as music, images, videos, audios, and hypertexts, the huge demand for extracting useful and interesting information from these data results in the fast development of multimedia mining algorithms in research and industrial communities. [139] presented a multimedia mining prototype, *MultiMediaMiner*, which is able to extract high-level multimedia information and knowledge from large multimedia data bases. In [36], a multimedia data mining framework for discovering important but previously unknown knowledge such as vehicle identification, traffic flow, and the spatial-temporal relations of the vehicles at the intersections from traffic video sequences is proposed.

Web mining mines patterns from world wide web. It can be roughly divided into three broad categories, i.e., web usage mining, web structure mining and web content mining [86]. Web usage mining is the process that extracts useful information from server logs, user histories for example. Web usage mining analyzes user interactions with a web server. Click stream analysis [99] is a good example of web usage mining. Web structure mining focuses on the structure of hyper links between webpages. One of the most active research areas in web structure mining are probably social networks [48, 49, 19]. Web content mining intends to mine contents stored and published in Internet, such as XML documents, HTML, and plaintext.

1.2. Various Data Sources and Types of Data

The most common data sources include relational databases, transactional databases, data warehouses, specialized database systems, flat files, images, data streams, multimedia data, and World Wide Web.

A relational database consists of a collection of interrelated data tables. Each table has one or more columns, which we call "attributes" or "features". Each row in a table is referred to as a record, object, example, data point, or instance. Sometimes a record is

also referred as a vector, where its attribute values serve as dimension values for the vector. Tabular datasets are the most common type of data used for data mining purpose. In Sec. 1.1, most data mining algorithms operate on tabular data sets.

A data warehouse is a repository of information collected from multiple sources, stored under a unified schema, and that usually resides at a single site [68]. The data in a warehouse are usually summarized data. For example, a data warehouse usually stores summarized data for the transactions (per item type or per category) for each store, instead of storing each transaction. Thus, a data warehouse typically follows a multidimensional database structure, in which each dimension corresponds to an attribute or a set of attributes in the schema, and each cell has aggregated information.

A transactional database consists of a file where each record represents a transaction, which typically includes a unique transaction identity number and a list of the items included in the transaction [132]. Additional tables maybe also associated with a transactional database, describing each transaction in more details. Frequent pattern mining and its related technique Association rule mining, which is discussed in Sec. 1.1, are often performed in transaction databases.

The advance of technology and the requirement of storing new types of data resulted in the development of specialized databases [68]. Currently, specialized databases mainly include object-relational databases which are constructed based on an object-relational data model, temporal databases that stores relational data that include time-related attributes, sequence databases which store sequences of ordered events, spatial databases that contain spacial-related information, spatiotemporal databases which store spatial objects that change with time, text databases that contain word descriptions for objects, multimedia databases in which image, audio, and video data are stored, heterogeneous databases that consist of a set of interconnected and autonomous component databases, data streams where data flow in and out of an observation platform dynamically, the world wide web

in which huge amount of web pages are connected.

1.3. Contributions of This Research

This study aims to mine important vector-item patterns hidden across diverse data sets. We consider sets of continuous attributes as vector data. Various types of data can be represented or transformed to vector data. For example, a time series can be viewed as a vector where each time point is considered as a dimension; In image processing, an image usually is analyzed through the extracted features which also can be considered as vector data; In text mining, a document can be represented using a word vector; Furthermore, a large amount of item information may also be available for these vector data set. Besides those common 'Item data', which are mentioned in many other research publications, such as those in market basket research [5] or protein domain information [111], we consider any data that can serve as group informations such as those from clustering algorithms as item data. Corresponding to the previous vector data examples, if the time series is the price change in stock, then item data could be the stock's sector information. The item information for an image vector may be the image's category, for instance 'animal' or 'building'; If the vector data is a document's word vector, then the document's type, 'Sports' or 'Scientific' for example, would be good item information.

If the vectors selected by an item[2] show an unexpected density, we consider the item to show a vector-item pattern. Figure 1.1 illustrates the concept. Each of the circles represents an object or transaction. The spatial position of the object corresponds to a vector attribute. In general, a vector attribute is composed of D continuous attributes that are assumed to form a vector space. In Figure 1.1, circles that are solid black represent the objects, for which a particular item is present. Only a single item is represented in this image, but the process can be applied to many items. In the left panel the solid black circles are close together. The right panel shows objects with the same vector data as the

[2]By *select* we mean that the vectors hold the item under consideration.

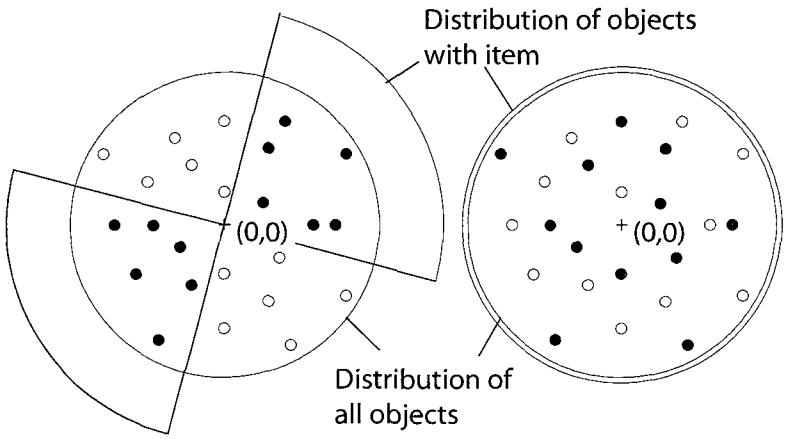


Figure 1.1: Schematic of objects that are characterized by vector data (position in plane) and item data (circles solid black). The angular distribution of all objects and of only those objects with the item are shown around the perimeter. The left panel shows an example of an existing vector-item pattern; no pattern is observed for the example in the right panel.

left panel, but the item data are associated with different objects. Although the vector data are identical, the item data look far more distributed. The setting on the left side illustrates what we consider a vector-item pattern.

In the following chapters, we will discuss several vector-item pattern mining algorithms and their applications. The vector-item pattern mining algorithms can be basically grouped into two categories: **histogram-based algorithms** and **point distribution algorithms**.

In Chapters 2 - 6, we will discuss the histogram-based vector-item pattern mining algorithms and their applications.

In Chapter 7, we will discuss the point distribution vector-item pattern mining algorithm and its application in sequential clustering.

1.4. Evolution of Vector-Item Pattern Mining Algorithms and Author's Contribution

This dissertation is composed of several published and submitted papers. The idea of vector-Item pattern was first proposed by Dr. Anne Denton in 2007. I developed the first vector-item pattern mining algorithm, namely the subspace-based vector-item pattern mining algorithm. The algorithm and its application for annotating protein domains were

published in [134]. The subspace-based vector-item pattern mining algorithm will be given in Chapter 3.

In a subsequent project for which we collaborated with Dr. Birgit M. Prüss' group, we related patterns of gene expression in a set of microarray experiments to functional groups of proteins. The vector-item pattern mining algorithm lent itself naturally to this project, when considering functional groups as item information and gene expression as vector data. The challenge of this project was that biologists are more interested in the genes that are differentially expressed. To find significant vector-item patterns in which genes are also differentially expressed, we developed the product similarity-based vector-item pattern mining algorithm and afterwards applied it on a previously published *Escherichia coli* data set [108]. The algorithm and its application were published in [47]. In this project, I did all the calculations including those of comparison software, created plots and tables to show experiment results, and participated in the writing of the paper[3]. The product similarity-based vector-item pattern mining algorithm will be discussed in Chapter 4.

I was very interested in identifying the instances within an item set that make this item set hold a significant vector-item pattern. These instances usually have a high density specific to the item set. In histogram-based vector-item pattern mining algorithms, by comparing observed and expected density histograms (definitions will be given in Sec. 2), we can identify these important instances [4]. In time series data sets, these instances tend to show coherent behavior. Furthermore, we developed an efficient quasi-clique mining algorithm which is further applied to these important instances. Through comparing DBScan and Kmeans clustering algorithms, we show the efficiency and effectiveness of the proposed algorithm. The algorithm was published in [133], and an extended version was submitted. In this project, I developed the algorithm and wrote a first draft of the paper. Thanks to Dr. Anne Denton for improving the writing of the paper. The algorithm will be

[3]All authors contributed equally in writing up the paper

[4]Details are given in Sec. 5

discussed in Chapter 5 in great detail.

There are two main drawbacks of using subspace-based and product similarity-based vector-item pattern mining algorithms. The first one is that it does not work well on small data sets[5]. The second one is that in some cases, the instances in significant vector-item patterns identified by subspace-based and product similarity-based algorithms are more sparse than those in a random sample[6]. To solve these problems, we adopted the "effect size" analysis concept, and substituted the χ^2 goodness-of-fit-test with effect size analysis when comparing the density histograms. In a project, for which we collaborated with the NDSU Center for Nanoscale Science and Engineering (CNSE), we developed an algorithm for identifying important design regions in a combinatorial design that has one or multiple responses. The importance of a design region is measured through a model that incorporates the effect size and confidence intervals of the design region. The important design regions will not only lead to low variances of multiple responses, but will also optimize them in the same step. By comparing with product similarity-based algorithm, we show the effectiveness and robustness of the proposed algorithm. In this project, I developed the algorithm, performed the experiments and wrote a first draft of the paper. The algorithm will be presented in Chapter 6.

To further improve the accuracy of vector-item pattern mining algorithms, we adopted information theory (Kullback Leibler divergence in particular), which resulted in the development of point distribution algorithm. While histogram-based vector-item pattern mining algorithms summarize point (vector) distributions first and then compare the density histograms, the point distribution algorithm calculates the Kullback-Leibler divergence of kernel densities. A vector-item pattern is considered to hold if the Kullback-Leibler divergence between the probability density function (PDF) of the sample and a random subsample is large. We demonstrate that the point distribution algorithm is

[5]Details are given in Sec. 6

[6]In some domains, these vector-item patterns could probably be interesting for the domains' experts.

more effective on gene expression and other data than any of the comparison algorithms. The algorithm was published in [45]. In this project, the algorithm was developed jointly by Dr. Anne Denton and me. Since the paper was solely written by Dr. Anne Denton, only a brief description of the point distribution algorithm will be given in Chapter 7.

The point-distribution algorithm provides us with a very effective way to compare subset distributions w.r.t. overall distributions. The "Sequential Data Clustering" algorithm, which was published in [135], shows that the idea can be extended to comparing two overall distributions. We applied the algorithm in the ICMLA2010 speech recognition challenge, and our solution ranked among the top 3 in the challenge. The Sequential Data Clustering algorithms was mainly developed by me and the corresponding paper was also largely written by me. The Sequential Data Clustering algorithm will be given in Chapter 7.

CHAPTER 2. HISTOGRAM-BASED VECTOR-ITEM PATTERN MINING ALGORITHMS

In histogram-based vector-item pattern mining algorithms, we use density histograms to summarize neighborhood relationships between objects, either from a group that has the same item data or from randomly selected samples.

Fig. 2.1 illustrates the problem of interest, records are represented by circles. The vector data determine the positions of the circles in the plane; Existence of the item is represented as solid black filling. Histograms in the two bottom panels summarize the distribution of neighbors with the items. The top left panel shows an item set in which objects are close to each other. We can notice from its corresponding histogram (the left bottom one), that all six objects with the item have five neighbors within the item set. The top right panel shows the same vector data. However, in contrast to the left panel, the objects with the item data are far more distributed, which can also be seen from the right bottom histogram. Only one object has four neighbors, three object have three neighbors and two objects have two neighbors within the item set. Since the histograms summarize the density of an item set, we will call them **density histograms**.

To determine whether an item set shows an unexpected density, we draw multiple random samples that have the same size as the item set, and construct density histograms for the item set as well as the random samples. If the density histogram for the item set is significantly different [7] from the one that is averaged from those of random samples, we determine that the item set is significant and thus shows a vector-item pattern.

Definition 1: Observed Density Histogram: The density histogram that is constructed for an item set under consideration.

Definition 2: Expected Density Histogram: The density histogram that is constructed by averaging multiple random samples which have the same size as the item set under

[7] We will use χ^2 goodness-of-fit test and effect size analysis

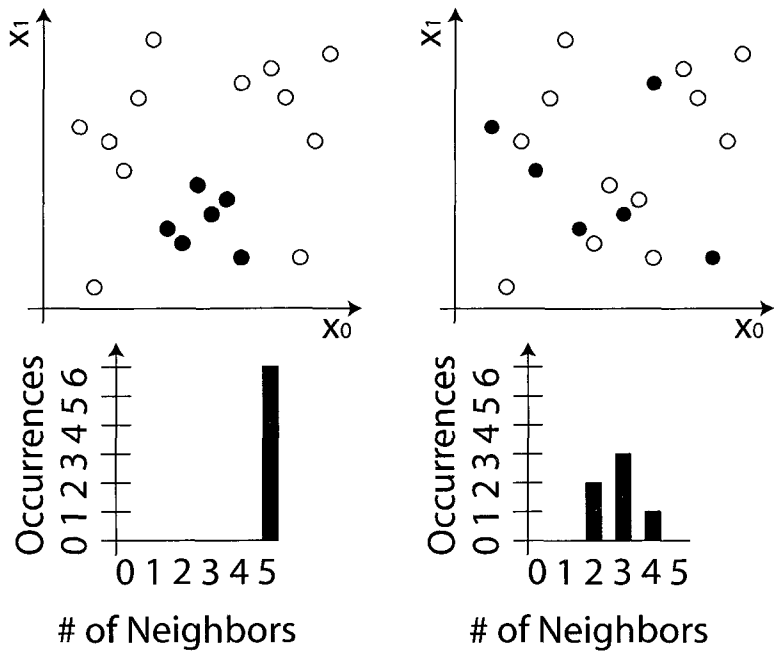


Figure 2.1: Sketch of a vector-item pattern between a 2-dimensional vector and one item. Records are represented by circles. The vector data determine the positions of the circles in the plane; existence of the item is represented as solid black filling. Bottom: Histograms summarize the distribution of neighbors with the item. Left: Records that have the item are close (vector-item pattern noticeable). Right: Same vector data as left but item data are distributed such that no vector-item pattern is noticeable.

consideration.

We call the density histogram for the item set **Observed Density Histogram**, and the density histogram that is constructed from random samples **Expected Density Histogram**, since the random samples serve as the expected distribution for the item set.

2.1. Outline of Histogram-Based Vector-Item Mining Algorithms

The histogram-based vector-item mining algorithms include the following steps:

(1) Normalization

Vector data set are first normalized so that the data from the respective noise models lead to homogeneous density distributions. We will discuss the normalization process in the following chapters in greater details.

(2) Density Histograms Construction

The observed density histogram as well as the expected density histogram are constructed for each item data, using various distance criteria.

Suppose a vector data set includes D continuous attributes, i.e., each vector has D dimensions, $x_i \in \mathbb{R}, 0 \leq i < D$. Binary item data $B^{(i)}, 0 \leq i < M$ represents the presence of item i, with M distinct items occurring in the database. A data set with a single vector attribute is defined through the relation $R(V, B^1 ... B^M)$. Each binary attribute $B^{(i)}$ defines a subset of the vector data. For each of these subsets, the density is summarized using the observed density histogram. The sub-relation $R^{(i)}$ defined by item $B^{(i)}$ is given through the following selection function (σ)

$$R^{(i)} = \sigma_{B^{(i)}}(R) \tag{1}$$

and the corresponding set of points is given through a projection (π) to vector attribute V:

$$V^{(i)} = \pi_V(R^{(i)}). \tag{2}$$

For each $V^{(i)}$, we summarize the density distribution through a histogram. We assume that a neighbor selector function c has been defined so that it has the following property:

$$c(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{y} \text{ is to be considered a neighbor of } \mathbf{x} \\ 0 & \text{else} \end{cases} \tag{3}$$

Given c(**x**,**y**), the observed density histogram of V can be defined such that each **x**, for which the sum of neighbors is equal to k, contributes 1 to the k^{th} bin h_k:

$$
\begin{aligned}
h_k &= \sum_{\mathbf{x} \in V} \delta_{k,s(\mathbf{x})} \quad \forall k \\
\text{with} \quad s(\mathbf{x}) &= \sum_{\mathbf{y} \in V} c(\mathbf{x}, \mathbf{y}),
\end{aligned} \tag{4}
$$

where $\delta_{i,j}$ is the Kronecker delta, which is 1 when $i = j$ and 0 otherwise.

To construct the expected density histogram, we select multiple random samples which have the same size as the item data, and for each sample we construct its density histogram using the same procedure as for the observed density histogram, as discussed above. Then the expected density histogram is constructed by averaging these random density histograms.

This construction requires creating density histograms for many random samples, and is therefore very time consuming. Thus we created theoretical models of expected density histograms for each noise model, which will be discussed in the following chapters.

(3) Determining Significance

We apply a χ^2 goodness-of-fit test or effect size analysis (Cohen's D is used in this study) to the density histograms. The item sets that are significant at 5% level or have a Cohen's D greater than 0.8 are considered significant/important, which means the item data and the vector data show a vector-item pattern.

Algorithm 1 summarizes the whole procedure of histogram-based vector-item mining algorithms. First the whole data set is normalized. For each item, the objects are selected within the item (line 4). Then the observed density histogram is constructed (line 5 to line 7). Switch $UsingTheoreticalModel$ is used to determine whether a theoretical model is used or multiple random samples are extracted to construct the expected density histogram. Function $SignificanceTest$ is used to compare the observed density histogram with the theoretical model (or the expected density histogram). We basically use two types of

significance test. In Sections 3, 4 and 5, we use a χ^2 goodness-of-fit test to test the two density histograms. In Section 6, we use an effect size analysis (Cohen's D) instead.

Algorithm 1: Density Histogram Algorithm

```
Data: pts;                                          /* nPts data points */
Data: items;
Data: UsingTheoreticalModel;
Result: significance;                                /* of each item */
1  normPts = normalize(pts);
2  hist = zeros(1,nPts);                             /* vector of zeros */
3  foreach it ∈ items do
4      itemPts = findPoints(normPts,it);
5      foreach x ∈ itemPts do
6          dens = NumberOfNeighbors(x);
7          hist(dens)++;
8      randHist = zeros(1, nPts);
9      if UsingTheoreticalModel==1 then
10         randHist=ConstructTheoreticalModel(normPts,occurrences(it));
11     else
12         for i=1:nAv do
13             randPts = randSubset(normPts,occurrences(it));
14             foreach x ∈ randPts do
15                 dens = NumberOfNeighbors(x);
16                 randHist(dens)++;
17         randHist/ = nAv;
18     significance(it) = SignificanceTest(hist, randHist);
19 return significance
```

CHAPTER 3. SUBSPACED-BASED VECTOR-ITEM PATTERN MINING ALGORITHM AND ITS APPLICATION IN MINING VECTOR-ITEM PATTERNS FOR ANNOTATING PROTEIN DOMAINS

3.1. Introduction

For many years, disk space has grown exponentially at rates higher than Moore's law for the number of microprocessors on a chip. A consequence is not only that larger tables are stored, but also that the diversity of data associated with any one entity has increased. In bioinformatics the increased storage capacity has been matched with developments in high-throughput experimentation. It is typical that any one protein is not only characterized by its sequence and derived sequence signatures, but also by results from high-throughput experimentation, text from scientific publications related to the protein, and other types of data.

Many algorithms for finding frequent or characteristic patterns have been developed for item data [5, 6, 73], continuous data [28], and combinations thereof [122, 116, 38]. The assumption behind such algorithms is that each of the continuous attributes represents a separate fact that may independently be related to any combination of the categorical attributes. This assumption, however, is not always valid. Multiple continuous attributes often represent a coherent concept that may or may not be related to items in the database. Gene expression experiments may be performed as time course experiments, in which different attributes correspond to the same overall experimental condition and differ only in the time that has passed since the beginning of the experiment. Feature vectors in image analysis and word vectors in text mining, are also examples of continuous data for which similarity is normally determined from a combination of many or all of the attributes.

Figure 2.1 illustrates the problem. Each of the circles represents an object or transaction. The spatial position of the object corresponds to a vector attribute, that is

— in this example — two-dimensional. In general, a vector attribute is composed of D continuous attributes that are assumed to form a vector space. In Figure 2.1, circles that are solid black represent objects, for which a particular item is present. Only a single item is represented in this image, but the process can be applied to many items. In the left panel the solid black circles are close together. For each of the solid circles all of the other circles can be considered "close" according to the closeness criteria we develop in Section 3.3.4. The histogram under the left panel reflects the observation that there are five objects that have the item of interest, each of which has six neighbors that have the item. The right panel shows objects with the same vector data as the left panel, but the item data are associated with different objects. Although the vector data are identical, the item data look far more distributed, and the histogram shows that relevant objects only have two to four neighbors. The setting on the left side illustrates what we consider a vector-item pattern.

A natural test as to whether the histograms that represent the data are exceptional is based on random permutations. If a random selection of an equal number data points leads to a similar histogram, then the selection criterion / item is not related to the vector data. We will show in Section 3.3.5 that the expected distribution of points can also be evaluated analytically.

So far, biologists who do time-course experiments are faced with cumbersome gene lists that are difficult to interpret. They may notice that several of the genes that are similarly expressed also have certain interesting domains or functional annotations. Such reasoning corresponds to a two-step process, in which the expression data is first analyzed with traditional techniques and relationships with domain or functional data are studied as a separate step. We use this hypothetical two-step process for comparison purposes in Section 3.4. Our approach removes the artificial separation of the two steps and directly returns the domain or functional information.

In our algorithm, the vector data are normalized based on rank order. We then use the

existence of a categorical attribute value as filter to derive subsets. For each of those subsets we create a density histogram that summarizes the number of neighbors to each point. A point is considered a neighbor if the number of dimensions in which it is within a predefined range exceeds a predefined threshold. We study the choice of range and threshold parameter extensively. We show that our algorithm is both effective and efficient for an example data set consisting of Interpro domain information [118], and gene expression data from cell-cycle experiments [121] for all yeast genes.

3.2. Related Work

Much work has been done on clustering gene expression data to find the genes that show a similar differential expression pattern under the given conditions [78, 55, 14, 82, 117, 39]. While this work does not directly relate genes to domain or functional information, it is worth noting that functional information is sometimes used to improve clustering results [80, 83]. Simultaneous identification of functional groups and the genes that belong to them has been achieved through biclustering techniques [37]. Such approaches assume that the vector data show multiple groups, which is not assumed in our approach. The significance of clustering results has been studied in the context of validation of clustering approaches [137], and functional information has been considered in this context [22]. Expression patterns have also been used to identify differentially expressed genes in time course experiments [44, 62, 15, 121].

It has frequently been observed that the Euclidean distance and other L_p norms are problematic in high dimensions [3], and new distance measures have been developed [2] that only consider a subset of dimensions over which points are close. We use a similar concept of requiring a fraction of dimensions to be within a given range. We show that such a distance measure can be interpreted as a subspace-based evaluation. In contrast to conventional subspace clustering based on axis-parallel projections [4], our distance measure evaluates similarity in any of the projections. More importantly our algorithm

searches for subsets of the data that show inhomogeneities rather than looking for clusters in the full set of points.

The normalization in this algorithm is related to quantile normalization [23] in that the quantile of a point determines its normalized value. In our normalization we consider a constant distribution as reference distribution. Such a normalization results in distances that are closely related to the mass-distance discussed in [138].

3.3. Algorithm

3.3.1. Vector and Item Data

We consider D continuous attributes, $x_i \in \mathbb{R}, 0 \leq i < D$, that are known to be related based on background knowledge, as one "vector" attribute $\mathbf{x}$ with domain $\mathrm{dom}(\mathbf{x}) = \mathbb{R}^D$. The set of occurring data points (extant domain) is $V \subset \mathbb{R}^D$. In principle, a data set can have arbitrarily many vector attributes. Each of the vector attributes is considered separately in the pattern mining step. Note that the continuous attributes that form vector attribute $\mathbf{x}$ do not have to come from the same source. Different combinations of attributes can, furthermore, be considered separately. In the evaluation, we determine patterns involving four sets of experiments as separate vector attributes and also search for patterns involving the full set of all continuous attributes.

Similar to the original formalism of Agrawal et al. [5] we consider item data as Boolean attributes $B^{(i)}, 0 \leq i < M$, that represent the presence of item i, with M distinct items occurring in the database. Conjunctions of the Boolean item attributes could be considered and, among item data, the usual downward closure statement based on support remains valid. However, the significance measure of vector-item patterns does not satisfy the downward closure property. For the evaluation in Section 3.4 we limit our discussion to patterns that involve individual item and vector attributes and, thereby, avoid using a support threshold beyond the requirements of the statistical analysis.

A data set with a single vector attribute is defined through the relation $R(V, B^{(1)} \ldots B^{(M)})$.

Generalization to multiple vector attributes is straightforward.

3.3.2. Outline of the Algorithm

To find patterns among vector and item data we perform the following steps

1. **Normalization:** Vector Data are normalized such that they homogeneously fill a given interval in each individual dimension. Details of the normalization are given in Section 3.3.3

2. **Density Histogram Computation:** For each item, a density histogram analogous to the bottom part of Figure 2.1 is calculated. Density histograms are at the center of the algorithm and are discussed in Section 3.3.4.

3. **Computation of Expected Histogram:** For each item, significance is determined in comparison with the expected histogram. The expected histogram can either be calculated based on theoretical considerations or through random sampling. Both approaches are discussed in Section 3.3.5.

4. **Determining Significance:** Once the observed and expected histograms are known, the significance is determined using a χ^2 test. Those domains that are significant at the 5% level are returned as having a strong pattern with the vector data under consideration.

5. **Considering Multiple Vector Data Sets:** The process can be repeated for different vector data sets.

3.3.3. Normalization

The basic idea for normalizing attributes such that they are evenly distributed in each dimension is to convert to rank order and scale the result accordingly. One may consider the following definition

$$x_i' = \frac{1}{N} \int_{-\infty}^{x_i - \epsilon} \sum_{j=1}^{N} \delta(x - x_j) \mathrm{d}x \tag{5}$$

where $\delta(x)$ is the Dirac delta function and $\epsilon > 0$ is a number smaller than all differences between attribute values. Note, however, that this transformation maps equal values into equal values. That means that some attribute values may occur multiple times, and the modeling through a constant distribution is less accurate.

This problem is more serious than it may initially appear: In practice, gene expression data are not available with arbitrary precision, and the same holds for most other types of data that are commonly called "continuous". The data set we used for the evaluation only lists two digits after the decimal point. Considering that the data set has over 7000 records, some values occur more than 100 times. We cannot uniquely decide on a rank order among these, nor do we expect that the experimental precision is high enough to make such a rank order meaningful. However, assigning the same normalized value multiple times results in a poor fit with the model of a constant distribution. In practice, we assign a random ordering to the records. This is not expected to increase the inherent error since experiments are typically designed such that the resolution of the output values is higher than the precision of the experiment.

3.3.4. Density Histograms

The goal of the density histogram computation is to summarize the distribution of data points with respect to the item of interest. Each Boolean attribute $B^{(i)}$ defines a subset of the vector data. For each of these subsets the density is summarized. The sub-relation $R^{(i)}$ defined by item $B^{(i)}$ is given through the selection (σ)

$$R^{(i)} = \sigma_{B^{(i)}}(R) \tag{6}$$

and the corresponding set of points is given through a projection (π) to vector attribute V

$$V^{(i)} = \pi_V(R^{(i)}). \tag{7}$$

For each $V^{(i)}$, we summarize the density distribution through a histogram. We assume that a neighbor selector function c has been defined that has the following property

$$c(\mathbf{x},\mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{y} \text{ is to be considered a neighbor of } \mathbf{x} \\ 0 & \text{else} \end{cases} \tag{8}$$

For the density definition of conventional density-based clustering with a uniform kernel, the neighbor selector would be

$$c_{\text{uniform}}(\mathbf{x},\mathbf{y}) = \begin{cases} 1 & \text{if} \quad |\mathbf{x}-\mathbf{y}| < d/2 \\ 0 & \text{else} \end{cases} \tag{9}$$

where d is the diameter of the hypersphere that defines a neighborhood. Note that c is of type integer and does not satisfy the normalization conditions of a kernel function.

Given c(**x**,**y**), a density histogram of V can be defined such that each **x**, for which the sum of neighbors is equal to k, contributes 1 to h_k

$$\begin{aligned} h_k &= \sum_{\mathbf{x}\in V} \delta_{k,s(\mathbf{x})} \quad \forall k \\ \text{with} \quad s(\mathbf{x}) &= \sum_{\mathbf{y}\in V} c(\mathbf{x},\mathbf{y}), \end{aligned} \tag{10}$$

where $\delta_{i,j}$ is the Kronecker delta, which is 1 when $i = j$ and 0 otherwise. Figure 3.1 shows the histogram for a real domain (black) and for an average over random subsets of transactions (red).

In the proposed algorithm, we use a neighbor selector for which we only require a fixed fraction of dimensions to match rather than the uniform one of equation (9). We consider two vectors as similar if the number of dimensions in which they are similar

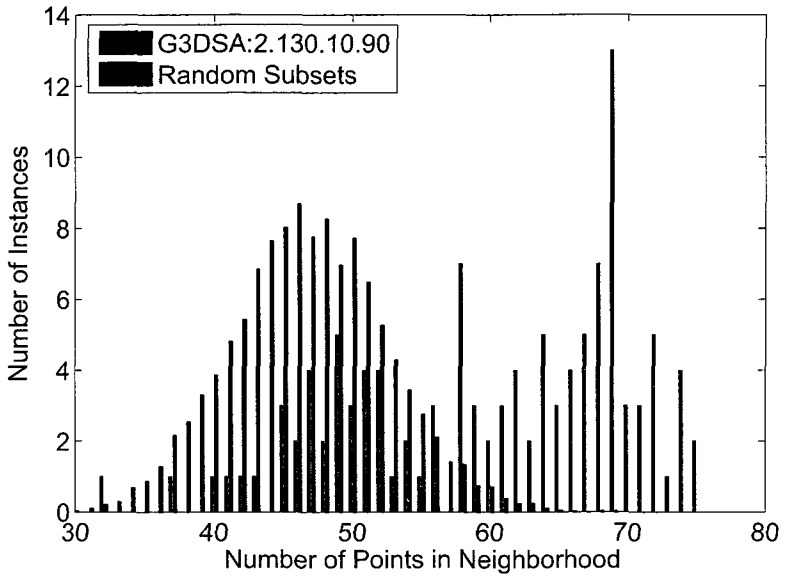


Figure 3.1: Histograms for a real domain from the Interpro database (black) and a random subset of genes (red) with the same number of elements.

exceeds a threshold tD, with D being the total number of dimensions and $t \in]0, 1]$.

$$c_{\text{subspace}}(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if} \quad \sum_i c_{1d}(x_i, y_i) \geq tD \\ 0 & \text{else,} \end{cases} \tag{11}$$

where $c_{1d}(x_i, y_i)$ is a one-dimensional neighbor selector. Note that for $t = 1$ this definition requires values in each dimension to satisfy $c_{1d}(x_i, y_i) = 1$. For $c_{1d}(x_i, y_i) = |x_i - y_i| < r/2$ this corresponds to defining distances on the basis of the MAX metric, i.e. the distance is determined by the maximum of distances in individual dimensions. In practice, we found it useful to limit t to $t \in [0.2, 0.8]$.

The neighbor selector of equation (11) can also be interpreted as being subspace-based. Requiring that a point be close in a fraction of t dimensions is equivalent to saying that the point should be within any one of the axis-parallel projections of a hypercube that satisfy the requirements of the one-dimensional selector $c_{1d}(x_i, y_i)$ and the maximum number of dimensions over which projections have been performed $(1 - t)D$.

Figure 3.2 illustrates the subspaces for two dimensions in which a minimum of 1 dimension is required to match, corresponding to a t in the interval $]0, 0.5]$. In the light shaded areas only one of the dimensions respectively corresponds to regions within the range of the data point $\mathbf{x}$.

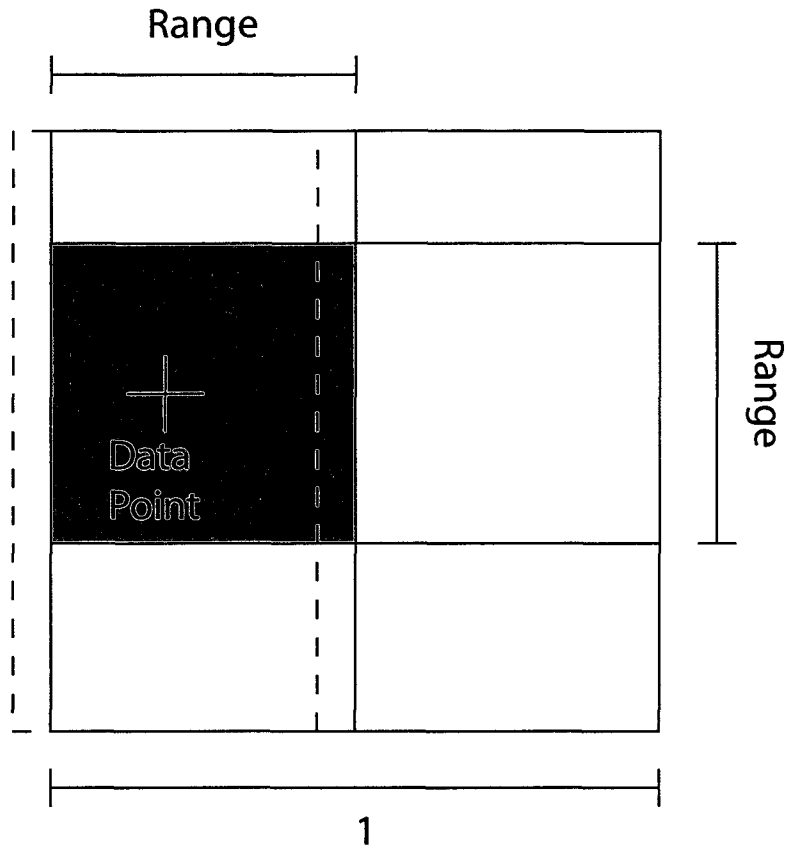


Figure 3.2: Illustration of subspaces in two dimensions. Subspaces that extend beyond the volume defined by the normalization procedure are shifted to fit completely inside.

It is important for our concept that neighbor distributions of randomly distributed points only depend on the number of points and not on the position of the central point with respect to the normalization volume. For that reason we have to ensure that the volume for which $c_{\text{subspace}}(\mathbf{x}, \mathbf{y}) = 1$ does not depend on $\mathbf{x}$. We do so by shifting this volume such that it does not extend beyond the boundaries of the volume defined by the normalization. The dashed lines in Figure 3.2 illustrate an area that is defined by $[x_1 - 0.5, x_1 + 0.5]$ in two

dimensions. Note how this area exceeds the boundaries of the square with side length 1 over which the normalized data extend. The shaded area illustrates the shifted area, which is entirely enclosed in the square. In general, the one-dimensional neighborhood selector $c_{1\mathrm{d}}(x_i, y_i)$ is defined as

$$c_{1\mathrm{d}}(x_i, y_i) = \begin{cases} 1 & \text{if} \quad (y_i > \min(\frac{1}{2} - r, x_i - \frac{r}{2})) \\ & \quad \wedge (y_i < \max(-\frac{1}{2} + r, x_i + \frac{r}{2})) \\ 0 & \text{else} \end{cases} \tag{12}$$

This definition is not symmetric in $\mathbf{x}$ and $\mathbf{y}$, i.e. point $\mathbf{y}$ may be within range of point $\mathbf{x}$ without point $\mathbf{x}$ being within range of point $\mathbf{y}$. We can use such a measure since we are only interested in the statistical properties of sets of points, and do not make statements about individual points. Figure 3.3 shows the subspace setup in 3 dimensions. An example volume with $r = 0.5$ and $t = 0.5$ is highlighted. In three dimensions, $t = 0.5$ means that a data point must be within the specified range for at least 2 dimensions.

