Data Mining Techniques in Gene Expression Data Analysis

Xin Xu

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2005

Data Mining Techniques in Gene Expression Data Analysis

by

Xin Xu, MS

Dissertation

Presented to the Faculty of the Graduate School of

National University of Singapore

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

National University of Singapore

July 2006

ACKNOWLEDGMENTS

I would like to thank my supervisor Dr. Anthony K.H. Tung for years of professional guidance and his invaluable advice and comments for the thesis during my ph.d. study.

Especially thanks go to Prof. Beng Chin Ooi and Prof. Kian Lee Tan for their guidance as well as helpful suggestions. I am also thankful to Prof. Lim Soon Wong for the constructive opinion for my study.

Also, my acknowledgements go out to all my friends Gao Cong, Kenny Chua, Qiang Jing, Tiefei Liu, Qiong Luo and Chenyi Xia for their warm-hearted help and beneficial discussions.

Finally, heartful thanks go for my family for their support with heart and soul.

XIN XU

National University of Singapore

July 2006

Data Mining Techniques in Gene Expression Data Analysis

Publication No. ___________________

Xin Xu, Ph.D.

National University of Singapore, 2006

Supervisor: Anthony K.H. Tung

With the rapid development of microarray chip technology, gene expression data are being generated in large throughput. The imdispensable task of data mining, as a result, is to effectively and efficiently extract useful biological information discussed above from gene expression data. However, the high-dimensionality and the complex relationships among genes impose great challenges for existing data mining methods.

In this thesis, we systematically study the existing problems of the state-of-the-art data mining algorithms for gene expression data in class association rule mining, associative classification and subspace clustering of genes of nonlinear and shifting-and-scaling correlation. Specifically, we propose the concept of top-*k* covering rule groups for each gene expression sample, TopKRGs and design an row-wise mining algorithm to discover the TopKRGs efficiently; we further develop a new associative classifier by combining the nl rules consisted of the most significant genes based on entropy test of the top k covering rule groups; to address the nonlinear correlation problem and shifting-and-scaling correlation problem, we introduce Curler and RegMiner algorithms respectively to identify the subset of genes which exhibit non-linear or shifting-and-scaling correlation patterns across a subset of conditions.

Extensive experimental studies are conducted on synthetic and real-life datasets. The experimental results show the effectiveness and efficiency of our algorithms. While we mainly use gene expression data in our study, our algorithms can also be applied to high-dimensional data of other domains.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Gene expression is the process of transcribing a gene's DNA sequence into mRNA sequences, which are later translated into amino acid sequences of proteins. The number of copies of produced RNA is called the expression level of the gene. The regulation of gene expression level is considered important for proper cell function. As an effective technology to study gene expression regulation, microarray gene expression profiling uses arrays with immobilized cDNA or oligonucleotide sequences to measure the quantity of mRNA based on hybridization. Microarray technologies provide the opportunity to measure the expression levels of tens of thousands of genes in cells simultaneously which are correlated with the corresponding protein made either under different conditions or during different time spots. Gene expression profiles generated by microarrays can help us understand the cellular mechanism of biological process. For instance, it provides information about the cancerous mutation of cells: which genes are most responsible for the mutation, how they are regulated, and how experimental conditions can affect cellular function. With these advantages, microarray technology has been widely used in post-genome cancer research studies. With the rapid advance of microarray technology, gene expression data are being generated in large throughput so that an imposing data mining task is to effectively and efficiently extract useful biological information discussed above from the huge and fast-growing gene expression data.

Essentially, data mining methods can be partitioned into two big categories: *supervised* and *unsupervised*. Supervised data mining methods assume each gene expression profile has a certain class label, i.e., the expression profile of each patient is associated with the specific disease the patient has, and supervised methods make use of the class information in the learning process. On the contrary, unsupervised data mining methods have no assumption about the class information of each gene expression profile. Specifically, for gene expression analysis, supervised data mining methods include class association rule mining and classification, while unsupervised data mining methods mainly refer to the various clustering methods.

Class association rule mining is one of the most famous traditional data mining methods. Each row of the expression data matrix involved in class association rule mining corresponds to a samples or a condition, while each column corresponds to a gene. Current class association rule mining methods like [11] follow the item-wise searching strategy of traditional association mining methods [5, 37, 60, 66]. After discretizing the expression levels of the genes correlated with class label into two or more intervals, the class association rule mining algorithm searches the combinations of gene expression intervals of high statistical significance w.r.t. a certain class label. The simple class association rule in the form of $gene_1[a1, b1]\ gene_2[a2, b2] \rightarrow cancer$ is not only easy for understanding but also useful in practice. By focusing on the subset of most discriminating genes involved in the rules, here $gene_1$ and $gene_2$, biologists can design the following experiments to understand the cancer mutation scheme. Going beyond this, the class association rule is also a reference to drug discovery. And, a considerable amount of research has demonstrated that accurate and inexpensive diagnosis can be achieved with class association rules [52–54] because of their informativeness and succinctness.

Classification is yet another important supervised data mining method for gene expression analysis. Many classification approaches, such as decision tree [71], KNN [29], SVM [50], neural network [33], have been applied on gene expression data. During the classification subroutine, the classifier is first trained on training samples, and then tested on test samples. After having been accessed to have enough correctness, the classifier could be classify samples of unknown class label. All these approaches have limitations when applied to gene expression data. The decision tree approaches like C4.5 (single tree [71], bagging [16] and boosting [30]) derive rules that are exclusive with each other and cover the training samples just once. These decision tree methods search class association rules by selecting the genes that contribute most for distinguishing a certain partitioned training samples, NOT genes that contribute most for distinguishing samples of different classes globally. Therefore, some biased rules may be generated by decision tree methods. Meanwhile the information contained in the limited decision tree rules is far from sufficient for biological research. KNN, too, provides little information about the disease scheme. Other classification methods like SVM and Neural network have demonstrated effectiveness in classifying test samples, however, their classification scheme is rather difficult to understand. A better alternative is the associative classification [55, 56], which makes the decision with the most significant class association rules. These class association rules, as we discussed above, are both informative and easy for understanding. PCL [52] is a representative associative classification method for gene expression data, which combines the discriminating powers of the emerging patterns of each class.

Unsupervised data mining methods mainly refer to the clustering method. The clustering subroutine typically groups the correlated genes or samples (conditions) together to find co-regulated and functionally similar genes or similarly expressed samples (conditions). Gene clustering and sample (conditions) clustering can also be combined to find the most important genes or samples (conditions). The most popular clustering algorithms adopted for gene expression data include the hierarchical clustering (iteratively joining the two closest clusters beginning with singleton clusters), K-mean (typically using the Euclidean distances to partition the space into K parts) [8], SOM (a neural network algorithm) [49] and graph theoretic approaches such as HCS [38]. However, these methods require the input parameter of cluster number which is difficult to determine in advance; and the clustering results are not steady in most cases. Besides, these algorithms are all full-space clustering algorithms which evaluate the similarity of gene expression profiles under all the samples (conditions). Other traditional full-space clustering methods include GDR (global dimension reduction) [77] and PCA (principle component analysis) [46].

This is actually wrong for gene expression data, since a group of genes can be correlated only in a subset of samples (conditions) rather than the whole space. In recent years, a number of subspace clustering algorithms have been proposed, such as CLIQUE [4], OptiGrid [39], ENCLUS [41], PROCLUS [3], DOC [68], ORCLUS [2] and 4C [14].

However, as we will discuss in the next Section, these state-of-the-art data mining methods in class association rule mining, classification and clustering are still problematic for gene expression data.

1.1 Motivation

The extremely high dimensionality and the complex correlations among the genes pose great challenges for successful application of existing class association rule mining [11], class associative classification [52, 55, 56] and the subspace clustering algorithms [2–4, 14, 39, 41, 68] to gene expression analysis.

• Challenge for Class Association Rule Mining: *Inefficiency and Huge Rule Number*

Traditional association mining methods are not able to work well on gene expression data for class association rule discovery due to their inefficiency. These item-wise association mining methods [5, 11, 37, 60, 66] which enumerate gene-intervals (items) iteratively may fail to finish running in days or even weeks when extended to search class association rules. The main cause of the inefficiency is the huge item-wise search space resulting from the thousands or tens of thousands of gene-intervals after discretization. Note that the item-wise searching space is as high as 2^n, exponential with the gene-interval (item) number n. As another drawback of item-wise methods, extremely large of class association rules will be output, owing to explosive item combinations.

• Challenge for Associative Classification: *Rule Selection*

The inefficiency in rule mining together with the huge rule number make the conventional associative classification methods like CBA [56] and CMAR [55] impractical. CBA and CMAR are built on class association rules discovered by the inefficient item-wise rule mining algorithms discussed above. It's rather difficult to select the significant rules for classifier building with these inefficient rule mining algorithms. Another recent associative classification method, PCL, avoids the problems of inefficiency and huge rule number by simply choosing a limited number of top-ranked genes based on the chi-square test to generate rules and ignoring those of lower ranks. However, the globally significant rules sometimes contain low-ranked genes. Furthermore, some genes of lower chi-square rank may also play a big role in cancer pathogenesis. For instance, MRG1 of rank 671 in the prostate cancer data may function as a coactivator through its recruitment of p300/CBP in prostate cancer cell [32, 47]. Eliminating such important genes during classification is not reasonable.

• Challenges for Subspace Clustering: *Nonlinear Correlation* and *Shifting-and-Scaling Correlation*

For high-dimensional data like gene expression data, a subset of data objects (genes) is probably strongly correlated only in a subset of conditions, while not correlated at all in the remaining ones. Besides, the orientation of these local correlation clusters can be arbitrarily oriented. The above problems have been addressed by several subspace clustering algorithms such as LDR [17], ORCLUS [2], and 4C [14] are proposed to identify local correlation clusters with arbitrary orientations, assuming each cluster has its own fixed orientation. However, they could only identify linear dependency among certain subset of conditions, i.e., the linear dependency of gene expressions in a time series gene expression data. To our knowledge, correlation between two or more genes (or other data objects) may be more complex than just a linear one. As one example reported in [34], gene $mGluR1$ and gene $GRa2$ have obvious nonlinear correlation pattern. Thus, finding nonlinear correlation clusters (clusters with varying orientations instead of a fixed orientation) in different subspaces is a necessary task for high-dimensional data such as gene expression data.

Both the linear correlation and the nonlinear correlation subspace clustering methods are density-based, requiring gene members to be close to each other in correlated subspace. However, correlated genes don't need to be close in correlated subspaces at all: positive-correlated genes and negative-correlated genes exhibit no spatial proximity; genes co-regulated together may exhibit pure shifting or pure scaling patterns across the subset of the correlated samples, as addressed in pCluster [80] and TRICLUSTER [85]. However, the shifting-and-scaling pattern, which includes both positive correlation and negative correlation, has received little attention.

In summary, the inefficiency of traditional rule discovery algorithms together with the resulting inappropriate rule selection strategy seriously limit the application of association rule mining and association classification on gene expression data; the diversified correlations among genes, nonlinear correlation and shifting-and-scaling correlation, have been disregarded by current clustering algorithms. These are the imposing problems of the state-of-the-art data mining methods.

1.2 Contributions

In this thesis, we systematically study and solve the existing problems of the state-of-the-art data mining algorithms when applying on gene expression data. We propose the concept of TopKRGs to handle the problems of inefficiency and huge rule number in class association rule mining; to address the problem of rule selection in associative classification, we present classifier RCBT based on TopKRGs; we design two algorithms, CURLER and Reg-Cluster, for finding nonlinear correlation clusters and shifting-and-scaling correlation clusters in subspace respectively. In particular, we make the following contributions.

TopKRGs: To cope with extremely large rule number, we propose the concept of top-*k* covering rule groups (TopKRGs) for each row of a gene expression dataset and have designed a row-wise mining algorithm to discover the top-*k* covering rule groups for each row. In this way, numerous rules have been clustered into a limited number of rule groups, bounded by $k * n$, where n is the number of rows of gene expression dataset and k is the user specified parameter. Our algorithm is specially efficient for gene expression data with extremely large number of genes but relatively small number of samples. Extensive experiments on real-life gene expression datasets show that our algorithm can be several order of magnitudes better than FARMER [20], CLOSET+ [81] and CHARM [84] which uses diff-sets.

RCBT: TopKRGs also facilitates rule selection for associative classification. Based on that, we combine the nl rules generated by the most significant genes from each discovered TopKRGs and further develop a new associative classifier called RCBT. Essentially, our RCBT classifier works in a committee-like way. Each test data is first classified by the main classifier built on rules of the top one covering rule groups for each class; if unclassified, the test data is further passed on to the subsequent ordered classifiers built on the rules from the top 2, 3, ..., j covering rule groups until it is classified or $j == k$. The committee-like scheme avoids many default class assignment cases. Extensive experimental studies show that our classifier is competitive with the state-of-the-art classifiers: C4.5 (single tree, bagging and boosting), CBA [56], IRG classifier [20] and even SVM [55]. To help biologists understand our rule selection scheme, we also implemented a demo to visualize the discovered rule groups effectively. Biologists can interactively explore and select the most significant rule groups with the demo.

CURLER: Detecting nonlinear correlation clusters is quite challenging. Unlike the detection of linear correlation in which clusters are of unique orientations, finding nonlinear correlation clusters of varying orientations requires merging clusters of possibly very different orientations. Combined with the fact that spatial proximity must be judged based on a subset of features that are not originally known, deciding which clusters to be merged during the clustering process becomes a challenge. To avoid the problems discussed above, we propose a novel concept called *co-sharing level* which captures both spatial proximity and cluster orientation when judging similarity between clusters. Based on this concept, we design an algorithm, Curler, for finding and visualizing such nonlinear correlation clusters in subspace. Our algorithm can also be applied to other high-dimensional database besides gene expression data. Experiments on synthetic data, gene expression data and benchmark biological data are done to show the effectiveness of our method.

Reg-Cluster: We propose a new model for coherent clustering of gene expression data called **reg-cluster**. The proposed model allows (1) the expression profiles of genes in a cluster to follow any shifting-and-scaling patterns in a certain subspace, where the scaling can be either positive or negative, and (2) the expression value changes across any two conditions of the cluster to be significant, when measured by a user-specified regulation threshold. We also develop a novel pattern-based biclustering algorithm for identifying shifting-

and-scaling co-regulation patterns, satisfying both regulation constraint and coherence constraint. Our experimental results show: (1) the reg-cluster algorithm is able to detect a significant amount of gene clusters missed by previous model, and these gene clusters are potentially of high biological significance; and (2) the reg-cluster algorithm can easily be extended to 3D $gene \times sample \times time$ dataset for identifying 3D reg-clusters.

While we focus on gene expression data mainly in this study, our methods can also be applied on other complex high-dimensional data in bioinformatics, industry, finance and so on. For instance, our reg-cluster algorithm can be adopted for identifying metabolites demonstrating complex shifting-and-scaling correlation patterns in a subset of conditions as well.

1.3 Organization of the Thesis

The remaining thesis is organized as follows: we introduce the concept of $TopKRGs$ and the $TopKRGs$ discovery algorithm in details in Chapter 2; the associative classifier built upon $TopKRGs$ will be presented in Chapter 3; In Chapter 4, we describe the concept of co-sharing level and then propose our nonlinear correlation clustering algorithm $Curler$; we propose our recluster model of shifting-and-scaling patterns and reg-cluster discovery algorithm in Chapter 5; we summarize and conclude our work in Chapter 6.