3.3.5. Determining Significance

We search for vector-item patterns by comparing observed histograms with their expected counterparts. Hence, we must first determine the expected distribution for the given number of transactions. In principle, a theoretical derivation can be used for this purpose. We first consider the probability of a data point being within the given range for exactly one axis parallel projection of the D-dimensional space. Let us assume that the projected space is d-dimensional. The probability of a point being within range r of the total interval $[-0.5, 0.5]$ for any one dimension is $p = r$. The probability of being within range r for d dimensions is $p = r^d$. We assume that the point is not within range for the remaining $D - d$ dimensions. The probability of being in precisely the subspace under consideration is

$$p_{\text{one subspace}} = r^d (1 - r)^{D-d} \tag{13}$$

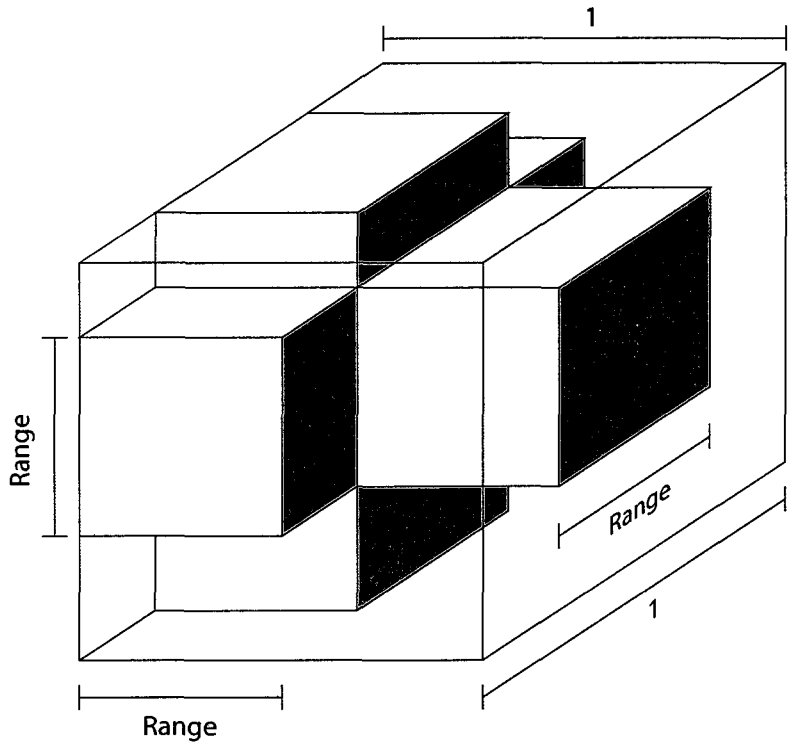


Figure 3.3: Illustration of the volume that defines the neighborhood of an example point in three dimensions, with range $r = 0.5$. Threshold t could be anywhere in the interval $]1/3, 2/3]$ to yield this image.

The number of subspaces with d dimensions is $D!/d!(D-d)!$. Hence, the probability of finding a point within any subspace of at least $d = \lceil tD$ dimensions, where "$\lceil$" is the ceiling operator, is given by the cumulative binomial probability density function

$$\begin{aligned} p &= \sum_{i=d}^{D} \binom{D}{i} r^i (1-r)^{D-i} \\ &= \text{binocdf}(D-d, D, 1-r) \end{aligned} \tag{14}$$

where Matlab notation was used in the second line.

Given this probability p, we can calculate the expected distribution for the density histogram h. This distribution depends on the number of transactions N_i that are selected

by the item attribute of interest b_i:

$$h_k = N_i \begin{pmatrix} N_i \\ k \end{pmatrix} p^k (1-p)^{N_i - k} \tag{15}$$

Although equations (14) and (15) are both governed by underlying binomial distributions, they represent entirely different properties of the system. Equation (14) makes a statement about one point relative to another point. The probability of the point being within the neighborhood of the other is given by a binomial distribution, because the D dimensions of the difference vector are considered to be independent. The point is only required to be within range for d out of a total of D dimensions. Equation (15) makes a statement about all points in one vector subset V_i. We evaluate the neighborhoods for each one of the points as center. For each center, we know that any one other point is in a neighborhood with probability p. The probability that k points out of the total of N_i points are within the neighborhood is given by a binomial distribution. We sample the distribution by picking every point as center once, hence the prefactor N_i.

We compare the observed and expected distribution using a standard χ^2 goodness-of-fit test. Bins at both ends of the distribution are merged until the expected number is at least 5. If intermediate bins have an expected number smaller than 5 then pairs of bins are merged until no more bins have an expected number < 5 (one recommended strategy according to [96]). A vector-item pattern is considered significant if the χ^2 goodness-of-fit test yields a p-Value < 0.05.

In practice, we derive the expected distribution by averaging over a large number of randomly picked subsets (in the evaluation we use 50). This step ensures that the expected distribution appropriately reflects the real distribution even when the distribution of the complete data set deviates from the fully random model. Such deviations can occur since we can only fix the distribution of individual dimensions. Correlations among attributes

can potentially still lead to inhomogeneous distributions in multi-dimensional subspaces and the full space. The data set we use also suffers from many missing values. In Figure 3.4 an averaged distribution is plotted against the theoretical binomial distribution. For this plot, imputation was used and missing values were replaced by zeros before normalization. It can be seen that, although the fit is not perfect, p-Values based on both types of evaluation are closely related.

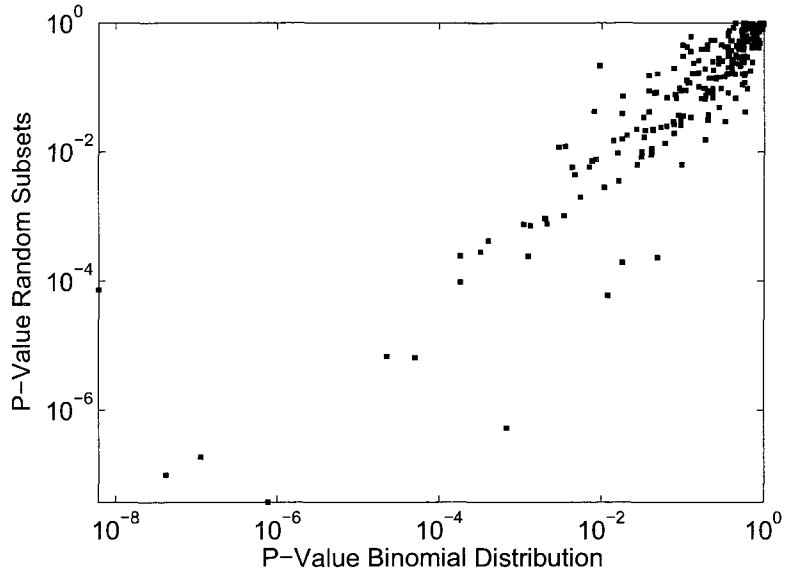


Figure 3.4: p-Value for all Interpro domains comparing with a binomial distribution (rightward) and an experimentally determined comparison distribution (upward).

The complete algorithm can be seen as Algorithm 2. An iteration is performed over all items and, for each item, densities are evaluated at the location of each point that has the given item. "findPts" selects those vectors for which the item is present, and "randSubset" selects a random subset of the same number of points. "NumberOfNeighbors" function is given by a sum over equation (11)

$$dens(x) = \sum_{y \in pts} c_{\text{subspace}}(\mathbf{x}, \mathbf{y}) \tag{16}$$

A faster version of the algorithm would use the equation 15 to determine the comparison

Algorithm 2: Density Histogram Algorithm

```
   Data: pts;                                   /* nPts data points */
   Data: items;
   Result: significance;                            /* of each item */
 1 normPts = normalize(pts);
 2 hist = zeros(1,nPts);                        /* vector of zeros */
 3 foreach it ∈ items do
 4 |   itemPts = findPoints(normPts,it);
 5 |   foreach x ∈ itemPts do
 6 |   |   dens = NumberOfNeighbors(x);
 7 |   |   hist(dens)++;
 8 |   randHist = zeros(1, nPts);
 9 |   for i=1:nAv do
10 |   |   randPts = randSubset(normPts,occurrences(it));
11 |   |   foreach x ∈ randPts do
12 |   |   |   dens = NumberOfNeighbors(x);
13 |   |   |   randHist(dens)++;
14 |   randHist/ = nAv;
15 |   significance(it) = SignificanceTest(hist, randHist);
16 return significance
```

distribution, but in this paper we opted for accuracy of the result.

3.3.6. Parameter Choices

Our algorithm allows, in principle, choice of the two parameters, r (range) and t (threshold). Fortunately, one of the parameters can be chosen based on fundamental considerations alone: Our algorithm is expected to be most sensitive to local density fluctuations if the probability that enters into equation (15) itself is sensitive to fluctuations. In the random model with uniform distribution, p depends on input parameters t and r as shown in Figure 3.5 for $D = 20$. The slope of the distribution is largest for $rD = d$, which can be seen as follows: The maximum of the slope of a cumulative density function is found at the maximum of the corresponding probability density function, which is at $D - d = (1 - r)D$, or $d = rD$ for the distribution in equation (14). Since all our data sets have $D > 10$, we ignore the ceiling operator in $d = \lceil tD$ and choose $r = t$ for all of our

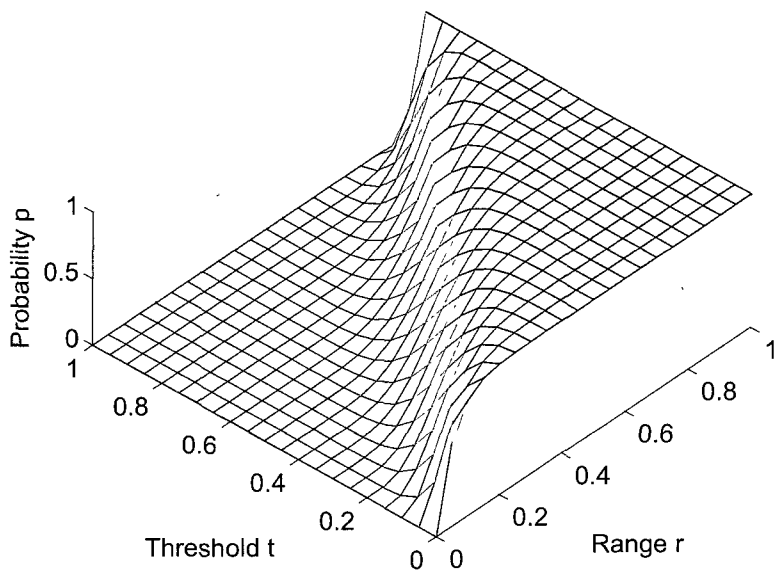


Figure 3.5: Probability p of a point satisfying r (range) and t (threshold) criteria for D=20.

experiments, regardless of whether tD is an integer. The choice of the remaining parameter $r = t$ is somewhat data set dependent. We evaluate different choices and find that typically $r = t = 0.5$ leads to the most stable results.

3.4. Experimental Evaluation

The algorithm is implemented in MATLAB, allowing us to access many of the statistics functions directly as part of the Statistics Toolbox. For simplicity, we use a bitvector representation for items, since vectors can be handled much more easily in MATLAB than sets. We did not encounter memory limitations for the data sets we considered.

3.4.1. Evaluation of Gene Expression Data

In our first evaluation we use gene expression data sets from cell cycle experiments on yeast [121] as vector data, which are available at [120]. Table 3.1 summarizes the properties of the four data sets from separate experiments.

Item data come from the Interpro database, in which information on protein domains, motifs, and other kinds of sequence signatures is collected and combined [100]. For

Table 3.1: Expression data sets

Name	Abbr.	No. of Att.	No. of Genes
Alpha	Alp	22	6177
Cdc15	C15	24	5995
Cdc28	C28	17	6147
Elu	Elu	14	6075
—	All	77	6178

simplicity, we refer to all sequence signatures as domains. Yeast domains are available at [118]. We limit our study those domains for which significance can be evaluated without violating the constraint of having at least 5 instances in at least 2 bins, limiting the set to 329 domains.

We evaluate effectiveness by applying the algorithm separately to the four data sets corresponding to different experiments. For each data set, we determine the domains that are significantly related to the vector data at the 5% level. We then compare the two sets using a χ^2-test on the contingency table of the results. We use a χ^2-test with Yates correction for all our tests on contingency tables to account for possibly small cell values [136].

As an example, the results for the Alpha data set and the Cdc15 data set have a contingency table 3.2.

Table 3.2: Contingency table for the Alpha data set and the Cdc15 data set.

	C15	$\overline{\text{C15}}$	tot
Alp	106	61	167
$\overline{\text{Alp}}$	54	108	162
tot	160	169	329

The contingency table shows that for almost two thirds of the predictions there is agreement between both data sets. This is a reasonably good result, considering, that the experiments were performed independently. The p-value for this contingency table

is 4.52E-8, indicating that it would be highly unlikely to get such an overlap accidentally. Table 3.3 summarizes the mutual overlap and p-values for all data sets we considered, using $r = 0.5$ as range. The combination of all data sets, resulting in a single data set with 77 columns ("All") is also shown. The diagonal represents the number of domains found as significant for each of the techniques. The upper triangular matrix shows the overlap between any two of the experiments. Set below is a lower triangular matrix that gives the p-value of the contingency table corresponding to the two data sets. p-values that are below the 5% significance level are rendered in bold face.

Table 3.3 furthermore shows a comparison with a traditional type of analysis. For this comparison we created contingency tables of genes that have a particular domain compared with genes that have been identified as cell cycle genes in [121]. Significance is determined using a χ^2 test with Yates correction. 54 domains are significant in this analysis. Note that the total number of domains that are considered significant is larger for our own evaluation, which identified up to 219 domains as significant.

Table 3.3: Results for $r = 0.5$

	Sp	All	Alp	C15	C28	Elu	
	54	41	34	38	40	34	Sp
		219	138	133	146	128	All
All	0.11		167	106	124	100	Alp
Alp	0.05	**5E-10**		160	114	104	C15
C15	**5E-4**	**8E-10**	**5E-8**		193	109	C28
C28	**0.01**	**5E-5**	**7E-9**	**8E-6**		159	Elu
Elu	**0.02**	**3E-7**	**3E-5**	**5E-9**	**5E-4**		

We now look at an example domain in more detail. The domain G3DSA:2.130.10.90, which was used as an example in Figure 3.1, occurs 117 times in the yeast data set. Comparing the occurrence of this domain with the Spellman subset, we get the contingency table 3.4.

The p-value based on χ^2-test with Yates correction is 0.071, which is not considered

Table 3.4: Contingency table for the domain G3DSA:2.130.10.90 and the Spellman subset

	G3DSA	$\overline{\text{G3DSA}}$	tot
Spellman	9	790	799
$\overline{\text{Spellman}}$	108	5271	5379
tot	117	6061	6178

significant, and the number of occurrences of domain G3DSA:1.25.40.20 in the genes from the Spellman set (9 times) is actually smaller than would be expected by chance (15.1 times). In our own analysis, using the combined data set, this domain has p-value that is below the smallest double precision value in MATLAB and is hence considered significant, which is in accordance with the impression gained from Figure 3.1.

The overall consistency among our results gives a strong indication that our algorithm is able to extract reliable expression - domain patterns. Table 3.3 shows that the relationship between results from our algorithm is consistently significant, even when expression data are from independent experiments. Overlap is particularly high for comparisons with the full data set. Naturally, any individual data set is dependent of the full data set. The high overlap in results is noteworthy rather because it shows that our algorithm performs as well or better for 77-dimensional vector data as for the sets with fewer attributes.

3.4.2. Evaluation of Time Series Data

The algorithm is also tested on time series subsequence data. For this evaluation, subsequences from a particular time series are associated with one item: the label of the time series from which they originate. The significance of the pattern time series label $\Longleftrightarrow$ subsequence vector is tested. This problem can be assumed to be difficult since clusters of time series subsequences are notorious for being independent from the time series that was used for clustering [85].

For this evaluation, 7 data sets from the UCR time series repository [84] are used and one independently collected one: The Ecg series (MIT-BIH Arrhythmia Database:

mitdb100) originates from PhysioBank [66]. Descriptions of the data sets from [84] are distributed with the data. Subsequences are extracted using a sliding window approach, and differences between successive data points are considered as the dimensions of the vector space. Windows of length 17 are used resulting in a 16-dimensional vector space. 1000 data points are collected from each time series and a randomly selected subset of 100 is used in the evaluation. The data are combined with the full set of points that can be extracted from the UCR random walk time series (65520 points). For comparison purposes, 7 labels are given to samples of 100 randomly picked points from the random walk time series.

A true positive result indicates a label that is associated with a non-random time series and is identified as significant at the 5% level. If a label is considered insignificant although it belongs to a non-random set, the result is treated as a false negative. A label that is associated with random data and is considered (in)significant at the 5% level is a false positive (true negative). Figure 3.6 shows the sensitivity (true positives / all positives) and specificity (true negatives / all negatives) for 50 runs of the data set described in the previous paragraph. It can be seen that the sensitivity is consistently above 0.78. The specificity is greater than 0.8 for range values of 0.5 or less.

It appears that for larger range values the result becomes unreliable, possibly due to the overall fluctuations in the data set. Given the challenging nature of time series subsequence data the result can be considered very promising.

3.4.3. Performance

The algorithm scales linearly with the number of domains. Scaling as a function of domain size is quadratic for large domains, since we construct N_i histograms, each of which represents N_i points. In our data sets most domains are so small that linear contributions related to the χ^2 test play an important role. Figure 3.7 shows the runtime per domain for the gene expression data set plotted as a function of the number of instances

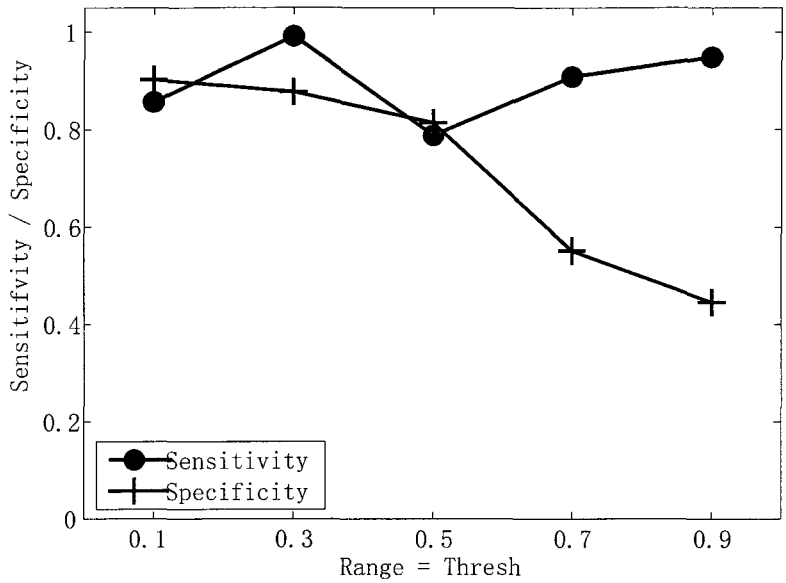


Figure 3.6: Sensitivity and Specificity for experiment on time series data. Sensitivity refers to the rate of recognizing labels that are associated with time series subsequences. Specificity denotes the rate, with which labels that are attached to random data are identified as insignificant.

in the domain, N_i. It can be seen that the initial slope is less than 2 indicating that the relationship is not quite quadratic. For larger domain size, the slope increases to a value close to 2. Overall, performance was not a major concern to us since individual runs for the complete set of domains and the entire yeast genome only took minutes, despite the computation-intensive choice of deriving the comparison histogram through averaging rather than through equation (15).

We have not yet tested the algorithm on combinations of domains. The density-histogram-based significance measure does not satisfy downward closure. Hence, a satisfactory performance of an itemset-based evaluation would have to be achieved through setting a support threshold. We are currently working on techniques for vector-item-patterns that are better suited to patterns involving itemsets and multiple vectors.

3.5. Conclusions

We have presented an algorithm for data mining of vector-item patterns based on

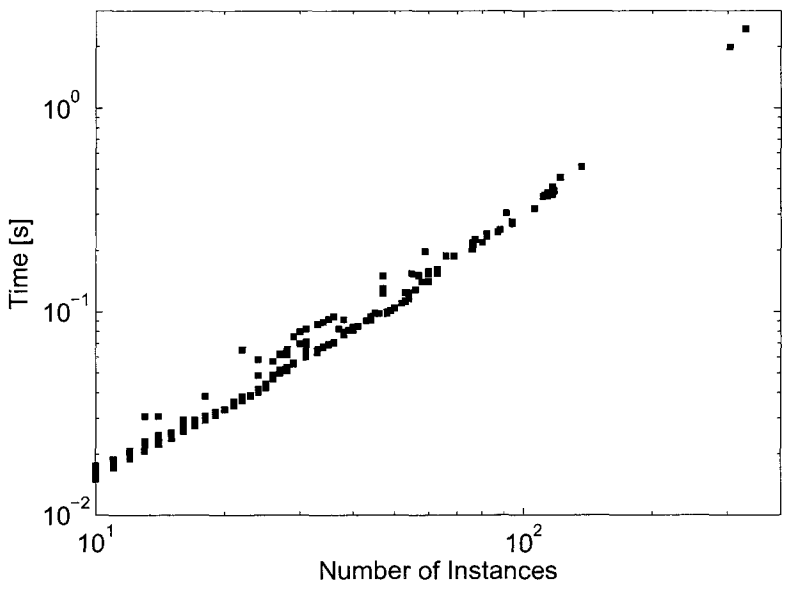


Figure 3.7: Runtime for evaluation of a single domain for one parameter with expected distribution derived from 50 examples.

density histograms. Subsets of the vector data, which are defined by the item data are the basis for the histograms. A density measure is used that considers subspaces. Histograms based on the observed densities at data points are compared with their expected counterparts. A normalization is chosen such that the expected density histograms approximately follow a binomial distribution. Effectiveness and efficiency of our algorithm have been demonstrated using cell-cycle gene expression data and Interpro domain annotations as well as time series subsequence data. We have shown that our algorithm produces results that are consistent among independent experimental data. Comparison with a conventional technique shows that overlap is significant for one of our data sets. For time series subsequence data we were able to show quantitatively that labels associated with non-random data can be recognized with sensitivity and specificity greater than 0.8. This work introduces a new approach toward relating continuous vector data and item data that is potentially valuable in many application areas.

3.6. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. IDM-0415190. Thanks to Alan Denton for proof-reading the manuscript.

CHAPTER 4. PRODUCT SIMILARITY-BASED VECTOR-ITEM PATTERN MINING ALGORITHM AND ITS APPLICATION IN RELATING PROTEIN FUNCTIONS TO GENE EXPRESSION DATA

In this study, we tried to relate patterns of gene expression in a set of microarray experiments to functional groups. Biologists are interested in patterns in which genes are differentially expressed [8]. Therefore, the subspace-based histogram vector-item pattern mining algorithm cannot be directly applied. To tailor the histogram-based vector-item pattern mining algorithm to this problem, we changed the noise model and used product-similarity as similarity measurement for vectors. Through experiments, we demonstrate that our new algorithm is able to find interesting and biologically meaningful relationships in previously analyzed data sets. Scaling of the algorithm to large data sets can be achieved based on a theoretical model.

4.1. Introduction

Microarray experiments are popular tools in functional genomics. Correspondingly, many techniques have been developed to analyze their results. Typical questions asked include which genes are differentially expressed [51], and which groups of genes show similar expression in multiple related experiments [78]. Biclustering techniques have been developed to group functions and experiments simultaneously [37], but those do not intend to identify the significance of the relationship between expression and functional designation. Gene expression information is also used to predict gene functions [29]. In experiments related to transcriptional regulation, the objective is to understand the regulation process rather than predicting protein function. That means that predictive techniques are not appropriate.

Recently, gene set analysis [125] has become a popular tool for relating expression

[8] In several other situations, domain experts favor this kind of pattern, in which vector data have attribute values that deviate from the mean. Financial data is a good example

values to properties that define sets of genes. Gene set analysis tests the relationship of a single gene expression experiment to any one of a number of possible grouping criteria. This is especially valuable in settings where a single gene expression experiment is expected to hold all relevant information such as a comparison between healthy and diseased tissue or between different strains in bacteria. Many gene expression experiments are, however, performed in groups. Examples are time course experiments that compare two strains or conditions over several time points or a set of related experiments. An example of the latter is the study of two-component systems [108] that is used in this manuscript. Gene set analysis does not consider the relationship among the gene expression experiments and thereby ignores relevant information.

Our approach is intended for finding those gene functions that are associated with patterns in the expression data. Fig. 4.1 shows the concept: The same set of curves are shown in the left and in the right panel of the figure. Each curve represents a gene expression profile over multiple related experiments. In each panel, a different subset of profiles is highlighted, corresponding to genes of a different functional designation. The highlighted profiles in the left panel show a clear pattern, while the ones in the right panel do not. We quantify the presence of patterns by identifying neighboring relationships among profiles using a product measure. If a gene has many neighbors with a similar expression profile – more than would be expected by random chance – then it supports the existence of a pattern. The number of neighbors, for all genes that show the function, is summarized in a histogram. In the left panel, two of the profiles have all other three genes as neighbors, and the others have two neighbors. In the right panel, in contrast, none of the profiles has any neighbor. It is expected that some profiles may have neighbors by random chance alone. For this reason, we compare the resulting histograms with ones that are generated for random subsets of the same size.

The algorithm is tested on a published set of microarray data [108], in which

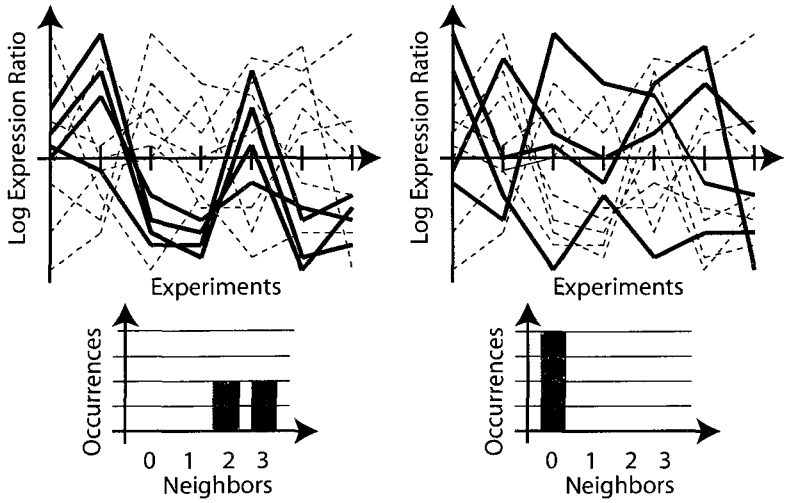


Figure 4.1: Sample expression profiles for two subsets of data. The top panels show gene expression profiles over multiple related experiments. The same set of curves are shown in the left and in the right panel. In each panel, a different subset of profiles is highlighted, corresponding to genes of a different functional designation. The bottom panels quantify the presence of patterns by identifying neighboring relationships among profiles using a product measure. The number of neighbors, for all genes that show the function, is summarized in a histogram.

each experiment corresponds to one knock-out mutant that represents one two-component system, compared to wild-type *E.coli*.

4.2. Algorithm

The algorithm has two objectives: (1) Finding those data points that have more neighbors than expected and (2) identifying those subsets that have a distribution that significantly differs from what would be expected for a random subset. We define a density measure that is evaluated for each data point, and is given by the number of neighbors that are close according to a product similarity measure. Product similarity is used, rather than cosine similarity or Euclidean distance, since we expect those vector pairs to be most relevant that exhibit a large absolute value of differential expression as well as a small angle between vectors. The product similarity measure for vectors $\mathbf{x}^{(j)}$ and $\mathbf{x}^{(k)}$, with coordinates

$x_i^{(j)}$ and $x_i^{(k)}$ respectively, is defined as follows

$$S_{jk} = \sum_{i=1}^{d} x_i^{(j)} x_i^{(k)} \quad 1 \leq j \leq N \quad \text{and} \quad 1 \leq k \leq N \tag{17}$$

N is the number of selected data points. Vectors for which S_{jk} exceeds a threshold t are considered neighbors. The threshold is given as

$$t = \mu D \qquad \mu \in (0,1) \tag{18}$$

where μ is chosen to be 0.09 in the evaluation. Each data point is associated with a density that is of type integer: the number of neighbors that satisfy the product similarity criterion. The occurring density values for all data points can be summarized using histograms. Density calculations are done on z-normalized data [34] i.e. for each attribute, the mean is subtracted and the attribute values are divided by the standard deviation. Fig. 4.2 and 4.3 show examples of histograms of the observed density values that are derived using the product similarity criterion (blue bars). These examples are derived as part of the evaluation on the Oshima data set on two-component systems [108], which is discussed in more detail in the next section.

Even randomly distributed vectors are expected to have some neighbors. We, therefore, have to evaluate the expected distribution of densities. This is done by randomly selecting a subset of genes that has the same number of elements as the protein function under consideration. A histogram is then constructed for the random subset. The process is repeated multiple times and the results averaged over 20 runs. Algorithm 3 summarizes the complete algorithm. Figs. 4.2 and 4.3 (white bars) show the distribution for random datasets in addition to the experimental ones.

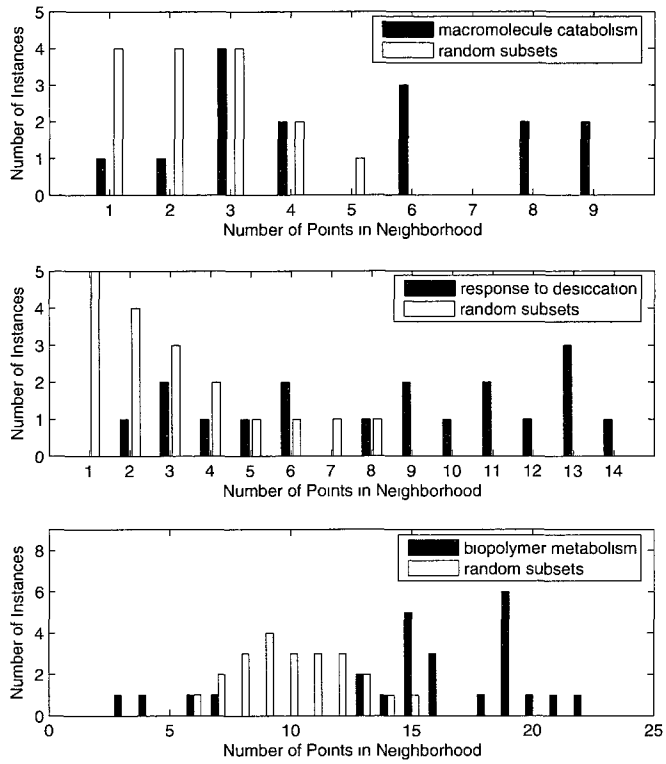


Figure 4.2: Experimental and randomized histograms for the functions macromolecule catabolism, response to desiccation, and biopolymer metabolism.

An observed histogram is considered significant if it would be unlikely to encounter it based on a randomly selected set of genes. A χ^2 goodness-of-fit test is used to compare the histogram with its randomized counterpart. We consider patterns as significant, if the comparison yields a p-value ≤ 0.05. The methods section provides details on the significance testing. Fig. 4.2 shows three vector-item patterns that are considered significant, while Fig. 4.3 shows one counter example that is considered not significant.

4.3. Application of the Algorithm to Gene Expression Data

We applied the algorithm to the gene expression data from Oshima and coworkers [108]. Throughout the entire dataset, gene expression ratios represent the expression of a gene in a particular mutant, divided by the expression of the same gene in the

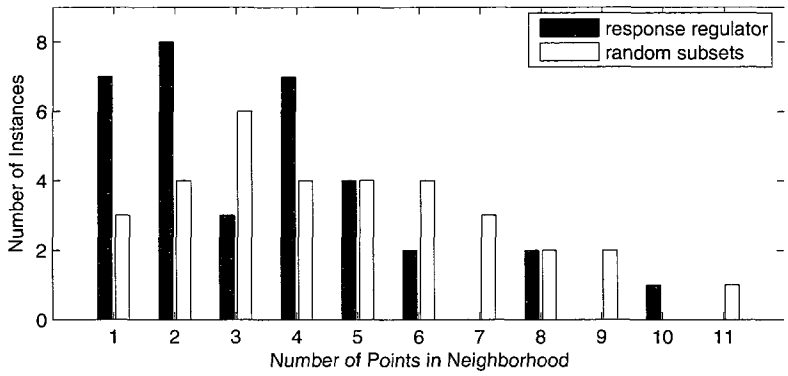


Figure 4.3: Experimental and randomized histograms for the function response regulators.

Algorithm 3: Resampling-based Algorithm

Data: $genes$; /* expression values */
Data: $functions$
Result: $significance, tailGenes$; /* for each function */
1 $normGenes$ = normalize($genes$);
2 $hist$ = zeros(1,nPts); /* vector of zeros */
3 **foreach** $f \in function$ **do**
4 $subset$ = findPoints($normGenes, f$);
5 **foreach** $x \in subset$ **do**
6 $dens$ = NumberOfNeighbors(x);
7 $hist(dens)$++;
8 $randHist$ = findRandomHistogram($1, nPts, normGenes$);
9 $significance(f)$ = chiSquaredGoodnessOfFit($hist, randHist$);
10 $tailGenes(f)$ = findTailGenes($hist, randHist$);
11 **return** $significance, tailGenes$

wild-type strain. Therefore, an expression ratio above 1 indicates that the respective gene is repressed by its corresponding two-component system that is knocked out in the mutant. Expression ratios below 1 indicate activation of that gene by the two-component system. Log expression ratios for individual experiments are considered dimensions in a d-dimensional vector space, where d is the number of experiments considered. Histograms for all functional groups were derived. Seven functional designations were found to be significant, as shown in Fig. 4.4. The histograms for the gene ontology terms GO:0009057=macromolecule_catabolism, GO:0009260=response_to_desiccation,

and GO:0043283=biopolymer_metabolism are presented in Fig. 4.2.

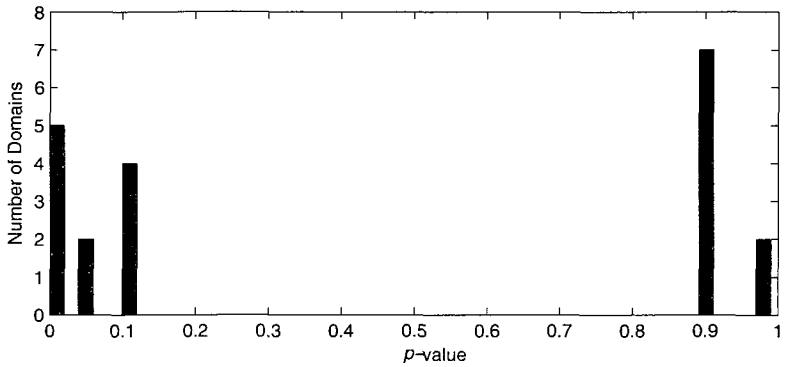


Figure 4.4: p-values of the χ^2 goodness-of-fit test were performed on all 20 domains. Seven domains were considered significant.

Fig. 4.5 shows all those genes in the above functional groups that have a larger number of points in the neighborhood than any of the points in the histograms for random subset (blue tails in the histograms). The individual genes that form this group are indicated in the inserted legends. The numbers on the x-axis symbolize the individual two-component systems. The order of two-component systems (attributes) is identical to the original data set [108]. For the purpose of this study, #3 is OmpR/EnvZ, #4 is BasSR, #9 is YfhA, #15 is UvrY, #16 is YpdAB, #19 is DcuSR, #20 is NtrBC, and #22 is ArcB. These are the two-component systems for which the grouping of expression ratios is most visible within the profiles.

4.4. Comparison with the GSEA Algorithm

We compared our algorithm with the gene set enrichment analysis algorithm, GSEA [5]. In order to use this algorithm, the gene expression data were transformed to GSEA format, then phenotype files as well as gene sets for each domain were created. For both the Oshima and Baev data sets, no domain was considered significantly enriched at nominal p-value ≤ 0.05 by the GSEA algorithm. All possible combinations of parameters, which include 'metric for ranking genes' and 'gene list sorting mode', were used.

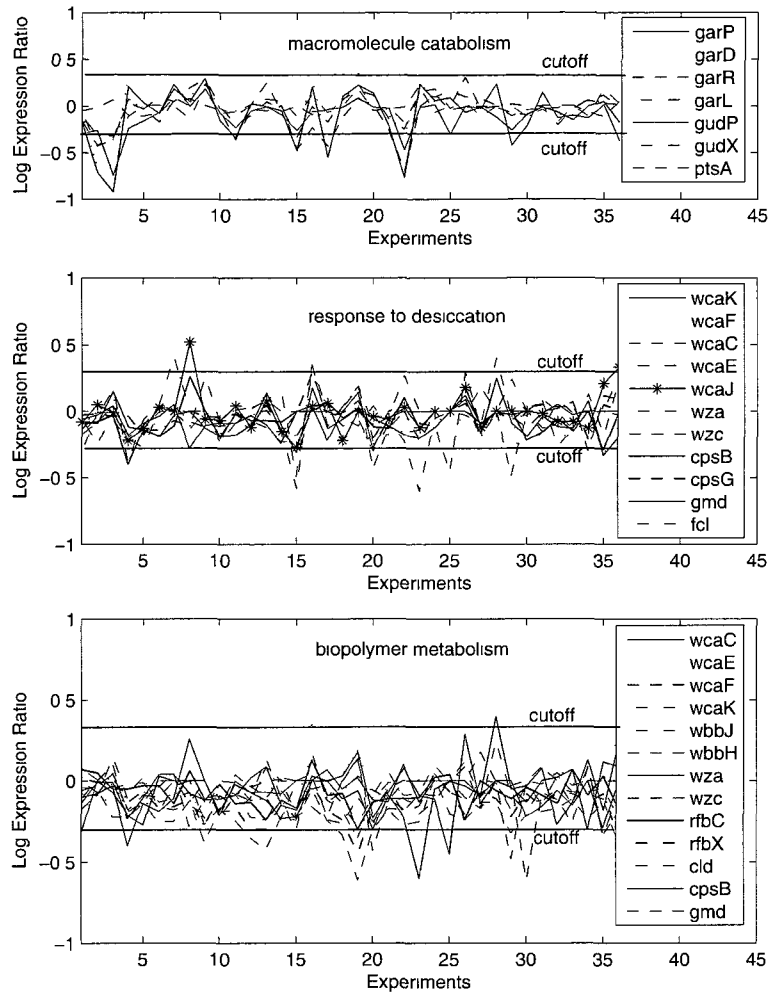


Figure 4.5: Gene expression profiles for the functions macromolecule catabolism, response to desiccation, and biopolymer metabolism

4.5. Performance

The algorithm scales linearly with the number of domains. For each domain, scaling as a function of domain size is approximately quadratic, as Fig. 4.6 shows. For the *E. coli* data set that was used in this study, the quadratic complexity is not a problem, and it can be seen that execution times are so small that performance is not expected to be a bottleneck even for larger genomes.

The runtime for one domain subset can be expressed as follow:

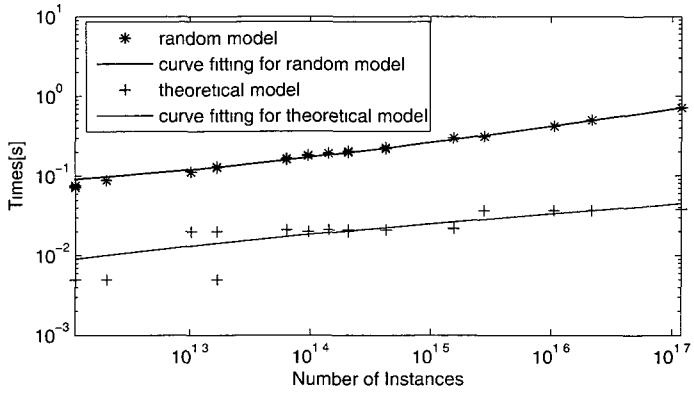


Figure 4.6: Runtime for evaluation of all significant domains.

$$T_c = (R+1) * (T_{extract} + T_{hist}) + T_{\chi^2-test} \tag{19}$$

Where T_c is the runtime of the algorithm for one domain subset, T_{hist} the time to create the histogram for one subset, $T_{extract}$ is the time for extracting the subset of genes, R is the number of runs for random subsets ($R = 20$ in the evaluation) and T_{χ^2-test} is the time for statistical analysis. Under the assumption that data sets are small enough such that all genes can be kept in memory, the time is dominated by T_{hist}, which is quadratic in the number genes in a set.

Notice that the main contribution to T_c comes from the histogram evaluation on random subsets. In the following section, we discuss a theoretical model for evaluating random histogram. The model ignores correlations that may occur even among unrelated genes, and is hence not expected to be as accurate as the randomized evaluation. Nevertheless, it provides an alternative if the data sets are very large.

4.5.1. Theoretical Model

For the random model, we assume that all experimental data follow a normal distribution. This assumption is not expected to be fully accurate, since gene expression experiments typically do not exactly follow a normal distribution. It should be noted that

the assumption only applies to the theoretical model and not to the resampling approach that is used in the remainder of the study. The calculation also assumes that dimensions for the random comparison model are unrelated, which also is an approximation. For these reasons, the resampling model is expected to be substantially more accurate, and the theoretical model should only be used if the computational complexity of the resampling model is considered prohibitive.

The coordinates of one experiment are denoted by the vector $\mathbf{x}$ and the coordinates of the other by $\mathbf{y}$. All those pair of genes are considered significantly related, for which the product is greater than threshold t. The expected probability that the product for any two experiments is beyond the threshold t can be calculated by integrating the following expression over the relevant Gaussian distribution functions:

$$p = \theta\left(\sum_i x_i y_i - t\right) \tag{20}$$

where θ is the Heaviside step function, which is 1 for a positive argument and 0 otherwise. We integrate over all directions of $\mathbf{x}$ and $\mathbf{y}$ with their respective weights

$$\int \theta\left(\sum_i x_i y_i - t\right)\left(\frac{1}{\sqrt{2\pi}}\right)^{2n} e^{-\frac{x^2}{2}}\, e^{-\frac{y^2}{2}}\, \mathrm{d}^n x\, \mathrm{d}^n y \tag{21}$$

Note the data are normalized using z-normalization, resulting in mean 0 and standard deviation 1 for both vector $\mathbf{x}$ and vector $\mathbf{y}$. The radius of the vector $\mathbf{x}$ will be denoted by r and the integration re-written

$$\int \left(\frac{1}{\sqrt{2\pi}}\right)^n e^{-\frac{x^2}{2}}\, \mathrm{d}^n x = \int S_n\, r^{n-1}\, e^{-\frac{r^2}{2}}\, \mathrm{d}r \tag{22}$$

where S_n is the surface of a hypervolume in n dimensions

$$S_n = \begin{cases} \frac{2^{\frac{n+1}{2}} \pi^{\frac{n-1}{2}}}{(n-2)!!} & \text{for } n \text{ odd} \\ \\ \frac{2\pi^{\frac{n}{2}}}{(\frac{n}{2}-1)!} & \text{for } n \text{ even} \end{cases} \tag{23}$$

The integration over $\mathbf{y}$ can be written as an integration over the coordinate in the direction of $\mathbf{x}$, which we will denote as z and the vector perpendicular to $\mathbf{x}$, represented by u. We can now rewrite the θ-function as follows

$$\theta\left(\sum_i x_i y_i - t\right) = \theta\left(rz - t\right) \tag{24}$$

This function does not depend on u. The $n-1$-dimensional integration over u, therefore only has a normalized $n-1$ dimensional Gaussian function as integral, and thereby trivially gives the result 1. The probability of a product beyond the threshold is

$$p = \int\int \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{r^2}{2}} \; S_n \; r^{n-1} \theta(rz - t) e^{-\frac{z^2}{2}} \mathrm{d}r \; \mathrm{d}z \tag{25}$$

The integration over z can be performed by recognizing that the θ function is 1 only for $z > \frac{t}{r}$ and 0 otherwise.

$$p = \int \frac{1}{2(\sqrt{2\pi})^{n-1}} S_n \; r^{n-1} \; e^{-\frac{r^2}{2}} \; \left[1 - \mathrm{erf}\left(\frac{t}{\sqrt{2}r}\right)\right] \; \mathrm{d}r \tag{26}$$

Given this probability p, we can calculate the theoretical distribution for the selected subsets:

$$h_k = N \binom{N}{k} p^k (1-p)^{N-k} \tag{27}$$

Fig. 4.7 shows the histograms of the theoretical distribution, expected distribution and the observed distribution for the subset selected by item biopolymer metabolism. The

resampled distribution is slightly more stretched than the theoretical one, which can be attributed to correlations among the experiments that are not considered in the theoretical model. Fig. 4.6 shows that the complexity of the algorithm is significantly decreased, although it is still roughly quadratic.

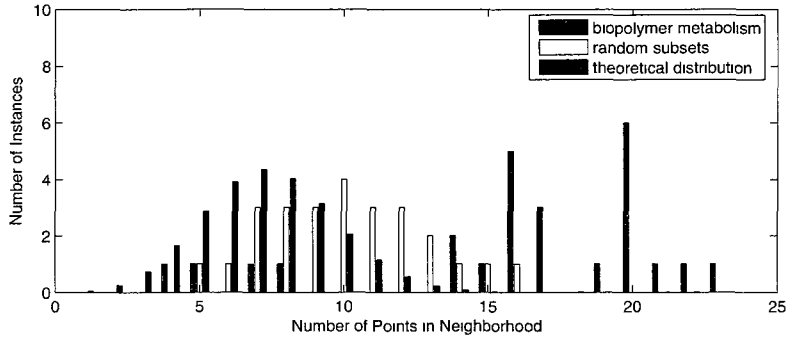


Figure 4.7: Resampled and theoretical histograms for the macromolecule catabolism function.

Using the theoretical model, Algorithm 3 can be modified to Algorithm 4.

Algorithm 4: Distribution-based Algorithm

```
   Data: genes;                                   /* expression values */
   Data: functions
   Result: significance, tailGenes;               /* for each function */
 1 normGenes = normalize(genes);
 2 hist = zeros(1,nPts);                          /* vector of zeros */
 3 foreach f ∈ function do
 4     subset = findPoints(normGenes,f);
 5     foreach x ∈ subset do
 6         dens = NumberOfNeighbors(x);
 7         hist(dens)++;
 8     if NunmberOf(genes) greater than a threshold then
 9         randHist = findTheoreticalHistogram(1, nPts, normGenes);
10     else
11         randHist = findRandomHistogram(1, nPts, normGenes);
12     significance(f) = chiSquaredGoodnessOfFit(hist, randHist);
13     tailGenes(f) = findTailGenes(hist, randHist);
14 return significance, tailGenes
```

4.6. Conclusions

We have introduced an algorithm that permits relating protein functions to gene expression data. It allows us to identify functions that are common in genes that are regulated similarly across the spectrum of two-component systems. Our analysis led to the development of biological hypotheses that suggest further biological experiments.

4.7. Acknowledgments

This study was funded by grant IDM-0415190 from the National Science Foundation and the USDA Cooperative State Research, Education and Extension Service (CSREES) Grant #2006-35604-16675.

CHAPTER 5. EXTRACT IMPORTANT INSTANCES USING HISTOGRAM-BASED VECTOR-ITEM PATTERN MINING ALGORITHMS, AND ITS APPLICATION IN MINING FOR CORE PATTERNS IN STOCK MARKET DATA

In histogram-based vector-item pattern mining algorithms, a vector-item pattern is identified through comparing observed density histogram and expected density histogram of an item set. Take Fig. 4.7 for example, there are several bins in the observed density histogram that locate to the right of the expected density histogram or theoretical model. The instances that contribute to these bins have a higher density than other instances in the item set (in this case the macromolecule catabolism function). To utilize this useful information, we developed a core pattern mining algorithm, which was applied to stock market data.

In this study, we introduce an algorithm that uses stock sector information directly in conjunction with time series subsequences for mining core patterns within the sectors of stock market data. The core patterns within a sector are representative groups of stocks for the sector when it shows coherent behavior. Multiple core patterns may exist in a sector at the same time. In comparison with clustering algorithms, the core patterns are shown to be more stable as the stock price evolves. The proposed algorithm has only one free parameter, for which we provide an empirical choice. We demonstrate the effectiveness of the algorithm through a comparison with the DBScan clustering algorithm using data from the Standard and Poor 500 Index.

5.1. Introduction

Considering the general interest in stock market data, it is not surprising that many stock mining algorithms have been developed. Clustering plays an import role in many stock market applications, such as in stock prediction [92, 71], stock selection [81, 12] and portfolio building [50]. An important question in the clustering of multiple stock time

series is how stable the clusters are as time advances. Collective behavior among stock prices has been observed [113], especially within the same sectors. The similar behavior among stocks within the same sectors explains also why clustering algorithms are likely to result in clusters involving stock of the same sectors [40, 106, 12].

In this study, we introduce an algorithm for identifying core patterns within sectors of stock market data. We use the sector data directly in conjunction with the stock time series for finding the core patterns. In contrast to standard machine learning approaches, which use Boolean labels in form of a class label (supervised) or not at all (unsupervised), our algorithm inherently tests the relationship between the Boolean data and the time series, and returns patterns only when the relationship is significant. For the purpose of this study, a core pattern is a representative group of stocks that show coherent behavior specific to their sector. Multiple core patterns may exist in one sector concurrently. Whether a sector is showing coherent behavior is determined through a modified version of the density histogram technique introduced in [134]. The distribution of stocks within a sector is compared with that of the overall data set, and the statistical significance is calculated. Core patterns of a significant, and hence coherent, sector can be extracted. A detailed discussion is given in Sect. 5.3.

Fig.5.1 illustrates the concepts for an example. The top two panels show the preprocessed stock open values in a time window. In the top left panel, the stocks in black highlight a sector of stocks. A randomly selected sample of the same size as the sector is highlighted using charcoal grey color in the top right panel. Two histograms (one in black, the other in charcoal grey) in the bottom panel summarize the neighboring relationships between stocks in the sector and in the random sample respectively. It can be observed that the stocks in the sector have more neighbors on average than those in the random sample. Furthermore, the stocks that are represented by the circled bins of the black histogram have more neighbors than any of the stocks in the random sample. They form a core pattern for

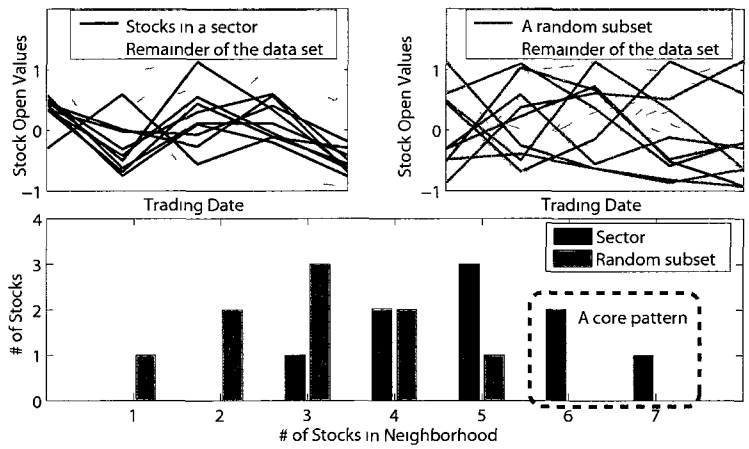


Figure 5.1: The top two panels show the preprocessed stock open values in a time window. The top left panel highlights a sector of stocks. The top right panel shows a randomly selected sample which has the same size with the sector. Two histograms (one in black, the other in charcoal grey) in the bottom panel summarizes the neighboring relationships between stocks in the sector and in the sample respectively. A core pattern is identified by comparing these two histograms.

this sector.

In Fig.5.1, only one random sample is shown. However, in the algorithm, which will be presented shortly, we use multiple random samples and construct a histogram that represents averages over those samples. This histogram acts as a reference for the sector that allows determining if the sector differs from what would be expected by random chance.

The histograms quantify the number of neighbors for a subset of stocks, which can be viewed as a measure of local density of the subset of stocks. We will call them density histograms. We refer to the histogram for a sector as observed density histogram, and the one that was averaged over multiple random samples as expected density histogram.

In this study, the stock data come from the Standard and Poor 500 Index from 07/30/07 to 07/29/08. We follow the suggestion of [64] and use the first derivatives instead of raw values, i.e., we take the differences between successive trading days of each stock. The evaluation shows that our algorithm is more effective than a comparison algorithm at

determining groups of sequences that show coherent behavior in a successive window. The reminder of the section is organized as follows: In Sect. 5.2 we introduce the related work. In Sect. 5.3 we present our algorithm in detail. In Sect. 5.4 we systematically evaluate our algorithm, and also compare with a density-based clustering algorithms, namely DBScan. In Sect. 5.6 we conclude this section.

5.2. Related Work

Many previous studies discuss cluster stability analysis for static data sets [16, 88, 130]. All of them separate the original data set into subsets and then evaluate consistency between clusters found in different subsets. In [16], Ben *et al* present a method for quantitatively assessing the presence of structure in a clustered data set. The method counts the number of similarities between the labels of the objects common to both subsamples. In [88], Tilman *et al* present an approach that evaluates the partitions of a data set of some clustering algorithm, by checking whether similar structures are identified under repeated applications of the clustering algorithm. Volkovicha *et al* present an approach to evaluate the goodness of a cluster by the similarity among the entire cluster and its core [130]. None of these discuss stability in a dynamic data set or time series data set. How stable clusters are expected to be over consecutive time windows depends on factors that influence the time dependence.

For stock data, previous studies [107, 40] have shown that stocks that belong to the same sector tend to move more coherently than stocks from different sectors. From a business point of view this can be expected, since stocks in the same sectors are likely to be influenced by the same external factors, and may also influence each other. That means that coherence that is specific to a sector may be an indication that the stocks are influenced by the same factors and may continue to do so over longer periods of time. We test this coherence using concepts developed in [134]. If the stocks show a statistically significant pattern with respect to their membership in a sector, we expect the core or cores of that

pattern to be stable over time.

The term core pattern has previously been used to describe a subset of a frequent itemset with a certain core ratio [142], which is a different concept usage.

5.3. Algorithm

The proposed algorithm has two main steps. In each time window, the algorithm first identifies significant sectors by comparing the two density histograms,d i.e., for each sector the observed density histogram is compared with the expected density histogram that is constructed from random samples. In a second step, the algorithm extracts core patterns from those significant sectors.

In this study, we use the stream sliding window concepts. Two adjacent sliding windows are used, namely training window and evaluation window, as Fig. 5.2. shows for an example. The algorithm detects significant sectors in the training window, and builds core patterns for those significant ones. Evaluation windows are used for testing how stable those core patterns are. The results of this evaluation are compared with a clustering algorithm. In the reminder of the section, the term "time window" is used to represent either a training window or an evaluation window. Table 5.1 lists the main symbols and their descriptions used in this section.

5.3.1. Outline of the Algorithm

The algorithm iteratively executes the following steps until all training and evaluation windows are processed:

1. **Detecting Significant Sectors in a training window**

    a *Normalization:* Stock data are first normalized within a training window, using a row-wise z-score normalization followed by a column-wise z-score normalization. More details will be given in Sect. 5.3.2.

    b **Significance test:** For each sector, an observed density histogram, together with an expected density histogram, is constructed. Then the statistical significance

Table 5.1: Description of notation

Symbol	Description
N	The ♯ of stocks in the data set
M	The ♯ of sectors
Z	The ♯ of trading days of the data set.
B	Stock sector filter. It is a $N \times M$ matrix. The (i,j) entry ($B_{i,j}$) indicates whether the j^{th} stock belongs to the i^{th} sector or not by using binary value 1 or 0 respectively
S	A subsets of stocks.
$\mathbf{V}$	Stock vectors. $\mathbf{V}_i^{k,L}$ indicates the i^{th} ($i \in [1, N]$) stock vector in time window $[k, (k + L - 1)]$ ($k \in [1, Z]$). Thus it has a dimensionality of L. $V_{i,j}^{k,L}$ indicates j^{th} dimension value of vector $\mathbf{V}_i^{k,L}$. $\mathbf{V}_S^{k,L}$ indicates stock vectors for subset S in the time window and $\mathbf{V}_:^{k,L}$ indicates all stock vectors in the time window.
L	The ♯ of trading days in a training window used for detecting significant sectors. Notice that L is the dimensionality of stock vectors in a training window.
L'	The ♯ of trading days in an evaluation window used for evaluating core patterns. Notice that L' is the dimensionality of stock vectors in an evaluation window.
P^L	Core patterns in a training window. $P_{i,j}^L$ indicates core patterns for sector i in the training window $[j, (j + L - 1)]$. A core pattern is a subset of the indices of all stocks.
$P^{L'}$	Core patterns in an evaluation window. $P_{i,j}^{L'}$ indicates core patterns for sector i in the evaluation window $[j + L, (j + L + L' - 1)]$. A core pattern is a subset of the indices of all stocks.
X, Y	Two example stock vectors
$Sim(\mathbf{X}, \mathbf{Y})$	Cosine similarity between stock vector **X** and stock vector **Y**.
τ	Cosine similarity threshold for two stock vectors.

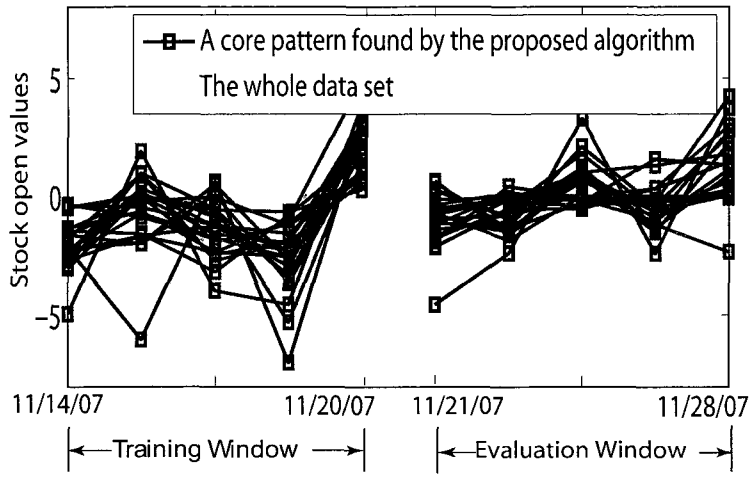


Figure 5.2: Illustration of training window and evaluation window.The training window is from 11/14/07 to 11/20/07, and the evaluation window is from 11/21/07 to 11/28/07. A core pattern, which is indicated in darker curves, is found in the training window.

of the sector is calculated by applying a χ^2 goodness of fit test on the two density histograms. The sectors that are significant at 0.1% level are used to form core patterns.

2. **Forming core patterns**

   a *Extract high-density stocks:* By comparing the two density histograms, stocks that have more neighbors than expected can be identified.

   b **Form core patterns:** Core patterns are extracted from high-density stocks of a sector that is considered to be significant. Details will be discussed in Sect. 5.3.4.

3. **Extract core patterns in the successive evaluation window** Core patterns in the successive evaluation window are extracted with the same procedures as in the training window.

5.3.2. Normalization

In each time window, We first perform a row-wise z-score normalization on each vector, then apply a column-wise z-score normalization on all dimension values of stock vectors along each dimension, as equation (28) and equation (29) show respectively.

$$\begin{aligned} \mathbf{V}_{i,j}^{k,L} &= \frac{\mathbf{V}_{i,j}^{k,L} - mean(\mathbf{V}_{i}^{k,L})}{std(\mathbf{V}_{i}^{k,L})} \\ mean(\mathbf{V}_{i}^{k,L}) &= \frac{\sum_{j=1}^{L} \mathbf{V}_{i,j}^{k,L}}{L} \\ std(\mathbf{V}_{i}^{k,L}) &= \sqrt{\frac{\sum_{j=1}^{L} (\mathbf{V}_{i,j}^{k,L} - mean(\mathbf{V}_{i}^{k,L}))^2}{L}} \end{aligned} \tag{28}$$

$$\mathbf{V}_{i,j}^{k,L} = \frac{\mathbf{V}_{i,j}^{k,L} - \frac{\sum_{i=1}^{N} \mathbf{V}_{i,j}^{k,L}}{N}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} (\mathbf{V}_{i,j}^{k,L} - \frac{\sum_{i=1}^{N} \mathbf{V}_{i,j}^{k,L}}{N})^2}} \tag{29}$$

N is the number of stock vectors in a time window. It is equal to the number of stocks in the data set. $\mathbf{V}_{i}^{k,L}$ is the i^{th} ($i \in [1, N]$) stock vector that starts from the k^{th} trading day and ends at the $(k + L - 1)^{th}$ trading day. It has a dimensionality of L. $\mathbf{V}_{i,j}^{k,L}$ is the j^{th} ($j \in [1, L]$) dimension value of the stock vector $\mathbf{V}_{i}^{k,L}$.

The row-wise z-score normalization was chosen in accordance with the study in [64]. Together with the column-wise z-score normalization, this normalization approach ensures that random data would result in a distribution that closely resembles the normal distribution that is assumed in the theoretical model.

All stock vectors are further projected to a unit hyper-sphere, as (30) shows, such that the cosine similarity between two stock vectors can be calculated efficiently.

$$\begin{aligned} V_{i,j}^{k,L} &= \frac{V_{i,j}^{k,L}}{|V_{i}^{k,L}|} \\ |V_{i}^{k,L}| &= \sqrt{\sum_{j=1}^{L} (V_{i,j}^{k,L})^2} \end{aligned} \tag{30}$$

5.3.3. Significance Test

In a time window, for each sector, an observed density histogram together with an expected density histogram is constructed. The observed histogram is used to summarize the neighboring relationships between stocks in a sector. We consider .a sector as coherent if stocks have more neighbors within the sector than would be expected. The distribution of expected neighbors is quantified through calculating neighbors of stocks in random samples. In other words, if stocks in a sector have more neighbors than those in random samples, we would expect that the sector is showing a coherent behavior.

In this study, cosine similarity is used to determine whether two stock vectors are neighbors of each other. If the cosine similarity of two stock vectors exceeds a threshold τ, they are considered neighbors. The cosine similarity between stock vector $\mathbf{X}$ and stock vector $\mathbf{Y}$ is defined as:

$$Sim(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{L} (X_i Y_i) \tag{31}$$

$\mathbf{X}$ and $\mathbf{Y}$ are two unit vectors both having a dimensionality of L. X_i and Y_i are the i^{th} dimension values for $\mathbf{X}$ and $\mathbf{Y}$ respectively.

Algorithm 5 shows how the observed and expected density histograms are constructed. Function $BuildObservedDensityHist$ builds an observed density histogram, and function $RandomSampling$ builds an expected desity histogram using random sampling. Both functions call a sub-function $BuildDensityHist$, which builds a density histogram for a subset of stock vectors.

The input of the function $BuildDensityHist$ are a subset of stock vectors $\mathbf{V}_S^{k,L}$ and the cosine similarity threshold τ. The output of the function $BuildDensityHist$ is the density histogram $hist$. To build a density histogram for a subset of stock vectors, the number of neighbors for each stock vector in the sector is calculated (Lines 16 to 20), and a contribution of 1 is added to a corresponding bin of the density histogram (Line 21). For

Algorithm 5: Building Density Histograms

Function: *BuildObservedDensityHist* ; /* Building Observed density histogram */
Data: $\mathbf{V}_S^{k,L}$; /* a subset of stock vectors. */
Data: τ; /* Cosine similarity threshold. */
Result: *hist*; /* Observed density histogram */
$hist = BuildDensityHist(\mathbf{V}_S^{k,L}, \tau)$;
return *hist*;
;
;
Function: *RandomSampling* ; /* Building Expected density histogram from Random samples */
Data: $\mathbf{V}_:^{k,L}$; /* All stock vectors */
Data: τ; /* Cosine similarity threshold. */
Data: $|S|$; /* Sample size. */
Data: T; /* Number of samples. */
Result: *hist*; /* Expected density histogram */
$hist = zeros(1, |S|)$;
for *i=1 to T* **do**
    $R = FindARandSet(N, |S|)$; /* Extract a random sample */
    $hist = hist + BuildDensityHist(\mathbf{V}_R^{k,L}, \tau)$;
$hist = hist/T$;
return *hist*;
;
;
Function: *BuildDensityHist* ; /* Building density histogram for a subset of stock vectors */
Data: $\mathbf{V}_S^{k,L}$; /* a subset of stock vectors */
Data: τ; /* Cosine similarity threshold. */
Result: *hist*; /* Density histogram */
$hist = zeros(1, |S|)$;
for *i=1 to* $|S|$ **do**
    neighbors=0;
    for *j=1 to* $|S|$ **do**
        if $Sim(\boldsymbol{V}_{S_i}^{k,L}, \boldsymbol{V}_{S_j}^{k,L}) \geq \tau$ **then**
            neighbors=neighbors+1;
    hist(neighbors)=hist(neighbors)+1;
return *hist*;

example, if vector **X** has n number of neighbors in a subset of stock vectors, then contribute 1 to the n^{th} bin of the density histogram.

To build an observed density histogram, function $BuildObservedDensityHist$ calls directly to function $BuildDensityHist$, with the stock sector under consideration (Line 1), and returns the result of function $BuildDensityHist$, which is the observed density histogram.

To build an expected density histogram, function $RandomSampling$ draws T random samples. A density histogram is built for each sample (Lines 7 to 9). Then the contribution of these random samples are averaged to create the expected density histogram for the sector (Line 10). In this study, T is set to 30.

Fig. 5.3 shows an example of the observed density histogram and the expected density histogram for the energy sector from 07/31/07 to 08/06/07. Notice the difference between the two density histograms in Fig. 5.3. The mean of the observed density histogram (the gray one) is greater than that of the expected density histogram (the black one), which implies that the stock vectors in the energy sector have a higher density than the overall data set.

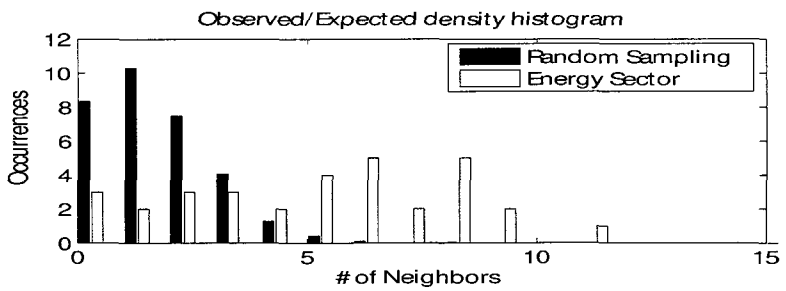


Figure 5.3: Density histograms for energy sector and random sampling from 07/31/07 to 08/06/07. Notice that the observed density histogram (the gray one) is greater than that of the expected density histogram (the black one), which implies that the stock vectors in the energy sector have a higher density than the overall data set.

After the two density histograms are constructed, statistical significance of the sector can be attained by using a χ^2 goodness of fit test on these two histograms. We follow a

conservative rule of thumb that each expected frequency (the height of a bin) should be greater than or equal to 5. Thus, we merge both tails of each histogram until the expected frequencies are at least 5. For intermediate bins that are less than 5, we merge then with adjacent bins until all bins has an expected frequency at least 5. Those sectors that are significant at 0.1% level (p-value of 0.001) are further used to extract core patterns.

5.3.4. Forming Core Patterns

Notice that in Fig. 5.3, there are several bins in the tail of the observed histogram that are located to the right of the expected histogram. The stock vectors that contribute to those bins are the ones that have high-densities in the sector.

Definition 1: High-density stock vectors: The stock vectors that can be extracted from the tail of the observed density histogram until the aggregated density of the expected density histogram is greater than or equal to 1.

We extract those high-density stock vectors from the tail of the observed histogram until the aggregation of bins of the expected histogram are greater than or equals to 1. For instance, in the observed histogram of Fig.5.3, all stock vectors that contribute to the bins that correspond to more than 5 neighbors are extracted.

Once those high-density stock vectors are extracted, core patterns can be identified among them. Quasi-clique detection [1, 93] is a state-of-art technique to mine dense subgraphs in a large graph. It has been used in many applications, including functional prediction of uncharacterized genes [74], mining highly correlated stocks [21], solving the entity resolution (ER) problem [105]. We use quasi-clique mining technique to extract core patterns from the high-density stock vectors: Let $G = (V, E)$ be a graph where V is a set of vertices representing the high-density stock vectors, and E is a set of edges representing neighboring relationships among the high-density stock vectors. There are two existing definitions of *quasi-cliques*. One definition that is given in [1] and some others is: A graph $G = (V, E)$ is $\gamma - clique$ if G is connected and the number of edges of G is greater than

or equal to γ times the number of edges of a complete graph with the same number of vertices. Another definition from [93] is: A graph $G = (V, E)$ is a $\gamma - quasi - clique$ if G is connected, and the number of edges connected to every vertex ($deg^G(v)$) is greater than or equal to the ceiling function of γ times the number of the rest vertices. Unfortunately, the problem of mining $\gamma - clique$ in a graph is a $NP - hard$ problem.

In this study, we use a modified version of the second definition of *quasi-cliques* mentioned above. We adopt an absolute cutoff for $deg^G(v)$, that is, we consider a threshold (*ThN*, which is an integer) for the number of edges that all vertices in a quasi-clique must possess. In other words, we try to find quasi-cliques in which all vertices have at least *ThN* edges (In our study *ThN* equals to 3). Such a modification greatly simplifies the quasi-clique mining procedure. Algorithm 6 depicts an algorithm for mining this type of quasi-cliques, and it has a time complexity of $O(n^2)$ in the worst case. Fig.5.4 illustrate an example of extracted quasi-cliques.

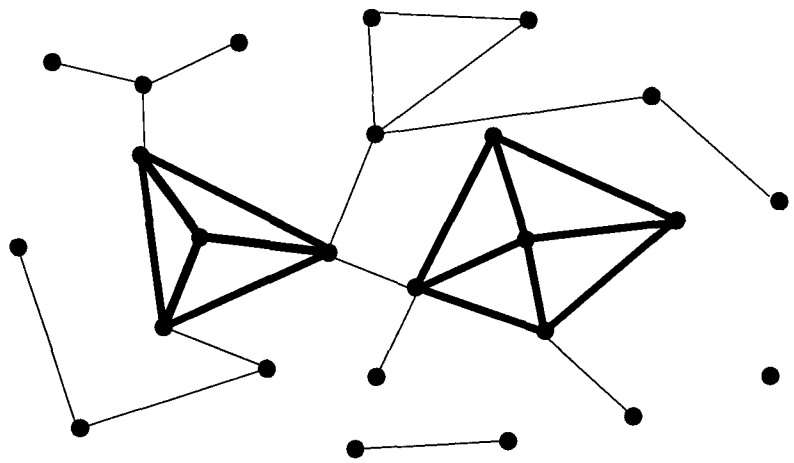


Figure 5.4: Mining quasi-cliques in a graph. The vertices connected by bold edges form quasi-cliques.

Algorithm 6 iteratively deletes the vertices that do not belong to a quasi-clique (i.e., delete the vertices that have less than *ThN* edges), until only those ones that belong to a quasi-clique are left.

The input of algorithm 6 are graph matrix ($graph\_Matrix$) in which entry value

Algorithm 6: Mining Quasi-Clique

Data: $graph\_Matrix$; /* graph matrix in which entry value equals to 1 indicates an edge, 0 other wise. */
Data: ThN; /* threshold for the minimum number of edges. */
Result: $quasi\_cliques$
1 $candidate\_vertices$ = findVertices($graph\_Matrix$,$\geq$,ThN); /* find vertices which have at least ThN edges. */
2 $graph\_Matrix$ = updateMarix($graph\_Matrix$,$candidate\_vertices$); /* update graph matrix by eliminating entries having value 1 that are not connecting vertices in $candidate\_vertices$ */
3 **while** *findVertices*($graph\_Matrix$,$<$,ThN) $\neq$ *null* **do**
4 $candidate\_vertices$ = findVertices($graph\_Matrix$,$\geq$,ThN);
5 $graph\_Matrix$ = updateMarix($graph\_Matrix$,$candidate\_vertices$)
6 $quasi\_cliques$ = $candidate\_vertices$;
7 **return** $quasi\_cliques$;

equaling to 1 indicates an edge and 0 otherwise, and the threshold ThN for the minimum number of edges. The output of algorithm 6 are the quasi cliques identified. Algorithm 6 first calls function $findVertices()$ finds the set of vertices, $candidate\_vertices$, which have at least ThN edges (Line 1). Then it calls a function $updateMarix()$ to updates the graph matrix $graph\_Matrix$ by eliminating entries having a value of 1 but not connecting vertices in $candidate\_vertices$ (Line 2). This step is to remove isolated edges. In the while loop (Lines 3 to 5), algorithm 6 checks if there is any vertices that have less than ThN edges. If there is any, the above two steps, i.e., the calling of functions $findVertices()$ and $updateMarix()$, will be repeated iteratively. Finally, the vertices in the set $candidate\_vertices$ form quasi cliques, and are returned.

Fig. 5.5 illustrates a simple example (*ThN* equals to 2 in this example). The original graph has 6 vertices. Before algorithm 6 proceeds to the while loop, vertices 5 and 6 are deleted after executing *Updatematrix* command. After one iteration of the while loop, vertex 4 is deleted, and a quasi-clique which includes vertices 1, 2 and 3 is found.

We use an array of bit sets to represent adjacency matrix, in which '1' indicates an

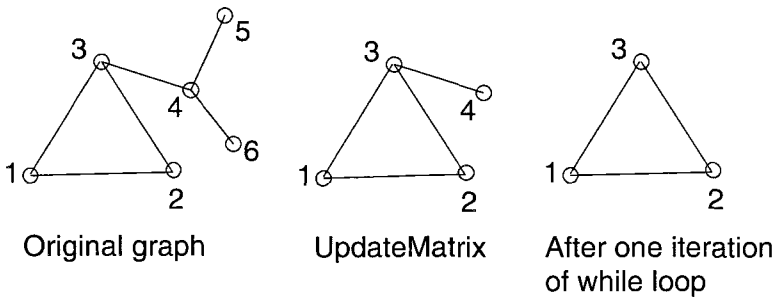


Figure 5.5: Illustration of the procedure to mine a quasi-clique.Nodes 1,2 and 3 form a quasi-clique.

edge existing in correspond vertices. This data structure allows us efficiently count the number of neighbors for a particular vertex, i.e., we can just count the number of "1"s in a bit set ([8] introduces several methods that are O(1) time complexity). In the worst case, the $while$ loop in algorithm 6 only removes 1 vertex in each iteration, which means there are at most $|V|$ iterations in the $while$ loop. In each iterations, we need to count the number of "1"s in at most $|V|$ bit sets. Therefore, algorithm 6 has a worst time complexity $O(n^2)$.