CHAPTER 2

TopKRGs: Efficient Mining of Top K Covering Rule Groups

High-dimensional gene expression data requires powerful computational analysis tools to extract the most significant and reliable correlation between gene expression patterns and disease outcomes and to translate the complex raw data into relevant and clinically useful diagnostic knowledge. Class association rule is the solution for the above requirements.

We define a class association rule as a set of items, or specifically a set of conjunctive gene expression level intervals (*antecedent*) with a single class label (*consequent*). The ***general*** form of a class association ***rule*** is: $gene_1[a_1, b_1]$, ..., $gene_n[a_n, b_n] \rightarrow class$, where $gene_i$ is the name of the gene and $[a_i, b_i]$ is its expression interval. For example, $X95735\_at[-\infty, 994] \rightarrow ALL$ is one rule discovered from the gene expression profiles of ALL/AML tissues.

2.1 Background

Association rule mining has attracted considerable interest since a rule provides a concise and intuitive description of knowledge. It has already been applied to biological data analysis, such as [22, 25, 67]. The unlabelled association rules can help discover the relationship between different genes, so that we can infer the function of an individual gene based on its relationship with others [22] and build the gene network. In this thesis, we discuss about class association rule, the consequent of which is a class label. Class association rules can relate gene expressions to their cellular environments or categories indicated by the class, thus they can be used to build accurate classifiers on gene expression datasets as in [26, 53].

Many association rule mining algorithms have been proposed to find the complete set of association rules satisfying user-specified constraints by discovering frequent (closed) patterns as the key step, such as [5, 36, 37, 62, 64, 65, 81, 84]. The basic approach of most existing algorithms is item enumeration in which combinations of items are tested systematically to search for association rules. Such an approach is usually unsuitable for class association rule mining on gene expression datasets, since the maximal enumeration space can be as large as 2^i, where i is the number of items and is in the range of tens of thousands for gene expression data. The high-dimensional gene expression data renders most of the existing algorithms impractical. On the other hand, the number of rows in such dataset is typically very small and the maximum row enumeration space 2^m (m is the number of rows) is significantly smaller.

There are also many proposals about mining interesting rules with various interestingness measures. Some of them do a post-processing to remove those uninteresting rules, such as [57]. Such methods cannot work on gene expression data since it is usually too computationally expensive to mine the huge association rules from gene expression data. Other works [10, 72] try to mine interesting rules directly. The proposed algorithm in [10] adopts item enumeration method and usually cannot work on gene expression data as shown in the experiments of [20]. FARMER [20] is designed to mine interesting rule groups from gene expression data by row enumeration. But it is still very time-consuming on some entropy-based discretized gene expression datasets. Although we also adopt the row enumeration strategy, our algorithm is different from FARMER: (1) we discover different kinds of rule groups; (2) we use top-k pruning; (3) we use a compact prefix-tree to improve efficiency while FARMER adopts in-memory pointer.

Two main challenges remain for mining class association rules from gene expression data.

First, it has been shown in [20, 22] that huge number of rules will be discovered from the high-dimensional gene expression dataset even with rather high minimum support and confidence thresholds. This makes it difficult for the biologists to filter out rules that can encapsulate very useful diagnostic and prognostic knowledge discovered from raw datasets. Although recent row-wise enumeration algorithms like FARMER [20] can greatly reduce the number of rules by clustering similar rules into rule groups, it is still common to find tens of thousands and even hundreds of thousands of rule groups from gene expression dataset, which are rather hard to interpret.

Second, the high dimensionality together with the huge number of rules results in extremely long mining process. Rule mining algorithms using item enumeration (combinations of items are tested systematically to search for rules), such as CHARM [84] and CLOSET+ [81], are usually unsuitable for gene expression datasets because searching in huge item enumeration space results in extremely long running time. Although FARMER efficiently clusters rules into rule groups and adopts an anti-monotone confidence pruning with a delicate row ordering strategy, it is still very slow when the number of rule groups is huge.

These two challenges greatly limit the application of rules to analyze gene expression data. It will be ideal to discover only a small set of the most significant rules instead of generating a huge number of rules.

2.2 Problem Statement and Preliminary

To address the problems we discussed in the above section, we propose to discover the **most significant top-*k* covering rule groups (TopkRGS) for each row of a gene expression dataset**. We will illustrate this with an example.

Example 2.2.1 TopkRGS

For the running example shown in Figure 2.1(a), given $minsup = 2$*, the top-1*

i	r_i	class
1	a, b, c, d, e	C
2	a, b, c, o, p	C
3	c, d, e, f, g	C
4	c, d, e, f, g	¬C
5	e, f, g, h, o	¬C

(a) Example Table

i_j	$\mathcal{R}(i_j)$	
	C	$\neg C$
a	1, 2	
b	1, 2	
c	1, 2, 3	4
d	1, 3	4
e	1, 3	4, 5
f	3	4, 5
g	3	4, 5
h		5
o	2	5
p	2	

(b) $TT|_{\emptyset}$ (or TT)

i_j	$\mathcal{R}(i_j)$	
	C	$\neg C$
a	2	
b	2	
c	2, 3	4
d	3	4
e	3	4, 5

(c) $TT|_{\{1\}}$

i_j	$\mathcal{R}(i_j)$	
	C	$\neg C$
c		4
d		4
e		4, 5

(d) $TT|_{\{1,3\}}$

Figure 2.1: Running Example

covering rule group for rows r_1 and r_2 is $\{abc \rightarrow C\}$ with confidence 100%, the top-1 covering rule group for row r_3 is $\{cde \rightarrow C\}$ with confidence 66.7%, and the top-1 covering rule group for rows r_4 and r_5 is $\{fge \rightarrow \neg C\}$ with confidence 66.7%. The support values of the above top-1 covering rule groups are all 2, which is equal to $minsup$.

While formal definition will be given later, we summarize the task of finding top-k covering rule groups as essentially doing the following:

- Define an interestingness criterion for rule group ranking.

- Based on the ranking, for **each** row r in the dataset, find the k highest ranked rule groups of the same class as r such that the antecedent of the k rule groups are all found in r (i.e. r is covered by these k rule groups).

The top-k covering rule groups are beneficial in several ways, as listed below:

- TopkRGS can provide a more complete description for each row. This is unlike previous proposals of interestingness measures like confidence which may fail to discover any interesting rules to cover some of the rows if the mining threshold is set too high. Correspondingly, information in those rows that are not covered will not be captured in the set of rules found. This may result in loss of important knowledge since gene expression datasets have small number of rows;

- Finding TopkRGS helps us to discover the complete set of useful rules for building a classifier while avoiding the excessive computation adopted by algorithms like the popular CBA classifier [56]. These algorithms first discover a large number of redundant rules from gene expression data most of which will be pruned in the later rule selection phase. We will prove later that the set of top-1 covering rule group for each row contains the complete set of rules required to build the CBA classifier while avoiding the generation of huge redundant rules;

- We do not require users to specify the minimum confidence threshold. Instead only the minimum support threshold and the number of top covering rule groups, k, are required. Such an improvement is useful since it is not easy for users to set an appropriate confidence threshold (we do not claim that specifying minimum support is easy here) while the choice of k is semantically clear. In fact, the ability to control k allows us to balance between two extremes. While rule induction algorithms like decision tree typically induce only 1 rule from each row and thus could miss interesting rules, association rule mining algorithms are criticized for finding too many redundant rules covering the same rows. Allowing users to specify k gives them control over the number of rules to be generated.

- The number of discovered top-k covering rule groups is bounded by the product of *k* and the number of gene expression data, which is usually quite small.

TopkRGS runs on discretized gene expression data.

Dataset: the gene expression dataset (or table) D consists of a set of rows, R=$\{r_1$, ..., $r_n\}$. Let I=$\{i_1, i_2, ..., i_m\}$ be the complete set of items of D (each item represents some interval of gene expression level), and $C = \{C_1, C_2, ..., C_k\}$ be the complete set of class labels of D, then each row $r_i \in R$ consists of one or more items from I and a class label from C.

As an example, Figure 2.1(a) shows a dataset with 5 rows, r_1, r_2, ..., r_5, the first three of which are labelled C while the other two are labelled $\neg C$. To simplify the notation, we use the *row id set* to represent a set of rows and the *item id set* to represent a set of items. For instance, “134” denotes the row set $\{r_1, r_3, r_4\}$, and “cde” denotes the itemset $\{c, d, e\}$.

As a mapping between rows and items, given a set of items $I' \subseteq I$, we define the **item support set**, denoted $\mathcal{R}(I') \subseteq R$, as the largest set of rows that contain I'. Likewise, given a set of rows $R' \subseteq R$, we define **row support set**, denoted $\mathcal{I}(R') \subseteq I$, as the largest set of items common among the rows in R'.

Example 2.2.2 $\mathcal{R}(I')$ **and** $\mathcal{I}(R')$

Consider again the table in Figure 2.1(a). Let I' be the itemset $\{c, d, e\}$, then $\mathcal{R}(I') = \{r_1, r_3, r_4\}$. Let R' be the row set $\{r_1, r_3\}$, then $\mathcal{I}(R')$=$\{c, d, e\}$ since this is the largest itemset that appears in both r_1 and r_3.

Based on our definition of item support set and row support set, we can redefine the association rule.

Association Rule: an **association rule** γ, or just **rule** for short, from dataset D takes the form of $A \rightarrow C$, where $A \subseteq I$ is the antecedent and C is the consequent (here, it is a class label). The **support** of γ is defined as the $|\mathcal{R}(A \cup C)|$, and its **confidence** is $|\mathcal{R}(A \cup C)|/|\mathcal{R}(A)|$. We denote the antecedent of γ as $\gamma.A$, the consequent as $\gamma.C$, the support as $\gamma.sup$, and the confidence as $\gamma.conf$.

As discussed in the introduction, in real biological applications, biologists are often interested in rules with a specified consequent C, which usually indicates the cancer outcomes or cancer status.

The rule group is a concept which helps reduce the number of rules discovered by identifying rules that come from the same set of rows and clustering them conceptually into rule groups.

Definition 2.2.1 Rule Group

Let D be the dataset with itemset I and C be the specified class label. $G = \{A_i \rightarrow C | A_i \subseteq I\}$ is a rule group with antecedent support set R and consequent C, iff (1) $\forall A_i \rightarrow C \in G$, $\mathcal{R}(A_i) = R$, and (2) $\forall \mathcal{R}(A_i) = R$, $A_i \rightarrow C \in G$. Rule $\gamma_u \in G$ (γ_u: $A_u \rightarrow C$) is an **upper bound** *of G iff there exists no $\gamma' \in G$ (γ':$A' \rightarrow C$) such that $A' \supset A_u$. Rule $\gamma_l \in G$ (γ_l: $A_l \rightarrow C$) is a* **lower bound** *of G iff there exists no $\gamma' \in G$ (γ': $A' \rightarrow C$) such that $A' \subset A_l$.*

Lemma 2.2.1 *Given a rule group G with the consequent C and the antecedent support set R, it has a unique upper bound γ (γ: $A \rightarrow C$).*

Based on lemma 2.2.1, we use upper bound rule γ_u to refer to a rule group G in the rest of this paper.

Example 2.2.3 Rule Group

Given the table in Figure 2.1(a). $\mathcal{R}(\{a\}) = \mathcal{R}(\{b\}) = \mathcal{R}(\{ab\}) = \mathcal{R}(\{ac\}) = \mathcal{R}(\{bc\}) = \mathcal{R}(\{abc\}) = \{r_1, r_2\}$. They make up a rule group $\{a \rightarrow C, b \rightarrow C, ..., abc \rightarrow C\}$ of consequent C, with the upper bound $abc \rightarrow C$ and the lower bounds $a \rightarrow C$ and $b \rightarrow C$.

It is obvious that all rules in the same rule group have the same support and confidence since they are essentially derived from the same subset of rows. Based on the upper bound and all the lower bounds of a rule group, it is easy to identify the remaining members. Besides, we evaluate the significance of rule groups consistently with the individual rule ranking criterion.

Definition 2.2.2 Significant

Rule group γ_1 is more significant than γ_2 if $(\gamma_1.conf > \gamma_2.conf) \vee (\gamma_1.sup > \gamma_2.sup \wedge \gamma_1.conf = \gamma_2.conf)$.

The top-k covering rule groups, as defined below, encapsulate the most significant information of the dataset while enabling users to control the amount of information in a significance-top-down manner.

Definition 2.2.3 Top-k covering Rule Groups (TopkRGS)

Given the database D and a user-specified minimum support $minsup$, the top-k covering rule groups for a row r_i is the set of rule groups $\{\gamma_{r_ij}\}$ $(1 < j < k)$, where $\gamma_{r_ij}.sup \geq minsup$, $\gamma_{r_ij}.A \subset r_i$ and there exists no rule group γ' $\gamma' \notin \{\gamma_{r_ij}\}$ such that γ' is more significant than γ_{r_ij}. For brevity, we will use the abbreviation TopkRGS to refer to top-k covering rule groups for each row.

2.3 Efficient Discovery of TopkRGS

The first problem that we address is to efficiently discover the set of top-k covering rule groups for each row (TopkRGS) of gene expression data given a user-specified minimum support $minsup$.

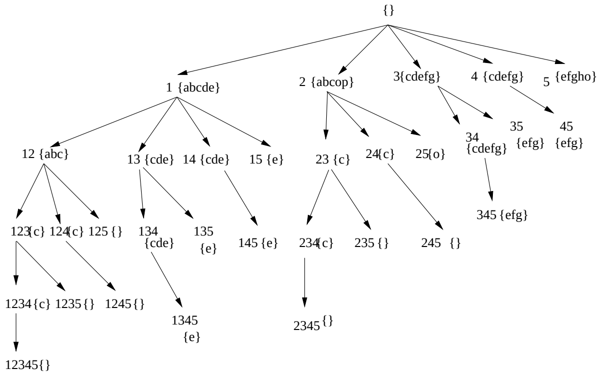


Figure 2.2: Row Enumeration Tree.

We first give a general review of how **row enumeration** takes place using the **(projected) transposed table** first proposed in [20] before proceeding to our TopkRGS discovery strategies. Implementation details will then be discussed.

Figure 2.1(b) is a transposed version TT of the table in Figure 2.1(a). In TT, the items become the row ids while the row ids become the items. The rows in the transposed tables are referred as *tuples* to distinguish from the so-called *rows* in the original table. Let X be a subset of rows. Given the transposed table TT, a X**-projected transposed table**, denoted as $TT|_X$, is a subset of tuples from TT such that: 1) For each tuple t in TT which contains all the row ids in X, there exists a corresponding tuple t' in $TT|_X$. 2) t' contains all rows in t with row ids larger than any row in X. As an example, the $\{13\}$-projected transposed table, $TT|_{13}$, is shown in Figure 2.1(d).

A complete **row enumeration tree** will then be built as shown in Figure 2.2. Each node X of the enumeration tree corresponds to a combination of rows R'

and is labelled with $\mathcal{I}(R')$ that is the antecedent of the upper bound of a rule group identified at this node. For example, node "12" corresponds to the row combination $\{r_1, r_2\}$ and "abc" indicates that the maximal itemset shared by r_1 and r_2 is $\mathcal{I}(\{r_1, r_2\}) = \{a, b, c\}$. An upper bound $abc \rightarrow C$ can be discovered at node "12". The correctness is proven by the following lemma in [20].