5.3.5. Theoretical Model

In the above proposed algorithm, it is very time consuming to create an expected density histogram, which has to be averaged from multiple density histograms of random samples. A theoretical model is built to approximate the expected density histogram to improve the efficiency. As we already mentioned in Sect. 5.3.2, performing a row-wise z-score normalization then a column-wise z-score normalization ensures that all dimension values of the stock vectors approximately follows a standard normal distribution along each dimension. Fig.5.6 shows a comparison between the distribution of dimension values of one randomly selected dimension of stock vectors with a standard normal distribution. We further assume that all dimensions of stock vectors are independent. This may not be strictly true but, since we take the first derivative of each stock, the dependency between dimensions is mitigated. Thus, the probability density function (PDF) for a stock vector $\mathbf{X}$ can be approximated as:

$$\begin{aligned}\varphi(\mathbf{X}) &= P(x_1, x_2 ... x_L) = \prod_{i=1}^{L} P(x_i) \\ &= \prod_{i=1}^{L} \frac{1}{\sqrt{2\pi}} e^{-\frac{x_i^2}{2}} = (\frac{1}{\sqrt{2\pi}})^L e^{-\frac{\sum_{i=1}^{L} x_i^2}{2}} \\ &= (\frac{1}{\sqrt{2\pi}})^L e^{-\frac{|\mathbf{X}|^2}{2}}\end{aligned} \tag{32}$$

Where x_i is the i^{th} dimension value of the stock vector $\mathbf{X}$, and $|\mathbf{X}|$ is the vector length of $\mathbf{X}$.

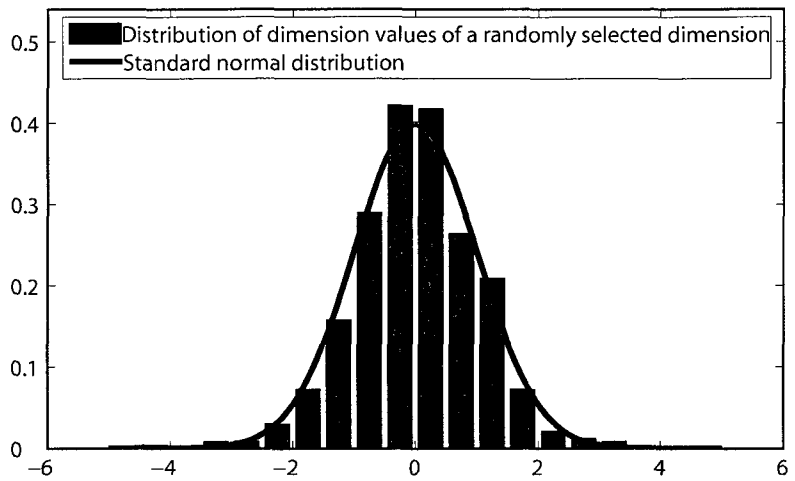


Figure 5.6: Compare the distribution of dimension values of a randomly selected dimension with standard normal distribution.

Under such an assumption, the distribution of a stock vector is independent of the orientation of the stock vector. A random distribution with the above PDF has the property that if the vectors are projected onto a hyper-sphear the resulting distribution will be uniform. Therefore, under this approximation the number of neighbors of a stock vector for a random data set can be estimated by computing the hyper-surface area of the cap, which is segmented by the hype-plane that is defined by threshold τ.

Fig. 5.7 illustrates the concept in two dimensions. In Fig. 5.7, suppose $\mathbf{X}$ is a stock vector in a random sample of size $|S|$. Then the number of neighbors of $\mathbf{X}$ within the random sample can be approximated by $(|S| - 1) * \frac{A_L}{A}$, where A is the hyper-surface area

of the unit hypersphere (in this example it is the circumference of the circle) and A_L is the hyper-surface area of the cap (in this example it is the length of the arc) and L is the dimensionality of the stock vector (in the example L=2).

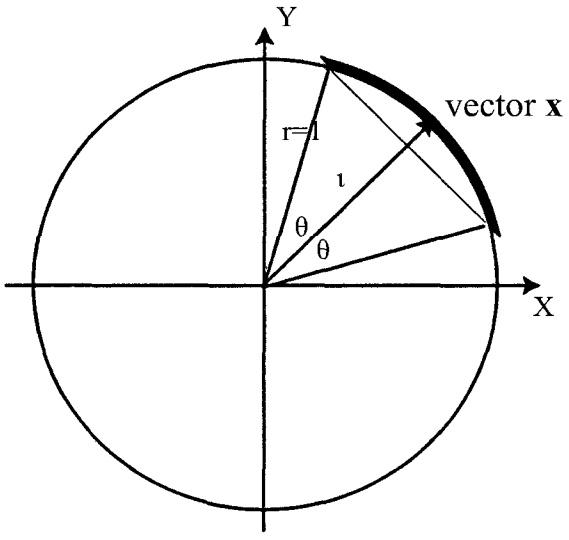


Figure 5.7: The relation between τ and surface area of the cap, which is indicated in bold.

Therefore, we try to find the relationship between the threshold τ and the hyper-surface area of the cap which is segmented by the hype-plane that is defined by threshold τ. The hyper-surface area of the cap divided by that of the hypersphere (i.e., $\frac{A_L}{A}$) is exactly the probability for another stock vector to be neighborhood of stock vector **X**.

The area of the cap can be calculated as follows:

$$\begin{aligned} A_L &= \int_{-a\cos\tau}^{a\cos\tau} \sin^{L-2}(\varphi_1)d\varphi_1 \int_0^{\pi} \sin^{L-3}(\varphi_2)d\varphi_2 \\ &\quad \ldots \int_0^{\pi} \sin(\varphi_{L-2})d\varphi_{L-2} \int_0^{2\pi} d\varphi_{L-1} \\ &= \Omega * \int_{-a\cos\tau}^{a\cos\tau} \sin^{L-2}(\varphi_1)d\varphi_1 \end{aligned} \tag{33}$$

where $\phi_1, \phi_2 \ldots \phi_{L-1}$ are angular coordinates, and

$\Omega = \int_0^{\pi} \sin^{L-3}(\varphi_2)d\varphi_2 \ldots \int_0^{\pi} \sin(\varphi_{L-2})d\varphi_{L-2} \int_0^{2\pi} d\varphi_{L-1}$. So the probability of two

vectors **X** and **Y** to be neighborhood is

$$\lambda = \frac{A_L}{A} = \frac{\Omega * \int_{-a\cos\tau}^{a\cos\tau} \sin^{L-2}(\varphi_1) d\varphi_1}{n * C_L} \tag{34}$$

where $C_L = \begin{cases} \frac{\pi^{(L/2)}}{(L/2)!} & \text{for even L} \\ \frac{2^{(L+1)/2}\pi^{(L-1)/2}}{L!!} & \text{for odd L} \end{cases}$

Equation (34) can be simplified:

$$\lambda = \frac{A_L}{A} = \Omega' * \int_{-a\cos\tau}^{a\cos\tau} \sin^{L-2}(\varphi_1) d\varphi_1 \tag{35}$$

where $\Omega' = \frac{\Omega}{L * C_L}$. To improve performance, λ can be saved to a table TB for different values of L and different thresholds τ. Because τ is of type float, we may need to find the closest representative in the table. From (35), we notice λ is a function of cosine similarity threshold τ and vector dimensionality L.

Fig.5.8 shows a surface plot of λ for different τ and L. Notice that it is exactly the surface plot of table TB.

Notice that λ is less than or equal to 0.5 when cosine similarity threshold $\tau \geq 0$. i.e.:

$$\lambda \leq 0.5 \tag{36}$$

Once λ for a specific threshold τ is calculated using (35), or simply by searching table *TB*, the theoretical model for approximating the expected density histogram is given through a binomial model as follows:

$$h_i = |S| * \begin{pmatrix} (|S|-1) \\ i \end{pmatrix} * \lambda^i * (1-\lambda)^{(|S|-1)-i} \tag{37}$$

h_i is the height of i^{th} bin in the expected density histogram, and $|S|$ is the number

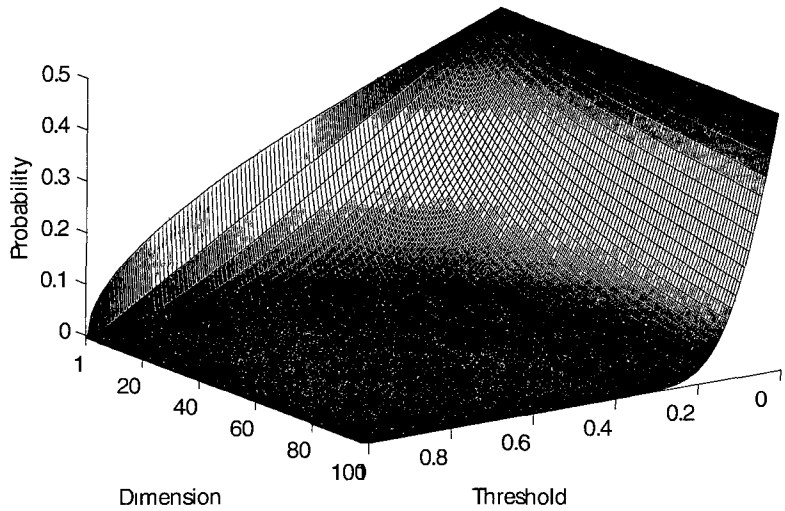


Figure 5.8: Surface plot of table TB: the values of λ for different cosine threshold τ and different vector dimension L

of stock vectors in a sector. Fig. 5.9 shows the result of theoretical model together with random sampling and observed histogram of the same stock vectors with those in Fig. 5.3.

5.3.6. Deriving the Best Choices of the Parameters

In the proposed algorithm with random sampling, there are three parameters, i.e., the number of random samples, the cosine similarity threshold (τ) and the cutoff of p-value for the χ^2 goodness of fit test. The last two parameters are used in the proposed algorithm with using theoretical model. We set the number of random samples to 30 and the cutoff of p-value for the χ^2 goodness of fit test to 0.001 in this study. Thus it leaves us one free parameter, i.e., the cosine similarity threshold (τ) for the proposed algorithm with random sampling as well as with using theoretical model. An universally good choice of the cosine similarity threshold τ can be derived based on the theoretical model:

If there is a small density fluctuation in a large random data set, we assume the observed density histogram for a particular τ, can be expressed as:

$$o_i(\lambda(\tau)) = h_i(\lambda(\tau) + \Delta) \tag{38}$$

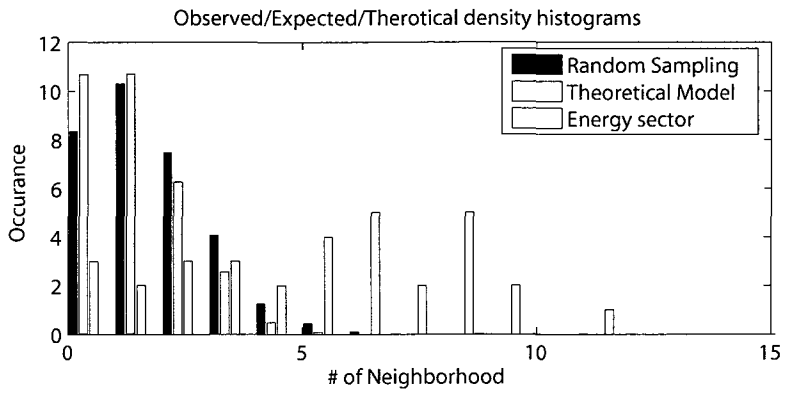


Figure 5.9: Density histogram for energy sector and random sampling and Theoretical model for the time window from 07/31/07 to 08/06/07. Notice that the theoretical model closely approximates the expected density histogram which is created using random sampling

where $o_i(\lambda(\tau))$ is the observed value for the i^{th} bin, $h_i(\lambda(\tau))$ is the expected value for the i^{th} bin, $\lambda(\tau)$ is the corresponding λ (equation (34)) with a particular τ (we omit $'(\tau)'$ part afterwards for simplicity), and Δ indicates a small density fluctuation. This assumption implies that an observation value corresponds to an infinitesimally larger value for its expected value. Fig. 5.10 illustrates the concept. The solid curve indicates the observed density histogram with a small density fluctuation. The dash curve indicates the expected density histogram.

Given a constant small Δ,

$$\begin{aligned} o_i(\lambda) - h_i(\lambda) &= h_i(\lambda + \Delta) - h_i(\lambda) \\ &= \Delta * \tfrac{h_i(\lambda+\Delta) - h_i(\lambda)}{\Delta} \approx \Delta * h_i'(\lambda) \end{aligned} \tag{39}$$

where $h_i'(\lambda)$ is the first derivative of $h_i(\lambda)$ with regard to λ.

The results of the χ^2 goodness of fit test heavily depend on how the cosine similarity threshold τ is chosen. With a suitably chosen τ, a small density fluctuation of a subset can be detected. With this motivation, the cosine similarity threshold τ should be chosen such that the theoretical model is the most sensitive, i.e., the p-value of the χ^2 goodness of

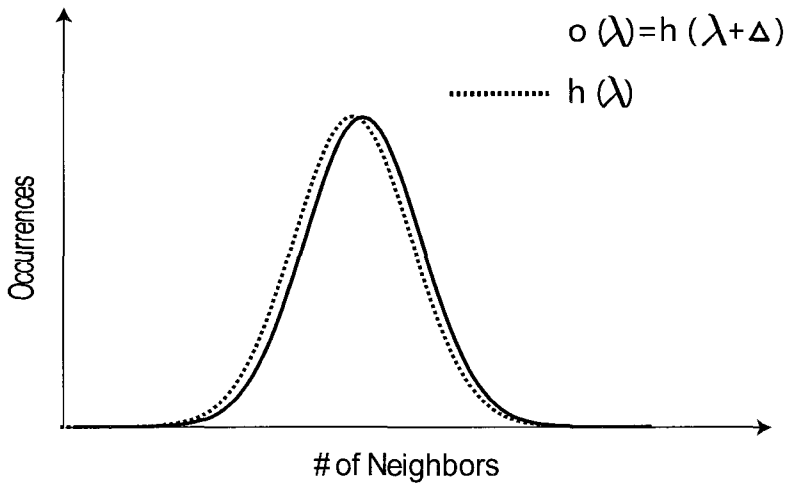


Figure 5.10: The solid curve indicates the observed density histogram with a small density fluctuation. The dash curve indicates the expected density histogram.

fit test is minimized. Thus in the following, through simulation we are going to find a τ that would minimize the p-values of the χ^2 goodness of fit test. To simplify the problem, we first find a λ that minimizes the p-values, and then the corresponding cosine similarity threshold τ can be attained by solving equation (34), or more simply, by searching the table *TB* mentioned in Sect. 5.3.5.

From equation (39) we get:

$$(h_i(\lambda) - o_i(\lambda))^2 \approx \Delta^2 * (h_i'(\lambda))^2 \tag{40}$$

Notice that when we constructing density histograms, we merge both tail bins until the expected frequencies are at least 5 and the intermediate bins for which the heights are less than 5 to their adjacent bins until the expected frequencies are no less than 5.

Let $F(x; |S|-1, \lambda)|_{x_1}^{x_2}$ be the cumulative function for theoretical model (equation 37) in $[x_1, x_2]$ inclusive:

$$F(x; |S|-1, \lambda)|_{x_1}^{x_2} = \sum_{i=x_1}^{x_2} \begin{pmatrix} S-1 \\ i \end{pmatrix} \lambda^i (1-\lambda)^{S-1-i} \tag{41}$$

(1) Suppose the left tail bins whose indices are in $[0, a]$ will be merged to the bin of index a, then a can be expressed as:

$$a = \arg\min_{x}(F(x; |S| - 1, \lambda)|_0^x \geq 5) \tag{42}$$

(2) Similarly, assume the right tail bins whose indices are in $[b, |S|]$ will be merged to the bin for which the index is equal to b, then b can be expressed as:

$$b = \arg\max_{x}(F(x; |S| - 1, \lambda)|_0^x \leq |S| - 5) \tag{43}$$

where b is the index of the bin that the right tail bins are merged to.

(3) For the intermediate bins:

$$d = \arg\min_{x}(F(x; |S| - 1, \lambda)|_c^x \geq 5) \tag{44}$$

where c is the index of an intermediate bin, d is the index of the bin that all the bins for which the indices are within $[c, d]$ are merged to. Notice that for an intermediate bin whose height is equal to or greater than 5, $c = d$.

Solving the first derivatives of the cumulative functions in equation (41) with respect to p, we can get $F'(x; |S| - 1, \lambda)|_{x_1}^{x_2}$ (details in Appendix 7.8), where $F'(x; |S| - 1, \lambda)|_{x_1}^{x_2}$ is the first derivative of the cumulative function $F(x; |S| - 1, \lambda)|_{x_1}^{x_2}$ with respect to λ, x_1 is equal to 0 for equations (42) and (43), and c for equation (44).

To be consistent, we use $F(x; |S| - 1, \lambda)|_i^i$ to represent the expected height of the i^{th} bin after the merging process. Therefore, Pearson's χ^2 test after merging process can be expressed as:

$$\chi^2 = \sum_{i=0}^{|S|-1} \frac{(O_i - E_i)^2}{E_i} \qquad substituting \quad in \quad equation(39), \quad we \quad get:$$
$$\approx \Delta^2 * \sum_{i=0}^{|S|-1} I_{\{F(x;|S|-1,\lambda)|_i^i>0\}}(F(x;|S|-1,\lambda)|_i^i) * \frac{(F'(x;|S|-1,\lambda)|_i^i)^2}{F(x;|S|-1,\lambda)|_i^i} \tag{45}$$

where $I_{\{F(x,|S|-1,\lambda)|_i^i>0\}}$ is an indicator function, i.e., when $F(x;|S|-1,\lambda)|_i^i > 0$,

$I_{\{F(x;|S|-1,\lambda)|_i^i>0\}}(F(x;|S|-1,\lambda)|_i^i) = 1$, otherwise

$I_{\{F(x;|S|-1,\lambda)|_i^i>0\}}(F(x;|S|-1,\lambda)|_i^i) = 0$.

Accordingly, the degree of freedom can be expressed as:

$$DFs = -1 + \sum_{i=0}^{|S|-1} I_{\{F(x,|S|-1,\lambda)|_i^i>0\}}(F(x;|S|-1,\lambda)|_i^i) \tag{46}$$

Algorithm 7 gives a straightforward algorithm to calculate the χ^2 values and the degree of freedom (DFs) for a theoretical model with $|S|$ and p. In algorithm 7, χ^2 values and DFs are calculated by summing up the contributions from merged left tail bin, merged right tail bin and intermediate bins separately. From line 1 to line 11, algorithm 7 calculates the DFs and χ^2 values contributions from left tail bins. A variable a records the index of the merged left tail bin. Similarly, DFs and χ^2 values contributions from right tail bins are calculated from line 12 to line 18. A variable b keeps track of the index of the merged right tail bin. If both tails are merged to the same bin (i.e., $a = b$ in line 19), χ^2 value as well as DFs is set to 0 and algorithm 7 returns. To process the intermediate bins, algorithm 7 proceeds to merge intermediate bins, beginning from the one right after the merged left tail bin (i.e. $a + 1$). Once the merged bin's height is greater than or equals to 5 (line 30 to line 34), its contributions to χ^2 values and DFs are added sequentially (line 40 to line 42). However it is possible that the last few bins are merged but the accumulated height is still less than 5, in this case we need to merge these intermediate bins with the bin that is merged from right tail bins (line 35 to line 39).

Algorithm 7: Calculate χ^2 values and DFs

Data: $|S|$
Data: λ
Result: $ChisquareValue$
Result: DF
$ChisquareValue$=0;
DF=0;
/*process left tail bins*/ ;
a=0 /* in equation (42) */,
acc=0 /* a counter */ ;
while $acc<5$ **do**
    acc+=h_a;
    a++;
a−−;
$ChisquareValue$+=$\Delta^2 * \frac{(F'(x,|S|-1,\lambda)|_0^a)^2}{F(x,|S|-1,\lambda)|_0^a}$;
DF++;
/*process right tail bins*/ ;
b=$|S|-1$ /* in equation (43) */;
acc=0;
while $acc<5$ && $b \geq a$ **do**
    acc+=h_b;
    b−−;
b++;
if $b==a$ **then**
    $ChisquareValue$=0 /*merged to one bin*/;
    DF=0; **return** *ChisquareValue and DF*,
else
    $hightertail$=$\Delta^2 * \frac{(F'(x,|S|-1,\lambda)|_b^{|S|-1})^2}{F(x,|S|-1,\lambda)|_b^{|S|-1}}$ /*save the χ^2 value of the merged higher tail*/;
    $ChisquareValue$+=$hightertail$
DF++;
/*process intermediate bins*/ ;
i=a+1;
lasti=i /* record last i*/ ;
while $i<b$ **do**
    acc=0;
    while $acc<5$ && $i<b$ **do**
        acc+=h_i;
        i++;
    i−−;
    if $i==b\text{-}1$ **then**
        /*merging to the merged right tail bin*/;
        $ChisquareValue$-=$hightertail$;
        $ChisquareValue$+=$\Delta^2 * \frac{(F'(x,|S|-1,\lambda)|_{lasti}^{|S|-1})^2}{F(x,|S|-1,\lambda)|_{lasti}^{|S|-1}}$,
        break;
    else
        $ChisquareValue$+=$\Delta^2 * \frac{(F'(x,|S|-1,\lambda)|_{lasti}^{i})^2}{F(x,|S|-1,\lambda)|_{lasti}^{i}}$;
    DF++;
    lasti=i;
DF − − /*the degree of freedom is equal to the number of merged bins minus 1*/ ;
return *ChisquareValue and DF*;

We obtain the best λ which minimizes the p-values of the χ^2 goodness of fit test for various $|S|$ through a simulation. The solid curve in the Fig. 5.11 shows the textitλ that would yield minimum p-values for different $|S|$. To simplify, we fit the curve using $1/(|S-1|)$ as shown in dotted curve in Fig. 5.11.

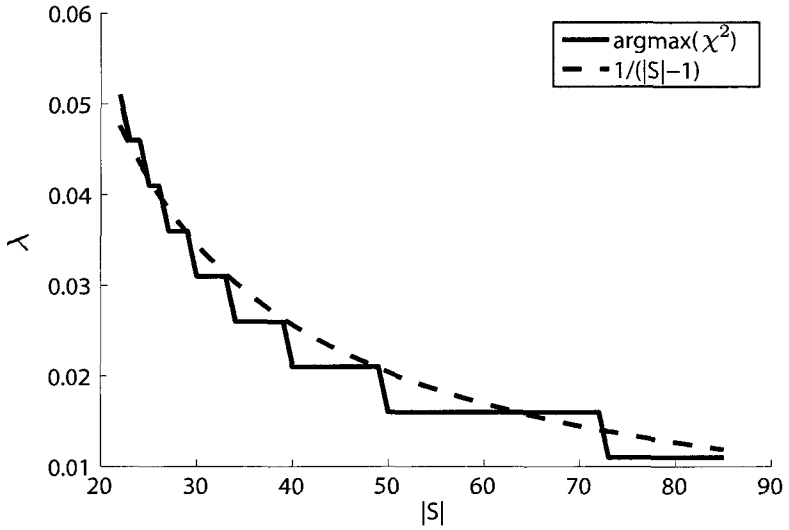


Figure 5.11: The solid curve shows the q that would yield minimum $p-values$ for different $|S|$, The dotted one illustes a fit with $\lambda = 1/(|S|-1)$

Thus, when we constructing the theoretical model for the expected density histogram of a sector of size $|S|$, we first calculate the λ ($textit\lambda = 1/(|S|-1)$). The corresponding cosine similarity threshold τ can be attained by solving equation (34), or more simply, by searching the table *TB* mentioned in Sect. 5.3.5.

5.3.7. Summary of the Algorithm

The proposed algorithm is summarized in algorithm 8. The input of algorithm 8 includes $\mathbf{V}_{:}^{k,L}$ which is an array of all stock vectors that start from k^{th} trading day and end at $(k+L-1)^{th}$ trading day, stock sector filter matrix B, and the algorithm switch $AlgoSwitch$ for which 0 indicates using theoretical model to construct expected density histogram, or 1 using random sampling. $\mathbf{V}_{:}^{k,L}$ is first normalized using (28) and (29) (Line 2). Then for each sector, the cosine similarity threshold is first calculated, and the observed

Algorithm 8: algorithm summary

```
Data: V_:^{k,L};          /* Stock vectors in a time window.  */
Data: B;                  /* Stock sector filter matrix.      */
Data: AlgoSwitch;         /* Algorithm switch, 0 for theoretical
    and 1 for random sampling */
Result: CorePatterns
1  CorePatterns = [];
2  V_:^{k,L} = normalize(V_:^{k,L});
3  foreach sector i do
4      S = σ(B_i);  /* Select the subset of stocks in sector i.
       */
5      λ = 1/(|S| − 1);
6      τ = LookUpTB(λ);
7      observed_hist = BuildObservedDensityHist(V_S^{k,L}, τ);
8      if AlgoSwitch == 0 then
9          expected_hist = BuildTheoreticalModel(|S|);
10     else
11         T = 30;
12         expected_hist = RandomSampling(V_:^{k,L}, τ, |S|, T);
13     signif = SignificanceTest(expected_hist, observed_hist);
14     if signif ⩽ 0.001 then
15         CorePatterns = AddNewCorePattern(CorePatterns,
           ExtractCorePatterns(expected_hist,observed_hist));
16 return CorePatterns;
```

histogram is built (Line 4 to 7). *AlgoSwitch* is used to switch between theoretical model and random sampling. When *AlgoSwitch* is equal to 0, the algorithm uses the theoretical model from equation (37) (Line 9), otherwise it uses random sampling with 30 random samples to build the expected histogram (lines 11 to 12). Then the statistical significance of each sector is calculated by applying a χ^2 goodness-of-fit test on the two density histograms (Line 13). Core patterns are extracted from those that are significant (Lines 15 to 16).

5.4. Evaluation

The algorithm is tested on *S&P500* stock database from 02/01/06 to 01/31/2007, 07/30/2007 to 07/29/2008, and from 08/21/2009 to 08/20/2010. With training window size of L, evaluation window size of L',and the number of trading days after taking derivative

equaling to $Z-1$, the algorithm is tested $(Z-L-L')$ times. For each test, if the algorithm identifies that a sector is coherent (significant) in a training window, core patterns are built separately in the training window and its corresponding evaluation window. Table 5.2 summarizes the fraction of coherent windows for each sector.

Table 5.2: Fraction of coherence windows for each sector

	02/01/06 to 01/31/2007		07/30/2007 to 07/29/2008		08/21/2009 to 08/20/2010	
Sector	$L=5$	$L=10$	$L=5$	$L=10$	$L=5$	$L=10$
Industrials	0.38	0.56	0.40	0.54	0.37	0.44
HealthCare	0.49	0.57	0.47	0.65	0.47	0.56
Consumer Discretionary	0.75	0.88	0.90	0.96	0.85	0.79
Financials	0.93	0.96	0.98	0.99	0.91	0.93
Information Technology	0.84	0.92	0.91	0.89	0.84	0.88
Utilities	0.92	0.87	0.83	0.95	0.89	0.89
Materials	0.56	0.69	0.37	0.62	0.32	0.54
Consumer Staples	0.27	0.44	0.40	0.55	0.37	0.33
Energy	0.96	0.97	0.91	0.99	0.91	0.94

It can be seen that the Consumer Discretionary sector, Financials sector, Information Technology sector, Utilities sector and Energy sector show coherent behavior in most training windows.

The stability of core patterns can be evaluated by examining whether the stock vectors in the core patterns found in a training window can still form core patterns in the consecutive evaluation window. We apply sensitivity, specificity and accuracy measurements to evaluate the stability of a core pattern. Notice that if there is more than one core pattern, we aggregate them for the purpose of simplifying the evaluation. Assume $P_{i,j}^{L}$ is a subset of stock vectors that forms a core pattern in the training window $[j, j+L-1]$ in sector i, and $P_{i,j}^{L'}$ is a subset of stock vectors that forms a core pattern in the evaluation window $[j+L, j+L+L'-1]$ in the same sector. Then the sensitivity, specificity and

accuracy for the core pattern $P_{i,j}^{L}$ in the training window can be calculated as:

$$sens = \frac{TP}{TP+FN} = \frac{|P_{i,j}^{L} \cap P_{i,j}^{L'}|}{|P_{i,j}^{L} \cap P_{i,j}^{L'}| + |P_{i,j}^{L'} - P_{i,j}^{L}|} \tag{47}$$

$$spec = \frac{TN}{TN+FP} = \frac{|\overline{P_{i,j}^{L}} \cap \overline{P_{i,j}^{L'}}|}{|\overline{P_{i,j}^{L}} \cap \overline{P_{i,j}^{L'}}| + |\overline{P_{i,j}^{L'}} - \overline{P_{i,j}^{L}}|} \tag{48}$$

$$\begin{aligned} acc &= \frac{TP+TN}{TP+TN+FP+FN} = \frac{|P_{i,j}^{L} \cap P_{i,j}^{L'}| + |\overline{P_{i,j}^{L}} \cap \overline{P_{i,j}^{L'}}|}{|S|} \\ S &= \sigma(B_i) \end{aligned} \tag{49}$$

B_i is the i^{th} row in the stock sector filter matrix B, and $\sigma(B_i)$ is a select function that selects stocks that are in the i^{th} sector. Thus S is the subset of stocks in the i^{th} sector. $\overline{P_{i,j}^{L}} = S - P_{i,j}^{L}$, and $\overline{P_{i,j}^{L'}} = S - P_{i,j}^{L'}$.

Fig. 5.12 illustrates an example. There are 5 stocks in this sector. In the training window, stocks 76, 13 and 7 form a core pattern, i.e.,$P_{i,j}^{L} = \{76, 13, 7\}$. While in the evaluation window, stocks 119, 7 and 13 form a core pattern, i.e., $P_{i,j}^{L'} = \{119, 7, 13\}$. Then $\overline{P_{i,j}^{L}} = \{119, 273\}$ and $\overline{P_{i,j}^{L'}} = \{76, 273\}$. Thus sensitivity, specificity and accuracy for this sector are 0.67, 0.5 and 0.6 respectively.

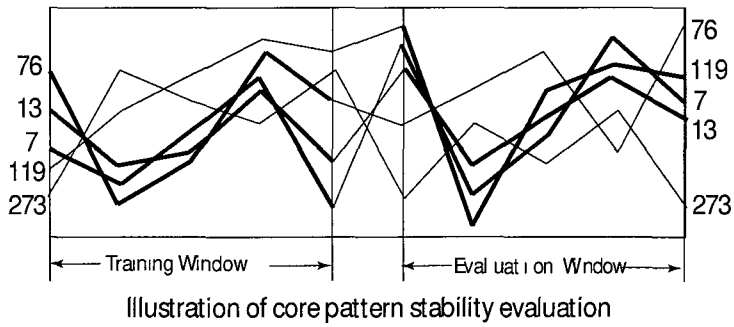


Figure 5.12: Illustration of a sector which has 5 stocks. In training window, stocks 76, 13 and 7 form a core pattern, while in evaluation window, stocks 119, 7 and 13 form a core pattern

The way to calculate these measures on all sectors for all time windows is as follows: Suppose that the set of training windows in which the i^{th} sector is coherent is $SigD^{iL}$. We label a training window using the start trading date of the training window. Thus $SigD^{iL}$ is a set of start trading dates of the training windows under consideration. Let $SigD^{iL}_j$ indicate the j^{th} ($j \in [1, |SigD^{iL}|]$) training window in $SigD^{iL}$. Then sensitivity, specificity and accuracy for all sectors in all time windows can be calculated as equations (50), (51) and (52) show.

$$sens = \frac{TP}{TP+FN} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{|SigD^{iL}|}|P^{L}_{i,SigD^{iL}_j} \cap P^{L'}_{i,SigD^{iL}_j}|}{\sum_{i=1}^{M}\sum_{j=1}^{|SigD^{iL}|}(|P^{L}_{i,SigD^{iL}_j} \cap P^{L'}_{i,SigD^{iL}_j}| + |P^{L'}_{i,SigD^{iL}_j} - P^{L}_{i,SigD^{iL}_j}|)} \tag{50}$$

$$spec = \frac{TN}{TN+FP} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{|SigD^{iL}|}|\overline{P^{L}_{i,SigD^{iL}_j}} \cap \overline{P^{L'}_{i,SigD^{iL}_j}}|}{\sum_{i=1}^{M}\sum_{j=1}^{|SigD^{iL}|}(|\overline{P^{L}_{i,SigD^{iL}_j}} \cap \overline{P^{L'}_{i,SigD^{iL}_j}}| + |\overline{P^{L'}_{i,SigD^{iL}_j}} - \overline{P^{L}_{i,SigD^{iL}_j}}|)} \tag{51}$$

$$acc = \frac{TP+TN}{TP+TN+FP+FN} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{|SigD^{iL}|}(|P^{L}_{i,SigD^{iL}_j} \cap P^{L'}_{i,SigD^{iL}_j}| + |\overline{P^{L}_{i,SigD^{iL}_j}} \cap \overline{P^{L'}_{i,SigD^{iL}_j}}|)}{\sum_{i=1}^{M}\sum_{j=1}^{|SigD^{iL}|}|\sigma(B_i)|} \tag{52}$$

$|SigD^{iL}|$ is the number of training windows in which the i^{th} sector is coherent; $P^{L}_{i,SigD^{iL}_j}$ and $P^{L'}_{i,SigD^{iL}_j}$ are core patterns of the i^{th} sector in the training window that starts from the $(SigD^{iL}_j)^{th}$ day and the evaluation window that starts from the $(SigD^{iL}_j + L)^{th}$ day respectively. $|\sigma(B_i)|$ is the number of stock vectors in sector i.

5.5. Comparison with Clustering Algorithms

In this study, two prominent clustering algorithms, DBScan and K-means, are used to compare with the proposed algorithm.

5.5.1. Comparison with DBScan

DBScan [56] is a well-known density-based clustering algorithm. It has two parameters: minimum points (*minPts*) and epsilon (*eps*).

We perform DBScan in the same stock data sets. Since stock vectors change over each time window, it is very inefficient to use a k-dist graph [56] to estimate proper values for *eps* and *minPts* in each time window. Therefore, we test DBScan with *minPts* being equal to 3,4,5,...,11 in each time window. A corresponding *eps* is chosen such that the average sizes of clusters found by DBScan algorithm are close to the average size of core patterns.

$$|C| = \sum_{i=1}^{M} |\sigma(B_i)| * \frac{\sum_{i=1}^{M} \sum_{j=1}^{|SigD^{iL}|} P^{L}_{i,SigD^{iL}_{j}}}{\sum_{i=1}^{M} \sum_{j=1}^{|SigD^{iL}|} |\sigma(B_i)|} \tag{53}$$

$$|C'| = \sum_{i=1}^{M} |\sigma(B_i)| * \frac{\sum_{i=1}^{M} \sum_{j=1}^{|SigD^{iL}|} P^{L'}_{i,SigD^{iL}_{j}}}{\sum_{i=1}^{M} \sum_{j=1}^{|SigD^{iL}|} |\sigma(B_i)|} \tag{54}$$

$|\sigma(B_i)|$ is the number of stock vectors in sector i. $|C|$ and $|C'|$ are the desired cluster sizes in a training window and an evaluation window. To achieve the desired cluster sizes, parameter *eps* can be attained by an interval halving method. Choosing *eps* in such a way is to make sure that both algorithms capture a similar number of high-density stocks.

DBScan is performed in the same stock data set, with $(L = 5, L' = 5)$, $(L = 10, L' = 5)$ and $(L = 10, L' = 10)$. Sensitivity, specificity and accuracy are calculated accordingly. Notice that if we find there are more than one cluster found by DBScan in a training window or an evaluation window, we aggregate them to one cluster, in order to be fair in the comparison with the proposed algorithm since core patterns are also aggregated.

5.5.2. Comparison with K-means

We also compare the proposed algorithm with a partitioning clustering algorithm, namely K-means algorithm. In K-means algorithm, K is chosen such that the average size of each cluster is equal to the average size of the aggregated core patterns in the training windows:

$$K = \left\lfloor \frac{\sum_{i=1}^{M} \sum_{j=1}^{|SigD^{iL}|} |\sigma(B_i)|}{\sum_{i=1}^{M} \sum_{j=1}^{|SigD^{iL}|} |P^{L}_{i,SigD^{iL}_j}|} \right\rfloor \tag{55}$$

where $\lfloor \rfloor$ is floor operation, M is the number of sectors, $|\sigma(B_i)|$ is the number of stock vectors in sector i, and $P^{L}_{i,SigD^{iL}_j}$ is the set of core patterns of the i^{th} sector in the training window that starts from the $(SigD^{iL}_j)^{th}$ day. K is appled to both a training window and an evaluation window. However, there is no strict correspondence of K-means clusters between two consecutive windows. For example, after runing K-means algorithm, we get 5 clusters in a training window $(C_1, C_2 ... C_5)$, as well as in an evaluation window $(C'_1, C'_2 ... C'_5)$. We do not know to which cluster C_1 in the training window corresponds in the evaluation window. One way to achieve that goal is to maximize $\sum_{i=1}^{K} C_i \cap C''_i$, where C_i is the i^{th} cluster in the training window and C''_i ($C''_i \subset \{C'_1, C'_2 ... C'_5\}$) is C_i 's hypothetical corresponding cluster in the evaluation window. To solve this problem, a bruteforce approach may work but with a complexity of $O(n^n)$. In this study, we follows a heuristic approach depicted in algorithm 9.

The input of algorithm 9 are clusters (C) in a training window and clusters (C') in an evaluation window. The output of algorithm 9 is the matched clusters (C'') in the evaluation window. In algorithm 9, an empty matrix *OvM* which is used for counting the matching stock vectors of each clusters between two partitions is first constructed. Entry (i,j) in matrix *OvM* indicates the number of matching stock vectors between

Algorithm 9: Heuristic approach for corresponding K-means clusters

Data: C; /* clusters in training window. */
Data: C'; /* clusters in evaluation window. */
Result: C''

```
1 OvM=BuildOverlapMatrix(|C|,|C'|); /* with C column-wise and C' row-wise. */
2 C''=[ ];
3 while OvM has a non-zero entry do
4     [i,j] = findMax(OvM); /* find the indices (i,j) of the cell which has the maximum value. */
5     C''_i = C'_j;
6     ResetEdge(OvM,i,j); /* set all cells in row i and column j of OvM to 0. */
7 return C'';
```

cluster C_i' in an evaluation window and cluster C_j in a training window. The algorithm iteratively finds two clusters, one in the training window and the other in the evaluation window, that have maximum matches, until all clusters in both windows are matched. Function $BuildOverlapMatrix()$ in the 1^{st} line is to build the matrix *OvM*. Function $findMax(OvM)$ in the 4^{th} line is to extract the row and column indices of the entry which has the maximum value. Function $ResetEdge(OvM, i, j)$ in the 6^{th} line sets all entries in the i^{th} row and the j^{th} column of OvM to 0.

5.5.3. Comparison Results

The comparisons of sensitivity, specificity and accuracy are shown in Fig. 5.13,Fig. 5.14 and Fig. 5.15. It can be seen that the proposed algorithm, using both random sampling and the theoretical model, outperforms both DBScan algorithm and the K-means algorithm. The proposed algorithm with theoretical model is comparable with the proposed algorithm with random sampling. The accuracies and specificities of the proposed algorithms are not significantly different among $(L = 5, L' = 5)$, $(L = 10, L' = 5)$ and $(L = 10, L' = 10)$. However, the sensitivity values for $(L = 5, L' = 5)$ for the proposed algorithms are relatively worse than those for $(L = 10, L' = 5)$ and $(L = 10, L' = 10)$.

Fig. 5.16 shows the comparison of the performance among the algorithms. The run times for the algorithms are averaged among the three data sets, and the run times for DBScan with different parameters are also averaged. Notice that the performance of the proposed algorithm with random sampling depends heavily on the number of samples used to create an expected density histogram. Although we only use 30 samples in this study, its performance is much worse than the other algorithms. However, using the theoretical model in the proposed algorithm achieves comparable performance with DBScan or K-means algorithms.

Summarizing, the core patterns extracted by the proposed algorithms with both random sampling and with theoretical model are more stable than the clusters found by DBScan algorithm or K-means algorithm. Furthermore, The proposed algorithm with theoretical model has a comparable performance with DBScan and K-means algorithms.

5.6. Conclusions

In this study, we presented an algorithm for mining core patterns in stock market data. The algorithm identifies whether a stock sector currently shows coherent behavior. When coherent behavior of a stock sector is detected, core patterns are extracted. The core patterns are more stable than clusters found by some clustering algorithms such as DBScan and K-means. Such core patterns can be used in many existing stock mining algorithms that would otherwise use clusters. Through comparing with DBScan and K-means, we show the effectiveness of the proposed algorithm.

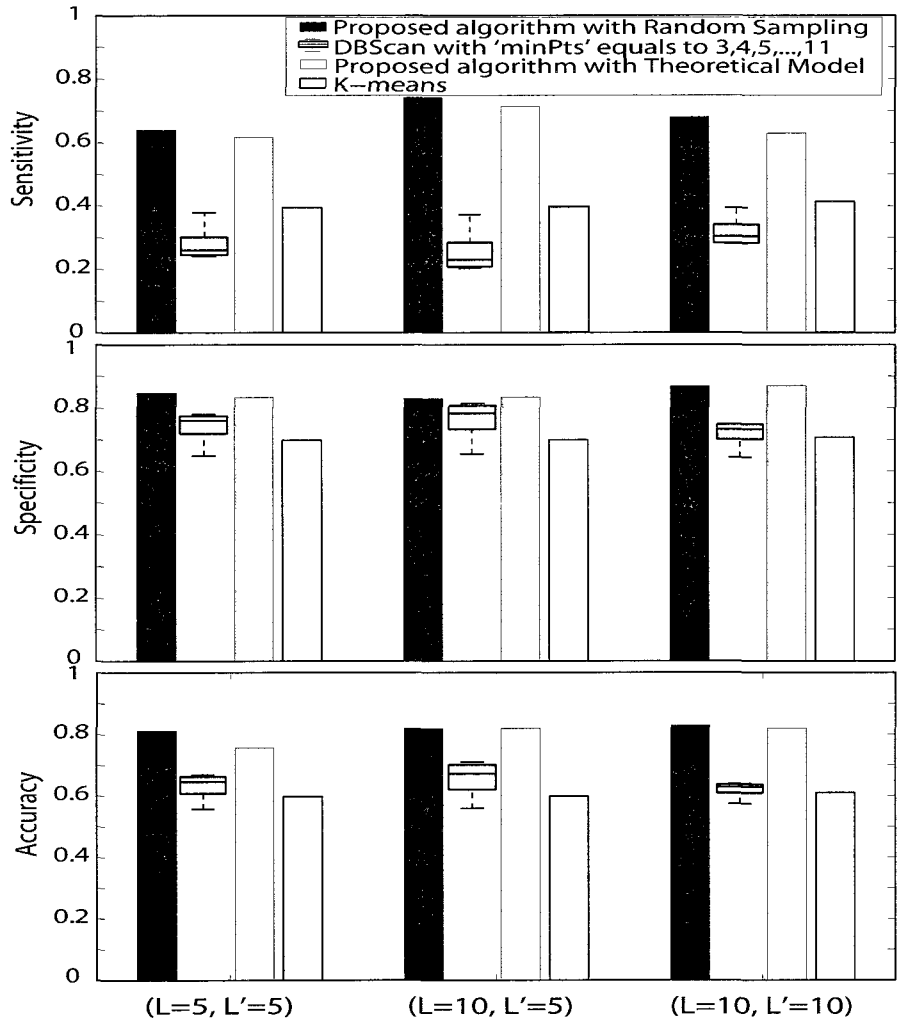


Figure 5.13: The comparisons of sensitivity, specificity and accuracy among algorithms for data set 02/01/06 to 01/31/2007. It can be seen that the proposed algorithm, using both random sampling and the theoretical model, outperforms both DBScan algorithm and K-means algorithm. The proposed algorithm with theoretical model is comparable with the proposed algorithm with random sampling.

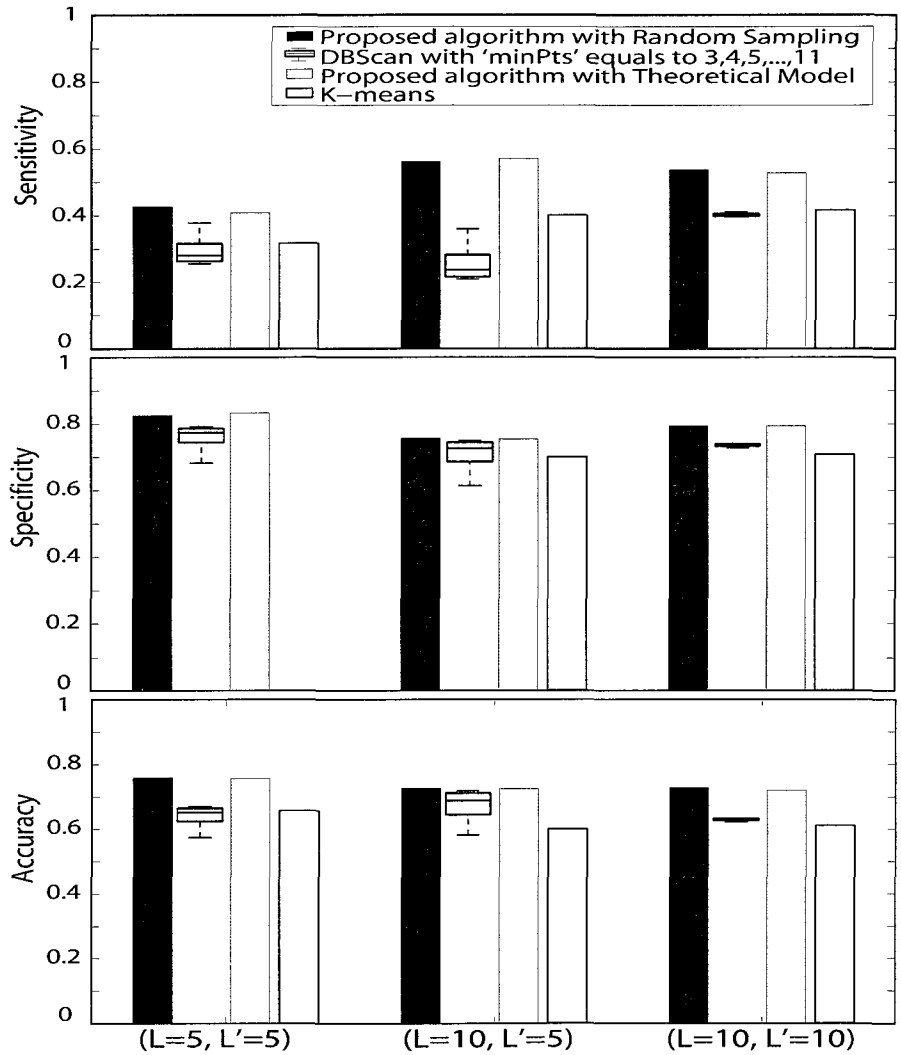


Figure 5.14: The comparisons of sensitivity, specificity and accuracy among algorithms for data set 07/30/2007 to 07/29/2008. It can be seen that the proposed algorithm, using both random sampling and the theoretical model, outperforms both DBScan algorithm and K-means algorithm. The proposed algorithm with theoretical model is comparable with the proposed algorithm with random sampling.

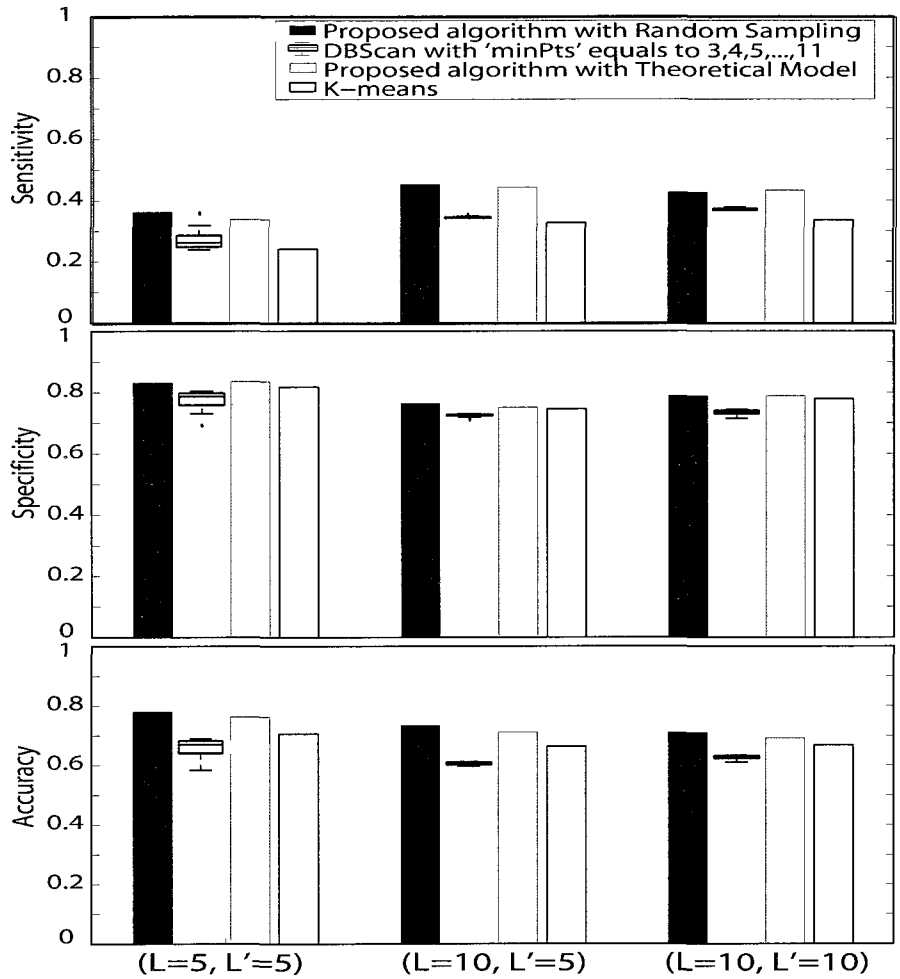


Figure 5.15: The comparisons of sensitivity, specificity and accuracy among algorithms for data set 08/21/2009 to 08/20/2010. It can be seen that the proposed algorithm, using both random sampling and the theoretical model, outperforms both DBScan algorithm and K-means algorithm. The proposed algorithm with theoretical model is comparable with the proposed algorithm with random sampling.

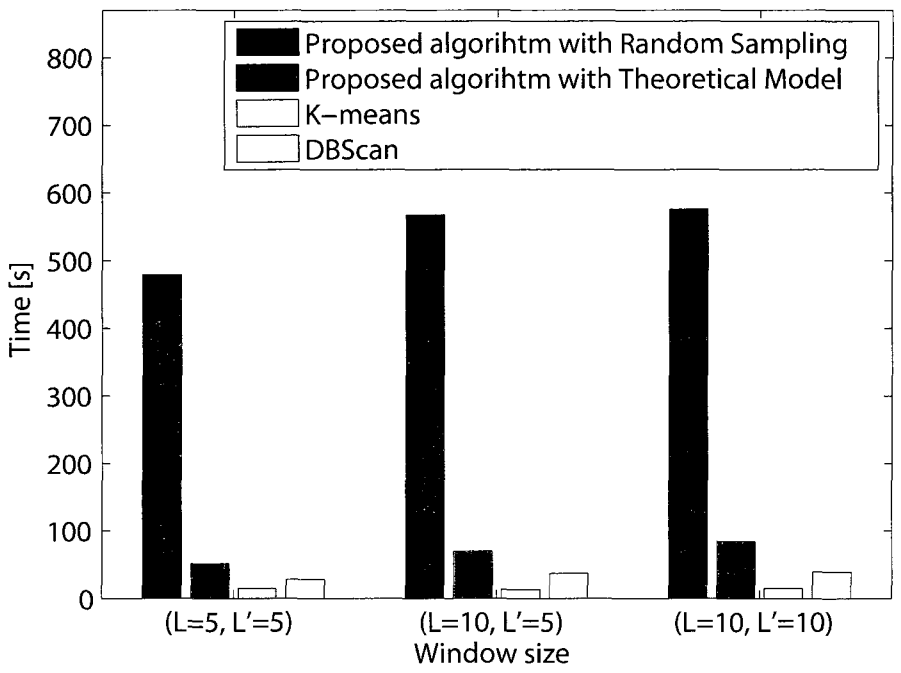


Figure 5.16: The run times for the algorithms are averaged among the three data sets, and the run times for DBScan with different parameters are also averaged.

CHAPTER 6. FINDING IMPORTANT DESIGN REGIONS IN A COMBINATORIAL DESIGN

In previous histogram-based vector-item pattern mining algorithms, bins in density histograms with a height of less than 5 are combined with adjacent bins till the height of the merged bin is greater than 5. This process in some cases causes the previous histogram-based vector-item pattern mining algorithms unable to handle small data set. Furthermore, another drawback of the previous histogram-based vector-item pattern mining algorithms is that they cannot rank the significance level of these vector-item patterns. We tackled these problems by substituting the "χ^2 goodness of fit test" with "Effect size analysis", which is more suitable in some cases, especially in those that need multiple-hypothesis testing.

In this project we collaborated with the NDSU Center for Nanoscale Science and Engineering. Our goal was to identify important design regions in a combinatorial design that has one or multiple responses. In this study, design regions, for which definitions will be given shortly, serve as item information. Experiments with multiple responses serve as vectors. Using the framework of vector-item pattern mining algorithms, we developed an algorithm for identifying important design regions in a combinatorial design. The importance of a design region is measured through a model that incorporates the effect size and confidence intervals of the design region. The important design regions will not only lead to low variances of multiple responses, but optimize them in the same step. The algorithm is tested on two published coating data sets. Experiment results show that the proposed algorithm identifies important design regions that match interesting findings of coating scientists.

The study presented here is motivated by experiments on polymeric coatings, in which combinations of different modifications of the polymers as well as multiple solvent choices are examined for their physical and chemical properties using robotic techniques.

The individual design choices that make up a coating design will be called "factors". Examples of factors are the molecular weight of the main polymer chain, the length of side chains, and the salt concentration of the solvent.

The factors and their design values together constitute the design space. Besides, one or more continuous responses are measured. These responses characterize certain features of a process or a product under different conditions specified by the combination of design values of each factor.

The goal of the proposed algorithm is to identify which design points or design regions are important and would be likely to yield desired response values. In this study, a design region is defined as one design point or a set of adjacent design points. When the subset of instances in a design region shows a clear pattern, the design region is considered to be important. The proposed algorithm adopts a modified version of the vector-item pattern mining algorithm introduced in [134]. To determine whether the subset of instances in a design region shows a clear pattern or not, two histograms that summarize neighboring relationships are created, one for the instances within the subset and the other for the instances in multiple random samples. We refer to the histogram for the subset in a design region as observed density histogram, and the one that is created by averaging over random samples as expected density histogram.

Fig. 6.1 shows an example of an observed density histogram for a design region and an expected density histogram constructed from random samples. Also included are results for a theoretical model that is used to approximate the expected density histogram in order to improve the efficiency, and which will be described shortly.

The highlighted response profiles in the top left panel of Fig. 6.1 indicate that the instances in the design region display a clear pattern. Notice there is an obvious difference between the observed and expected density histograms that are shown in the bottom left panel. In contrast, the highlighted response profiles in the top right panel do not show a

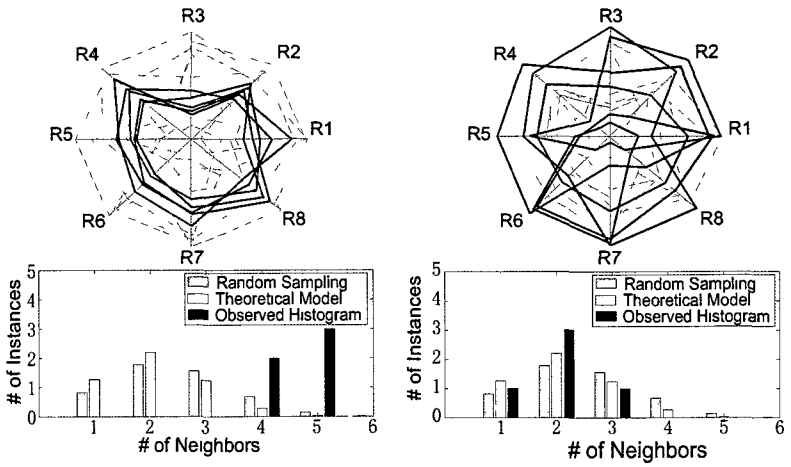


Figure 6.1: The top-left panel shows a spider plot for the instances in a design region. The top-right panel shows a spider plot of a random sample. In the bottom-left and bottom-right panels, the black density histogram summarizes the neighboring relationship of the instances in the design region and random sample respectively. The gray density histograms are constructed from random samples. The blank histograms shows the theoretical model which will be discussed shortly.

clear pattern and the two density histograms are very similar.

Thus, by analyzing the difference between the observed density histogram and the expected density histogram of a design region, we can evaluate whether the design region shows a clear pattern or not. In the proposed algorithm, the effect size (we use Cohen's D) of the two histograms are analyzed. If effect size analysis yields a value greater than 0.8 (a "large" effect [41]), a pattern is identified for the design region under consideration.

A clear pattern is identified by comparing the neighborhood of the instances in a design region with that of the instances in random samples. The instances in the design region, hence, are more close to each other than those in random samples, indicating a low variance when considering all the responses. In other words, such a pattern mining procedure identifies the settings of factors that will result in low variance of multiple responses.

Furthermore, by adopting a product similarity measure, the proposed algorithm was able to identify factor settings that will lead to desired multiple responses. More details

will be described shortly.

The contribution of the proposed algorithm are:

(1) It provides an efficient and effective method to find important design regions that tend to yield interesting and important results for domain experts.

(2) The design regions identified by the algorithm will not only lead to low variances of multiple responses, but can be used as design region bounds for other multiple responses optimization algorithms (more details will be given shortly).

6.1. Related Work

Frequent pattern mining is one of the fundamental techniques in data mining. Many efficient and effective pattern mining algorithms [75, 69, 5] have been developed, and the techniques have been applied to many domains [10, 33, 52, 90]. [134] presents an algorithm that identifies patterns that exist between vector data and item sets. Significant core patterns between sector information and stock time series have been identified using the algorithm presented in [133].

Design of Experiments [59] is a widely used method for controlled experiments, aiming to optimize a manufacturing process. The most popular designs include two-level factorial, fractional factorial designs, and classical response surface designs [128, 7]. These designs enable researchers and experimenters to identify settings that are optimal for yielding desired response values.

The goal of robust design [127] is to reduce response variations in a manufacturing process by setting the factors to provide the best performance and to make the responses insensitive to noise factors. [127] employs an orthogonal array (or crossed array) to arrange the experiments, and to evaluate the response of an experimental run using signal-to-noise ratios.

The Design of Experiments and Robust design are usually used to model each response with regard to the input factors. When there are multiple responses, it is common

that setting the factors to values that result in optimal performance of one response will not necessarily lead to good performance of other responses. Thus several other methods were developed to determine the best settings for the factors, in order to simultaneously optimize multiple response variables. [32] proposed an multiple responses optimization algorithm that consists of four stages based on artificial neural networks, desirability functions and a simulated annealing algorithm. [11] provides an algorithm to solve a multiple responses optimization problem using a neuro-fuzzy model and the Taguchi method of experimental design. [98] presents an approach for the multiple response robust parameter design problem based on a Bayesian method that maximizes the posterior predictive probability that the process satisfies a set of constraints on the responses. [115] takes the modeling stage into consideration besides the optimization stage. [5] presents an improved genetic algorithm approach to conduct multi-objective optimization of simulation modeling problems.

Most of the available methods fit a polynomial model for the multiple responses w.r.t. control factors, and find optimal factor settings to maximize a desirability function. However, most of these approaches do not take into consideration the variance of responses when seeking the optimal factor settings.

6.2. The Proposed Algorithm

The proposed algorithm includes the following steps:

1. **Test region construction:** Test regions are first constructed in order to determine which design regions are important. Details are given in Section 6.2.1.

2. **Building density histograms:** For each test region, the two density histograms are built. More details will be given in Section 6.2.3.

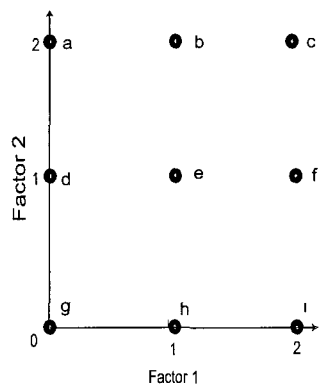
3. **Effect size analysis:** Effect size (Cohen's D) of each test region is analyzed by analyzing the effect size of the two density histograms. Details will be given in

sections 6.2.4 and 6.2.5.

6.2.1. Test Region Enumerating

A test region is a design region with each factor having no more instances than a predefined threshold. For simplicity, we only consider hype-cubic test regions that are orthogonal to factor axes.

For example, in Fig. 6.2 design points e and f can form a test region that is defined by $(Factor1 == 1 \cup Factor1 == 2) \cap Factor2 == 1$; However, design points g, e and h cannot form a test region since they do not form a hype-cubic (a rectangle in this two dimensional case) region. The proposed algorithm allows users to define how many design values of each factor can be merged to form a test region.



	Factor1	Factor2	Response1	Response2
instance1	0	0	0 11	0 23
instance2	0	1	0 75	0 39
instance3	0	2	0 22	0 75
instance4	1	0	1 36	1 47
instance5	1	1	1 79	3 56
instance6	1	1	1 82	3 64
instance7	1	2	1 32	2 77
instance8	2	0	0 75	2 83
instance9	2	1	1 22	0 93
instance10	2	2	2 11	1 00

Figure 6.2: Left panel shows a design space of 2 factors, each of which has 3 different values. Right panel shows a response table in which there are ten instances with different combinations of factors.

6.2.1.1 Indexing of design value combinations for each factor using bitvectors

Suppose the p^{th} factor has $|Y^p|$ values, and given a threshold θ^p indicating the maximum number of adjacent design values in the factor that can be included to form a test region. Then we can index all combinations of these design values of the factor. Fig. 6.3 indexes the combinations of different design values of Factor1 and Factor2 in Fig. 6.2 with θ^1 (threshold of Factor1) equaling 2 and θ^2 (threshold of Factor2) equaling 1. The bitvectors

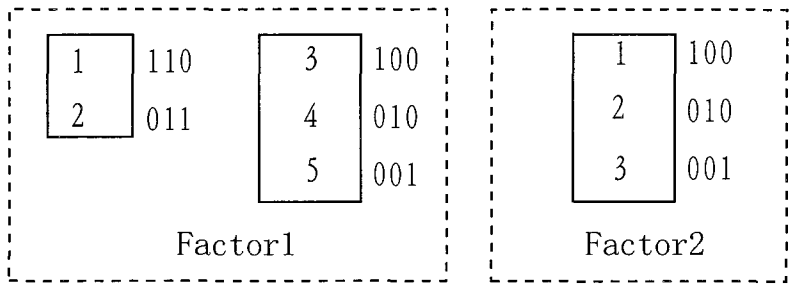


Figure 6.3: Enumerating the combinations of different design values of Factor1 and Factor2 in Fig. 6.2 with θ^1 (threshold of Factor1) equaling 2 and θ^2 (threshold of Factor2) equaling 1. The numbers in boxes are the indices.

represent the selection of design values. The right-most bit to the left-most bit of a bitvector indicate the selection of the smallest to the biggest design values, where '1' means the corresponding design value is selected and '0' otherwise. For example, the bitvector '110' in the left panel of Fig. 6.3 means design values '2' and '1' of Factor1 in Fig. 6.2 are selected, and design value '0' of Factor1 is deselected. The numbers in the boxes are the indices of the bitvectors (indexing this way facilitates the pruning procedure, which will be discussed shortly). We can easily generalize the enumeration to a factor that has $|Y^p|$ design values with a threshold θ^p. And the total number of combinations can be expressed as:

$$S^p = \sum_{i=0}^{\theta^p - 1} (|Y^p| - i) = \theta^p * |Y^p| - \frac{(\theta^p - 1) * \theta^p}{2} \tag{56}$$

where S^p is the number of combinations for the p^{th} factor, $|Y^p|$ is the number of design values of the p^{th} factor, and θ^p is the maximum number of adjacent design values that can be merged for the p^{th} factor. In Fig. 6.3, p, $|Y^p|$ and θ^p are equal to 1 and 3 and 2 respectively for Factor1, 2 and 3 and 1 for Factor2. From equation (56) we get S^1 equaling 5 and S^2 equaling 3, which is consistent with the total number of bitvectors shown in Fig. 6.3.

6.2.1.2 Indexing of test regions for multiple factors A test region is created by intersecting the design value combinations from all factors. The setting from Fig. 6.2 will continue to be used as an example. Test region (e and f) is defined by $(Factor1 == 1 \cup Factor1 == 2) \cap Factor2 == 1$. Since we have a method to index all combinations of design values for each factor, we can construct a generalized number system to index all test regions straightforwardly.

Fig. 6.4 enumerates all test regions in Fig. 6.2 with θ^1 (threshold of Factor1) equaling 2 and θ^2 (threshold of Factor2) equaling 1. The first column holds the indices of each test region. The second column and the third column are the indices of design value combinations for Factor1 and Factor2 respectively. The fourth column shows the set of design points within each test region as in Fig. 6.2. The last column shows the bitvectors as in Fig. 6.3.

Index	Factor1	Factor2	Test Region	Bit strings
1	1	1	bc	110 ∩ 100
2	2	1	ab	011 ∩ 100
3	3	1	c	100 ∩ 100
4	4	1	b	010 ∩ 100
5	5	1	a	001 ∩ 100
6	1	2	ef	110 ∩ 010
7	2	2	de	011 ∩ 010
8	3	2	f	100 ∩ 010
9	4	2	e	010 ∩ 010
10	5	2	d	001 ∩ 010
11	1	3	hi	110 ∩ 001
12	2	3	gh	011 ∩ 001
13	3	3	i	100 ∩ 001
14	4	3	h	010 ∩ 001
15	5	3	g	001 ∩ 001

Figure 6.4: Enumerating the test regions in Fig. 6.2 with θ^1 (threshold of Factor1) equaling 2 and θ^2 (threshold of Factor2) equaling 1.

We can generalize the enumeration of test regions to M factors that have $|Y^p|$ design values with a threshold θ^p for the $p^{th}(p \in [1, M])$ factor. The total number of test regions can be expressed as:

$$|B| = \prod_{p=1}^{M} S^p \tag{57}$$

where B is the set of all test regions, $|B|$ is the number of all test regions, M is the number of factors, and S^p is the number of combinations of design values for the p^{th} factor (in equation (56)).

Therefore, given an experimental design for which we know the number of factors, M, and the number of design values ($|Y^p|$) for the $p^{th}(p \in [1, M])$ factor, after we define the maximum number of adjacent design values (θ^p) in the $p^{th}(p \in [1, M])$ factor that can be included to form a test region, we can index each test region using numbers from 1 to $|B|$, which can be calculated using equations (57) and (56).

6.2.1.3 Converting the index over all test regions back to indices of design value combinations of individual factors When enumerating test regions, given an index $q(q \in [1, |B|])$, we need to know the indices of design value combinations for each factor p. For example, in Fig. 6.4, for the index of test region (the first column) $q = 6$, we need to get the index of the combination of design values for Factor1 ($I^1 = 1$) and Factor2 ($I^2 = 2$) (the second and the third columns). Given an index q $(q \in [1, |B|])$ of test regions, the indices for the combinations of design values for each factor p can be calculated as follows:

$$I^p = \left\lceil \left(q - (\prod_{i=1}^{p} S^i) * \left\lfloor q/(\prod_{i=1}^{p} S^i) \right\rfloor \right) /(\prod_{i=1}^{p-1} S^i) \right\rceil \tag{58}$$

I^p is the index of the combination of design values for the p^{th} factor, for a test region index $q \in [1, |B|]$, S^i is the number of combinations of design values for the i^{th} factor (in equation (56)), $\lfloor\rfloor$ is a floor operation and $\lceil\rceil$ is a ceiling operation.

6.2.1.4 Converting single-factor indices back to bitvectors that characterize design regions After extracting the index I^p (in equation (58)), we then try to obtain the selected

design values of the p^{th} factor.

Take Fig. 6.4 for example, previously we know when the test region index (the first column) $q = 6$, from equation (58) we get the index of the combination of design values for Factor1 $I^1 = 1$ and Factor2 $I^2 = 2$ (the second and the third columns). We need to know that the second and the third design values of Factor1 and the second design value of Factor2 are selected (the fifth column), in order to know that the test region index $q = 6$ indexes the design region $\{e, f\}$ (the fourth column).

We solve the problem in a geometric way:

Notice that in Factor1 of Fig. 6.3, the bitvectors in the rightmost column have only one '1' bit, from bottom to top starting from the least significant bit to most significant one. And the next rightmost column have two consecutive '1' bits, also from bottom to top starting from the least significant bits to most significant ones, and so on. There are $\theta^1 == 2$ columns in total. The index for Factor1 starts from the top-left bitvector to right-bottom one, from top to bottom and from left to right. Fig. 6.5 shows a geometric representation for the bitvectors for the p^{th} factor. Each dot in Fig. 6.5 represents a bitvector.

Given an index I^p, we first try to find its representing dot, for example P in Fig. 6.5. The number of dots between dot P and dot G, i.e., L, is exactly the position of the least significant '1' bit in the corresponding bitvector. The number of columns from column EG to rightmost column BD (denote by $|EB|$), is the number of '1' bits of corresponding bitvector, which are indexing the selected design values.

L is equal to the number of dots from dot P to dot D minus the number of dots from dot F to dot D (EG and FH are two consecutive columns). Let T^p denote the number of dots from dot P to dot D:

$$T^p = S^p - I^p + 1 \tag{59}$$

where S^p (equation (56)) is the total number of dots from dot A to dot D, and I^p is

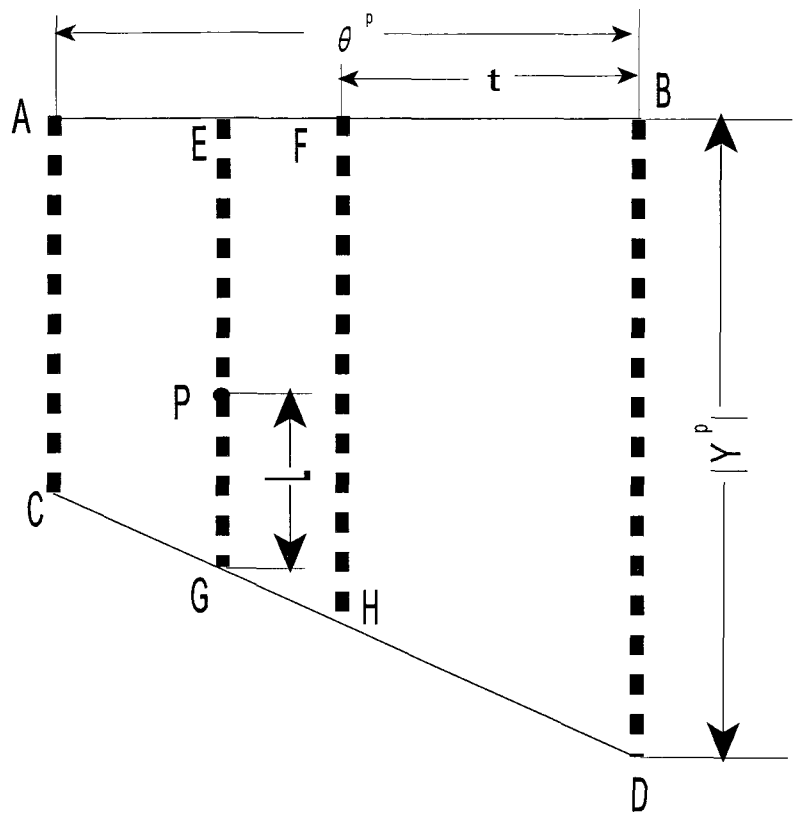


Figure 6.5: Geometric representation for the bitvectors. Each dot represents a bitvector.

the index which is equal to the number of dots from dot A to dot P. Let t equal the number of columns from FH to BD, then the number of dots from dot F to dot D is equal to $\frac{(|Y^p|-(t-1)+|Y^p|)t}{2}$ (the number of dots from dot F to dot H is equal to $|Y^p|-(t-1)$). Thus:

$$L = T^p - \frac{(|Y^p| - (t-1) + |Y^p|)t}{2} \tag{60}$$

where T^p is in equation (59), and t has the following relationship with T^p:

$$\frac{(|Y^p|-(t-1)+|Y^p|)t}{2} < T^p \leq \frac{(|Y^p|-t+|Y^p|)(t+1)}{2}$$

i.e., the number of dots from dot P to dot D is greater than the number of dots from dot F to dot D but less than or equals to the number of dots from dot E to dot D. Thus we get:

$$t = \left\lfloor \frac{(2 * |Y^p| + 1) - \sqrt{(2 * |Y^p| + 1)^2 - 8 * T^p}}{2} \right\rfloor \tag{61}$$

The number of '1' bits in the corresponding bitvector is eqal to $t + 1$, i.e., $|EB| == t + 1$.