Lemma 2.3.1 *Let X be a subset of rows from the original table, then $\mathcal{I}(X) \rightarrow C$ must be the upper bound of the rule group G whose antecedent support set is $\mathcal{R}(\mathcal{I}(X))$ and consequent is C.*

By imposing a **class dominant order** order $\mathcal{ORD}$ on the set of rows, FARMER [20] performs a systematic search by enumerating the combinations of rows based on the order $\mathcal{ORD}$. For example, let "$1 \prec 2 \prec 3 \prec 4 \prec 5$" be the $\mathcal{ORD}$ order, then the depth-first order of search in Figure 2.2 will be $\{$"1", "12", "123", "1234", "12345", "1235",...,"45", "5"$\}$ in absence of any optimization strategies. Ordering the rows in class dominant order is essential for FARMER to apply its confidence and support pruning efficiently. Class dominant order is also essential for efficient pruning based on the top-k dynamic minimum confidence, as we will discuss later.

Definition 2.3.1 Class Dominant Order

A **class dominant order** $\mathcal{ORD}$ *of the rows in the dataset is an order in which all rows of class C are ordered before all row of class $\neg C$.*

Given the row enumeration strategies introduced above, a naive method of deriving the top-k covering rule groups is to first obtain the complete set of upper bound rules in the dataset by running the row-wise algorithm FARMER [20] with a low minimum confidence threshold and then picking the top-k covering rule groups for each row in the dataset. Obviously, this is not efficient. Instead, our algorithm will maintain a list of top-k covering rule groups for each row during the depth-first search and keep track of the k-th highest confidence of rule group at each enumeration node dynamically. The dynamic minimum confidence will be used to prune the search space. That is, whenever we discover that the rule groups to be discovered in the subtree rooted at the current node X will not contribute to the top-k covering rule groups of any row, we immediately prune the search down node X. The reasoning of our pruning strategies is based on the following lemma.

Lemma 2.3.2 *Given a row enumeration tree T, a minimum support threshold $minsup$, and an $\mathcal{ORD}$ order based on specified class label C, suppose at the current node X, $\mathcal{R}(\mathcal{I}(X)) = X$, X_p and X_n represent the set of rows in X with consequent C and $\neg C$ respectively, and R_p and R_n are the set of rows ordered after rows in X with consequent C and $\neg C$ respectively in the transposed table of node X, $TT|_X$. Then, we can conclude that the maximal set of rows that the rule groups to be identified in the subtree rooted at node X can cover is $X_p \cup R_p$.*

Proof: *As $\mathcal{R}(\mathcal{I}(X)) = X$, the maximal antecedent support set of the rule groups to be identified at the subtree rooted at node X is $(X \cup R_p \cup R_n)$. In addition, as the rule groups are labelled C, the maximal set of rows covered by these rule groups is $(X_p \cup R_p)$.*

Combined with Definition 2.2.2, we compute $minconf$ and sup, the cutting points of the TopkRGS thresholds for the rows in $(X_p \cup R_p)$, where $minconf$ is the minimum confidence value of the discovered TopkRGS of all the rows in $X_p \cup R_p$, assuming the top-k covering rule groups of each row r_i are ranked in significance such that $\gamma_{r_i 1} \prec \gamma_{r_i 2} \prec \ldots \prec \gamma_{r_i k}$,

$$minconf = \min_{r_i \in (X_p \cup R_p)} \{\gamma_{r_i k}.conf\}, \tag{2.1}$$

and sup is the support value of the corresponding covering rule group with confidence $minconf$,

$$sup = \gamma_{r_c k}.sup,\ where\ \gamma_{r_c k}.conf = minconf. \tag{2.2}$$

According to the definition of the top-k covering rule groups (Definition 2.2.3), we can further obtain Lemma 2.3.3 below.

Lemma 2.3.3 *Given the current node X, $minconf$ and sup computed according to Equations 2.1 and 2.2, if the rule group identified inside the subtree rooted at node X is less significant (according to Definition 2.2.2) than $\gamma_{r_c k}$ ($\gamma_{r_c k}.conf = minconf$ and $\gamma_{r_c k}.sup = sup$), then the rule group cannot become a rule group in the top-k covering rule group list of any row.*

Naturally our **top-k pruning** will proceed in the following way:

- If the upper bound of the confidence value of the rule groups to be identified in the subtree rooted at node X is below $minconf$ which is dynamically calculated at node X, then prune the search down node X;

- If the upper bound of the confidence value of the rule groups to be identified in the subtree rooted at node X is equal to $minconf$ which is dynamically calculated at node X and the upper bound of the support value of the rule groups to be identified in the subtree rooted at node X is smaller than sup, then prune the search space down node X.

The reasoning of our TopkRGS discovery is clearly that the rule groups to be discovered down node X will not contribute to the TopkRGS of any row. The top-k pruning strategy introduced above can be perfectly integrated with the backward pruning, loose and tight upper bound pruning of confidence or support values of FARMER, which further speeds up our mining process. The following is an example.

Example 2.3.1 Discovery of Top-1 Covering Rule Groups

For the running example in Figure 2.1(a) where $k = 1$, specified class is C, and $minsup = 2$, when the depth-first traversal comes to node $\{1, 2\}$, the top-1 covering rule group for both r_1 and r_2 is dynamically updated to $abc \rightarrow C$ (conf:100%, sup:2). At node $\{1, 3\}$, when $X_p = \{1, 3\}$ and $R_p = \emptyset$, as the identified top-1 covering rule group for r_1 has confidence 100% while no top-1 covering rule group of r_3 has been discovered yet, we get $minconf = 0$ and $sup = 0$. Since the rule group $cde \rightarrow C$ identified at node $\{1, 3\}$ has confidence 66.7% and support 2, which is above the $minconf$ and $minsup$ thresholds, it is output to update the top-1 covering rule group of r_3. The estimated upper bound of the confidence values of the sub-level nodes down node $\{1, 2\}$ and $\{1, 3\}$ are all below the corresponding

$minconf$ and are simply pruned. The consequent search down node $\{2\}$ and $\{3\}$ is pruned using the backward pruning because of the rule groups down these nodes are identified already in previous enumerations.

2.3.1 Algorithm

Our algorithm performs a depth-first traversal of the row enumeration tree, where each node X will be associated with X-projected transposed table. As an example, when visiting node 1 in the enumeration tree, the 1-projected transposed table will be formed as shown in Figure 2.1(c). Also, it is important to note that the projected transposed table at a node can in fact be computed from the projected transposed table of its parent node. To compute the 13-projected transposed table as shown in Figure 2.1(d), we can simply scan $TT|_1$ and extract those tuples in $TT|_1$ that contain r_3. Since the enumeration order is such that parent node will always be visited before the child node, this gives rise naturally to a recursive algorithm where each parent node will call its children passing the relevant projected transposed table to the children nodes.

Formally, the algorithm is shown in Figure 2.3. There are four input parameters of the algorithm, the original dataset D, class label *C*, the minimum support $minsup$ and *k*. The algorithm will scan through the dataset D to count the frequency of each item and remove infrequent items from each row in D. D will then be transformed into the corresponding transposed table. At the same time, the top-k covering rule groups for each row r_i with consequent C denoted as ζ_{r_i}=[$\gamma_{r_i 1}$, $\gamma_{r_i 2}$,

..., $\gamma_{r_i k}$] will be initialized. Then the procedure Depthfirst() is called to perform the depth-first traversal of row enumeration tree.

The procedure Depthfirst() takes in six parameters at node X: $TT'|_X$, X_p, X_n, R_p, R_n, and $minsup$. $TT'|_X$ is the X-projected transposed table at node X. X_p and X_n represent the the set of rows in X with consequent C and $\neg C$ respectively. R_p is the set of candidate enumeration rows with consequent C that appear in $TT'|_X$ and R_n is the set of enumeration candidate rows with $\neg C$ appearing in $TT'|_X$. Among the steps in Depthfirst(), only steps 10, 12 and 14 are necessary if no pruning strategies are adopted. Step 10 scans the projected table $TT'|_X$ and computes $freq(r_i)$, the frequency of occurrence of each row r_i in $TT'|_X$. Based on $freq(r_i)$, rows that occur in all tuples (i.e. $freq(u) = \mathcal{I}(X)$) of $TT'|_X$ are found. These rows will appear in all descendant nodes of X and are thus added directly into X. Correspondingly, X_p and X_n are updated based on the consequent of these rows and they are removed either from R_p or R_n at step 12. Step 14 moves on into the next level enumerations in the search tree by selecting each row r_i that is either in R_p or R_n, creating a new $\{X \cup \{r_i\}\}$-projected transposed table and then passing the updated information to another call of MineTopkRGS.

Note that Step 14 implicitly does some pruning since it is possible that there is no row available for further enumeration, i.e. $R_p \cup R_n = \emptyset$. It can be observed from the enumeration tree that there exist some combinations of rows, X, such that $\mathcal{I}(X) = \emptyset$.

```
Algorithm MineTopkRGS ($D$, $C$, $minsup$, $k$)

1.  Scan database $D$ to find the set of frequent items $F$ and remove the infrequent items in each row $r_i$ of $D$;

2.  Let $D_p$ be the set of rows in $D$ with consequent $C$ and $D_n$ be the set of rows in $D$ without consequent $C$;

3.  Convert table $D$ into transposed table $TT|_\emptyset$;

4.  Initiate a list of k dummy rule groups with both confidence and support values of 0, $\zeta_{r_i}$=[$\gamma_{r_i1}$, ...., $\gamma_{r_ik}$], for each
    row $r_i$ in $D_p$;

5.  Call Depthfirst ($TT|_\emptyset$, $\emptyset$, $\emptyset$, $D_p$, $D_n$, $minsup$);

6.  Return $\zeta_{r_i}$ for $\forall r_i \in D_p$.
    Procedure: Depthfirst($TT'|_X$, $X_p$, $X_n$, $R_p$, $R_n$, $minsup$)

7.  Backward Pruning: If there is a row $r'$ that appears in every tuple w.r.t $\mathcal{I}(X)$ and does not belong to $X$, Then
    return.

8.  Threshold Updating: Check the kth covering rule group $\gamma_{r_ik}$ for each row $r_i \in X_p \cup R_p$ to find the lowest
    confidence $minconf$ and the corresponding support $sup$.

9.  Threshold Pruning: If prunable with the loose upper bounds of support or confidence, Then return.

10. Scan $TT'|_X$ and count the frequency, $freq(r_i)$, for each row, $r_i \in R_p \cup R_n$.

    Let $Y_p \subset R_p$ be the set of rows such that $freq(u) = |\mathcal{I}(X)|, u \in R_p$ and $Y_n \subset R_n$ be the set of rows such that
    $freq(u) = |\mathcal{I}(X)|, u \in R_n$;
    $X_p = X_p \cup Y_p$, $X_n = X_n \cup Y_n$ and $X = X_p \cup X_n$;

11. Threshold Pruning: If prunable with the tight upper bounds of support or confidence, Then return.

12. $R_p = R_p - Y_p,\ R_n = R_n - Y_n$.

13. $c$ = $|X_p|/(|X_p| + |X_n|)$; //compute confidence
    If (($|X_p| \geq minsup$) $\wedge$ ($c > minconf$)) $\vee$ (($c = minconf$)$\wedge$ ($|X_p| > sup$)) Then
        For each $r_i \in X_p$ Do
            If $\exists \gamma_{r_ij} \in \zeta_{ri}, j \leq k$ such that
                ($\gamma_{r_ij}.conf < c$) or
                (($\gamma_{r_ij}.conf = c$) $\wedge$ ($\gamma_{r_ij}.sup < |X_p|$)),
            Then update $\zeta_{ri}$ with $\mathcal{I}(X) \rightarrow C$;

14. For each $r_i \in R_p \cup R_n$ Do
        If $r_i \in R_p$ Then $R_p = R_p - \{r_i\}$, $X_p = X_p \cup \{r_i\}$;
        If $r_i \in R_n$ Then $R_n = R_n - \{r_i\}$, $X_n = X_n \cup \{r_i\}$;
        $Depthfirst(TT'|_{X \cup r_i}, X_p, X_n, R_p, R_n, minsup)$;
```

Figure 2.3: Algorithm MineTopkRGS

2.3.2 Pruning Strategies

In MineTopkRGS, top-k pruning is the main pruning strategy, and other pruning techniques first introduced in [20] are the supplementary pruning that we have seamlessly combined with our top-k pruning.

We first briefly introduce how to estimate the support upper bounds at an enumeration node X. At Step 9, it is obvious that the support of any rule groups enumerated along X cannot be more than $|X_p|+|R_p|$. The maximal number of rows with consequent C in one row, denoted as m_p ($m_p \leq R_p$), among all the branches under node X can be obtained at Step 10. As a result, we can get a tighter support upper bound at Step 11, i.e. $|X_p| + m_p$.

The estimation of confidence upper bounds is a bit complicated. For a rule γ discovered in the subtree rooted at X, its confidence is computed as $|\mathcal{R}(\gamma.A \cup C)|/(|\mathcal{R}(\gamma.A \cup C)| + |\mathcal{R}(\gamma.A \cup \neg C)|)$. This expression can be simplified as $x/(x+y)$, where $x = |\mathcal{R}(\gamma.A \cup C)|$ and $y = |\mathcal{R}(\gamma.A \cup \neg C)|$. This value is maximized with the largest x and the smallest y. The smallest y is $|R_n|$ at node X and the largest x can be $|R_p|$ or m_p as we just discussed. Therefore, we can get a loose confidence upper bound $|R_p|/(|R_p| + |R_n|)$ at Step 9 and a tight confidence upper bound $m_p/(m_p + |R_n|)$ at Step 11.

Top-k Pruning

Step 8 is a very important step in our algorithm. In this step, the $minconf$ threshold is dynamically set for enumeration down X, which makes it possible to use the confidence threshold to prune the search space at steps 9 and 11. The $minconf$ threshold is obtained according to Equation 2.1. Steps 9 and 11 perform pruning by utilizing the user-specified minimum support threshold, $minsup$ and the dynamic minimum confidence threshold, $minconf$ (generated dynamically at step 8). If the estimated upper bound of either measure at X is below either $minsup$ or $minconf$, we stop searching down node X. At Step 9, we will perform pruning using the two loose upper bounds of support and confidence that can be calculated without scanning $TT'|_X$. At Step 11, we compute the tight upper bounds of support and confidence after scanning $TT'|_X$.

The corresponding support sup information is also recorded for computation at Step 13. Note that $sup \geq minsup$. Whenever a new rule group $\mathcal{I}(X) \rightarrow C$ is discovered at node X, a check is made to see whether the new rule is more significant than one or more rule groups in the list of top-k covering rule group for some rows in X_p, the top-k covering rule groups of such rows will be updated dynamically. This is done at Step 13.

Two additional optimization methods are utilized in our top-k pruning.

- First, because we can easily know the confidence of the rule whose antecedent is a single item at Step 1 of algorithm MineTopkRGS, we use these confidence values to initiate the confidence and support values of the list of TopkRGS at Step 4 instead of initiating them with zero. Such an optimization may cause a problem. That is, if a single item is a lower bound of an upper bound rule, the result set will not include the upper bound rule because they have the same support and confidence. We need to update the single item with the upper bound rule by adapting step 13 of algorithm MineTopkRGS. Another technical detail here is that we need to ensure that any two single items to be used to initiate the top-k rule groups for one row cannot be the lower bounds of the same upper bound rules.

- Second, we dynamically increase the user-specified $minsup$ threshold if we find that all TopkRGS have 100% confidence and the lowest support value of the k rule groups is larger than the user-specified one.

MineTopkRGS outputs the most significant information for each row, as well as dramatically improving the efficiency and reducing the memory usage, compared to FARMER.