Summarizing equations (59), (60) and (61), given an index I^p, the selected design values of the p^{th} factor can be expressed as:

$$\sigma(p, I^p) = \{Y^{p,j} | j \in [L, L + t]\} \tag{62}$$

where $\sigma(p, I^p)$ is a function to extract a set of design values $Y^{p,j}$ (the j^{th} design value of the p^{th} factor) for the p^{th} factor when the index of the combination of design values of p^{th} factor is equal to I^p, L is in equation (60) and t is in equation (61).

Going back to the example shown in Fig. 6.4, previously we know when the test region index (the first column) $q == 6$, from equation (58) we get the index of the combination of design values for Factor1 $I^1 == 1$ and Factor2 $I^2 == 2$ (the second and the third columns). Using the function in equation (62) we get $\sigma(1, I^1) == \{Y^{1,2}, Y^{1,3}\}$ and $\sigma(2, I^2) == Y^{2,2}$, i.e., the second and the third design values of Factor1 and the second design value of Factor2 are selected. Thus we know the test region index $q == 6$ indexes the design region $\{e, f\}$.

Through the procedure discussed above, we can enumerate all the test regions ($q \in [1, |B|]$) that satisfy the threshold $\theta^p (p \in [1, M])$ for each factor.

6.2.2. Pruning Searching Space

Since we are only looking at those test regions in which the numbers of instances are greater than a threshold φ, we can adopt a pruning technique that is similar to the apriori algorithm in association rule mining [112, 5]. Fig. 6.6 shows an example of design space of 3 factors, a, b and c, which have 5 and 4 and 2 design values respectively.

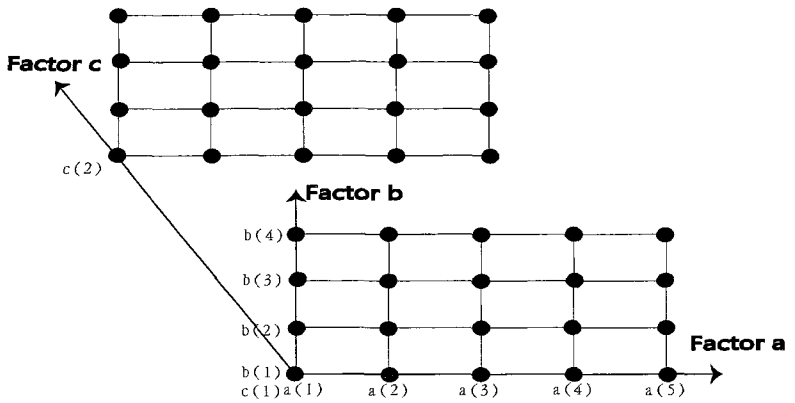


Figure 6.6: An example design space of 3 factors, which have 5, 4 and 2 design values respectively.

For each factor, we can build a lattice in which each node represents a design value combination that could possibly participate in forming a valid test region. In Fig. 6.7, the threshold θ is set to 3 for all the three factors. Thus all the nodes with bold fonts, such as "a(5)a(4)a(3)" and "b(2)b(1)", will participate in forming valid test regions. Once we find the number of instances in the intersection of multiple nodes, each from different lattices, is less than the threshold φ, then:

Pruning (i): all their offspring's intersections will also have less than φ instances.

Pruning (ii): adding more factors to the intersection results in even less instances.

Take nodes "a(3)a(2)a(1)" and "b(2)b(1)" in Fig. 6.7 for example. If the intersection of the two nodes has less than φ instances, then (i) all the intersections of any two nodes, one from each dash circle, will also be less than φ and (ii) adding another factor to the intersection, for example "a(3)a(2)a(1)" $\cap$ "b(2)b(1)" $\cap$ "c(2)", will have less than φ instances either.

From design space point of view, if the number of instances in all the nodes within the gray cube in Fig. 6.6 is less than φ, then the number of instances in any of its subset of nodes will also be less than φ.

This observation provides us an efficient way to prune the test region search space.

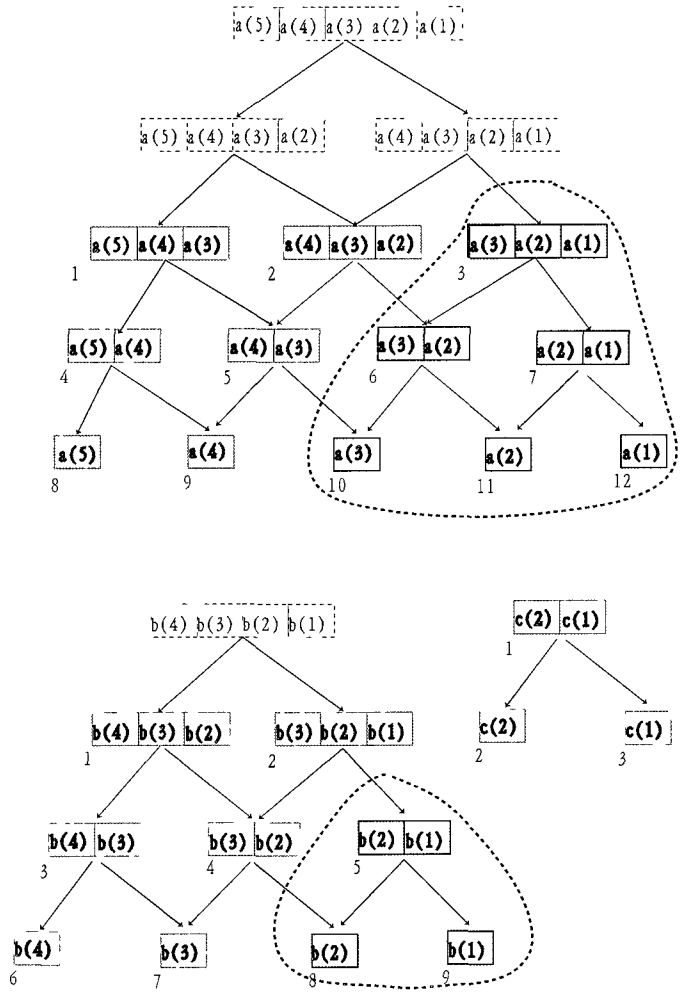


Figure 6.7: All design value combinations that could possibly form a valid test region. In this example $\theta == 3$ for factor a and factor b and factor c. Thus all the nodes with bold fonts, such as "a(5)a(4)a(3)" and "b(2)b(1)", will participate in forming valid test regions.

Notice that the numbers at the left-bottom of the nodes in a lattice (Fig. 6.7 for example) indicate the indices of that factor, i.e., the I^p in equation (58). Ordering the indices this way, i.e, the index increments from left to right and level by level, facilitates the pruning algorithm.

When enumerating test regions using index q ($q \in [1, |B|]$ where $|B|$ is the number of all test regions as in equation (57)), once there is a test region indexed by q that has less than φ instances, all its sub test regions' indices can be discarded.

Given the I^p-th node for the p^{th} factor, we can know immediately the indices of its two children, $\{I^p$+L, I^p+L+1$\}$ (L is the number of nodes at the same level). For example, the two children of the 3^{rd} node of factor a in Fig. 6.7 are [3+3, 3+3+1] = [6, 7]. Notice that $L = |Y^p| - |\sigma(p, I^p)| + 1$, where $|Y^p|$ is the number of design values in the p^{th} factor, i.e., the number of leaf nodes, and $|\sigma(p, I^p)|$ (equation (62)) is the number of different design values in the node. Furthermore, node I^p's children's children (there will be three children if it has any), can be expressed as $[I^p + L + (L + 1), I^p + L + 1 + (L + 1) + 1]$, i.e., from the left child's left child to right child's right child. For example, the three children of the 3^{th} node's children are $[3 + 3 + 4, 3 + 3 + 1 + 4 + 1] = [10, 12]$, i.e, the 10^{th} node to 12^{th} node. The indices of the l^{th} level offspring nodes for the p^{th} node can be expressed as:

$$\begin{aligned} \mathrm{O}^{\mathrm{p},\mathrm{I}^\mathrm{p},l} = [I^p + \sum_{t=1}^{l} (|Y^p| - |\sigma(p, I^p)| + 1), \\ I^p + \sum_{t=1}^{l} (|Y^p| - |\sigma(p, I^p)| + t) + 1] \end{aligned} \tag{63}$$

where $|\sigma(p, I^p)|$ (equation (62)) is the number of different design values in the node.

Notice that node I^p has $|\sigma(p, I^p)| - 2$ levels of *non-leaf* offspring nodes. Thus all offspring nodes of the I^p's node can be expressed as:

$$\delta(\mathrm{p}, \mathrm{I}^\mathrm{p}) = \bigcup_{l=1}^{|\sigma(p,I^p)|-1} \mathrm{O}^{\mathrm{p},\mathrm{I}^\mathrm{p},l} \tag{64}$$

where $\delta(p, I^p)$ is the function to extract all the offspring nodes of the $I^p - th$ node for the p^{th} factor, $\mathrm{O}^{\mathrm{p},\mathrm{I}^\mathrm{p},l}$ is in equation (63).

After we get all the offspring nodes in each lattice, we can calculate all the sub test regions' indices as follows:

$$Q = \{t | t = 1 + \sum_{p=1}^{M} (I^r - 1) * \prod_{i=1}^{p-1} S^i, I^r \in \{I^p \cup \delta(p, I^p)\}\} \tag{65}$$

where M is the number of factors, I^p is the index of a test region in the p^{th} factor, S^i

(in equation (56)) is the number of combinations for the i^{th} factor, and $\delta(p, I^p)$ (in equation (64)) is the nodes in sub lattices of the I^p-th node of the p^{th} factor.

Alg. 10 shows the algorithm to enumerate all test regions. Alg. 10 first determines if there is a corresponding sub-lattice table file (Line 1), which will be discussed shortly. If the sub-lattice table file exists, then it initializes the lookup table $SubLatticeTable$, otherwise it sets the lookup table $SubLatticeTable$ to be empty (Lines 2 to 4). Then Alg. 10 calculates $|B|$, the number of all combinations of design values for all factors (Line 5). For each number q in [1, $|B|$], the Alg. 10 first determines if the test region represented by the number q is valid or not. If the test region is not valid, which means the test region is a sub test region of an invalid test region that is already calculated, then the test region is simply discarded (Lines 9 to 10). Otherwise for each factor, the algorithm first calculates the index I^p using equation (58), and then using equation (62) to extract which design values are selected by I^p in the p^{th} factor (Lines 14 to 16). Once function $FindNumberOfInstance$ finds that the number of the instance in the intersection of the nodes of the calculated lattices(factors) is less than φ, 'Pruning (i)' as well as 'Pruning (ii)' is applied: if the lookup table $SubLatticeTable$ is empty, equation (65) is used to extract the complete sub test region set V (Lines 19 to 20). Otherwise the sub test region set V is read from the lookup table $SubLatticeTable$ (Lines 21 to 22).

To make the pruning procedure even more efficient, we can save to disk sub-lattice tables w.r.t. the number of factors, the number of design points in each factor, and the threshold θ for each factor in advance. When processing a data set we can simply load the corresponding sub-lattice tables (lines 1 and 2 in Alg. 10). To create such a sub-lattice table, we first calculate the number of combinations of all the lattices (factors) using equation (57), i.e. N in line 5 of Alg. 10, and for each number in $[1, N]$ we extract all the sub test regions using equation (65). This procedure can be easily parallelized, and we can save multiple files for each sub-lattice table, and load the corresponding file by checking the

Algorithm 10: Enumerate test regions

```
Function GetTestRegion;                          /* get all test regions.  */
Data: Y;     /* The array for storing link list of each factor
     values.  */
Data: θ;            /* number of design values can be merged */
Result: B;                        /* the set of all test regions */
1  if exist(corresponding sub-lattice table file) then
2      SubLatticeTable=load(corresponding sub-lattice table file);      /* speed up
       pruning process in line 22.  */
3  else
4      SubLatticeTable=[];
5  calculate N (N==|B|) using equation (57);
6  B=[] ;                                                  /* initialize B */
7  validTestRegion = ones(1, N) ;        /* Initialize a look up table
   indicating whether a test region is valid or not.  */
8  for q = 1 : N do
9      if validTestRegion(q)==0 then
10         continue; ;    /* if the test region, indexed by q, is not
           valid, disregard it.  */
11     validSet=[];
12     nodeIndices=ones(1,M);
13     for p = 1 : M do
14         calculate I^p using equation (58);
15         nodeIndices(p)=I^p;
16         validSet=[validSet, σ(p, I^p)] using equation (62);
17         if FindNumberOfInstanceIn(validSet) < φ then
18             validSet=[];
19             if isempty(SubLatticeTable) then
20                 V=Q using equation (65).;        /* find all test regions
                   constructed by the nodes in sub-lattices */
21             else
22                 V=SubLatticeTable(q);  /* look up sub test regions for
                   the test region indexed by q */
23             validTestRegion(V)=0;
24             break;
25     B = [B,validSet];
```

value of the test region index q (in line 8 of Alg. 10) to save memory. Whenever there is a test region dose not meet the threshold φ, all the sub test regions can be discarded, as shown in line 22 of Alg. 10.

6.2.3. Building Density Histograms

To prevent some response values from dominating the results of the proposed algorithm, response values are transformed by applying a z-score normalization:

$$X_i^j = \frac{X_i^j - \overline{X^j}}{\sigma(X^j)} \tag{66}$$

where X_i^j is the j^{th} response value of the i^{th} instance, $\overline{X^j}$ is the mean of the j^{th} response and $\sigma(X^j)$ is the standard deviation of the j^{th} response.

The proposed algorithm adopts a modified version of the vector-item pattern mining algorithm [134]. Two density histograms are built for the subset of instances in each test region. The observed density histogram summarizes the distribution of the neighboring relationship for the subset of instances in the test region under consideration; The expected density histogram provides the expected distribution of the neighboring relationship for the subset.

In the proposed algorithm, product similarity is applied:

$$Sim(\mathbf{X}_i, \mathbf{X}_j) = \sum_{l \in [1,D]} (X_i^l \times X_j^l) \tag{67}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ are the i^{th} and j^{th} instance vectors, X_i^l is the l^{th} response value of i^{th} instance vector $\mathbf{X}_i$, and D is the number of responses. If two instance vectors $\mathbf{X}_i$ and $\mathbf{X}_j$ satisfy the following condition, they are considered to be neighbors:

$$Sim(\mathbf{X}_i, \mathbf{X}_j) >= D \times ThS \tag{68}$$

where $D \times ThS$ (ThS is a constant value, 0.1 in our study) serves as a threshold for the similarity measurement.

Alg. 11 shows how the observed and expected density histograms are constructed. Function $BuildObservedDensityHist$ builds an observed density histogram, and function $RandomSampling$ builds an expected desity histogram using random sampling. Both functions call a sub-function $BuildDensityHist$, which builds a density histogram for a subset of instance vectors.

The input of the function $BuildDensityHist$ are a subset of instance vectors $\mathbf{X}_S$ and the product similarity threshold ThS. The output of the function $BuildDensityHist$ is the density histogram $hist$. To build a density histogram for a subset of vectors, the number of neighbors for each instance vector in the test region is calculated (Lines 15 to 18), and a contribution of 1 is added to a corresponding bin of the density histogram (Line 19).

To build an observed density histogram, function $BuildObservedDensityHist$ directly calls function $BuildDensityHist$, with the test region under consideration (Line 1), and returns the result of function $BuildDensityHist$, which is the observed density histogram.

To build an expected density histogram, function $RandomSampling$ draws T random samples. A density histogram is built for each sample (Lines 7 to 8). Then the contribution of these random samples are averaged to create the expected density histogram for the test region (Line 9). In this study, T is set to 30.

Fig. 6.1 shows an example of the observed density histogram and the two versions of the expected density histogram, one constructed from random samples and the other using the theoretical model, which will be described in Section 6.2.6.

6.2.4. Effect Size Analysis

The data sets for a combinatorial design sometimes are relatively small. The sizes of the two density histograms hence are small too, which makes statistical significance

Algorithm 11: Building Density Histograms

/*********BuildObservedDensityHist**********/;
Function: *BuildObservedDensityHist* ; /* Building Observed density histogram */
Data: $\mathbf{X}_{S_{B_k}}$; /* the subset of instance vectors in the k^{th} test area. */
Data: ThS; /* $size(\mathbf{X}_{S_{B_k}}, 2) \times ThS$ is the product similarity threshold. */
Result: *hist*; /* Observed density histogram */
$hist = BuildDensityHist(\mathbf{X}_{S_{B_k}}, ThS)$;
return *hist*;
/*********RandomSampling*********/ ;
Function: *RandomSampling* ; /* Building Expected density histogram from Random samples */
Data: X; /* the matrix for all attribute values */
Data: ThS; /* $size(X, 2) \times ThS$ is the product similarity threshold. */
Data: $|S_{B_k}|$; /* Sample size. */
Data: T; /* Number of samples. */
Result: *hist*; /* Expected density histogram */
$hist = zeros(1, |S_{B_k}|)$;
for *i=1 to T* **do**
    $R = FindARandSet(N, |S_{B_k}|)$; /* Extract a random sample. N is the number of instances. */
    $hist = hist + BuildDensityHist(\mathbf{X}_R, ThS)$;
$hist = hist/T$;
return *hist*;
/*********BuildDensityHist*********/;
Function: *BuildDensityHist* ; /* Build density histogram for a subset of instances in a test area */
Data: $\mathbf{X}_S$; /* a subset of vectors */
Data: ThS; /* $size(\mathbf{X}_S, 2) \times ThS$ is the product similarity threshold. */
Result: *hist*; /* Density histogram */
$hist = zeros(1, |S|)$;
for *i=1 to* $|S|$ **do**
    neighbors=0;
    for *j=1 to* $|S|$ **do**
        if $i! = j \&\& \; Sim(\mathbf{X}_{S_i}, \mathbf{X}_{S_j}) \geq size(\mathbf{X}_S, 2) \times ThS$ **then**
            neighbors=neighbors+1;
    hist(neighbors)=hist(neighbors)+1;
return *hist*;

analysis such as a χ^2 goodness of fit test unsuitable. Furthermore, the proposed algorithm detects patterns in test regions of various sizes, for which some include several others. For example, test region $\{a, b, d, e\}$ in Fig. 6.2 includes test regions $\{a\}$, $\{b\}$, $\{d\}$, $\{e\}$, $\{a, b\}$, $\{a, d\}$, $\{b, e\}$ and $\{d, e\}$. To obtain the familywise error rate when using a statistical hypothesis testing, The Bonferroni correction has to be applied. Several previous works [110, 63, 17] have addressed the drawbacks of the Bonferroni correction.

In the proposed algorithm, effect size (we use Cohen's D) together with confidence intervals (CI) of a subset of instances in a test region under consideration is analyzed by comparing the two density histograms.

Cohen's D of the two density histograms is as follows:

$$d = \frac{\overline{H_1} - \overline{H_2}}{\sqrt{\frac{(n_1-1)s_1^2-(n_2-1)s_2^2}{n_1+n_2}}} \tag{69}$$

where $\overline{H_1}$ is the mean of the observed density histogram for a subset of instances under consideration, $\overline{H_2}$ is the mean of the expected density histogram created from multiple random samples, n_1 and n_2 are the sizes of the two density histograms and s_1 and s_2 are the standard deviations of the two density histograms respectively. Since $n_1 == n_2$, the above equation can be transformed:

$$d = \sqrt{\frac{2n_1}{n_1 - 1}} \times \frac{\overline{H_1} - \overline{H_2}}{\sqrt{s_1^2 - s_2^2}} \tag{70}$$

Upward bias caused by small sample size is further corrected as suggested by [72]:

$$d = \left[1 - \frac{3}{8(n_1 - 1) - 1}\right] \sqrt{\frac{2n_1}{n_1 - 1}} \times \frac{\overline{H_1} - \overline{H_2}}{\sqrt{s_1^2 - s_2^2}} \tag{71}$$

The calculation of confidence interval (CI) of Cohen's D involves the calculation of

noncentral parameters (ncp). To simplify, an approximate estimation [101] is used instead:

$$95\%CI = d - 1.96se \quad to \quad d + 1.96se \tag{72}$$

where effect size d is in equation (71), $95\%CI$ is the approximate width of 95% confidence interval of d and se is the asymptotic standard error for the effect size:

$$se = \sqrt{\frac{2}{n_1} + \frac{d^2}{4(n_1 - 1)}} \tag{73}$$

The test regions for which the effect size of the corresponding density histograms exceeds 0.8 ("large" effect [41]) are extracted. The confidence interval values will be used for ranking the importance of test regions.

6.2.5. Ranking the Importance of Test Regions

The importance of a test region consists two parts. One is the effect size of the corresponding density histograms, the other is confidence interval. A simple linear model is created to evaluate the importance of a test region:

$$IM^q = \begin{cases} a \times d + b \times 3.92se & \text{d} \geq 0.8 \\ 0 & \text{d} < 0.8 \end{cases} \tag{74}$$

where IM^q is the importance value of the q^{th} test region, d is the effect size(equation (71)) and se is in equation (73). a and b are the weights for the effect size and the confidence interval respectively. It is desired to have a big effect size value and a small confidence interval value.

The importance of an individual design point can be calculated from all the test regions that span this design point:

$$IMP^i = \frac{\sum_{k=1}^{|B|} IM^q * I_{B^q}(i)}{\sum_{q=1}^{|B|} I_{B^q}(i)} \tag{75}$$

where IMP^i is the importance of the i^{th} design point, $I_{B^q}(i)$ is an indicator function which equals to 1 if the i^{th} design point belongs to the q^{th} test region otherwise 0, and $|B|$ is the number of test regions.

The importance of an individual design point thus can be seen as being modeled from the design point and its neighboring design points within different sizes of hype-volumes. Therefore, the further a neighbor to this design point, the less the neighbor contributes to the importance of the design point under consideration. Fig. 6.8 illustrates the degree of contributions of neighboring design points to the one under consideration, the darker the higher.

Notice that the above design point importance modeling method enables us to calculate the importance of the design points in which there is no instances at all, by only considering the contributions from neighboring design points.

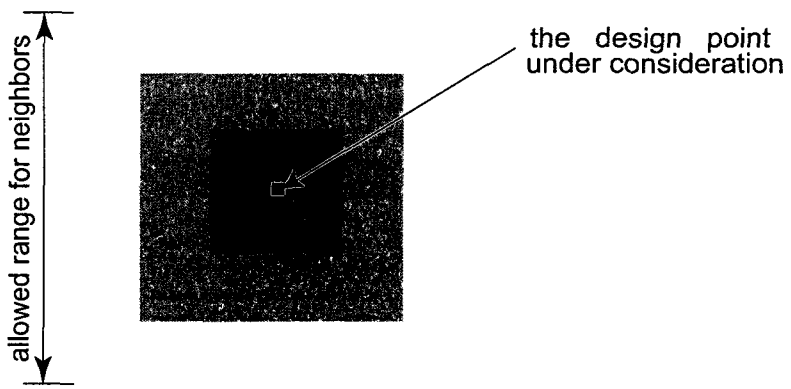


Figure 6.8: Illustration of the weights for the importance of neighboring design points. The darker the higher.

6.2.6. Theoretical Model

In many cases, domain experts expect that the responses would roughly follow a

normal distribution. A theoretical model of the expected histogram can be used to replace random sampling for the purpose of improving efficiency.

Let $\mathbf{X}_i$ and $\mathbf{X}_j$ be two arbitrary instance vectors. The probability that the product similarity of $\mathbf{X}_i$ and $\mathbf{X}_j$ exceeds the threshold t can be calculated by integrating over all dimensions of $\mathbf{X}_i$ and $\mathbf{X}_j$ with their respective weights (note that the two arbitrary vectors roughly follow a standard normal distribution along all the dimensions after z-score normalization (equation (66)), provided that all the responses roughly follow a normal distribution):

$$p = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \theta(\sum_k X_i^k X_j^k - t)(\frac{1}{\sqrt{2\pi}})^{2D} e^{-\frac{\mathbf{x}_i^2}{2}} e^{-\frac{\mathbf{x}_j^2}{2}} d^D\mathbf{X}_i d^D\mathbf{X}_j \tag{76}$$

where θ is an Heaviside step function, D is the number of responses (i.e., the dimensionalities of $\mathbf{X}_i$ and $\mathbf{X}_j$), and X_i^k and X_j^k are the k^{th} response values of $\mathbf{X}_i$ and $\mathbf{X}_j$ respectively.

Suppose S_D is the hype-surface region of an D-sphere of unit radius, then:

$$\begin{aligned} S_D \int_0^\infty e^{-r^2} r^{D-1} dr &= \underbrace{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty}}_{D} e^{-(x_1^2 + \ + x_D^2)} dx_1 ... dx_D \\ &= \left(\int_{-\infty}^{\infty} e^{-x^2} dx\right)^D \end{aligned} \tag{77}$$

where

$$\mathrm{S_D} = \begin{cases} \frac{2^{\frac{D+1}{2}} \pi^{\frac{D-1}{2}}}{(D-2)!!} & \text{for D odd} \\ \frac{2\pi^{\frac{D}{2}}}{(\frac{D}{2}-1)!} & \text{for D even} \end{cases} \tag{78}$$

Let r be the length of the vector $\mathbf{X}_i$. Vector $\mathbf{X}_j$ can be represented by two vectors $\mathbf{Z}$ and $\mathbf{U}$ such that $\mathbf{X}_j = \mathbf{Z} + \mathbf{U}$, where vector $\mathbf{Z}$ points to the same direction as $\mathbf{X}_i$ and vector $\mathbf{U}$ is perpendicular to vector $\mathbf{X}_i$. Rewrite the Heaviside step function:

$$\theta(\sum_k X_i^k X_j^k - t) = \theta(r\mathbf{Z} - t) \tag{79}$$

To simplify the integration over $\mathbf{X}_j$ in equation (76), the coordinate system of vector $\mathbf{X}_j$ is rotated such that one axis points to the direction of $\mathbf{Z}$. Therefore the integrations in the new coordinate system of $\mathbf{X}_j$ over the other $D-1$ dimensions other than the one that has the same direction as $\mathbf{Z}$ yield 1, since the Heaviside step function in these integrations are independent from $\mathbf{Z}$. Also substitute equations (77) and (79) to equation (76):

$$\begin{aligned}
&p = \int_0^{+\infty}\int_{-\infty}^{+\infty}\theta(r\mathbf{Z}-t)(\tfrac{1}{\sqrt{2\pi}})^n S_D r^{D-1} e^{-\frac{r^2}{2}}\tfrac{1}{\sqrt{2\pi}}e^{-\frac{\mathbf{Z}^2}{2}}d\mathbf{Z}dr \\
&\overset{let\ \mathbf{Z}=\sqrt{2}\mathbf{Y}}{=} \tfrac{1}{2}(\tfrac{1}{\sqrt{2\pi}})^D \\
&\int_0^{+\infty}\int_{-\infty}^{+\infty}\theta(r\sqrt{2}\mathbf{Y}-t)S_D r^{D-1}e^{-\frac{r^2}{2}}\tfrac{2}{\sqrt{\pi}}e^{-\mathbf{Y}^2}d\mathbf{Y}dr \\
&= \tfrac{1}{2}(\tfrac{1}{\sqrt{2\pi}})^D\int_0^{+\infty}\int_{\frac{t}{\sqrt{2}r}}^{+\infty} S_D r^{D-1}e^{-\frac{r^2}{2}}\tfrac{2}{\sqrt{\pi}}e^{-\mathbf{Y}^2}d\mathbf{Y}dr \\
&= \tfrac{1}{2}(\tfrac{1}{\sqrt{2\pi}})^D\int_0^{+\infty} S_D r^{D-1}e^{-\frac{r^2}{2}}erfc(\tfrac{t}{\sqrt{2}r})dr
\end{aligned} \tag{80}$$

where $erfc$ is a complementary error function.

After calculating this probability p, the theoretical model for the expected density histogram is given by a binomial distribution:

$$h_i = |S_{B_k}|\begin{pmatrix} |S_{B_k}| \\ i \end{pmatrix} p^i(1-p)^{|S_{B_k}|-i} \tag{81}$$

where $|S_{B_k}|$ is the size of the subset of instances and h_i is the height of the i^{th} bin in the expected density histogram.

6.2.7. Summary of Algorithm

Algorithm 12 summarizes the proposed algorithm. Firstly, Algorithm 10 is performed to get the set of all valid test regions (Line 1). Then for each valid test region, the observed density histogram is constructed (Line 5). If the test region includes many instances, the theoretical model (equation (81)) is used to build the expected density histogram, otherwise multiple random samples are created to construct the expected density histogram (Lines 6 to 9). Finally, those test regions for which the Cohen's D is greater than 0.8 are returned

Algorithm 12: Algorithm Summary

```
Function MainAlgorithm;                    /* the main algorithm.  */
Data: Y;        /* The array for storing link list of each
    factor values.  */
Data: θ;        /* number of design values can be merged */
Result: C;    /* return turples, [important design region,
     important value] */
1  Using Algorithm 10 to get B ;          /* get the set of all test
   regions */
2  ;
3  C = {};
4  foreach valid B_q do
5      Construct observed density histogram;
6      if the |S_{B_q}| > a threshold then
7          Construct Expected Density histogram using Theoretical Model.
           (equation (81));
8      else
9          Construct Expected Density histogram using Random samples.;
10     Calculate Cohen's D using equation (71). ;
11     if the Cohen's D ≤ 0.8 then
12         continue;
13     Calculate 'importance' using equation (74);
14     C = [C, {q, importance}];
```

with corresponding importance values that are calculated from equation (74) (Lines 10 to 14).

6.3. Experimental Evaluation

The proposed algorithm is tested on 2 coating data sets, which were published in [53, 54, 95].

The first data set [53, 54] has three factors: M_n for PDMS (molecular weight of polydimethylsiloxane, PDMS), PCL Length (Length of polycaprolactone blocks attached to PDMS) and Siloxane Content (percentage of PDMS). The factors and their values are shown in Table 6.1.

There are six responses in the data set, measuring three physical properties before

Table 6.1: Factors for the first data set

Factor	Design values
MN_1	M_n for PDMS = 2500
MN_2	M_n for PDMS = 5000
MN_3	M_n for PDMS = 7500
MN_4	M_n for PDMS = 10000
MN_5	M_n for PDMS = 15000
MN_6	M_n for PDMS = 20000
MN_7	M_n for PDMS = 25000
MN_8	M_n for PDMS = 30000
MN_9	M_n for PDMS = 35000
SC_1	Siloxane Content = 10
SC_2	Siloxane Content = 20
SC_3	Siloxane Content = 30
SC_4	Siloxane Content = 40
PCL_1	PCL Length = 0
PCL_2	PCL Length = 2
PCL_3	PCL Length = 3
PCL_4	PCL Length = 4

and after water immersion (three for each). The researchers may be more interested in the difference of these properties before and after water immersion, thus we take the difference of the pairs of the responses, resulting in three new features: WCAI (Water Contact Angle Increment after water immersion), MICAI(Methylene Iodide Contact Angle Increment after water immersion) and AFI (Average Force at release Increment after water immersion).

For this data set, we set the threshold of the number of instances in a test region θ be equal to 10.

Fig. 6.9 shows a screen shot of spider plot outputted by the related software for important test region in this data set.

Notice that the profiles of the design region (the darker curves) shows a clear pattern, from which we can know that the water contact angle and Methylene Iodide contact angle are clearly increasing after water immersion, and the Average Force at release basically

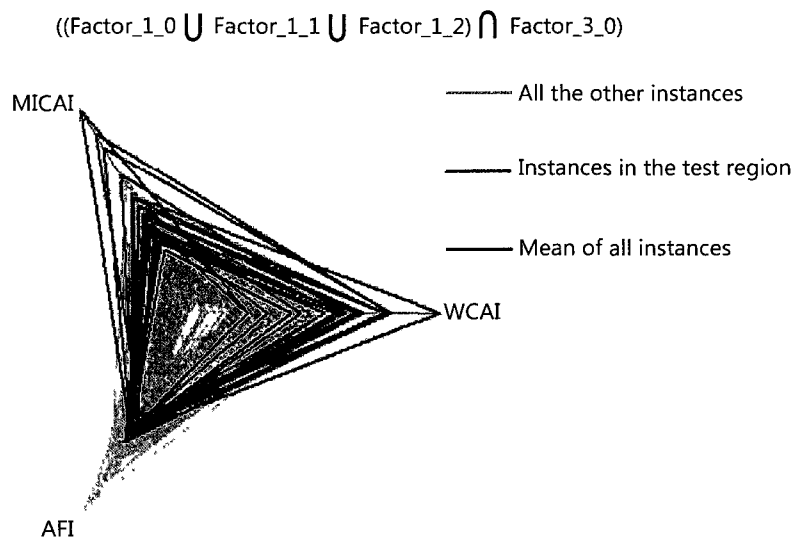


Figure 6.9: The most important design region identified.

have no changes after water immersion, which are considered good properties of a coating.

We also tested the product algorithm [47] in all test regions. Table 6.2 shows the confusion matrix for comparing the two algorithms. Among the 1470 test regions, 32 test regions are considerded to be important, and 28 test regions are significant at 5% (p-value ≤ 0.05) level by the product algorithm [47].

Table 6.2: Confusion matrix for the first data set

	Important	Not Important
Significant	21	7
Not Significant	11	1431

We analyzed the 11 test regions that are important by the proposed algorithm while not significant by the product algorithm. We find that the number of instances in these 11 test regions are equal to or just slightly greater than 10 (10 is the threshold for product algorithm [47]), which cause all the bins in the two density histograms aggregated to a single bin [47], failing to get significant results.

Fig. 6.10 shows an example that is considered to be important by the proposed

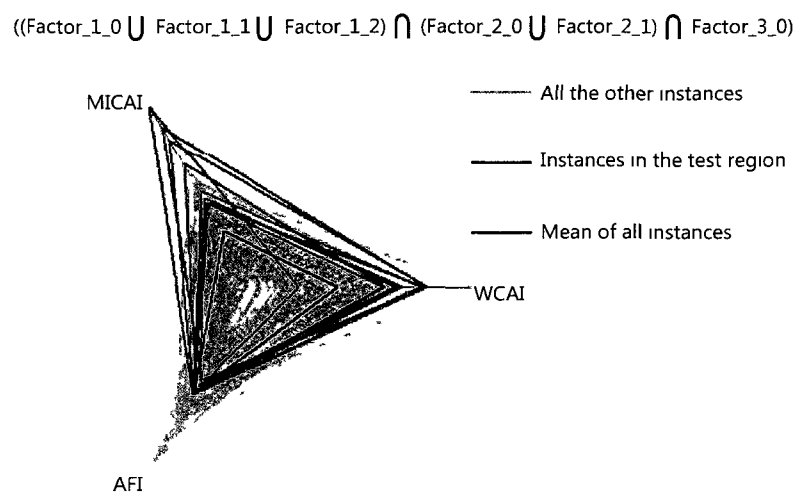


Figure 6.10: The spider plot of an important test region which is not significant.

algorithm but not significant by the product algorithm. From Fig. 6.10 we can see a clear pattern, and its two density histograms shown in Fig. 6.11 are clearly seperated.

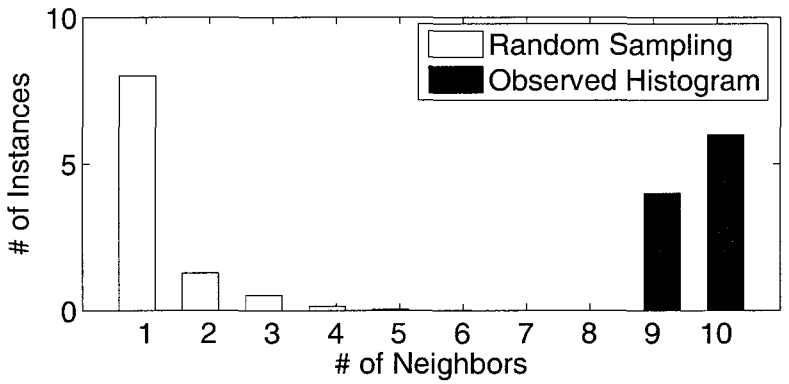


Figure 6.11: The density histograms of the important test region which is not significant.

Furthermore, we analyzed the 7 test regions that are significant based on the product algorithm but not important based on the proposed algorithm. We find that the sets of instances in the 7 test regions are even less dense than random sets, making them aggregate to a small number of bins to the left of the observed density histograms, which are consequently separating from their corresponding expected density histograms.

Fig. 6.12 shows an example that is not important based on the proposed algorithm

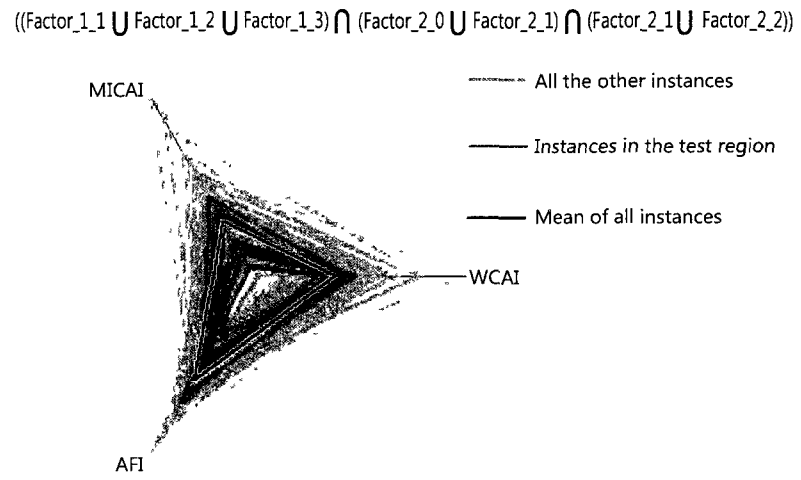


Figure 6.12: The spider plot of a significant test region which is not important.

but significant based on the product algorithm. Notice that the instances in the test region (the green curves) are located around the means of each responses (the black curve), which make the instances distant from each other as we using product simularity measurement (equation (67)). This can also be observed in Fig. 6.13, in which the observed density histogram has only two bins that are at "$\#\quad of\quad Neighbors == 1\quad or\quad 2$", resulting in a false "significant" by the product algorithm [47]).

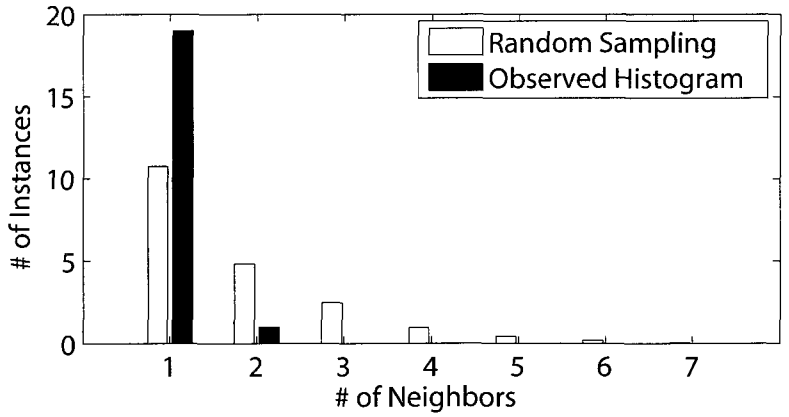


Figure 6.13: The density histograms of the significant test region which is not important.

The second data set [95] has three factors, namely polydimethylsiloxane (MN) which has 3 design values, Quaternary ammonium salt composition (SC) which has 4

design values, and Quaternary ammonium salt concentration (ST) which has 6 design values. Six responses including Water Contact Angle Hysteresis, Average Water Contact Angle, Average methyleneiodide Contact Angle, Surface Energy, C.lytica reduction, and N.incerta Reduction are measured. [95] investigated Polysilhasoxane coatings containing chemically-bound ("tethered") quaternary ammonium salt (QAS) moieties. Table 6.3 details the factors and their design values.

Table 6.3: Factors for the second data set

Factor	Design values
MN_1	M_n for PDMS = 2000
MN_2	M_n for PDMS = 18000
MN_3	M_n for PDMS = 49000
SC_1	salt composition = C1
SC_2	salt composition = C14
SC_3	salt composition = C18
SC_4	salt composition = Ph
ST_1	salt concentration = 0.005
ST_2	salt concentration = 0.01
ST_3	salt concentration = 0.015
ST_4	salt concentration = 0.02
ST_5	salt concentration = 0.025
ST_6	salt concentration = 0.03

There are only 72 instances in this data set. We set the threshold of the number of instances in a test region θ be equal to 5.

The proposed algorithm identifies the most important test regions to be $MN\_2 \cap (SC\_2 \cup SC\_3) \cap (ST\_2 \cup ST\_3 \cup ST\_4)$. In this test region (or its sub- test regions), [95] noted several interesting observations. The following are a few of them:

(1) In test region $MN\_2 \cap SC\_3 \cap ST\_4$ [95] find the coating with the roughest surface, and it also shows large scale phase separation.

(3) Test region $MN\_2 \cap SC\_3 \cap (ST\_3 \cup ST\_4)$ displays highest Ulva sporeling removal (90%).

(4) Test region $MN\_2 \cap SC\_2 \cap ST\_4$ shows a very high (98%) reduction in N. incerta biofilm growth.

We also tested the product algorithm [47] in all test regions in this data set. Table 6.4 shows the confusion matrix for comparing the two algorithms. Among the 525 test regions, 27 test regions are considerded to be important, and 12 test regions are significant at 5% (p-value ≤ 0.05) level by the product algorithm [47].

In this data set, we identified 19 test regions that are important by the proposed algorithm but not significant by the product algorithm [47]. All these 19 important and also effective (we check with coating scientists) test regions have less than 10 instances (a minimum requirement in product algorithm [47]), highlighting the capability of processing small data set for the proposed algorithm.

Table 6.4: Confusion matrix for the second data set

	Important	Not Important
Significant	8	4
Not Significant	19	494

6.3.1. Performance

The performance of the algorithm mainly depends on the number of factors, the average number of design values in each factor of a data set, and the number of samples when using random samples to create the expected density histogram. To achieve a higher efficiency, the theoretical model is suggested. Based on our experience, even the response values in a data set do not strictly follow a normal distribution and the responses are not independent from each other. The theoretical model can still roughly approximate the expected density histogram that is created using multiple random samples.

Without pruning the proposed algorithm would scale exponentially with the number of factors in the data set. Luckily, the test region enumerating and pruning algorithm (Alg. 10) significantly decreases the search space. To illustrate the effectiveness of the pruning

procedure, we test the Alg. 10 with and without pruning procedure on an artificial data set which has 2000 instances and 6 design values in each factor, with setting θ (the threshold for how many design values can be merged) to 4 for each factor, and varying φ (the threshold of the number of instances for a test region to be valid) from 5 to 1000.

Fig. 6.14 shows the performances of the Alg. 10 with and without pruning procedure, for which both 'pruning using equation (65)' and 'pruning using sub-lattice table' are tested. When factor equals to 4 (the bottom panel), we only tested φ equals to 5 to 365, because the design region enumerating algorithm without pruning procedure takes too much time.

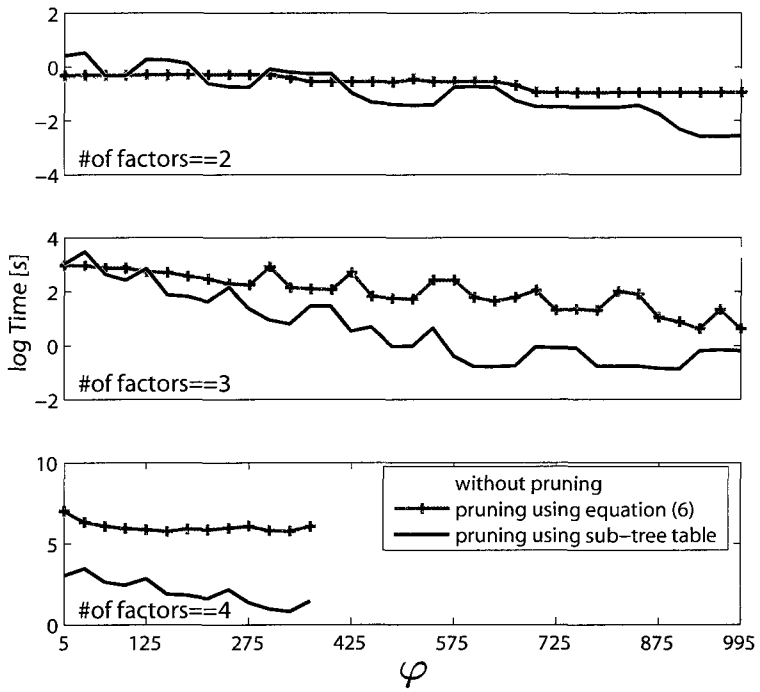


Figure 6.14: The comparison between the design region enumerating algorithm with and without pruning procedure. When factor equals to 4 (the bottom panel), we only tested φ equals to 5 to 365, for the design region enumerating algorithm without pruning procedure takes too much time.

From the top panel of Fig. 6.14 it can be seen that, when the number of factors is equal to 2 and φ is less than 300, the test region enumerating with pruning procedure is even worse than that of without pruning procedure. The reason behind is that, in these cases all

design points will have more than φ instances which makes the pruning procedure invalid while costing extra time to perform the pruning procedure or load corresponding sub-lattice table file. With the number of factors increases, the benefit of test region enumerating with pruning procedure becomes more obvious. When the number of factors is only equal to 4, the pruning procedure saves a lot of time even for a reasonably small φ.

The experiments are performed on a personal laptop, which has a 'Intel Pentium Dual CPU T3200 2.00GHz' processor, a 4GB memory, and 32-bit Windows Vista Home Premium operating system.

6.3.2. From Multiple Response Optimization Point Of View

The proposed algorithm can also be viewed as related to multiple response optimization [126, 32, 5, 11, 98, 115].

In the proposed algorithm, a z-score normalization (equation (66)) is first performed on the data set, which transforms the smallest response values in each response to biggest absolute negative values, and the biggest response values to biggest positive values. Thus most responses in an identified clear pattern will either have biggest or smallest original response values, since the proposed algorithm uses the product similarity measurement (equation (67)) to calculate the two density histograms, via which the clear pattern is identified.

In some cases, a particular response value rather than the biggest or the smallest response values is favored by domain expert, trivial transformation can be applied as a preprocessing step, such as those in desirability functions [65].

The dataset shown in Table 6.5 is from [30]. The goal of the experiment was to reduce the amount of by-products from a chemical process. There are five control factors, five responses and the experimental design used was a 2^{5-1} fractional factorial with two center points. The acceptable response region was definded by:

$$A = \{y = [y_2 \quad y_3 \quad y_4 \quad y_5] : y_2 \geq 91.0, \quad y_3 \leq 11.5, \quad y_4 \leq$$

$6.5, \quad y_5 \leq 5.5\}$

Table 6.5: Experimental results for the chemical process

x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3	y_4	y_5
-1	1	-1	-1	-1	80.0	93.7	5.1	1.2	2.6
-1	1	-1	1	1	80.0	88.7	10.9	0.4	4.0
1	-1	-1	1	1	91.0	90.8	9.0	0.2	1.9
1	-1	-1	-1	-1	86.0	94.3	3.5	2.2	1.2
0	0	0	0	0	75.0	92.7	7.1	0.2	2.5
1	1	1	-1	-1	89.0	95.0	4.4	0.6	1.4
-1	1	1	1	-1	84.0	91.7	8.3	0.0	2.4
-1	-1	1	-1	-1	80.0	82.8	2.3	14.9	0.6
-1	-1	1	1	1	83.0	90.0	4.1	5.9	0.7
-1	1	1	-1	1	84.0	94.6	5.4	0.0	1.5
1	-1	1	-1	1	89.0	96.2	3.8	0.0	1.6
1	-1	1	1	-1	92.0	94.5	5.5	0.0	1.5
0	0	0	0	0	96.0	94.1	5.9	0.0	2.8
1	1	1	1	1	88.0	86.9	13.9	0.0	7.9
-1	-1	-1	1	-1	89.0	81.2	4.8	14.0	2.6
-1	-1	-1	-1	1	81.0	87.4	4.1	8.5	0.5
1	1	-1	-1	1	100.0	92.1	7.9	0.0	5.1
1	1	-1	1	-1	90.0	89.3	10.7	0.0	9.2

The most important design region of the data set identified by the proposed algorithm is $x_2 == 1 \cap x_4 == -1 \cap x_5 == -1$, which is exactly the optimal solution found by the Bayesian approach [98].

Compare with other multiple responses optimization algorithms [126, 32, 5, 11, 98, 115], the drawback of the proposed algorithm used for multiple responses optimization purpose is that there is no model built for responses w.r.t. control factors, and in most cases the most important design region identified by the proposed algorithm is usually not a design point, which may or may not the optimal design point. However, besides tend to yield desired responses values, the important design region identified by the proposed algorithm to some extend also balances the variances of the responses.

6.4. Conclusions

In this study, we presented an algorithm to identify important design regions in a combinatorial design, which has one or multiple responses. The importance of a design region is measured through a model, which incorporates the effect size and confidence intervals of the design region. Through comparing with the product algorithm [47], we show the effectiveness and robustness of the proposed algorithm. Two pruning techniques are developed to speed up the enumeration of all test regions, ensuring the efficiency of the proposed algorithm. The important design regions identified by the proposed algorithm will not only lead to low variances of multiple responses, but tend to yield desired response values for most of responses.

6.5. Acknowledgments

This project is funded by Department of Energy under award # DE-FG52-08NA28921, the Office of Naval Research through ONR grant # N000014-08-1-1149 and the National Science Foundation under grant # 0415190.

CHAPTER 7. POINT DISTRIBUTION ALGORITHM AND ITS EXTENSION IN SEQUENTIAL DATA CLUSTERING

7.1. From Histogram-based Vector-Item Pattern Mining Algorithms to Point Distribution Algorithm

To determine whether the distribution of the subset of objects that is selected by an item is not random [9], histogram-based vector-item pattern mining algorithms first summarize the distributions of the item set as well as the random sub samples through density histogram technique, and then compare the observed density histogram with the expected density histogram using the χ^2 goodness of fit test or effect size analysis techniques. In point distribution vector-item pattern mining algorithms, we first draw multiple random samples and create a population of Kullback-Leibler divergence values between each sample and the overall data set, and then use this population of Kullback-Leibler divergence values as reference, to determine whether the Kullback-Leibler divergence value between the item set under consideration and the overall data set significantly different from the null hypothesis of a random subset. In contrast to histogram-based item-vector pattern mining algorithms, the point distribution computes Kullback-Leibler divergence values of both the item set and the random samples directly towards the overall data set at the level of each data point [45], which explains the significant increase of accuracy against previous histogram-based vector-item pattern mining algorithms. The point distribution algorithm will be briefly described in Section 7.5. For greater details of point distribution algorithm, please refer to [45].

The point distribution algorithm provides us a very efficient and effective way to determine whether the distribution of a subset of objects is significantly different from its overall distribution. One natural extension of point distribution algorithms is to tailor it to compare the distributions of multiple populations.

[9] Which means it differs from the overall distribution

In this chapter, an algorithm is presented for clustering sequential data in which each unit is a collection of vectors. An example of such a type of data is speaker data in a speaker clustering problem. The algorithm first constructs affinity matrices between each pair of units, using a modified version of the point distribution algorithm. The subsequent clustering procedure is based on fitting a Gaussian mixture model on multiple random projection matrices. The final class label of each unit is determined by voting from the results of the random projection matrices.

7.2. Challenge Introduction

Clustering on sequential data, such as time series data, has recently attracted increasing attention. In many sequential clustering problems each unit in the data set is a collection of vectors rather than one individual vector of attributes. The speaker clustering problem, which was the topic of the ICMLA competition is an example of a sequential clustering problem, in which each time point in a speech segment is represented by a 26-dimensional vector, and each speech segment consists of about 130 such vectors. This section discusses an algorithm that was designed for the first task which was 2-class speaker clustering. Seven datasets were provided, each consisting of speech segments coming from two different speakers. Notice that although the proposed algorithm was developed for clustering speech segments from two different speakers, it can be used in many similar applications.

We start by constructing a similarity matrix between speech segments. For this purpose we use a modified version of the point distribution algorithm [45], which was initially developed for mining patterns between vector and item data. The distance between two speech segments is measured through the KL-divergence between the distribution of the vectors for each of the speech segments and the overall expected distribution. We will discuss the construction and comparison of the two distributions in detail shortly. The subsequent clustering procedure is based on fitting a Gaussian mixture model on

multiple random projection matrices. The final class label of each speech segment, i.e. the identification of the speaker, is determined by vote from the results of these random projection matrices.

7.3. Related Works

The key challenge of clustering speakers is to find a proper affinity or distance measure between speakers. One common approach is to measure their underlying probability density function (pdf). Among the most popular distance measures between pdfs are the KL-divergence [87] and Bhattacharya distance [18]. Jebara [77] developed a kernel that incorporates the advantages of discriminative learning algorithms and kernel machines.

The approach used in this study is based on concepts that were developed for finding patterns between vector and item data [134, 46, 47, 133]. In that work, the goal is to identify subsets of data, defined by the presence of some item, that have a distribution that differs significantly from the overall distribution in the data set. A recent version of such a vector-item mining algorithm [45] compares the distribution of the subset with that in the whole data set. The density of point in the full data set serves as the expected density for the data point. The underlying PDF is unexpected if the KL-divergence between the pdf and the whole data set is significantly different from what would be expected by random chance alone.

7.4. Task Description and Data Preprocessing

The first task in the ICMLA 2010 Speaker Clustering Challenge includes seven data sets. Each data set contains speech segments coming from two different speakers, each speech segment comprises approximately 130 time points, and each time point is characterized by 26 parameters. Our task is to group the speech segments in each of the seven data sets into two clusters, representing the two different speakers.

As a preprocessing step, we apply a row-wise z-score normalization to the complete

data set:

$$X_k^{(i,j)} = \frac{X_k^{(i,j)} - \overline{X^j}}{\sigma(X^j)} \tag{82}$$

where $X_k^{(i,j)}$ is the k^{th} value of the j^{th} parameter ($j \in [1, 26]$) in the i^{th} speech segment, and $\overline{X^j}$ and $\sigma(X^j)$ are the mean and standard deviation, respectively, of the j^{th} parameter for all speech segments combined.

We further do a column-wise z-score normalization

$$X_k^{(i,j)} = \frac{X_k^{(i,j)} - \overline{X_k^i}}{\sigma(X_k^i)} \tag{83}$$

where $\overline{X_k^i}$ and $\sigma(X_k^i)$ are the mean and standard deviation of k^{th} column in the 26-dimensional data set of time points within the i^{th} speech segment respectively. Besides, each data point is normalized to a unit hype-sphere, i.e., $|\mathbf{X}| = 1$, so that each data point is positioned on the surface of the unit hype-sphere.

7.5. Clustering Based on Point Distribution Algorithm

The proposed algorithm has two steps. In the first step we modify the point distribution algorithm to construct an affinity matrix, on which the subsequent clustering step is performed.

7.5.1. Construction of The Affinity Matrix

7.5.1.1 Distance measurement The idea behind the point distribution algorithm [45] is to compare subset distributions with respect to an overall distribution to identify those subsets that are more inhomogeneous than would be expected by random chance.

Fig 1.1 illustrates the concept. Objects are characterized by vector data (position in plane) and an item that determines the subset membership (circles that are solid black). The angular distribution of those objects that have the item, as well as the overall distribution, are shown around the perimeter. The left panel shows an example of a subset distribution that differs significantly from the overall distribution, while the right panel shows a case,

in which the subset distribution follows the overall distribution closely. Notice that the pdf of the subset objects (circles solid black) in the left panel differs clearly from that of the whole data set.

To determine whether the pdf of the subset significantly differs from the overall pdf, the point distribution algorithm first calculates the KL-divergence between the subset pdf and the overall pdf (denoted as KL_{item}). Then multiple random subsets are extracted, which have the same size as the subset that is defined by the item data, and the KL-divergence is calculated between each random subset and the whole data set (denote it as KL^i_{sample} for the i^{th} sample). If KL_{item} differs more than two standard deviations from the mean of the KL^i_{sample}s, then the pdf of the item data is considered significant.

We modify the point distribution algorithm to construct an affinity matrix between speech segments of each data set. For the data set "challenge_data_1", which comprises 50 speech segments, the affinity matrix is a $50 * 50$ matrix, in which each entry (i, j) indicates the similarity between speech segments i and j.

To calculate the affinity between speech segments i and j ($i \neq j$) in each data set, we first temporarily construct a sub-dataset $S^{\{i,j\}}$, $S^{\{i,j\}} = S^i \cup S^j$ where S^i and S^j are the two 26-dimensional data sets for the i^{th} and j^{th} speech segments respectively. $S^{\{i,j\}}$ corresponds to the whole data set as in Fig. 1.1.

To determine whether speech segments i and j are from different speakers, i.e., S^i and S^j are drawn from different distributions, we randomly select N (N is equal to 200 in this study) samples from data sets S^i and S^j separately. Each sample has a size of $|S^i|/4$ or $|S^j|/4$, respectively. Suppose the two sets of samples are Φ^i and Φ^j, and Φ^i_m and Φ^j_m ($m \in [1 \quad N]$) are the m^{th} sample for the i^{th} and j^{th} speech segment respectively. Each sample corresponds to one subset defined by an item as shown in solid black in Fig. 1.1.

Another two sets of N random samples are selected from the aggregated data set $S^{\{i,j\}}$ accordingly. In one set (denoted as Ψ^i), each sample has a size of $|S^i|/4$, and each

sample in the other set (denoted as Ψ^j) has a size of $|S^j|/4$. Similarly, suppose Ψ^i_m and Ψ^j_m ($m \in [1, N]$) are the m^{th} samples for the i^{th} and j^{th} speakers respectively.

Similar to the point distribution algorithm, we calculate the KL-divergence between each sample (in Φ^i, Φ^j, Ψ^i and Ψ^j) and the whole data set $S^{\{i,j\}}$, denoted as $D_{KL}(\Phi^i_m||S^{\{i,j\}})$, $D_{KL}(\Phi^j_m||S^{\{i,j\}})$, $D_{KL}(\Psi^i_m||S^{\{i,j\}})$ and $D_{KL}(\Psi^j_m||S^{\{i,j\}})$ for each sample m respectively. This yields 4 new distributions of KL-divergence values for sets Φ^i, Φ^j, Ψ^i and Ψ^j. $D_{KL}(\Psi^j_m||S^{\{i,j\}})$ ($m \in [1, N]$) serves as expected distribution to $D_{KL}(\Phi^j_m||S^{\{i,j\}})$, so does $D_{KL}(\Psi^i_m||S^{\{i,j\}})$ to $D_{KL}(\Phi^i_m||S^{\{i,j\}})$.

Note that the greater the difference between the distributions $D_{KL}(\Psi^j_m||S^{\{i,j\}})$ and $D_{KL}(\Phi^j_m||S^{\{i,j\}})$ and between $D_{KL}(\Psi^i_m||S^{\{i,j\}})$ and $D_{KL}(\Phi^i_m||S^{\{i,j\}})$, the more likely speech segments i and j are actually from different people. Thus the distance between speech segments i and j is modeled by:

$$\begin{aligned} Dis_{\{i,j\}} &= D_{KL}(\{D_{KL}(\Phi^i_m||S^{\{i,j\}})\}_{m=1}^N \\ &\quad ||\{D_{KL}(\Psi^i_m||S^{\{i,j\}})\}_{m=1}^N) \\ &+ D_{KL}(\{D_{KL}(\Phi^j_m||S^{\{i,j\}})\}_{m=1}^N \\ &\quad ||\{D_{KL}(\Psi^j_m||S^{\{i,j\}})\}_{m=1}^N) \end{aligned} \tag{84}$$

where N is the number of samples.

7.5.1.2 Estimation of the pdf For the pdf estimation we use a uniform kernel with cosine similarity as similarity measure. If the cosine similarity between two vectors exceeds a threshold τ, they are considered to be neighbors.

Fig. 7.1 shows an example, in which τ is the cosine similarity threshold, and vectors $\mathbf{X}_k$, $\mathbf{X}_p$, and $\mathbf{X}_q$ represent three data points. In this example, vector $\mathbf{X}_p$ is a neighbor of vector $\mathbf{X}_k$, while vector $\mathbf{X}_q$ is not. The highlighted segment on the hype-sphere is the

neighborhood region for vector $\mathbf{X}_k$, i.e., if another vector is within this region, it is a neighbor of vector $\mathbf{X}_k$.

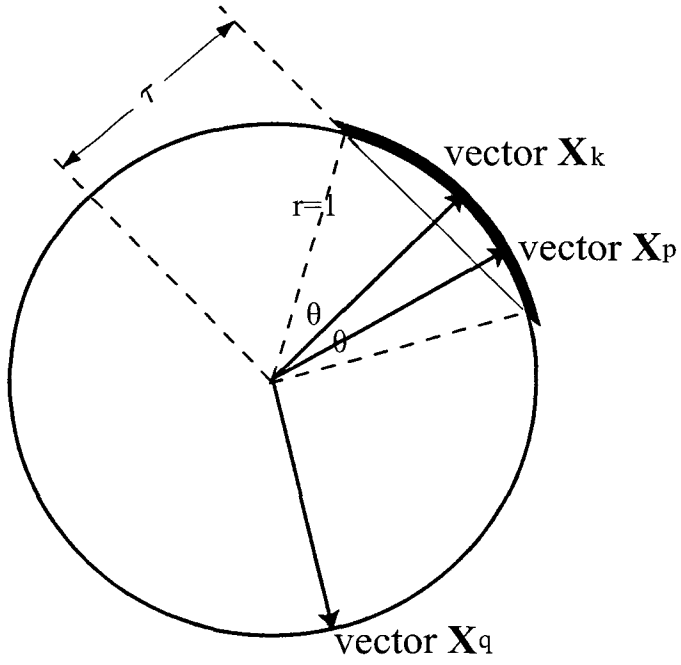


Figure 7.1: An example of three vectors.τ is the cosine similarity threshold, vector $\mathbf{X}_k$, vector $\mathbf{X}_p$ and vector $\mathbf{X}_q$ are three points. In this example, vector $\mathbf{X}_p$ is a neighbor of vector $\mathbf{X}_k$ while vector $\mathbf{X}_q$ is not. The highlighted segment on the hype-sphere is the neighborhood region for vector $\mathbf{X}_k$.

For each data point $\mathbf{X}_k$ in $S^{\{i,j\}}$, the expected density is estimated using a uniform kernel [45] since Gaussian kernels are known to be problematic in high dimensions [129]:

$$\hat{f}(X_k) = \frac{1}{|S^{\{i,j\}}|} \sum_{p=1}^{|S^{\{i,j\}}|} K(X_k, X_p) \tag{85}$$

where $K(.)$ is the uniform kernel function:

$$K(X_k, X_p) = \frac{\theta(X_k \cdot X_p - \tau)}{R_d} \tag{86}$$

and τ is the cosine similarity threshold, θ is the Heaviside step function and R_d

is the section of the d-hypersphere surface such that the kernel function is appropriately normalized:

$$\int K(X_k, X)dX = R_d \tag{87}$$

Let A^{cap} be the area of the neighborhood region of a vector X_k. Let A_d be the area of the unit hype-sphere in d dimensions. The area of the cap can be calculated as follows:

$$\begin{aligned} A^{cap} &= \int_{-a\cos\tau}^{a\cos\tau} \sin^{d-2}(\varphi_1)d\varphi_1 \int_0^{\pi} \sin^{d-3}(\varphi_2)d\varphi_2 \\ &\quad \ldots \int_0^{\pi} \sin(\varphi_{d-2})d\varphi_{d-2} \int_0^{2\pi} d\varphi_{d-1} \\ &= \Omega * \int_{-a\cos\tau}^{a\cos\tau} \sin^{d-2}(\varphi_1)d\varphi_1 \end{aligned} \tag{88}$$

where $\phi_1, \phi_2 \ldots \phi_{d-1}$ are angular coordinates, and $\Omega = \int_0^{\pi} \sin^{L-3}(\varphi_2)d\varphi_2 \ldots \int_0^{\pi} \sin(\varphi_{d-2})d\varphi_{d-2} \int_0^{2\pi} d\varphi_{d-1}$. Notice that the probability of two vectors X_k and X_p being neighbors is

$$p = \frac{A^{cap}}{A_d} = \frac{\Omega * \int_{-a\cos\tau}^{a\cos\tau} \sin^{d-2}(\varphi_1)d\varphi_1}{n * C_d} \tag{89}$$

where $C_d = \begin{cases} \frac{\pi^{(d/2)}}{(d/2)!} & \text{for even d} \\ \frac{2^{(d+1)/2}\pi^{(d-1)/2}}{d!!} & \text{for odd d} \end{cases}$

Equation (89) can be simplified:

$$p = \frac{A_d}{A} = \Omega' * \int_{-a\cos\tau}^{a\cos\tau} \sin^{d-2}(\varphi_1)d\varphi_1 \tag{90}$$

where $\Omega' = \frac{\Omega}{d * C_d}$. To improve performance, p can be saved to a table TB for different values of d and different thresholds τ. Because τ is of type float, we may need to find the closest representative in the table. From (90), we notice p is a function of cosine similarity threshold τ and vector dimensionality d.

Given a sample $\bar{S}$ (one from Φ^i, Φ^j, Ψ^i or Ψ^j in equation (84)), to determine how

unexpected the pdf of the sample $\bar{S}$ is, the point distribution algorithm first computer the KL divergence between $\bar{S}$ and $S^{\{\imath,\jmath\}}$:

$$D_{KL}^{(d)}(\bar{S}||S^{\{\imath,\jmath\}}) = \int p(X_k) \log \frac{p(X_k)}{q(X_k)} d^d X_k \tag{91}$$

where $D_{KL}^{(d)}(\bar{S}||S^{\{\imath,\jmath\}})$ is KL divergence between $\bar{S}$ and $S^{\{\imath,\jmath\}}$, $p(X_k)$ and $q(X_k)$ are the observed density and the expected density of vector $\mathbf{X}_k$ within $\bar{S}$ respectively. Using the law of large numbers, equation (91) can be approximated by:

$$D_{KL}^{(d)}(\bar{S}||S^{\{\imath,\jmath\}}) = \frac{1}{|\bar{S}|} \sum_{k=1}^{|\bar{S}|} \log \frac{p(X_k)}{q(X_k)} \tag{92}$$

Since we are using uniform kernel (equation (85)),

$$q(X_k) = \frac{|\bar{S}|}{|S^{\{\imath,\jmath\}}|} * \hat{f}(X_k) \tag{93}$$

where $\hat{f}(X_k)$ is the expected density of vector $\mathbf{X}_k$ w.r.t. the whole data set $S^{\{\imath,\jmath\}}$ and the prefix $\frac{|\bar{S}|}{|S^{\{\imath,\jmath\}}|}$ is the relative support for subset $\bar{S}$. Since $p(X_k)$ and $q(X_k)$ are calculated from the same neighborhood region of vector X_k, i.e., the cap that is segmented by the cosine similarity threshold τ, as shown in Fig. 7.1, we can simplify $\frac{p(X_k)}{q(X_k)}$ to the ratio of the observed number of neighbors against the expected one for each vector $\mathbf{X}_k \in \bar{S}$.

$$D_{KL}^{(d)}(\bar{S}||S^{\{\imath,\jmath\}}) = \frac{1}{|\bar{S}|} \sum_{k=1}^{|\bar{S}|} \sum_{p=1}^{|\bar{S}|} \frac{\theta(X_k \cdot X_p - \tau)}{\frac{|\bar{S}|}{|S^{\{\imath,\jmath\}}|} \sum_{q=1}^{S^{\{\imath,\jmath\}}} \theta(X_k \cdot X_q - \tau)} \tag{94}$$

To be more efficient, we can first calculate the term $\sum_{q=1}^{S^{\{\imath,\jmath\}}} \theta(X_k \cdot X_q - \tau)$ for each point X_k ($X_k \in \bar{S}$) as a preprocessing step.

In [45] we show that a good choice of kernel width assumes that only one data point is expected to be found in the hyper-volume. That means, the probability of two vectors X_k

and X_p (X_k,$X_p \in \bar{S}$) to be neighborhood is exactly $\frac{1}{|S|}$. Substituting it to p in equation (90), we can solve the equation to get the value of τ, or more efficiently, we can search table TB to get the τ according to the value of p. After getting the cosine similarity threshold τ, we can summarize the number of neighbors for each vector X_k ($X_k \in \bar{S}$) w.r.t. $\bar{S}$ and $S^{\{i,j\}}$ respectively (the Heaviside step functions in equation (94)), and then calculate the KL divergence using equation (94).

Thus, the KL-divergence between samples Φ^i, Φ^j, Ψ^i and Ψ^j in equation (84) and the whole data set $S^{\{i,j\}}$ can be calculated using equation (94). Moreover, the distance between speech segments i and j $Dis_{\{i,j\}}$ (equation (84)) can be calculated.

7.5.1.3 Construction of the affinity matrix Entry $Dis_{\{i,j\}}$ of the affinity matrix is the distance between speech segments i and j, while row $Dis_{\{i,\}}$ indicates the distances between the i^{th} speech segment and all the others. The correlation between rows $Dis_{\{i,\}}$ and $Dis_{\{j,\}}$ hence measures the distance between the i^{th} and the j^{th} speech segment by considering their distances with all the others. This can be viewed as evaluating the distance between speech segments based on shared similarities. Thus we further transform the distance matrix Dis by taking correlations between each pairs of rows, hoping to enhance the within-class similarity and between-classes dissimilarity:

$$Sim_{\{i,j\}} = corr(Dis_{\{i,\}}, Dis_{\{j,:\}}) \quad \{\forall i,j \in [1,|Dis|]\} \tag{95}$$

where *corr* is a Poisson correlation function, Sim is a similarity matrix, $Dis_{i,:}$ and $Dis_{j,:}$ are the i^{th} and j^{th} rows of matrix Dis respectively, using Matlab notation. Notice that Sim is not considered as a symmetric matrix but rather as a new data matrix where each column is a new feature. The subsequent clustering procedure is performed on matrix Sim. Experiment (Section 7.7) results show that using the correlation to form a new similarity matrix increases the accuracy of the presented algorithm.

Algorithm 13: Sampling Point Distribution Algorithm

Data: S; /* The data matrix for a data set */
Result: Sim; /* similarity matrix between speakers */

1 **for** *i=1:# of speakers* **do**
2 **for** *j=1:# of speakers* **do**
3 **if** *i=j* **then**
4 continue;
5 Construct $S^{\{i,j\}}$;
6 **foreach** $k \in [1, S^{\{i,j\}}]$ **do**
7 calculate expected density $q(X_k)$;
8 **for** *m=1:N* **do**
9 select samples Φ_m^i, Φ_m^j, Ψ_m^i and Ψ_m^j;
10 **foreach** k^{th} *data point* $\in \Phi_m^i$, Φ_m^j, Ψ_m^i *and* Ψ_m^j **do**
11 calculate densities $p(X_k^{\Phi_m^i})$, $p(X_k^{\Phi_m^j})$, $p(X_k^{\Psi_m^i})$ and $p(X_k^{\Psi_m^j})$;
12 **foreach** m^{th} *sample* $\in \Phi^i$, Φ^j, Ψ^i *and* Ψ^j **do**
13 calculate each samples KL-divergence w.r.t. $S^{\{i,j\}}$: $D_{KL}(\Phi_m^i||S^{\{i,j\}})$, $D_{KL}(\Phi_m^j||S^{\{i,j\}})$ $D_{KL}(\Psi_m^i||S^{\{i,j\}})$, $D_{KL}(\Psi_m^j||S^{\{i,j\}})$;
14 using equation (94);
15 calculate $Dis_{i,j}$ using equation (84);
16 **for** *i=1:# of speakers* **do**
17 **for** *j=1:# of speakers* **do**
18 calculate $Sim_{i,j}$ using equation (95);

Alg. 13 summarizes the modified point distribution algorithm. The input of the algorithm S is one single data set in Task 1 of the Challenge. For each pair of speech segments i and j, Alg. 13 first constructs a new data set $S^{\{i,j\}}$(Line 5). Then the expected density for each data point $\mathbf{X}_k \in S^{\{i,j\}}$ is calculated. As in the denominator in equation (94), $\sum_{q=1}^{S^{\{i,j\}}} \theta(X_k \cdot X_q - \tau)$, we simplify the problem of calculating density to that of summarizing the number of neighbors. Four groups of N number of subsamples are then selected from $S^{\{i,j\}}$ (Line 8 and 9).Φ_m^i and Φ_m^j are sampled from the i^{th} and j^{th} speech segments separately, while Ψ_m^i and Ψ_m^j are sampled from the aggregated data $S^{\{i,j\}}$ as mentioned in section 7.5.1.1. The KL-divergence values for each subsample w.r.t. $S^{\{i,j\}}$ are then calculated using equation (94) (Lines 10 to 14). The distance between each pair of speech

segments is then calculated using equation (84) (Line 15). Finally, the similarity matrix between speech segments is computed by taking row-wise correlation from the distance matrix (Lines 16 to 18).