Backward Pruning

Step 7 implements the *backward pruning* first introduced in [20]. If there exists a row r' that appears in each prefix path w.r.t the set of nodes contributing to $\mathcal{I}(X)$ and does not belong to row set X, the rule groups $\mathcal{I}(X) \rightarrow C$ and all rule groups below X must have already been discovered below some enumeration node containing r' as proved in [20]. The principle is the same but our integration with the prefix tree makes TopkRGS more efficient. For example, at node $\{2\}$ in Figure 2.4 (b), we just need to do a back scan along the corresponding pointer list of node $\{2\}$ and can quickly find that there exists no such r'.

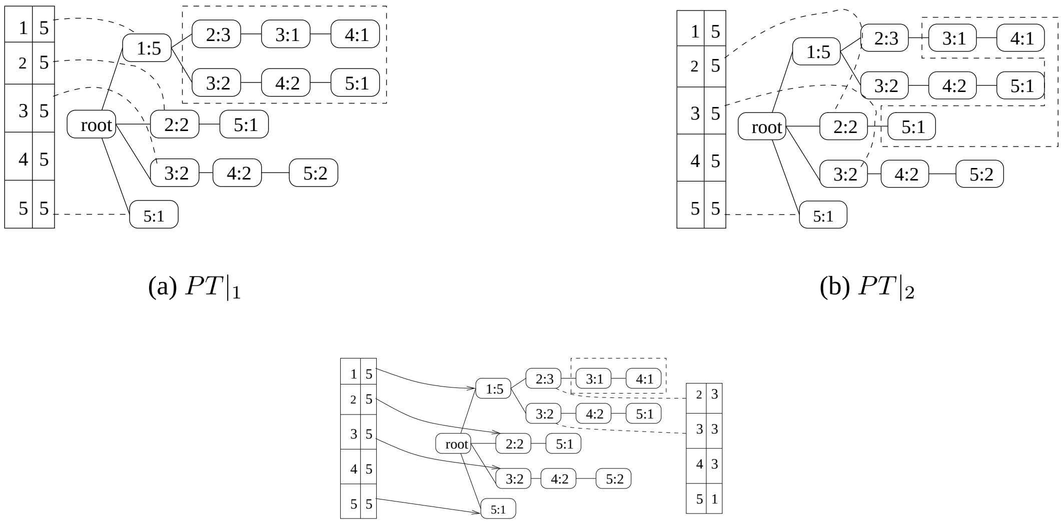
In addition, in $\mathcal{ORD}$, the rows from the same class are sorted in the ascending order of the number of frequent items contained in each row. This will improve the efficiency of algorithm MineTopkRGS.

2.3.3 Implementation

Next, we will illustrate how to represent (projected) transposed tables with prefix trees. The transposed table in Figure 2.1(b) is represented with the prefix tree shown in Figure 2.4 (a) (corresponding to the root node). The left head table in the figure records the list of rows in the transposed table and their frequencies. At each node of the prefix tree, we record the row id and the frequency of a row in the prefix path (separated by ":" in Figure 2.4 (a)). Additional information recorded at each node but not shown in the figure is the set of items represented at the node, such as items a, b, c, d and e at node "1:5". Such information will help to determine quickly the rule group w.r.t. a projected transposed table.

Example 2.3.2 *Projected Prefix Tree*

The part of nodes enclosed by dotted line in Figure 2.4(a) is the 1-projected prefix tree, $PT|_1$. Note that there are pointers linking the child nodes of the root with the corresponding rows in the head table. By following the pointer starting from row 1 of the header table, we can get the $PT|_1$. After $PT|_1$ has been mined recursively, the child paths of the node with label 1 will be assigned to other rows of the header table after row 1 (i.e. rows 2, 3, 4 and 5) and we get the 2-projected prefix tree, $PT|_2$. In Figure 2.4(b), the part enclosed by dotted line is $PT|_2$. By following the pointer from row 2 in the header table, we can get $PT|_2$.



(a) $PT|_1$

(b) $PT|_2$

(c) $PT|_{12}$

Figure 2.4: Projected Prefix Trees

2.4 Experimental Studies

Dataset	# Genes	# Genes after Discretization	Class 1	Class 0	# Training	# Test
ALL/AML (ALL)	7129	866	ALL	AML	38 (27 : 11)	34
Lung Cancer (LC)	12533	2173	MPM	ADCA	32 (16 : 16)	149
Ovarian Cancer (OC)	15154	5769	tumor	normal	210 (133 : 77)	43
Prostate Cancer (PC)	12600	1554	tumor	normal	102 (52 : 50)	34

Table 2.1: Gene Expression Datasets

We evaluate the efficiency of our algorithm in discovering TopkRGS on four real-life gene expression datasets. All our experiments were performed on a PC with a Pentium IV 2.4 Ghz CPU, 1GB RAM and a 80GB hard disk. Algorithms were coded in Standard C.

Datasets: We use 4 popular gene expression datasets for experimental studies.

The 4 datasets are the clinical data on ALL-AML leukemia (ALL) [1], lung cancer (LC)[2], ovarian cancer(OC) [3], and prostate cancer (PC) [4]. In such datasets, the rows represent clinical samples while the columns represent the activity levels of genes/proteins in the samples. There are two categories of samples in these datasets.

We adopt the entropy-minimized partition [5] to discretize gene expression datasets. The entropy discretization algorithm also performs feature selection as part of its process. Table 2.1 shows the characteristics of the four discretized datasets: the number of original genes, the number of genes after discretization, the two class labels (class 1 and class 0), and the number of rows for training and test data. All experiments presented here use the class 1 as the consequent; we have found that using the other consequent consistently yields qualitatively similar results.

We compare algorithm MineTopkRGS with FARMER, CLOSET+ and CHARM (which uses diff-sets). But CLOSET+ is usually unable to run to completion within reasonable time (for several hours without results) and CHARM will report errors after using up memory on the entropy discretized datasets. Therefore, we only report the runtime of MineTopkRGS and FARMER in discovering the upper bounds of discovered rule groups. The reported time here includes the I/O time. We should point out that MineTopkRGS discovers different kinds of rules from all these existing methods.

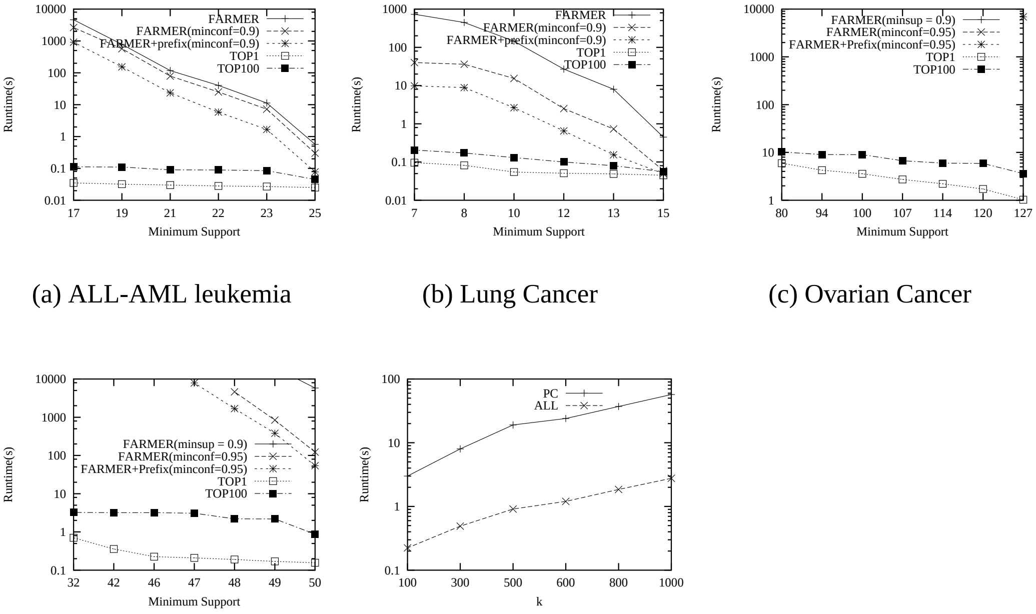
[1] http://www-genome.wi.mit.edu/cgi-bin/cancer

[2] http://www.chestsurg.org

[3] http://clinicalproteomics.steem.com/

[4] http://www-genome.wi.mit.edu/mpr/prostate

[5] the code is available at http://www.sgi.com/tech/mlc/



(a) ALL-AML leukemia

(b) Lung Cancer

(c) Ovarian Cancer

(d) Prostate Cancer

(e) Varying k

Figure 2.5: Comparisons of Runtime on Gene Expression Datasets

Figure 2.5 (a-d) shows the effect of varying minimum support threshold $minsup$. The graphs plot the runtime for the two algorithms at various settings of minimum support. Note that the y-axes in Figure 2.5 are in logarithmic scale. We run algorithm MineTopkRGS by setting the parameter k at 1 and 100 respectively on all the datasets. For FARMER algorithm, we run it by setting minimum confidence $minconf$ at 0.9 and 0 (which disables the pruning with confidence threshold) on datasets ALL, and LC. Due to the relatively large number of rows in the other two datasets, FARMER is slow even when we set $minconf$ at 0.9 and 0.95 respectively. For dataset PC, the runtime curve of FARMER at $minconf$ =0.9 is at the upper right corner. We do not show the runtime of FARMER on dataset OC because it cannot finish in several hours even at $minconf$ =0.95. To further show the effect of prefix tree structure on the runtime and thus the improvement of top-k prunning alone on the runtime, we also implemented FARMER with prefix tree structure and the runtime curve is labelled as "FARMER+prefix". Note that the minimum supports shown in Figure 2.5 are absolute values. We usually vary minimum support from 95% to 60% when measured with a relative value. We begin with a high minimum support in order to allow FARMER to finish in reasonable time.

Figure 2.5 (a-d) shows that MineTopkRGS is usually 2 to 3 orders of magnitude faster than FARMER. Especially at low minimum support, MineTopkRGS outperforms both FARMER+Prefix and FARMER substantially. This is because FARMER discovers a large number of rule groups at lower minimum support while the number of rule groups discovered by MineTopkRGS is bounded. This also explains why MineTopkRGS is not sensitive to the change of minimum support threshold as shown in Figure 2.5. Besides, Figure 2.5 (a-d) demonstrates that the combination of row enumeration and the prefix tree technique speeds up the mining process successfully, by which, FARMER+prefix can improve the efficiency of FARMER by about one order of magnitude.

Figure 2.5 (e) shows the effect of varying k on runtime. We observe similar tendencies on all datasets and report results on datasets ALL and PC only. It is quite reasonable that runtime of MineTopkRGS is monotonously increasing with k.

The impressive performance of MineTopkRGS can be contributed to four main factors. First, TopkRGS bounds the number of discovered rule groups. Second, the row enumeration strategy fits the problem of mining TopkRGS very well.

Third, the prefix tree structure speeds up frequency computation. Fourth, the dynamically generated minimum confidence helps in pruning search space although MineTopkRGS does not require users to specify minimum confidence.

2.5 Summary

In this chapter, we proposed the concept of top-k covering rule groups for each row of gene expression data and an algorithm called MineTopkRGS to find the TopkRGS. Experiments showed that MineTopkRGS outperforms existing algorithms like CHARM, CLOSET+ and FARMER by a large order of magnitude on gene expression datasets.

Although it is true that current gene expression datasets have small number of rows, we may extend TopkRGS to other large datasets that are characteristic of both long columns and a large number of rows by utilizing column-wise mining first, then switching to row-wise enumeration in later levels to mine top-k covering rules in the partitions, and finally aggregating the top-k covering rules in all partitions. It is well known that some item-wise mining algorithms have linear scalability with dataset size. Another method for MineTopkRGS to deal with the memory limitation problem is to utilize the database projection (disk-based) techniques as suggested in [36].

CHAPTER 3

RCBT: Classification with Top K Covering Rule Groups

The pioneering associative classification method is CBA [56]. However, CBA is unable to adapt to gene expression data not only because of its inefficiency in rule mining and excessively huge rule number but also its rule selection scheme. CBA always selects a single one rule of highest significance for each training data. When the generated CBA classifier does not cover a test data, CBA simply outputs the default class. Such case happens quite often for CBA when applying on gene expression data. In fact, discussion with biologists revealed that they are usually reluctant to believe in the classification made by selecting a default class which is done without giving any deciding factors.

IRG classifier in [20] is the first classifier to classify with rule groups. The rule groups selected by IRG classifier are the interesting ones such that no subset rule group of higher significance exists. During the rule selection step, IRG classifier simply chooses the longest rule, the upper bound rule, of each interesting rule group for classifier building. However, according to our experiments, the number of interesting rule groups (IRGs) are still too huge to handle, especially when the confidence and support thresholds are low.

Inspired by CBA and IRG classifier, we propose a new classifier, RCBT, built on the rules delicately selected from top covering k rule groups. We significantly reduce the default class decision cases by building a series of standby classifiers apart from the main one. We also improve the classification accuracy by aggregating the discriminating powers of carefully selected rules. As another benefit of RCBT, CBA classifier can be easily built with the top-1 covering rule groups of RCBT, as we will prove later.

Experiments on benchmark gene expression datasets show that RCBT outperforms or is competitive with CBA [56], IRG classifier [20], SVM [45], and C4.5 family algorithms [70] (single tree, bagging and boosting). Furthermore, we show that our method does provide knowledge of biological significance.

3.1 Background

Recent studies have shown that class association rules are very useful in classification. Due to their relative simplicity, they can be easily interpreted by biologists, providing great help in the search for gene predictors (especially those still unknown to biologists) of the data categories (classes). Moreover, it is shown in [20, 26, 53] that classifiers built from association rules are rather accurate in identifying cancerous cell. RCBT is one novel associative classifier built on class association rules.

Traditional statistical and machine learning methods typically rely on the feature selection (ranked according to measures such as gain ratio, chi-square and etc.) to reduce the number of dimensions for computational efficiency. So does a recent associative classification method PCL [52]. However, the feature selection is problematic: first, it is difficult to determine how many top-ranked genes to be used for classification model; second, as observed in [53] and our experiments, low-ranked genes are often contained in significant rules that are sometimes necessary for perfect classification accuracy.

Our work is closely related with previous associative classification methods [20, 26, 56]. These algorithms first try to mine all rules satisfying minimum support and minimum confidence thresholds, and then sort and prune the discovered rules to get the classification rules. The high-dimensional gene expression data renders these algorithms impractical because of the huge number of discovered rules.

Another related rule-based classification method is decision tree, such as C4.5. The rules generated by C4.5 are exclusive to each other and cover the training data just once. As a result, C4.5 only produces a small set of classification rules, some of which may be biased, and C4.5 may miss global significant rules for perfect prediction. Decision tree method C4.5 has also been criticized for fragmentation problem [63]: many locally important but globally unimportant rules are generated in the process of building decision tree. Committee decision tree techniques of bagging [16] and boosting [30] have been proposed to alleviate the above problems by applying a base C4.5 classifier multiple times using bootstrapped data to generate a committee of classifiers. However, the rules produced by bagging or boosting methods may not be correct for the original training data since they are generated from pseudo training data, and thus may deteriorate classification accuracy. Instead, our method generates a set of globally significant classification rules and aggregates their discriminating powers for classification, thus avoiding the above problems.

3.2 Comparison of RCBT with CBA and IRG Classifier

The pioneering associative classification method CBA suffers serious computational efficiency problem on gene expression data. Comparatively, the recent IRG classifier is adapted much better to gene expression day by systematically grouping class association rules into interesting rule groups. However, IRG classifier conducts the classification with the upper bound rules of interesting rules, the number of which may still be huge. RCBT further improves over the two methods. In this section, we discuss the relationships between RCBT and these two methods.