7.6. Clustering

We try to estimate a Gaussian mixture model that best fits the new data matrix. However, in order to improve accuracy, we adopt a clustering ensemble method, i.e., we first perform a random projection [60] on matrix Sim:

$$Sim^t = \frac{1}{\sqrt{q}} Sim * P \quad t \in [1, M] \tag{96}$$

where *Sim* is the affinity matrix (equation (95)), Sim^t is the t^{th} ($t \in [1, M]$) new projection matrix, M is the number of projection matrices, q is the desired number of dimensions and P is a random matrix, in which the number of rows equals the number of speech segments and the number of columns equals q. Each entry of P is either 1 or -1 with probability of 0.5.

For the first task in the challenge, we know there are exactly 2 clusters in each data sets. We can easily fit a q-component (q is equal to 3 for all data sets in this study) Gaussian mixture model of order 2. We would like to take the chance to thank [24] for providing nice software that uses the EM algorithm to fit a Gaussian mixture model. Our final results are then bagged from all the clustering results of Sim^t ($t \in [1, M]$):

$$\begin{aligned} &Class_i = \theta(\sum_{t=1}^{M} C_i^t - Thr) \quad s.t. \\ &\sum_{i=1}^{size(C,1)} \theta(1 - C_i^p + C_i^q) \geq \tfrac{M}{2} \qquad \forall p, q \in [1, \quad M] \end{aligned} \tag{97}$$

where $Class_i$ ($Class_i \in \{0, 1\}$) is the final predicted class for the i^{th} speech segment, $\theta(.)$ is Heaviside function, Thr is a threshold for determining which class a speech segment belongs to, C is a matrix where the number of rows is equal to the number of speech

segments and the number of columns is equal to the number of projected matrices and to which we will refer as clustering result matrix. C_i^t is the predicted class for the i^{th} speech segment from the clustering result of the t^{th} projected matrix Sim^t, and $size(C,1)$ is the number of speech segments using Matlab notation. $\sum_{i=1}^{size(C,1)} \theta(1 - C_i^p + C_i^q) \geq \frac{M}{2}$ is to make sure that the clustering results are indexed in the same way (0 or 1). This constraint may not always be satisfied. We used a simple heuristic approach to make certain that most of results are matched. The input of Alg. 14 is the clustering result matrix C. For each result of projection matrix, Alg. 14 first calculate the total number of class labels for all speech segments that it matches with the results of other projection matrices (Lines 1 to 8), the find the result of a projection matrix, "max_index", that matches most of the others [Line 9]. For each projection matrix, if it does not match more that half of class labels for all speech segments with the "max_index", then just inverse class labels [Lines 10 to 16].

The threshold Thr in equation (97) is thus estimated as:

$$Thr = \arg\min_{thr}\left(\left|\theta\left(\sum_{t=1}^{M} C_i^t - thr\right) - \frac{1}{M}\sum_{t=1}^{M}\sum_{i=1}^{size(C,1)} C_i^t\right|\right) \tag{98}$$

where M is the number of projected matrix and $size(C,1)$ is the number of speech segments. Thr in equation (98) can be easily attained using an interval-halving algorithm.

The intuition of using an ensemble-based approach that aggregates clustering results of multiple projection matrices is that the boundary between clusters may reside in different subspaces. By using ensembles of projection matrices, the structure of the boundary can be more accurately captured. Similar techniques are used in Random Forest Classification algorithm [123].

7.7. Experiments

7.7.1. Data Setup

Besides the first task of the challenge, we also perform the presented clustering algorithm on 8 data sets from UCR time series repository [84], namely, *buoy sensor*,

Algorithm 14: Adjust resulting matrices

```
Data: C;  /* clustering results of each projected matrix */
Result: C;                                  /* clustering results */
T = zeros(1,size(C,1));  /* vector of zeros.  size(C,1) is the
number of speech segment */
for i=1:size(C,2) do
    for j=1:size(C,2) do
        t=0;
        for k=1:size(C,1) do
            if C_k^i = C_k^j then
                t=t+1;
    T(i)=t;
max_index= find index of max(T);
for i=1:size(C,2) do
    t=0;
    for k=1:size(C,1) do
        if C_k^i = C_k^{max_index} then
            t=t+1;
    if t ≤ M/2 then
        C^i = 1 − C^i;
```

balloon, *glass furnace*, *steamgen*, *speech*, *earth quake*, *ocean*, and *darwin*. The *buoy sensor* series was preprocessed by averaging over 4 consecutive values and the ecg series by averaging over 20 consecutive values. Since we focus on the first task of the challenge, in which each data set includes the speech data from two speakers, we combine two different data sets from UCR time series repository into data sets to imitate speech data as in the First Task, leaving us 36 artificial speech data sets.

Fig. 7.2 shows a schematic for constructing a speech data set from all pairs of data sets we consider.

In Fig. 7.2, for both time series "darwin" and "ocean", after taking the difference between adjacent points in each time series, a sliding window of size M-26, where M is the length of original time series, is used to create 26 new times series for both "darwin" and

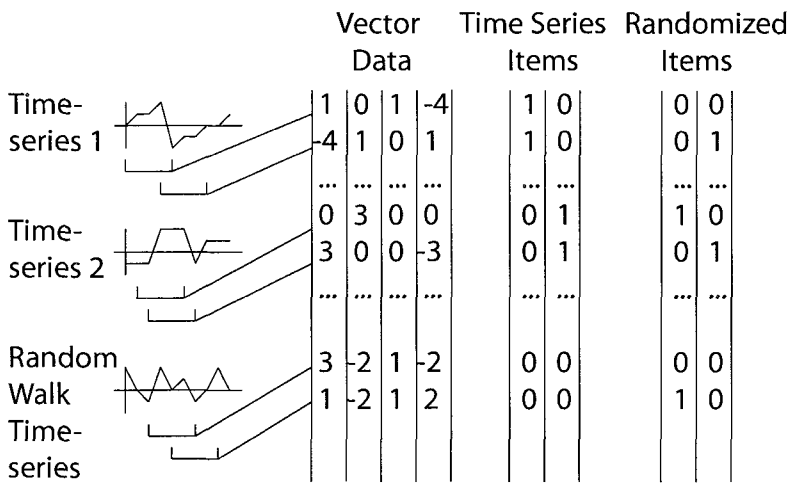


Figure 7.2: A schematic on constructing a simulated data set from time series data.

"ocean", Then we partition each group of 26 time series into N=25 parts of the same length. Each part imitates a speech segment, such as "Speech 1" shown in Fig. 7.2. Speech 1 to N will have the same Class label, since they come from the same time series ("darwin"), so do speech segments N+1 to 2N.

7.7.2. Comparison Algorithms

We try to analyze the effect of the two individual steps, i.e., affinity matrix construction and actual clustering, for the performance of the proposed speaker clustering algorithm. Furthermore, we also analyze the effect of taking row-wise correlations in the affinity matrix as in equation (95).

For the affinity matrix construction step, we compare the modified point distribution algorithm with anther algorithm [141](using KL-divergence). For the actual clustering step, we compare the proposed "random projection plus fitting Gaussian mixture model" procedure with Spectral Clustering algorithm [103];

Besides, we also perform Support Vector Machine algorithm (using libSvm [31]) on each affinity matrix. We split each affinity matrix to a training set and a test set, both having a size of half the number of rows in the affinity matrix. We test both linear kernel

and radial basis function, with a set of parameters $gamma \in \{0.001, 0.01, 0.05, 0.1, 1\}$ and $C \in \{1000, 200, 1, 0.1, 0.01\}$. The comparison results are shown in Table. 7.1.

Table 7.1: Comparison of algorithms

Algorithm	Accuracy
{Algorithm [141] + NC + SP }	0.9910
{Algorithm [141] + NC + RP&GM}	0.9858
{Algorithm [141] + C + SP}	0.9797
{Algorithm [141] + C + RP&GM}	0.9948
{MPD + NC + SP}	0.8435
{MPD + NC + $RP\&GM$}	0.9594
{MPD + C + SP}	0.9929
{MPD + C + $RP\&GM$}	0.9949
{Algorithm [141] + NC + SVM}	0.6471
{Algorithm [141] + C + SVM}	0.9982
{MPD + NC + SVM}	0.7556
{MPD + C + SVM}	0.9978
MPD:Modified Point Distribution $RP\&GM$:"random projection plus fitting Gaussian mixture model" procedure SP:Spectral Clustering NC:No correlation C:Correlation	

In table 7.1, we can see that the best two clustering algorithms are {Algorithm [141] + Correlation + "random projection plus fitting Gaussian mixture model"} and {Modified Point Distribution + Correlation + "random projection plus fitting Gaussian mixture model"}. The latter one is slightly better. Notice that from the SVM classification results, the affinity matrices, for which row-wise correlation was performed, have much higher accuracy. Overall speaking, using the "random projection plus fitting Gaussian mixture model" algorithm to do the actual clustering step is slightly better than using Spectral Clustering.

7.8. Conclusion

A clustering algorithm that is based on the Point Distribution algorithm is proposed

in the study. The algorithm has two steps. In the first step, a similarity matrix for the speech segments is constructed using sampling of point distributions. In the subsequent clustering step, we create multiple random projection matrices, each of which is fit on a Gaussian mixture model. The final class label of each speech segment is derived by voting from the results on these random projection matrices. The proposed algorithm does not require users to set any parameters. The only free parameter in the sampling-based point distribution algorithm is provided through a theoretically justified choice. Experimental results show that the proposed algorithm is comparable with available state-of-the-art algorithms.

REFERENCES

[1] J. Abello, M. G. Resende, and S. Sudarsky. Massive quasi-clique detection. In *In Latin American Theoretical Informatics (LATIN)*, pages 598–612, 2002.

[2] C. Aggarwal. Re-designing distance functions and distance-based applications for high dimensional data. *SIGMOD Record*, 30(1):13–18, 2001.

[3] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. *Lecture Notes in Computer Science*, 1973:420–434, 2001.

[4] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data. *Data Mining and Knowledge Discovery Journal*, 11(1):5–33, July 2005.

[5] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., May 26-28 1993. ACM Press.

[6] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[7] K. AI and C. JA. *Response Surface: Designs and Analyses (2nd edn)*. Dekker, 1996.

[8] S. E. Anderson. Bit twiddling hacks, http://graphics.stanford.edu/ seander /bithacks.html.countbitssetnaive.

[9] M. Ankerst, M. M. Breunig, H. peter Kriegel, and J. S. Optics: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 49–60. ACM Press, 1999.

[10] O. Z. Anne and O. R. Zaïane. Mining recurrent items in multimedia with progressive resolution refinement. pages 461–470, 2000.

[11] J. Antony, R. B. Anand, M. Kumar, and M. Tiwari. Multiple response optimization using taguchi methodology and neuro-fuzzy based model. *Journal of Manufacturing Technology Management*, 17(8):908–925, 2006.

[12] S. Asogawa and N. Akamatsu. An efficient algorithm for clustering data and best model selection using aic. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference*, 1994.

[13] N. Balakrishnan. *Handbook of the Logistic Distribution.* Dekker, 1991.

[14] Z. Bar-Joseph, G. Gerber, T. Jaakkola, D. Gifford, and I. Simon. Continuous representations of time series gene expression data. *J. Comput. Biol.*, 10(3-4), 2003.

[15] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *RECOMB '02: Proc 6th Annual Int'l Conf. on Computational Biology*, New York,NY, 2002.

[16] A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. In *Pacific Symposium on Biocomputing*, pages 6–17, 2002.

[17] R. Benden and S. Lange. Adjusting for multiple testing–when and how? *Journal of Clinical Epidemiology*, 54:343–349, April 2001.

[18] Bhattacharyya.A. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–109, 1943.

[19] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *Proceedings of the 2006 international workshop on Mining software repositories*, MSR '06, pages 137–143, New York, NY, USA, 2006. ACM.

[20] T. Bittner and B. Smith. A taxonomy of granular partitions. In *Proceedings of the International Conference on Spatial Information Theory: Foundations of Geographic Information Science*, COSIT 2001, pages 28–43, London, UK, UK, 2001. Springer-Verlag.

[21] V. Boginski, S. Butenko, and P. M. Pardalos. Mining market data: a network approach. *Comput. Oper. Res.*, 33(11):3171–3184, 2006.

[22] N. Bolshakova, F. Azuaje, and P. Cunningham. A knowledge-driven approach to cluster validity assessment. *Bioinformatics*, 21(10):2546–7, 2005.

[23] B. Bolstad, R. Irizarry, M. Astrand, and T. Speed. A comparison of normalization methods for high density oligonucleotide array data based on bias and variance. *Bioinformatics*, 19(2):185–193, 2003.

[24] C. A. Bouman. https://engineering.purdue.edu/ bouman/software/cluster/.

[25] P. S. Bradley, U. M. Fayyad, C. A. Reina, P. S. Bradley, U. Fayyad, and C. Reina. Scaling em (expectation-maximization) clustering to large databases, 1999.

[26] J. Branke. Evolutionary algorithms for neural network design and training. In *IN PROCEEDINGS OF THE FIRST NORDIC WORKSHOP ON GENETIC ALGORITHMS AND ITS APPLICATIONS*, pages 145–163, 1995.

[27] L. Breiman and E. Schapire. Random forests. In *Machine Learning*, pages 5–32, 2001.

[28] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: generalizing association rules to correlations. In *SIGMOD '97: Proc. of the 1997 ACM SIGMOD Int'l Conf. on Management of Data*, pages 265–276, New York, NY, USA, 1997. ACM Press.

[29] M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. Furey, M. Ares, J. q, and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci.*, 97(1):262–267, 2000.

[30] W. R. Carlson and B. Skagerberg. Statistical optimization as a means to reduce risks in industrial processes. *The Environmental Professional*, 11:127–131, 1989.

[31] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines, 2001.

[32] H.-H. Chang. A data mining approach to dynamic multiple responses in taguchi experimental design. *Expert Syst. Appl.*, 35(3):1095–1103, 2008.

[33] J. H. Chang and W. S. Lee. Finding recent frequent itemsets adaptively over online data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge descovery and data mining*, pages 487–492, 2003.

[34] C. Cheadle, M. Vawter, W. Freed, and K. Becker. Analysis of microarray data using z score transformation. *J. Mol. Diagn.*, 5(2), 1999.

[35] M.-S. Chen, J. Han, and P. S. Yu. Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8:866–883, 1996.

[36] S.-C. Chen, M.-L. Shyu, C. Zhang, and J. Strickrott. Multimedia data mining for traffic video sequences. In *MDM/KDD*, pages 78–86, 2001.

[37] Y. Cheng and G. Church. Biclustering of expression data. In *Proc. 8th Int'l Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 93–103, 2000.

[38] R. Chiang, C. H. Cencil, and E.-P. Lim. Linear correlation discovery in databases: a data mining approach. *Data and Knowledge Engineering*, 53:311–337, 2005.

[39] D. Chudova, C. Hart, E. Mjolsness, and P. Smyth. Gene expression clustering with functional mixture models. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2003.

[40] R. Coelho, S. Hutzler, P. Repetowicz, and P. Richmond. Sector analysis for a ftse portfolio of stocks. *Physica A Statistical Mechanics and its Applications*, 373:615–626, Jan. 2007.

[41] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Routledge Academic, July 1988.

[42] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[43] W. H. E. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1:7–24, 1984.

[44] A. Denton and A. Kar. Finding differentially expressed genes through noise elimination. In *Proc. Data Mining for Biomedical Informatics workshop in conjunction with the 7th SIAM Int'l Conf. on Data Mining*, Minneapolis, MN, April 2007.

[45] A. Denton, J. Wu, and D. H. Dorr. Point-distribution algorithm for mining vector-item patterns. In *Useful Patterns Workshop in conjunction with the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul 2010.

[46] A. M. Denton and J. Wu. Data mining of vector-item patterns using neighborhood histograms. *Knowledge and Information System*, 3:819–844, 2009.

[47] A. M. Denton, J. Wu, M. K. Townsend, P. Sule, and B. M. Prüß. Relating gene expression data on two-component systems to functional annotations in *escherichia coli*. *BMC Bioinformatics*, 9, 2008.

[48] P. Domingos. Mining Social Networks for Viral Marketing. *IEEE Intelligent Systems*, 20(1):80–82, 2005.

[49] P. Domingos and M. Richardson. Mining the network value of customers. In *In Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, pages 57–66. ACM Press, 2002.

[50] C. Dose and S. Cincotti. Clustering of financial time series with application to index and enhanced index tracking portfolio. *Physica A: Statistical Mechanics and its Applications*, 355(1):145–151, September 2005.

[51] S. Dudoit, Y. Yang, T. Speed, and M. Callow. Statistical methods for identifying differentially expressed genes in replicated cdna microarray experiments. *Statistica Sinica*, 12(1):111–139, 2002.

[52] M. Eirinaki and M. Vazirgiannis. Web mining for web personalization. *ACM Trans. Internet Technol.*, 3:1–27, February 2003.

[53] A. Ekin and D. C. Webster. Combinatorial and high-throughput screening of the effect of siloxane composition on the surface properties of crosslinked siloxane-polyurethane coatings. *Combinatorial Chemistry*, 9:178–188, 2007.

[54] A. Ekin, D. C. Webster, J. W. Daniels, S. J. Stafslien, F. Casse, J. A. Callow, and M. E. Callow. Synthesis, formulation and characterization of siloxane-polyurethane coatings for underwater marine applications using combinatorial high-throughput experimentation. *Coatings Tech*, 4:435–451, 2007.

[55] J. Ernst, G. Nau, and Z. Bar-Joesph. Clustering short time series gene expression data. *Bioinformatics*, 21(Supplement 1), 2005.

[56] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[57] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165:91–134, June 2005.

[58] R. Feldman and J. Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006.

[59] R. A. Fisher. *The Design of Experiments, 9th Edition,*. Macmillan, 1971.

[60] D. Fradkin and D. Madigan. Experiments with random projections for machine learning. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–522, New York, NY, USA, 2003. ACM.

[61] Y. Freund and R. Schapire. A short introduction to boosting. *Japonese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[62] B. Gao, O. Griffith, M. Ester, and S. Jones. Discovering significant opsm subspace clusters in massive gene expression data. In *Proc. 2006 ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, Philadelphia,PA, 2006.

[63] L. Z. Garamszegi. Comparing effect sizes across variables: Generalization without the need for bonferroni correction. *Behavioral Ecology*, 17(4):682–687, 2006.

[64] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani. Mining the stock market (extended abstract): which measure is best? In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 487–496, New York, NY, USA, 2000. ACM Press.

[65] D. George and S. Ronald. Simultaneous optimization of several response variables. *Journal of Quality Technology*, 12(4):214–219, 1980.

[66] A. Goldberger, L. Amaral, L. Glass, et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000. Circulation Electronic Pages: [http://circ.ahajournals.org/cgi/content/full/101/23/e215].

[67] A. Guénoche, P. Hansen, and B. Jaumard. Efficient algorithms for divisive hierarchical clustering with the diameter criterion. *Journal of Classification*, 8:5–30, 1991.

[68] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufman Publishers, 2nd edition, 2006.

[69] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *Sigmod Record*, 29:1–12, 2000.

[70] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[71] H. He, J. Chen, H. Jin, and S.-H. Chen. Stock trend analysis and trading strategy. In *JCIS*, pages 89–93, 2006.

[72] L. Hedges and I. Olkin. *Statistical Methods for Meta-Analysis*. Academic Press, New York, NY, 1985.

[73] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, July 2000.

[74] H. Hu, X. Yan, Y. Huang, J. Han, and X. J. Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 21(1):213–221, 2005.

[75] S. ichi Minato and H. Arimura. Frequent pattern mining and knowledge indexing based on zero-suppressed bdds. 4747:152–169, 2007.

[76] P. E. J. *Multiple regression in behavioral research: Explanation and prediction (2nd ed)*. New York: Holt, Rinehart and Winston.

[77] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *J. Mach. Learn. Res.*, 5:819–844, 2004.

[78] D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Trans. Knowl. Data Eng.*, 16(11):1370–1386, 2004.

[79] T. Joachims. Text categorization with support vector machines: Learning with many relevant features, 1998.

[80] P. Jonsson, K. Laurio, Z. Lubovac, et al. Using functional annotation to improve clusterings of gene expression patterns. In *Proc. 6th Joint Conference on Information Science*, pages 1257–1262, 2002.

[81] N. D. C. Jr, J. Cunha, and S. D. Silva. Stock selection based on cluster analysis. Finance 0509022, EconWPA, Sept. 2005.

[82] K. Kailing, H. Kriegel, P. Kroeger, and S. Wanka. Ranking interesting subspaces for clustering high dimensional data. In *Proc. PKDD Conference*, pages 241–252, 2003.

[83] S. Kaski, J. Sinkkonen, and J. Nikkilä. Clustering gene expression data by mutual information with gene function. In *Proc. Int'l Conf. on Artificial Neural Networks (ICANN)*, pages 81–86, 2001.

[84] E. Keogh and T. Folias. The ucr time series data mining archive, accessed 2003. [http://www.cs.ucr.edu/ ~eamonn/TSDMA/index.html].

[85] E. Keogh, J. Lin, and W. Truppel. Clustering of time series subsequences is meaningless: implications for previous and future research. In *Proc. IEEE Int'l Conf. on Data Mining*, pages 115–122, Melbourne, FL, 2003.

[86] R. Kosala and H. Blockeel. Web mining research: A survey. *SIGKDD Explorations*, 2:1–15, 2000.

[87] S. Kullback and R. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[88] T. Lange, V. Roth, M. L. Braun, and J. M. Buhmann. Stability-based validation of clustering solutions. *Neural Comput.*, 16(6):1299–1323, 2004.

[89] A. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes [microform]: second supplément.* 1820.

[90] Z. Li and Y. Zhou. Pr-miner: automatically extracting implicit programming rules and detecting violations in large software code. In *In Proc. 10th European Software*

Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE05, pages 306–315. ACM Press, 2005.

[91] N. Linial, A. Magen, and M. E. Saks. Low distortion euclidean embeddings of trees. *Israel Journal of Mathematics*, 106:339–348, 1998.

[92] F. Liu, P. Du, F. Weng, and J. Qu. Use clustering to improve neural network in financial time series prediction. In *ICNC '07: Proceedings of the Third International Conference on Natural Computation (ICNC 2007)*, pages 89–93, Washington, DC, USA, 2007. IEEE Computer Society.

[93] G. Liu and L. Wong. Effective pruning techniques for mining quasi-cliques. In *ECML PKDD '08: Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, pages 33–49, Berlin, Heidelberg, 2008. Springer-Verlag.

[94] B. Luo, R. Wilson, and E. Hancock. Spectral embedding of graphs. *Pattern Recognition*, 36:2213–2223, 2003.

[95] P. Majumdar, E. Lee, N. Patel, K. Ward, S. J. Stafslien, J. Daniels, B. J. Chisholm, P. Boudjouk, M. E. Callow, J. A. Callow, and S. E. Thompson. Combinatorial materials research applied to the development of new surface coatings ix: an investigation of novel antifouling/fouling-release coatings containing quaternary ammonium salt groups. *Biofouling*, 24(3):185–200, 2008.

[96] MATLAB. Documentation http://www.mathworks.com/access/helpdesk/help/toolbox/stats/chi2gof.html, accessed 02/07.

[97] K. R. McKeown, V. Hatzivassiloglou, R. Barzilay, B. Schiffman, D. Evans, and S. Teufel. Columbia multi-document summarization: Approach and evaluation. In *In Proceedings of the Document Understanding Conference (DUC01*, 2001.

[98] G. Miró-quesada, E. D. Castillo, and J. J. Peterson. A bayesian approach for multiple response surface optimization in the presence of noice variables. *Journal of Applied Statistics*, pages 251–270, 2004.

[99] A. L. Montgomery, S. Li, K. Srinivasan, and J. C. Liechty. Modeling online browsing and path analysis using clickstream data. *Marketing Science*, 23:579–595, 2004.

[100] N. Mulder, R. Apweiler, and T. Attwood. New developments in the interpro database. *Nucleic Acids Research*, 35:D224–228, 2007.

[101] S. Nakagawa and I. C. Cuthill. Effect size, confidence interval and statistical significance: a practical guide for biologists. *Biol Rev Camb Philos Soc*, 82(4):591–605, November 2007.

[102] K. Nakata, A. Voss, M. Juhnke, and T. Kreifelts. Collaborative concept extraction from documents. In *In Proceedings of the 2nd Int. Conf. on Practical Aspects of Knowledge management (PAKM 98*, pages 29–30, 1998.

[103] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2001.

[104] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 144–155, San Francisco, CA, USA, 1994.

[105] B.-W. On, E. Elmacioglu, D. Lee, J. Kang, and J. Pei. Improving grouped-entity resolution using quasi-cliques. In *ICDM '06: Proceedings of the Sixth International*

Conference on Data Mining, pages 1008–1015, Washington, DC, USA, 2006. IEEE Computer Society.

[106] J. P. Onnela, A. Chakraborti, K. Kaski, J. Kertész, and A. Kanto. Dynamics of market correlations: Taxonomy and portfolio analysis. *Phys. Rev. E*, 68(5):056110+, November 2003.

[107] J.-P. Onnela, J. Saramäki, K. Kaski, and J. Kertéesz. Financial market - a network perspective. In *Proceedings of the Third Nikkei Econophysics Symposium*, pages 302–306. Springer Tokyo, 2006.

[108] T. Oshima, H. Aiba, Y. Masuda, S. Kanaya, M. Sugiura, B. Wanner, H. Mori, and T. Mizuno. Transcriptome analysis of all two-component regulatory system mutants of *Escherichia coli* K-12. *Mol. Microbiol.*, 46(1):281–291, 2002.

[109] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2:1–135, January 2008.

[110] T. V. Perneger. What's wrong with bonferroni adjustments. *BMJ (Clinical research ed.)*, 316(7139):1236–1238, April 1998.

[111] D. C. Phillips. The three-dimensional structure of an enzyme molecule. *Scientific American*, 215(5):78–90, November 1966.

[112] G. Piatetsky-Shapiro. *Discovery, Analysis, and Presentation of Strong Rules.* AAAI/MIT Press, Cambridge, MA, 1991.

[113] V. Plerou, P. Gopikrishnan, B. Rosenow, L. Amaral, and H. Stanley. Collective behavior of stock price movements-a random matrix theory approach. *Physica A*, 299:175–180, October 2001.

[114] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[115] M. Ramezani, M. Bashiri, and A. C. Atkinson. A goal programming-topsis approach to multiple response optimization using the concepts of non-dominated solutions and prediction intervals. *Expert systems with applications*, (8):9557–9563, 2011.

[116] R. Rastogi and K. Shim. Mining optimized support rules for numeric attributes. *Information Systems*, 26(6):425–444, 2001.

[117] V. Roth, M. Braun, T. Lange, and J. Buhmann. A resampling approach to cluster validation. In *Computational Statistics (COMPSTAT'02)*, 2002.

[118] Saccharomyces Genome Database. Interproscan results using *s. cerevisiae* protein sequences ftp://genome-ftp.stanford.edu/pub/yeast/sequence_similarity/domains/domains.tab.

[119] P. P. shan Chen. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1:9–36, 1976.

[120] P. Spellman. Yeast cell cycle analysis project, http://cellcycle-www.stanford.edu/, accessed 2007.

[121] P. Spellman, G. Sherlock, M. Zhang, et al. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.

[122] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. 1996 ACM SIGMOD Int'l Conf. on Management of Data*, pages 1–12, Montreal, Quebec, Canada, 4–6 1996.

[123] L. B. Statistics and L. Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001.

[124] D. Stotts, J. Prins, L. Nyland, and T. Fan. Cobweb: Tailorable, analyzable rules for collaborative web use. Technical report, 1998.

[125] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 102:15545–15550, 2005.

[126] L.-I. Tong, C.-H. Wang, C.-C. Chen, and C.-T. Chen. Dynamic multiple responses by ideal solution analysis. *European Journal of Operational Research*, 156(2):433 – 444, 2004.

[127] K.-L. Tsui. Modeling and analysis of dynamic robust design experiments. *IIE Transactions*, 31:1113–1122, December 1999.

[128] R. Unal, R. A. Lepsch, M. L. Mcmillin, and H. Va. Response surface model building and multidisciplinary optimization using overdetermined d-optimal designs. Technical report, 1998.

[129] M. Verleysen and D. Franois. The curse of dimensionality in data mining and time series prediction. *Lecture Notes in Computer Science*, 3512, 2005.

[130] Z. Volkovich, Z. Barzily, and L. Morozensky. A statistical model of cluster stability. *Pattern Recognition*, 41(7):2174–2188, 2008.

[131] W. Wang, J. Yang, and R. Muntz. Sting: A statistical information grid approach to spatial data mining. pages 186–195. Morgan Kaufmann, 1997.

[132] G. Weikum and G. Vossen. *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery (The Morgan Kaufmann Series in Data Management Systems).* Morgan Kaufmann, 2001.

[133] J. Wu, A. Denton, O. el Ariss, and D. Xu. Mining for core patterns in stock market data. In *ICDM Workshops*, pages 558–563, 2009.

[134] J. Wu and A. M. Denton. Mining vector-item patterns for annotating protein domains. In *Proc. of the Workshop on Mining Multiple Information in conj. with the ACM SIGKDD Int'l Conf. on Data Mining (KDD)*, San Jose, Aug 2007.

[135] J. Wu, L. A. Nimer, O. A. Azzam, C. Chitraranjan, S. Salem, and A. M. Denton. Sequential data clustering. In *ICMLA*. IEEE Computer Society, 2010.

[136] F. Yates. Contingency table involving small numbers and the χ^2 test. *Journal of the Royal Statistical Society (Supplement)*, 1:217–235, 1934.

[137] K. Yeung, D. R. Haynor, and W. L. Ruzzo. Validating clustering for gene expression data. *Bioinformatics*, 17, 2001.

[138] G. Yona, W. Dirks, S. Rahman, and D. Lin. Effective similarity measures for expression profiles. *Bioinformatics*, 22(13):1616–1622, 2006.

[139] O. R. Zaïane, J. Han, Z.-N. Li, and J. Hou. Mining multimedia data. In *Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative research*, CASCON '98, pages 24–. IBM Press, 1998.

[140] Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis. Technical report, 2002.

[141] S. K. Zhou and R. Chellappa. From sample similarity to ensemble similarity: Probabilistic distance measures in reproducing kernel hilbert space. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(6):917–929, 2006.

[142] F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng. Mining colossal frequent patterns by core pattern fusion. In *ICDE*, pages 706–715, 2007.

APPENDIX A.

$$
\begin{aligned}
&F'(x;|S|-1,\lambda)|_{x_1}^{x_2} = \\
&(\sum_{\imath=x_1}^{x_2} |S| \begin{pmatrix} |S|-1 \\ \\ \imath \end{pmatrix} \lambda^{\imath}(1-\lambda)^{(|S|-1-\imath)})' = \\
&-(((x_2+1)\lambda^{(x_2-1)} + (x_2+1)\lambda^{(x_2-1)}(x_2-1) - \tfrac{|S|\lambda^{(x_2+1)}(x_2+1)}{\lambda}) \\
&(x_2+2-|S|)O_{12}/((x_2+1)!O_8(1-\lambda)^{(x_2+4-|S|)}(x_2+2)) \\
&+\tfrac{((x_2+1)\lambda_2^x-|S|\lambda^{(x_2+1)})(x_2+2-|S|)O_{12}(x_2+4-|S|)}{(x_2+1)!O_8(1-\lambda)^{(x_2+4-|S|)}(x_2+2)(1-\lambda)}+ \\
&((x_2+1)\lambda_2^x - |S|\lambda^{(x_2+1)})(x_2+2-|S|)(2x_2+6-2|S|) \\
&hypergeom([3,x_2+4-|S|],[x_2+4],\tfrac{\lambda}{\lambda-1})(\tfrac{1}{\lambda-1}-\tfrac{\lambda}{(\lambda-1)^2}) \\
&/((x_2+1)!O_8(1-\lambda)^{(x_2+4-|S|)}(x_2+2)(x_2+3) + (O_{10}\lambda^{(x_2-1)} \\
&+\tfrac{\lambda^{(x_2+2)}(x_2+2)(|S|-1)^2}{\lambda})O_7/((x_2+1)!O_8(1-\lambda)^{(x_2+4-|S|)})+ \\
&\tfrac{O_{11}O_7(x_2+4-|S|)}{(x_2+1)!O_8(1-\lambda)^{(x_2+4-|S|)}(1-\lambda)} + \tfrac{O_{11}(x_2+2-|S|)O_{12}(\frac{1}{\lambda-1}-\frac{\lambda}{(\lambda-1)^2})}{(x_2+1)!O_8(1-\lambda)^{(x_2+4-|S|)}(x_2+2)} \\
&-(O_5\lambda^{(x_1-2)} + \tfrac{(O_5\lambda-x_1^2)\lambda^{(x_1-2)}(x_1-2)}{\lambda} + \tfrac{(|S|-1)^2\lambda^{(x_1+1)}(1+x_1)}{\lambda} \\
&-2\tfrac{\lambda_1^x x_1 O_4}{\lambda})O_3/(x_1!O_1(1-\lambda)^{(x_1+3-|S|)}) - \tfrac{O_6(x_1+3-|S|)}{x_1!O_1(1-\lambda)^{(x_1+3-|S|)}} \\
&\tfrac{O_3}{(1-\lambda)} - \tfrac{O_6(x_1+1-|S|)O_2(\frac{1}{\lambda-1}-\frac{\lambda}{(\lambda-1)^2})}{x_1!O_1(1-\lambda)^{(x_1+3-|S|)}(x_1+1)} - (2x_1+4-2|S|) \\
&hypergeom([3,x_1+3-|S|],[x_1+3],\tfrac{\lambda}{\lambda-1})(\tfrac{1}{\lambda-1}-\tfrac{\lambda}{(\lambda-1)^2}) \\
&(x_1+1-|S|)(x_1\lambda^{(x_1-2)}\lambda - |S|\lambda_1^x)/((2+x_1)x_1!(1-\lambda)^{(x_1+3-|S|)} \\
&O_1(1+x_1)) - \tfrac{O_2(x_1+1-|S|)(x_1\lambda^{(x_1-2)}\lambda-|S|\lambda_1^x)(x_1+3-|S|)}{x_1!O_1(1-\lambda)^{(x_1+3-|S|)}(x_1+1)(1-\lambda)}) \\
&|S|(|S|-1)!/(\lambda-1) + (\tfrac{((x_2+1)\lambda_2^x-|S|\lambda^{(x_2+1)})(x_2+2-|S|)O_{12}}{(x_2+1)!O_8(1-\lambda)^{(x_2+4-|S|)}(2+x_2)} \\
&+\tfrac{O_{11}O_7}{(x_2+1)!O_8(1-\lambda)^{(x_2+4-|S|)}} - \tfrac{O_6O_3}{x_1!O_1(1-\lambda)^{(3+x_1-|S|)}} - \\
&\tfrac{O_2(1+x_1-|S|)(x_1\lambda^{(x_1-2)}\lambda-|S|\lambda_1^x)}{x_1!O_1(1-\lambda)^{(3+x_1-|S|)}(1+x_1)})|S|(|S|-1)/(\lambda-1)^2
\end{aligned}
\tag{99}
$$

where $F'(x;|S|-1,\lambda)|_{x_1}^{x_2}$ is the first derivative of the accumulative function $F(x;|S|-1,\lambda)|_{x_1}^{x_2}$ with respect to λ, x_1 is equal to 0 for equations (42) and (43), and c for equation (44), *hypergeom* is the generalized hypergeometric using Matlab notation,

and

$$O_1 = (|S| - 1 - x_1)$$

$$O_2 = hypergeom([2, x_1 + 2 - |S|], [2 + x_1], \frac{\lambda}{\lambda - 1})$$

$$O_3 = hypergeom([1, 1 + x_1 - |S|], [1 + x_1], \frac{\lambda}{\lambda - 1})$$

$$O_4 = \frac{|S|^2}{2} + (x_1 - 3/2)|S| + 1 - \frac{x_1}{2}$$

$$O_5 = (2x_1 - 1)|S| + x_1^2 - x_1 + 1$$

$$O_6 = (O_5\lambda - x_1^2)\lambda^{(x_1 - 2)} + (|S| - 1)^2\lambda^{(1 + x_1)} - 2\lambda_1^x O_4$$

$$O_7 = hypergeom([1, x_2 + 2 - |S|], [2 + x_2], \frac{\lambda}{\lambda - 1})$$

$$O_8 = (|S| - 2 - x_2)!$$

$$O_9 = -|S|^2 + (1 - 2x_2)|S| + x_2 - 1$$

$$O_{10} = (2x_2 + 1)|S| + x_2^2 + x_2 + 1$$

$$O_{11} = (O_{10}\lambda - (x_2 + 1)^2)\lambda^{(x_2 - 1)} + O_9\lambda^{(x_2 + 1)} + \lambda^{(x_2 + 2)}(|S| - 1)^2$$

$$O_{12} = hypergeom([2, 3 + x_2 - |S|], [3 + x_2], \frac{\lambda}{\lambda - 1})$$