3.2.1 RCBT and CBA Classifier

We first prove that the set of top-1 covering rule groups for each row contain the set of rules required to build CBA classifier. The basic idea of CBA can be summarized as the following steps:

Step 1: Generate the complete set of class association rules CR for each class that satisfy the user-specified minimum support and minimum confidence. [1]

Step 2: Sort the set of generated rules CR according to the relations ”$\prec$”. Given two rules, r_i and r_j, $r_i \prec r_j$ if and only if one of the following three conditions is satisfied (1) $r_i.conf > r_j.conf$; (2) $r_i.conf = r_j.conf \land r_i.sup > r_j.sup$; or (3) $r_i.conf = r_j.conf \land r_i.sup = r_j.sup$ and r_i is discovered before r_j. Because CBA discovers rules in breadth-first manner, CBA will always assign the shortest rule a higher rank when several rules have the same values of support and confidence.

Step 3: Select rules from sorted rule set CR. For each rule r in CR, if it can correctly classify some training data in D, CBA puts it into classifier C', removes those training data covered by r and continues to test the rules after r in CR. Meanwhile, CBA selects the majority class in the remaining data as default class and computes the errors made by current C' and default class. This process continues until there are no rules or no training data left.

[1] Note that CBA algorithm employs an Apriori-like algorithm for this task and will fail at this step on gene expression data. Likewise, newly proposed column enumeration algorithms, such as CHARM and CLOSET+ also failed.

As can be seen, in CBA, the rule generation scheme using fixed support and confidence thresholds at Step 1 and the rule selection scheme based on coverage test at Step 3 are simply NOT compatible with each other. Because of the extremely high dimensionality of gene expression data, even when the confidence threshold is set as high as 95%, CBA cannot finish running at Step 1 in several days. It is even more ridiculous that most of the time spent is used to generate redundant rules which will eventually be pruned away at Step 3. The following lemma proves that the rules selected by CBA for classification are actually a subset of rules of TopkRGS with $k = 1$.

Lemma 3.2.1 *Given a minimum support. Let Ψ be the set of discovered top-1 covering rule groups for each training data, Ψ_s be the set of shortest lower bounds of Ψ, and C' be the set of rules selected at Step 3 of CBA method. We get $C' \subseteq \Psi_s$.*

Proof: *For each rule $r \in C'$, it must correctly classify some training data. Because of the sorting at step 2 of CBA method, r must be the top-1 covering rule of a training data if it correctly classifies the training data. This means that r must be in Ψ_s. We get the proof.*

Note that mining top-1 covering rule group does not require a *minimum confidence* threshold while CBA algorithm needs one when generating rules at Step 1. Setting too high a confidence threshold will result in some rows not being covered by the discovered rule while lowering the confidence threshold will result in substantial increase in running time. This is unlike our approach which will still find the most significant top-1 covering rule for each training data without specifying an appropriate confidence threshold in advance.

In order to build CBA, we need to discover one of the shortest lower bounds from each top-1 covering rule group. [20] proposed a method to discover all lower bounds of a rule group. However, in entropy-based discretized gene expression datasets, a rule group may contain tens of thousands of lower bounds and discovering all these lower bounds is not only unnecessary but also computationally expensive. Instead of discovering all the lower bounds, we propose a straightforward method to search only a given number of lower bounds for classification purpose.

Lemma 3.2.2 *Rule γ' is a* lower bound rule *of* rule group G *with* upper bound rule γ *iff (1) $\gamma'.A \subseteq \gamma.A$, (2)$|\mathcal{R}(\gamma'.A)| = |\mathcal{R}(\gamma.A)|$ and (3) there is no other rule member γ'' of G such that $\gamma'.A \supset \gamma''.A$.*

With Lemma 3.2.2, we derive the algorithm FindLB() in Figure 3.1. It takes in four parameters: training data D, the upper bound rule γ, the set of rows covered by γ (denoted as $rowset$ and can be recorded when generating γ in algorithm MineTopKRGS), and the number of required shortest lower bounds nl (nl=1 for CBA classifier). At Step 1, we first rank genes based on their discriminant ability in classification measured by entropy score [9], and then rank the items in an upper bound rule based on the rankings of their corresponding genes (one gene may be discretized into several intervals, each represented by an item). In this way, we discover the shortest lower bound rules that contain items from the most discriminant genes to build CBA classifier. At step 2, for a candidate lower bound combination c_{lb}, we first test the condition (3) in Lemma 3.2.2; if condition (3) is satisfied, we continue to test condition (2), which is satisfied only if there does not exist a row $r \in D \wedge r \notin rowset$ that c_{lb} is contained in r. If both (2) and (3) are satisfied, c_{lb} is a lower bound. This process continues until we get the nl lower bound rules.

```
Algorithm FindLB($D$, $\gamma$, $rowset$, $nl$)
  1. Rank the items in $\gamma.A$ according to the descending order of the entropy scores of the corresponding genes;
  2. Perform a breadth-first search in the search space formed by the list of items $\gamma.A$ until we get $nl$ lower bound rules;
```

Figure 3.1: Algorithm FindLB

Both dataset D and candidate lower bound combinations are represented with bitmap to speed up the containment test. The discovered lower bounds usually contain 1-5 items while the upper bounds usually contain hundreds of items in the data we tested. We use one heuristic rule to speed-up the algorithm FindLB. Consider two upper bound rules, γ_1 and γ_2. Let $A' = \gamma_1.A \cap \gamma_2.A$. The lower bound rules of γ_2 will contain at least one item in $\gamma_2.A - A'$ if $\gamma_2.A - A' \neq \emptyset$, and the lower bound rules of γ_1 will contain at least one item in $\gamma_1.A - A'$ if $\gamma_1.A - A'$ $\neq \emptyset$. We can prune the unpromising search space with this strategy.

With the set of lower bound rules, we can build CBA classifier using the method presented in Section 2.2. Note that a minimum confidence threshold can be imposed on the set of lower bounds to filter out rules that do not satisfy the threshold to be consistent with CBA method in [56]. According to our experiments, for some training data, all the covering rules are beneath the specified confidence threshold and will be pruned off totally. This will certainly cause information loss. Comparatively, RCBT requires no specified confidence threshold and is more flexible for use.

3.2.2 RCBT and IRG Classifier

Association rules can reveal biological relevant relationships between genes and environments / categories. However, most existing association rule mining algorithms are rendered impractical on gene expression data, which typically contains thousands or tens of thousands of columns (gene expression levels), but only tens of rows (samples). The main problem is that these algorithms have an exponential dependence on the number of discretized items, which are approximately proportional to the number of columns. Another shortcoming is evident that too many associations are generated from such kind of data. These problems result in extremely long rule discovery runtime.

To address the two problems, the depth-first row-wise algorithm FARMER [20] is specially designed to efficiently discover and cluster association rules into *interesting rule groups* (*IRGs*) satisfying user-specified minimum support, confidence and chi-square value thresholds on biological datasets as opposed to finding association rules individually. Based on the IRGs discovered by FARMER, IRG classifier is built by aggregating the discriminating power of upper bound rules for gene expression data classification. IRG classifier is at present the classifier most related to RCBT.

To have a rough idea of IRG classifier, let’s look at a simple example. Sup-

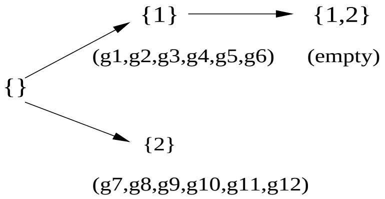


Figure 3.2: Row Enumeration Tree

pose there is a two-row discretized dataset, 1:$\{g_1$, g_2, g_3, g_4, g_5, g_6, $Cancer\}$, 2: $\{g_7$, g_8, g_9, g_{10}, g_{11}, g_{12}, $\neg Cancer\}$, where **item** g_i $(i = 1$, 2, ..., $12)$ is the discretized value of the original gene expression level. We could generate 63 association rules in the form of "$A \rightarrow Cancer$" from the same row set $\{1\}$, where A is any combination of g_1, g_2, ..., g_6, and 63 association rules in the form of "$B \rightarrow \neg Cancer$" from the same row set $\{2\}$, where B is any combination of g_7, g_8, ..., g_{12}. Obviously, many of them are redundant.

IRG classifier utilizes the following three main core techniques.

• *Interesting Rule Groups*: All the above 126 rules of the running example belong to two **rule groups**. One *rule group* is identified with a unique *antecedent support set* [2] $\{1\}$, a unique *upper bound rule* $g_1g_2g_3g_4g_5g_6 \rightarrow Cancer$, and 6 *lower bound rules* $g_i \rightarrow Cancer$, $i = 1$, 2, ..., 6. The other *rule group* is identified with another *antecedent support set* $\{2\}$, a unique *upper bound rule* $g_7g_8g_9g_{10}g_{11}g_{12} \rightarrow \neg Cancer$, and 6 *lower bound rules* $g_i \rightarrow \neg Cancer$, $i = 7$, 8, ..., 12. The rules between the *upper bound rule* and the *lower bound rules* are the remaining members of the corresponding *rule group*. In this way, we only need to generate 2 *upper bound rules*

[2] The *antecedent support set* of a rule is the complete set of rows that contain the antecedent of the rule

and 12 *lower bound rules* instead of all the 126 rules. As can be seen, the rules in the same *rule group* share the same *antecedent support set* and the same consequent, thus the same support, confidence and chi square values. From this point of view, the *rule group* is a lossless compression of the association rules. FARMER only outputs **interesting rule groups (IRGs)**. For two *rule groups* of the same consequent, rg_1 and rg_2, if $rg_1.upperbound \subset rg_2.upperbound$ and rg_1 has a higher confidence, then FARMER only outputs rg_1, because rg_1 is defined to be more interesting.

• *Row Enumeration Combined with Efficient Pruning Strategies*: As the row enumeration space is orders smaller than the column enumeration space in gene expression data, FARMER performs search by a depth-first traversal of a **row enumeration tree**. Each node corresponds to a certain row enumeration, where a **transposed table** is set up and a new *IRG* may be identified. For the simple example, the *row enumeration tree* without applying pruning strategies is shown in Figure 3.2. The traversal starts from the root node $\{\}$, goes through node $\{1\}$ and node $\{1, 2\}$ in sequence, and ends at node $\{2\}$. Figure 2.1 lists the corresponding three non-empty transposed tables, where $R(g_i)$ represents the complete set of rows that contain item g_i. In this way, the *upper bound rule* $g_1g_2g_3g_4g_5g_6 \rightarrow Cancer$ is discovered at node $\{1\}$, and the *upper bound rule* $g_7g_8g_9g_{10}g_{11}g_{12} \rightarrow \neg Cancer$ is discovered at node $\{2\}$. To avoid redundancy and to comply with the minimum measure thresholds, efficient pruning strategies of minimum confidence, support and chi-square are applied to further speed up the mining process.

• *Upper Bound Rules*: Like CBA classifier, after mining upper bound rules of interesting rule groups, IRG classifier first ranks the upper bound rules in statistical significance, then uses the *upper bound rules* of the most significant interesting rule groups to classify unknown test data.

There are still some problems with IRG classifier for gene expression data classification. Although with the concept of interesting rule group, numerous rules discovered from gene expression data are clustered into significantly smaller number of *IRGs*, the number of *IRGs* sometimes can still be quite huge, i.e., tens of thousands especially when the minimum support or minimum confidence thresholds are low. Another drawback is that the IRG classifier coarsely classifies the test data with a single one upper bound rules. That would probably be biased in some occasions. Our RCBT classifier performs a much more significant pruning on the discovered rule groups with the top k covering constraint and combines the discriminating powers of finely selected rules to build a committee of classifiers.

3.3 Rule Group Visualization

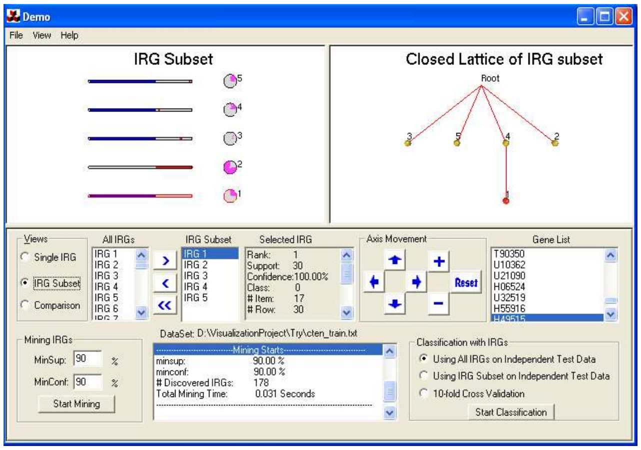
In this section, we introduce visualization technique to effectively interpret and compare the semantics of rule groups. The graphic interface enables users to conduct semantic explorations over the rule groups and identify the most discriminating rule groups rapidly. Besides, the visualization techniques can help understand our rule selection scheme in RCBT classifier.

Figures 3.3, 3.4, and 3.5 show our system interfaces of rule group visualization. The rule groups are sorted based on their rank (descending) as evaluated first by confidence (descending), next by support (descending), and last by # item (ascending). The top 5 rule groups ($RG_1 \prec RG_2 \prec RG_3 \prec RG_4 \prec RG_5$) are specified as the **rule group subset**. Meanwhile the order of the items in the specified rule group subset and the rows in the dataset are determined based on their memberships in the *itemsets*[3] and *antecedent support sets* of the rule groups respectively. An item i will be ranked higher than an item j if the highest ranked rule group that contain i is above the highest ranked rule group that contain j in the rule group ranking. Likewise, a row r will have a higher rank than a row s if the highest ranked rule group that is matched by r is above the highest ranked rule group that is matched by s based on the rule group ranking.

For each rule group, we can visualize its *antecedent support set* and its *itemset* with a "**barcode**" and a "**flower**" separately, or with a "**matrix**" jointly. A "**closed lattice**" graph is also proposed to summarize the rule groups in the rule group subset based on the subset/superset relationship of their *antecedent support sets*.

- *Antecedent Support Set Visualization*: The "**barcode**" (left hand of Figures 3.3 and 3.4) is the identification number of the rule group. The "bar" consists of several

[3] the *itemset* of a rule group is the complete set of items that appear in at least one of the antecedents of the association rules in the rule group



IRG Subset: IRG 1, IRG 2, IRG 3, IRG 4, IRG 5

Selected IRG:

```
Rank:         1
Support:      30
Confidence:100.00%
Class:        0
# Item:       17
# Row:        30
```

Gene List: T90350, U10362, U21090, H06524, U32519, H55916, H49515

MinSup: 90 %

MinConf: 90 %

DataSet: D:\VisualizationProject\Try\cten_train.txt

```
----------------------------Mining Starts----------------------------
minsup:                  90.00 %
minconf:                 90.00 %
# Discovered IRGs:       178
Total Mining Time:       0.031 Seconds
-----------------------------------------------------------------------
```

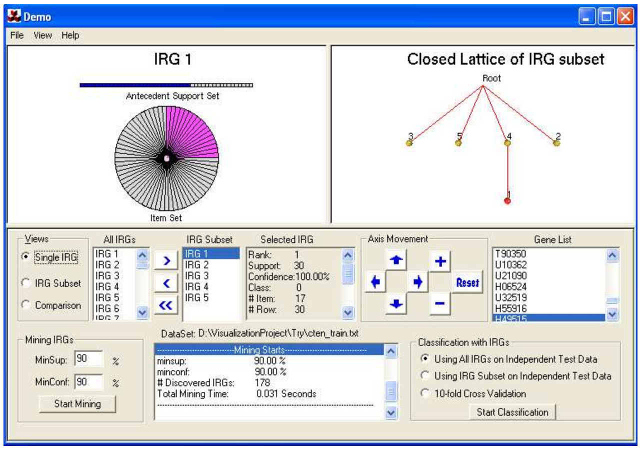
● Using All IRGs on Independent Test Data

○ Using IRG Subset on Independent Test Data

○ 10-fold Cross Validation

Figure 3.3: Semantic Visualization of the rule group Subset Using the Barcode View and the Flower View

small grids, each mapping to one ordered row of the dataset. If the mapped row is a member of the rule group's *antecedent support set*, the grid is dyed according to the class label of the row (i.e., red for "negative", blue for "positive"). In this way, the semantics of the rule group, like support and confidence, can be obtained by a snapshot. The overall "barcode" view (left hand of Figure 3.3) suggests that the *antecedent support set* of RG_1 occupies only the "negative" tissue samples (all red, no blue), while the *antecedent support set* of RG_2 occupies only the "positive" tissue samples (all blue, no red). They are the only two rule groups of confidence 100% in the rule group subset. The "**closed lattice**" (right hand of Figures 3.3 and 3.4) is another summarization based on the superset/subset relationships of the *antecedent support sets* of rule group in the rule group subset. Each node in the lattice except the root node maps to the *antecedent support set* of one rule group in the rule group subset. The *antecedent support set* of the parent node includes that of the child node. The root node corresponds to the set of all the 47 rows.



● Single IRG
○ IRG Subset
○ Comparison

All IRGs: IRG 1, IRG 2, IRG 3, IRG 4, IRG 5, IRG 6, IRG 7

IRG Subset: IRG 1, IRG 2, IRG 3, IRG 4, IRG 5

```
Rank:       1
Support:    30
Confidence: 100.00%
Class:      0
# Item:     17
# Row:      30
```

Gene List: T90350, U10362, U21090, H06524, U32519, H55916, H49515

MinSup: 90 %

MinConf: 90 %

DataSet: D:\VisualizationProject\Try\cten_train.txt

```
----------------------------Mining Starts----------------------------
minsup:                 90.00 %
minconf:                90.00 %
# Discovered IRGs:      178
Total Mining Time:      0.031 Seconds
--------------------------------------------------------------------
```

● Using All IRGs on Independent Test Data
○ Using IRG Subset on Independent Test Data
○ 10-fold Cross Validation

Figure 3.4: Semantic Visualization of a Single rule group Using the Barcode View and the Flower View

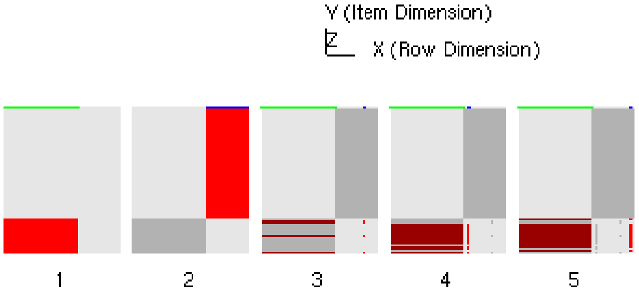


Figure 3.5: Rule Group Comparisons Using the Matrix View

- *Itemset Visualization*: We visualize the *itemset* of the rule group in the user-specified rule group subset as a "**flower**" (left hand of Figures 3.3 and 3.4). Each "flower" corresponds to the same set of ordered items that appear in the rule group subset and each item is represented by a "petal" of the "flower". The "petal" is dyed if the corresponding item appears in the current rule group, otherwise it is left blank.

• *Joined Visualization*: The x-dimension of the “matrix” represents the set of rows in the dataset while the y-dimension of the “matrix” represents the set of items in the rule group subset. The items and rows along each dimension are ordered. Given a “matrix” representing a rule group RG_i, a cell valued (x, y) in the “matrix” will be colored red if item y is in the antecedent of the *upper bound rule* for RG_i and row x matches the *upper bound rule* of RG_i. Due to the ordering of the items and rows, the red cells in the “matrix” of the highest ranked rule group (i.e. RG_1) will always be clustered at the bottom left corner of the “matrix” as can be seen from Figure 3.5.

To compare RG_i against other higher ranked rule groups, a cell in the “matrix” for RG_i will be colored dark grey if it has been colored red in any “matrix” of higher ranked rule groups. For example, the dark grey patch in the “matrix” of RG_2 indicates that these cells have been colored red in the “matrix” of RG_1. In the case in which the cell also has to be painted red to represent RG_i, the color of dark red will be used to paint the cell. Finally, the top most cells in each “matrix” are used to represent the class labels of the corresponding rows. By looking at the highest cells in the “matrix” of RG_1, we can see that RG_1 has a 100% confidence prediction for a certain class. Overall, we can see that RG_1 and RG_2 are the most discriminating rule groups with the largest number of non-overlapped red cells.

With the effective visualization techniques, we can identify the most discriminating rule groups graphically. Intuitively, the most discriminating ones are those with red cells in the matrix view which correspond to the top-1 covering rule groups; the ones with gray red cells in the matrix view correspond to the top-i,

$i > 1$, covering rule groups. By specifying the value of k for top k covering rule groups, we can flexibly make a trade-off between the number of rule groups and the redundancy among them. This is our motivation of top k covering rule group selection scheme. Combined with the entropy rule selection measure, RCBT classifier further identifies a small subset of most significant rules from selected rule groups.

3.4 RCBT Classifier

In this section, we present a refined classification method based on TopkRGS, called RCBT. RCBT improves over CBA method in two aspects:

- RCBT reduces the chance that a test data is classified with default class;
- RCBT uses a subset of rules to make a collective decision.

As discussed earlier, RCBT tries to reduce the chance of classifying test data with default class by building a series of stand-by classifiers apart from the main classifier. Moreover, RCBT carefully combines a subset of lower bound rules to make a collective decision instead of selecting only one shortest lower bound rule as CBA does. The subset of lower bound rules are selected based on the discriminant ability of genes. In this way, RCBT will not miss globally significantly rules which are unable to be identified because of advance feature selection, while concentrate on a small number of informative genes.

Building Classifier: RCBT has two input parameters, k, the number of covering rule groups for each row and nl, the number of lower bound rules to be used.

Let RG_j denote the *set* of rules groups, each of which is a top-j rule group for at least one training data of a certain class. We will thus have k sets of rule groups RG_1, RG_2, ..., RG_k. These k sets of rule groups are used to build k classifiers CL_1, CL_2, ...,CL_k with CL_j being built from RG_j. We call CL_1 the main classifier and CL_2, ..., CL_k backup classifiers. For each rule group in RG_j, RCBT finds its nl shortest lower bound rules by calling algorithm FindLB(). The union of the lower bound rules will be sorted and pruned (as in Step 3 of Section 2.2) to form CL_j.

Besides main and backup classifiers, we set a default class like in CBA, the majority class of the remaining training data.

Prediction: Given a test data t, we will go through CL_1 to CL_k to see whether t can be classified and stop once t is classified. In the case that t cannot be handled by any of the k classifiers, the default class will be assigned to t.

Instead of predicting a test data with the first matching rule as CBA does, RCBT tries to match all rules with an individual classifier (the main classifier or individual standby classifiers) and makes a decision by aggregating voting scores. We design a new voting score for a rule γ^{c_i} by considering both confidence and support as

$$S(\gamma^{c_i}) = \gamma^{c_i}.conf * \gamma^{c_i}.sup/d_{c_i},$$

where d_{c_i} is the number of training data of the class $\gamma.C$, i.e. c_i. Note that $0 \leq S(\gamma^{c_i}) \leq 1$. By summing up the scores of all rules in each class c_i, we get score $S^{c_i}_{norm}$ for normalization. Given a test data t, we suppose that t matches m_i rules of

Dataset	RCBT	CBA	IRG Classifier	C4.5 family			SVM
				single tree	bagging	boosting	
AML/ALL (ALL)	91.18%	91.18%	64.71%	91.18%	91.18%	91.18%	97.06%
Lung Cancer(LC)	97.99%	81.88%	89.93%	81.88%	96.64%	81.88%	96.64%
Ovarian Cancer(OC)	97.67%	93.02%	-	97.67%	97.67%	97.67%	97.67%
Prostate Cancer(PC)	97.06%	82.35%	88.24%	26.47%	26.47%	26.47%	79.41%
Average Accuracy	95.98%	87.11%	80.96%	74.3%	77.99%	74.3%	92.70%

Table 3.1: Classification Results

class c_i: $\gamma(t)^{c_i}_1, \gamma(t)^{c_i}_2, ...\gamma(t)^{c_i}_{m_i}$,. The classification score of class c_i for the test data t is calculated as:

$$Score(t)^{c_i} = (\sum_{i=1}^{m_i} S(\gamma(t)^{c_i}_i))/S^{c_i}_{norm}$$

. We make a prediction for t with the highest classification score.

3.5 Experimental Studies

We evaluate the performance of RCBT on the four gene expression datasets shown in Table 2.1. In term of classification accuracy, we compare the performance of RCBT classifier with CBA, IRG classifier, C4.5 family algorithms (single tree, bagging and boosting), and support vector machine (SVM). For the C4.5 family algorithms, we use the open-source software Weka version 3.2. We use SVM^{light} 5.0 for the SVM algorithm. To keep the comparisons fair, SVM and the C4.5 family algorithms are run using the same genes selected by entropy discretization, but with the original real values of the gene expression levels. Besides, we report the best accuracy of SVM when varying between the linear and polynomial kernel functions. The open-source-code CBA usually cannot finish after running several days. We set the minimum support at 0.7 of the number of instances of the specified class to generate top-1 covering rule group of each row to build CBA classifier. The same minimum support is set for IRG classifier and RCBT. We set minimum confidence 0.8 for IRG Classier (the same threshold is applied to CBA but we find all top-1 covering rule groups satisfy the threshold in our experiments). We set parameters $k = 10$ (TopkRGS) and $nl = 20$ (the number of lower bound rules) for RCBT.

Because the test data of all the benchmark datasets are not biased, the classification accuracy on the independent test data is used to evaluate these classification methods. Table 3.1 lists the classification results on the five datasets.

We first look at the last row of Table 3.1 to have a rough idea of these classifiers on gene expression datasets by comparing their average accuracy on four datasets. We see that the RCBT classifier has the highest average accuracy. Note that the result of IRG classifier on OC is not available since FARMER cannot finish in one day on OC and the average is computed on the other three data.

Comparison with SVM: RCBT outperforms the SVM significantly on dataset PC. SVM achieves the best results on dataset ALL although RCBT is still comparable to SVM on ALL. However, the complexity together with the distance model of SVM is much more complicated than our RCBT classifier and it is hard to derive understandable explanation of any diagnostic decision made by SVM. No doubt, these problems limit the practical use of SVM in biological discovery and clinical practice. In contrast, the RCBT classifier is very intuitive and easy to understand.

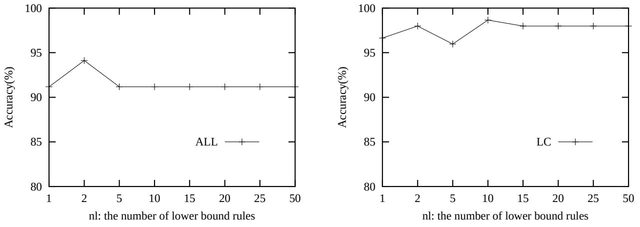
Comparison with C4.5 family algorithms: RCBT usually outperforms the C4.5

family algorithms. The C4.5 family algorithms fail on the PC data while RCBT classifier still performs well. This is because C4.5 always considers the top-ranked genes first when generating the rules to construct the decision trees, and it misses the globally significant rules on the PC data containing lower-ranked genes, as discovered by RCBT.

Comparison with CBA, IRG Classifier and RCBT: RCBT performs better than both CBA and IRG Classifier. Compared with CBA, RCBT classifies much fewer test data using default class. CBA classifies 5 test data (2 errors) on OC and 16 test data (5 errors) on PC using default class while RCBT classifies 1 test data (0 error) on OC, and 1 test data (0 error) on PC using default class. There is no test data classified using default class on ALL and LC for both CBA and RCBT.

For SVM and C4.5, we also try to use only the top 10, 20, 30, or 40 entropy-ranked genes when building the classifier. In both cases, the performances of SVM and C4.5 often become worse. There are two main reasons that contribute to the performance of RCBT classifier. The first is that we build a series of standby classifiers besides the main classifier. The second is that we use a subset of lower bound rules in building classifier. Next, we analyze the effect of both factors in detail and explain how we can set the parameters for RCBT.

Usefulness of Standby Classifiers in RCBT: In our experiments, we set $k = 10$ for TopKRGS to build RCBT classifiers, which means that we build 9 standby clas-



(a) ALL-AML leukemia (b) Lung Cancer

Figure 3.6: Effect of Varying *nl* on Classification Accuracy

sifiers besides a main classifier for each dataset. We find that the standby classifiers classify 2 test data of OC (no error) and 2 test data of PC (no error). On datasets ALL and LC, the main classifier makes all decision. This shows the usefulness of standby classifiers. We would like to stress that these standby classifiers not only improve the classification accuracy but also make the results more convincing to biologists since most test data are not classified by default class.

We also find that only the first 4 standby classifiers are used to classify some test data on all the four datasets. Therefore, RCBT is quite insensitive to the value of k as long as k is set to a sufficiently large value.

Sensitivity Analysis of nl for RCBT: We set $nl = 20$ to build RCBT classifier. Figure 3.6 shows the effect of varying nl on the classification accuracy on datasets ALL and LC. Both curves are quite plain especially when $nl > 15$ (changing nl does not affect accuracy). We observe similar trend on other datasets and only report results on ALL and LC. Again, as long as nl value is set reasonably large, RCBT will not be affected by it.

We also study the effect of varying minimum support thresholds from 0.6 to 0.8 on accuracy and find that the performance of both CBA and RCBT are not affected for all datasets.

As can be seen, the discovered TopkRGS are shown to be useful for classification for both CBA and RCBT. RCBT is both accurate and easy to understand. The parameters for RCBT are also easy for tuning. Besides, experimental results show that some important genes used in RCBT are really responsible for cancer pathogenesis.

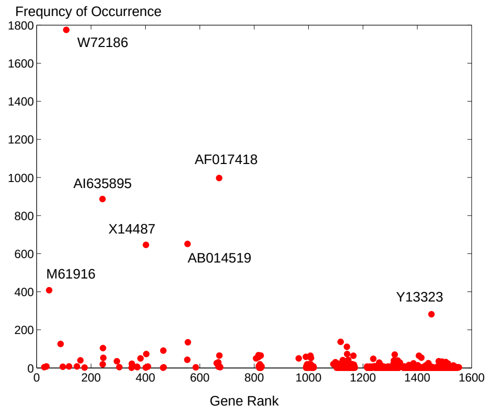


Figure 3.7: Chi-square based Gene Ranks and the Frequencies of Occurrence of the 415 Genes which Form the Top-1 Covering Rules of RCBT on the Prostate Cancer Data. Genes whose Frequencies of Occurrence are Higher than 200 are Labelled.

Biological Meaning: As the lower bound rules RCBT selected from the Prostate Cancer data contain genes of lower-ranks, it is interesting to have a further study of the relationship between gene ranks and usefulness in the lower bound rules. We assume that the more important genes are more likely to be used in the globally significant rules. Figure 3.7 illustrates the chi-square based gene ranks and the frequencies of occurrence of 415 genes (which are involved in forming the top-1 rule groups) in the shortest lower bound rules of top-1 rule groups. As can be seen, most of the genes that occur frequently in the rules are those that are ranked high in the chi-square based ranking (most are ranked 700^{th} and above).

This includes six genes which occur more than 200 times in the discovered lower bound rules of the Prostate Cancer data: M61916 (408 times), W72186 (1775 times), AI635895 (887 times), X14487 (646 times), AB014519 (651 times), and AF017418 (997 times). Among the lower ranked gene, only gene Y13323 occurs for a large number of times (282).

This indicates that the genes of lower ranks generally serve as a certain supplementary information provider for the genes of higher ranks. The large proportion of lower-ranked genes also suggests their necessity for globally significant rules. Based on the experiment, we suspect that the 7 most active genes, M61916, W72186, AI635895, X14487, AB014519, AF017418, and Y13323, are most likely to be correlated with the disease outcomes. Interestingly, gene AF017418 of rank 671 corresponds to MRG1 which has been reported to be useful in detecting glycosphingolipid antigen present in normal epithelium and superficial bladder tumor in patients with blood group A or AB, but absent in the invasive type of bladder (essentially prostate) tumor [51]. Also stated in [6, 15, 32, 47], MRG1 may function as a coactivator through its recruitment of p300/CBP in prostate cancer cell lines and stimulate glycoprotein hormone α-subunit gene expression. Gene AB014519 is related to Rock2 under certain cancer pathway known as the Wnt/planar cell polarity pathway [4]. X14487 is also a cancer-related gene for acidic (type I) cytokeratin. As reported in [61], X14487 shows consistently different expression levels in OSCC tissues and is one of the potential biomarkers for lymph node metastasis.

3.6 Summary

Our RCBT method has addressed an open problem of default class of previous associative classification methods [20, 26, 56] with the backup classifiers. RCBT also improves classification accuracy over CBA and IRG classifier by aggregating the discriminating powers of a subset of rules selected w.r.t. gene discriminant ability from global significant rule groups.

This chapter also showed that the set of top-1 covering rule group for each row makes it feasible to build CBA classifier. Our experiments showed RCBT has the highest average accuracy compared with CBA, IRG classifier, SVM and C4.5 family. Moreover, RCBT classifier is more understandable for biologists than SVM because rules themselves are intuitive.

In the future, we can investigate to mine top-k rule group for traditional datasets with large number of rows and relatively small number of columns to avoid wasting computation in generating a large number of useless rules. We also plan to test the performance of RCBT on such datasets.

[4] http://www.csl.sony.co.jp/person/tetsuya/Pathway/Cancer-related/cancer-related.html, http://www.csl.sony.co.jp/person/tetsuya/Pathway/Cancer-related/Wnt/Wnt-planar

CHAPTER 4

CURLER: Finding and Visualizing Nonlinear Correlation Clusters

Like data objects in other high-dimensional data, genes are NOT globally correlated in all conditions because of the inherent sparsity of high dimensionality. Instead, a cluster of genes may be strongly correlated only in a subset of conditions. Furthermore, the nature of such correlation is usually local to a subset of the genes, and it is possible for another subset of the genes to be correlated in a different subset of conditions. Traditional clustering methods of detecting correlations like GDR [77] and PCA [46] are not applicable in this case since they can detect only correlations in whole databases.

To handle the above problem, several subspace clustering algorithms such as

ORCLUS [2] and 4C [14] have been proposed to identify local correlation clusters with arbitrary orientations, assuming each cluster has a fixed orientation. They identify clusters of data objects which are linearly correlated in some subset of the features.

Correlation between genes or other data objects in high-dimensional data could however be nonlinear, depending on how the data is normalized and scaled [35]. Physical studies have shown that the pressure, volume and temperature of an ideal gas exhibit nonlinear relationships. In biology, it is also known that the co-expression patterns of genes in a gene network can be nonlinear [34]. Without any detailed domain knowledge of a dataset, it is difficult to scale and normalize the dataset such that all nonlinear relationships become linear. It is even possible that the scaling and normalization themselves cause linear relationships to become nonlinear in some subset of features.

In this chapter, we focus on detecting and visualizing nonlinear correlation clusters in subspace. Not restricted to gene expression data, our method can be applied to other high-dimensional data with complex correlation as well.

Detecting nonlinear correlation clusters is challenging because the clusters can have both **local** and **global** orientations, depending on the size of the neighborhood being considered. As an example, consider Figure 4.1, which shows a 2D sinusoidal curve oriented at 45 degrees to the two axes. Assuming the objects cluster around the curve, we will be able to detect the global orientation of this cluster if we consider a large neighborhood which is represented by the large circle centered at point p. However, if we take a smaller neighborhood at point q, we will only find

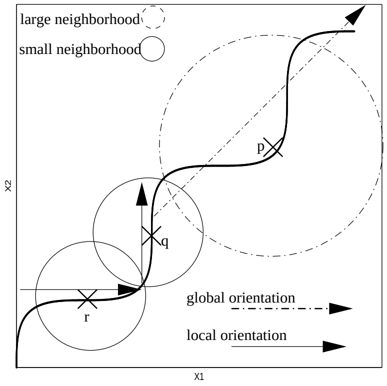


Figure 4.1: Global vs Local Orientation

the local orientation which can be very different from the global one. Furthermore, the local orientations of two points that are spatially close may not be similar at the same time, as can be seen from the small neighborhoods around q and r.

We next look at how the presence of local and global orientations may pose problems for existing correlation clustering algorithms like ORCLUS [2] and 4C [14]. These algorithms usually work in two steps. First, small clusters called **microclusters** [78, 79] are formed by grouping small number of objects that are near each other. Second, microclusters that are close both in proximity and orientation are merged in a bottom-up fashion to form bigger clusters. For nonlinear correlation clusters, such approaches will encounter two problems:

1) **Determination of Neighborhood**

Given that the orientation of a microcluster is sensitive to the size of the neighborhood from which its members are drawn, it is difficult to determine a neighborhood size in advance such that both the local and global orientations of the clusters are captured. Combined with the fact that spatial proximity must be judged based on a subset of the features that are not originally known, forming microclusters that capture the orientation of their neighborhood becomes a major challenge.

2) **Judging Similarity between Microclusters**

Since the orientations of two microclusters in close proximity can be very different, judging the similarity between two microclusters becomes non-trivial. Given a pair of microclusters which have high proximity [1] but very different orientations and another pair with similar orientations but low proximity, the order of merging for these two pairs cannot be easily determined. This in turn affects the final clustering result. One way to avoid this problem is to assign different weights to the importance of proximity and orientations, and then compute a combined similarity measure. However, it is not guaranteed that there will always be a unique weight assignment that gives a good global clustering result.

In this chapter, we aim to overcome the above problems in finding nonlinear correlation clusters. Our contributions are as follows:

1. We highlight the existence of local and global orientations in nonlinear correlation clusters and explain how they pose problems for existing subspace clustering algorithms like ORCLUS [2] and 4C [14], which are designed to find linear correlation clusters.

[1] Note that as mentioned earlier, judging proximity by itself is a difficult task since the two microclusters could lie in different subspaces. We assume that the problem is solved here for ease of discussion.

2. We design an algorithm called CURLER [2], for finding and visualizing complex nonlinear correlation clusters. Unlike many existing algorithms which use a bottom-up approach, CURLER adopts an interactive top-down approach for finding nonlinear correlation clusters so that both global and local orientations can be detected. A fuzzy clustering algorithm based on expectation maximization (EM) [43] is adopted to form microclusters so that neighborhoods can be determined naturally and correctly. The algorithm also provides a similarity measure called **co-sharing level** that avoids the need to judge the importance of proximity and orientation when merging microclusters.

3. We present extensive experiments to show the efficiency and effectiveness of CURLER.

4.1 Background

Existing clustering algorithms can be grouped into two large categories: full space clustering, to which most traditional clustering algorithms belong, and subspace clustering.

The clustering strategies utilized by full space clustering algorithms mainly include *partitioning-based clustering*, which favors spherical clusters such as the k-medoid [48] family and EM algorithm like [43]; and *density-based clustering*, represented by DBSCAN [28], DBCLASD [82], DENCLUE [1] and the more recent OPTICS [7]. EM clustering algorithms such as [73] compute probabilities of cluster

[2] CURLER stands for CURve cLustERs detection.

memberships for each data object according to certain probability distribution; the aim is to maximize the overall probability of the data. For density-based algorithms, OPTICS is the algorithm most related to our work. OPTICS creates an augmented ordering of the database, thereby representing the density-based clustering structure based on ‘core-distance’ and ‘reachability-distance’. However, OPTICS has little concern for the subspace where clusters exist or the correlation among a subset of features.

As large amounts of high-dimensional data have resulted from various application domains, researchers argue that it is more meaningful to find clusters in a subspace. Several algorithms for subspace clustering have been proposed in recent years.

Some subspace clustering algorithms like CLIQUE [4], OptiGrid [39], ENCLUS [41], PROCLUS [3], and DOC [68] only find axis-parallel clusters. More recent algorithms such as ORCLUS [2] and 4C [14] can find clusters with arbitrarily oriented principle axes. However, none of them addresses our issue of finding nonlinear correlation clusters. All these algorithms address clusters with linear orientation only.

4.2 Algorithm

Our algorithm, CURLER, works in an interactive and top-down manner. It consists of the following main components.

1. EM Clustering: A modified expectation-maximization subroutine $EMCluster$

is applied to convert the original dataset into a sufficiently large number of refined microclusters with varying orientations. Each microcluster M_i is represented by its mean value μ_i and covariance matrix Σ_i. At the same time, a similarity measure called co-sharing level between each pair of microclusters is computed.

2. Cluster Expansion: Based on the co-sharing level between the microclusters, a traversal through the microclusters is carried out by repeatedly choosing the nearest microcluster in the co-shared $\epsilon - neighborhood$ of a currently processed cluster. We denote this subroutine as $ExpandCluster$.

3. NNCO plot (Nearest Neighbor Co-sharing Level & Orientation plot): In this step, nearest neighbor co-sharing levels and orientations of the microclusters are visualized in cluster expansion order. This allows us to visually observe the nonlinear correlation cluster structure and the orientations of the microclusters from the NNCO plot.

4. According to the NNCO plot, users may specify clusters that they are interested in and further explore the local orientations of the clusters with regard to their global orientation.

In the next subsections, we will explain the algorithm in details and the reasoning behind it.

4.2.1 EM-Clustering

Like k-means, the EM-clustering algorithm is an iterative k-partitioning algorithm which improves the conformability of the data to the cluster model in each iteration and typically converges in a few iterations. It has various attractive characteristics that make it suitable for our purpose. This includes the **clustering model** it uses, the fact that it is a fuzzy clustering one with iterative refinement.

Clustering Model

In EM-clustering, we adopt a Gaussian mixture model where each microcluster M_i is represented by a probability distribution with density parameters, θ_i={μ_i, $\sum_i$}, μ_i and $\sum_i$ being the mean vector and covariance matrix of the data objects in M_i respectively. Such representation is sufficient for any arbitrarily oriented clusters. Furthermore, the orientation of the represented cluster can be easily computed.

Banfield and Raftery [43] proposed a general framework for representing the covariance matrix in terms of its eigenvalue decomposition:

$$\Sigma_i = \lambda_i D_i A_i D_i^T, \tag{4.1}$$

where D_i is the orthogonal matrix of eigenvectors, A_i is a diagonal matrix whose elements are proportional to the eigenvalues of Σ_i, and λ_i is a scalar. D_i, A_i and λ_i together determine the geometric features (shape, volume, and orientation respectively) of component θ_i.

Fuzzy Clustering

Unlike ORCLUS and 4C in which each data object either belongs or not belongs to a microcluster, EM-clustering is a fuzzy clustering method in which each data object has a certain probability of belonging to each microcluster.

Given a microcluster with density parameters θ_k, we compute the probability of a data object x's occurrence given θ_k as follows:

$$PR(x|\theta_i) = \frac{1}{\sqrt{(2\pi)^d |\sum_i|}} exp[-\frac{1}{2}(x-\mu_i)^T(\Sigma_i)^{-1}(x-\mu_i)], \tag{4.2}$$

where x and mean vector μ_i are column vectors, $|\Sigma_i|$ is the determinant of Σ_i, and $(\Sigma_i)^{-1}$ is its inverse matrix.

Assuming the number of microclusters is set at k_0, the probability of x occurrence given the k_0 density distributions will be:

$$PR(x) = \sum_{i=1}^{k_0} W_i PR(x|\theta_i), \tag{4.3}$$

The coefficient W_i (matrix weights) denotes the fraction of the database given microcluster M_i. The probability of x belonging to a microcluster with density parameters θ_i can then be computed as:

$$PR(\theta_i|x) = \frac{W_i PR(x|\theta_i)}{PR(x)}. \tag{4.4}$$

There are two reasons for adopting fuzzy clustering to form microclusters. First, fuzzy clustering allows an object to belong to multiple correlation clusters when the microclusters are eventually merged. This is entirely possible in real life

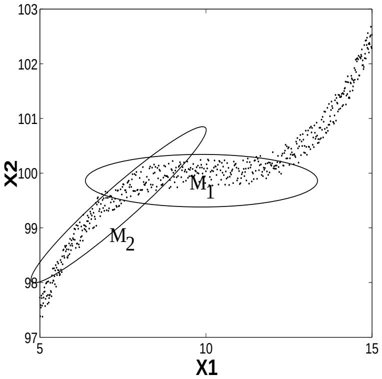


Figure 4.2: Co-sharing between Two Microclusters

datasets. For example, a hospital patient may suffer from two types of disease A and B, and thus his/her clinical data will be similar to other patients of disease A in one subset of features and also similar to patients of disease B in another subset of features. Second, fuzzy clustering allows us to indirectly judge the similarity of two microclusters by looking at the number of objects that are co-shared between them. More specifically, we define the following similarity measure:

Definition 4.2.1 *Co-sharing Level*

The co-sharing level between microclusters M_i and M_j is:

$$coshare(M_i, M_j) = \sum_{x \in D}[PR(M_i|x) * PR(M_j|x)], \tag{4.5}$$

where x is a data object in the dataset D, $PR(M_j|x)$ and $PR(M_i|x)$ are the probabilities of object x belonging to microcluster M_i and microcluster M_j respectively. $PR(M_j|x)$ and $PR(M_i|x)$ are calculated according to Equations 4.4 and 4.2. □

Given each data object in the database, we compute the probabilities of the object belonging to both M_i and M_j at the same time and sum up these probabilities over all the data objects. In this way, the co-sharing level takes both the orientation and spatial distance of two microclusters into account without needing to explicitly determine their importance in computing the similarity. A high co-sharing value between two microclusters indicates that they are very similar while a low co-sharing value indicates otherwise. As an example, consider Figure 4.2 where two microclusters, M_1 and M_2, are used to capture the bend in a cubic curve. Since M_1 and M_2 are neighboring microclusters, points that belong to both the Gaussian distributions will increase the co-sharing level between them.

Note that this similarity measure is important here simply because we are handling nonlinear correlation clusters [3]. For linear correlation algorithms like ORCLUS and 4C, this measure is unnecessary as they can simply not merge two microclusters which are either too far apart or very dissimilar in orientation.

On the basis of our new *co-sharing level*, we will define the co-shared $\epsilon - neighborhood$ and nearest neighbor co-sharing level (NNC) for microclusters.

Definition 4.2.2 *Co-shared* $\epsilon - neighborhood$

For a microcluster M_c***, its co-shared*** $\epsilon - neighborhood$ ***refers to all the microclusters whose co-sharing level from*** M_c ***is no smaller than some non-negative real number*** ϵ***:*** $\{\forall M_i|$ ***coshare(***M_c***,*** M_i***)*** $\geq \epsilon\}$. □

We will explain how these definitions will be useful in the section on cluster expansion later.

[3] As an analogy, consider how soft metals like iron, copper, etc., can be easily bended because of their stretchable bond structures. Correspondingly, we can now 'stretch' data objects across microclusters because of fuzzy clustering so that the merged microclusters can conform to the shape of the nonlinear correlation clusters.

Iterative Refinement

Like the well-known k-means algorithm, EM-clustering is an iterative refinement algorithm which improves the quality of clustering iteratively towards a local optimality. In our case, the quality of clustering is measured by the log likelihood for the Gaussian mixture model as follows:

$$E(\theta_1, \ldots, \theta_{k_0}|D) = \sum_{x \in D} log[\sum_{i=1}^{k_0} W_i \cdot PR(M_i|x)] \tag{4.6}$$

The EM-clustering algorithm can be divided into two steps: E-Step and M-Step. In E-Step, the memberships of each data object in the microclusters are computed. The density parameters for the microclusters are then updated in M-Step. The algorithm iterates between these two steps until the change in the log likelihood is smaller than a certain threshold between one iteration and another. Such iterative change of memberships and parameters is necessary in order to break the catch-22 cycle described below:

1. Without knowing the relevant correlated dimensions, it is not possible to determine the actual neighborhood of the microclusters.

2. Without knowing the neighborhood of the microclusters, it is not possible to estimate their density parameters i.e., the mean vectors and the covariance matrixes of the microclusters.

By sampling the mean vectors from the data objects and setting the covariance matrix to the identity matrix initially, the iterative nature of EM-clustering conforms the microclusters to their neighborhood through the iterations. Again, we note that our approach here is different from that of ORCLUS and 4C. ORCLUS does not recompute the microcluster center until two microclusters are merged, while 4C fixes its microclusters by gathering objects that are within a distance of ϵ of an object in full feature space. Our approach is necessary as we are finding more complex correlations. Incidentally, both ORCLUS and 4C should encounter the same catch-22 problem as us, but they are relatively unaffected by their approximation of the neighborhood.

The $EMCluster$ subroutine is illustrated in Figure 4.3. First, the parameters of each microcluster M_i ($M_i \in MCS$) are initialized as follows: $W_i = 1/k_0$, $\Sigma^0_{M_i}$ is the identity matrix, and the microcluster centers are randomly sampled from the dataset. The membership probabilities of each data object x ($x \in D$), $PR(M_i|x)$, are computed for each microcluster M_i. Then the mixture model parameters are updated based on the calculated membership probabilities of the data objects. The membership probability computation and density parameter updating iterate until the log likelihood of the mixture model converges, or if the maximum number of iterations, $MaxLoopNum$, is reached. The output of the EM clustering is the means and covariance matrices of the microclusters, and also the membership probabilities of each data object in the microclusters. These results are passed on to the $ExpandCluster$ subroutine.

```
EMCluster($D$, $MCS$, $\epsilon_{likelihood}$, $MaxLoopNum$)
1. Set the initial iteration Num. $j = 0$,
    initialize the mixture model parameters,
    $W_i$, $\mu_i^0$ and $\Sigma_i^0$, for each microcluster $M_i \in MCS$.
2. (E-Step) For each data object $x \in D$:
    $PR^j(x) = \sum_{M_i \in MCS} W_i PR^j(x|M_i)$,
    $PR^j(M_i|x) = \frac{W_i * PR^j(x|M_i)}{PR^j(x)}, M_i \in MCS$,
    $W_i' = \sum_{x \in D} PR^j(M_i|x)$.
3. (M-Step) Update mixture model parameters for $\forall M_i \in MCS$:
    $\mu_i^{j+1} = \frac{\sum_{x \in D}(x \cdot PR(M_i|x))}{\sum_{x \in D} PR(M_i|x)}$,
    $\Sigma_i^{j+1} = \frac{\sum_{x \in D} PR(M_i|x)(x - \mu_i^{j+1})(x - \mu_i^{j+1})^T}{\sum_{x \in D} PR(M_i|x)}$
    $W_i = W_i'$
4. if $|E^j - E^{j+1}| \leq \epsilon_{likelihood}$ or $j > MaxLoopNum$
        Decompose $\Sigma_i$ for $\forall M_i \in MCS$ and return
    else set $j = j + 1$ and go to 2.

  Note:
$E^j$: the log likelihood of the mixture model at iteration $j$, $PR^j(x|M_i) = \frac{1}{\sqrt{(2\pi)^d|\sum_i^j|}} exp[-\frac{1}{2}(x - \mu_i^j)^T(\Sigma_i^j)^{-1}(x - \mu_i^j)]$.
```

Figure 4.3: EMCluster Subroutine

4.2.2 Cluster Expansion

Having formed the microclusters, our next step is to merge the microclusters in a certain order so that the final nonlinear correlation clusters can be found and visualized.

Definition 4.2.3 *Co-sharing Level Matrix*

The co-sharing level matrix is a $k_0 \times k_0$ matrix with its entry (i, j) representing the *co-sharing level between microclusters M_i and M_j (coshare(M_i, M_j)).* □

We calculate the co-sharing level matrix at the beginning of the cluster expansion procedure based on the membership probabilities $PR(M_i|x)$ for each data object x and each microcluster M_i. To avoid the complexity of computing $k_0 \times k_0$ entries for each data object x, we instead maintain for each x, a list of l_{top} microclusters that x is most likely to belong to. This reduces the number of entries update to l_{top}^2. We argue that x has 0 or near 0 probability of belonging to most of the microclusters and thus our approximation should be accurate.

As shown in Figure 4.4, the ExpandCluster subroutine first initializes the current cluster C as $\{M_c\}$, where M_c is the first unprocessed microcluster in the set of microclusters MCS. It then merges all other microclusters that are in the co-shared ϵ-neighborhood of M_c into N_C through the function call to neighbors(M_c, ϵ, MCS). M_c is then output together with its co-sharing level value with C. From among the unprocessed microclusters in N_C, the next M_c with the highest co-sharing level is found. C_{new} is then formed by merging M_c and C. We then update the co-sharing level matrix according to Equation 4.7.

$$coshare(C, M_k) = Max(coshare(C, M_k),\ coshare(M_c, M_k)), \tag{4.7}$$

where M_k is any of the remaining unprocessed microclusters.

C is then updated to become C_{new} and unprocessed microclusters in the co-shared ϵ-neighborhood of MC are added to N_C. This process continues until N_C is

```
ExpandCluster($MCS$, $\epsilon$, OutputFile)
1. Calculate the co-sharing level matrix;
2. $M_c$=$MCS$.NextUnprocessedMicroCluster
   $C$ =$\{M_c\}$ ;
3. $N_C$ = neighbors($M_c$, $\epsilon$, $MCS$);
   $M_c$.processed = True;
   Output $M_c$ to OutputFile;
   **while** $|N_C| > 0$ **do**
      From $N_C$, remove nearest microcluster to $C$,
      and set it as $M_c$;
      $M_c$.processed = True;
      Output $M_c$ and coshare($M_c$,$C$) to OutputFile;
      Merge $C$ and $M_c$ to form new $C_{new}$;
      Update the co-sharing level matrix;
      $C$=$C_{new}$;
      $N_C$=$N_C$ + neighbors($M_c$, $\epsilon$, $MCS$);
4. **if** there exist unprocessed microclusters goto 2;
End.
```

Figure 4.4: ExpandCluster Subroutine

empty and then a C is re-initialized to another unprocessed microcluster by going to Step 2.

4.2.3 NNCO Plot

In the NNCO (Nearest Neighbor Co-sharing Level & Orientation) plot, we visualize the nearest neighbor co-sharing levels together with the orientations of the microclusters in cluster expansion order. The NNCO plot consists of a NNC plot above and an orientation plot below, both sharing the same horizontal axis.

NNC Plot

The NNC plot is inspired by the reachability plot of OPTICS [7]. The horizontal axis denotes the microcluster order in the cluster expansion, and the vertical axis above denotes the co-sharing level between the microcluster M_c and the cluster being processed C when M_c is added to C. We call this value the NNC (Nearest Neighbor Co-sharing) value of MC. Intuitively, the NNC plot represents a local hill-climbing algorithm which moves towards the local region with the highest similarity at every step. As such, in the NNC plot, a cluster will be represented with a hill shape with the up-slope representing the movement towards the local high similarity region and the down-slope representing the movement away from the high similarity region after it has been visited. Note that an NNC level of zero or nearly zero represents a complete separation between two clusters, i.e., the two clusters are formed from two sets of microclusters that do not co-share any data objects.

Orientation Plot

Below the NNC plot is the orientation plot, a bar consisting of vertical black-and-white lines. For each microcluster, there is a vertical line of d segments where d is the dimensionality of the data space, and each provides one dimension value of the microcluster's orientation vector, as defined below.

Definition 4.2.4 *Cluster Orientation*

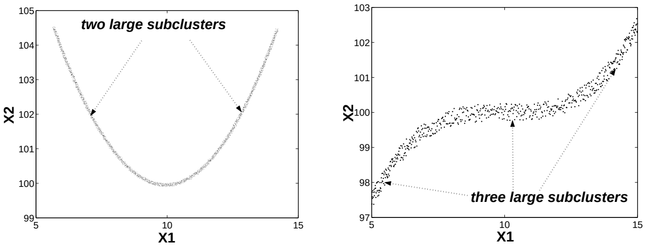
The cluster's orientation is a vector along which the cluster obtains maximum variation, that is, the eigenvector with the largest eigenvalue. □

Each dimension value y of the microcluster orientation vector is normalized to the range of [-127.5, 127.5] and mapped to a color ranging from black to white according to Equation 4.8.

$$Color(y) = [R(y + 127.5), G(y + 127.5), B(y + 127.5)] \tag{4.8}$$

Therefore, the darkest color ([$R(0)$, $G(0)$, $B(0)$], when $y = -127.5$) indicates the orientation parallel but opposite the corresponding dimension axis while the brightest color ([$R(255)$, $G(255)$, $B(255)$], when $y = +127.5$) indicates the orientation parallel and along the dimension axis. Gray ([$R(127.5)$, $G(127.5)$, $B(127.5)$], when $y = 0$) suggests no variation at all in the dimension. Obviously, similarly oriented microclusters tend to have similar patterns in the orientation plot. In this way, the clusters' specific subspaces can be observed graphically.

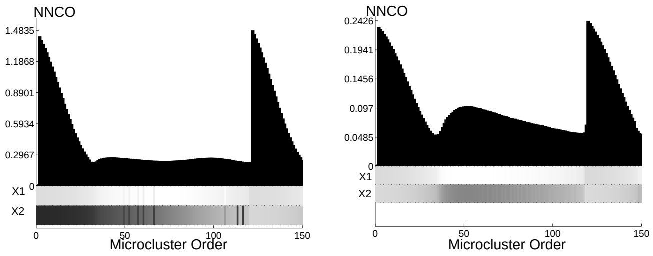
Examples



(a) Quadratic Cluster (b) Cubic Cluster

Figure 4.5: Quadratic and Cubic Clusters

Figure 4.5 shows a quadratic cluster and a cubic cluster. The nonlinear cluster structures are detected successfully, as shown in the NNCO plots in Figure 4.6. According to Definition 4.2.1, the more similar in orientation the microclusters are, the larger the co-sharing level value they have. As our microclusters are assumed to



(a) Quadratic

(b) Cubic

Figure 4.6: NNCO Plots

be evenly distributed, the microclusters which are similar in orientations and close to each other have larger NNC values and tend to be grouped together. Here, the microcluster orientations are approximately the tangents along the curves. There are two humps, indicating two large subclusters of similar orientations in the quadratic NNC plot (Figure 4.6(a)). Likewise, there are three humps, indicating three large subclusters of similar orientations in the cubic NNC plot (Figure 4.6(b)).

Generally, the tangent projection along the quadratic curve in X_2 dimension increases from negative to positive while the tangent projection on the X_1 dimension increases and decreases symmetrically. The simple mathematic reasoning behind this is that, given the 2D quadratic curve

$$x_2 = a * (x_1 - b)^2 + c,$$

where $a > 0$, the changing ratio of the tangent slop, $x_2^{''} = 2 * a$, is a positive constant. The maximum tangent projection on the X_1 dimension is achieved when the tangent slope is 0. That is why we see in the orientation plot that as a whole, the bar color in dimension X_2 brightens continuously (tangent slope changes from

```
CURLER($D$, $k_0$, $l_{top}$, $\epsilon$, $\epsilon_{likelihood}$, $MaxLoopNum$)
1.Randomly Sample $k_0$ number of seeds from $D$
    as $MCS$;
2.EMCluster($D$, $MCS$, $\epsilon_{likelihood}$, $MaxLoopNum$);
3.Select one microcluster in $MCS$ as $c$;
4.ExpandCluster($MCS$, $\epsilon$, $OutputFile$);
5. For any interesting cluster $C_i$
    Transform $D^{C_i}$ into $D_{new}$ in the subspace $\varepsilon_l^{C_i}$;
    CURLER($D_{new}$, $k_0'$, $l_{top}$, $\epsilon$, $\epsilon_{likelihood}$, $MaxLoopNum$)
End.
```

Figure 4.7: CURLER

negative to positive) while the bar color in dimension X_1 brightens first and darkens mid-way.

For the cubic curve $x_2 = a * (x_1 - b)^3 + c$, the tangent slope changes from positive to zero, then back to positive again. Again, as the tangent projection on dimension X_1 increases and decreases symmetrically while the tangent projection on dimension X_2 decreases and increases symmetrically. For this reason, the bar color in dimension X_1 brightens and darkens symmetrically while the bar color of dimension X_2 darkens and brightens symmetrically in the orientation plot.

4.2.4 Top-down Clustering

Having identified interesting clusters from the orientation plot, it is possible to perform another round of clustering by focusing on each individual cluster. The reason for doing so is that the orientation captured by the initial orientation plot could only represent the global orientation of the clusters.

As we know, each data object is assumed to have membership probabilities for several microclusters in CURLER. We define the **data members** represented by a discovered cluster C which consists of microcluster set MCS as the set of data objects whose highest membership probabilities are achieved in the microcluster among MCS, $\{\forall x | x \in D$ and $\exists M_c \in MCS$ such that $Max_{1 \leq i \leq k_0}\{PR(M_i|x)\} = PR(M_c|x)\}$. Based on the data members of cluster C, we can further compute the cluster existance space of C.

Definition 4.2.5 *Transformed Clustering Space*

Given the specified cluster C and l, we define the **transformed clustering space** *of C as a space spanned by l vectors, denoted as ε_l^C, in which the sum of the variances along the l vectors is the least among all possible transformations. In other words, the l vectors of the transformed clustering space ε_l^C are the l eigenvectors with minimum eigenvalues, computed from the covariance matrix of the data members of C. We denote the l vectors as e_1, e_2, ..., and e_l, where l may be much smaller than the dimensionality of the original data space d.* □

Given the dimensionality of the original data space, d, a correlation cluster C_i, and l, we can further project data members of C_i, D^{C_i}, to the subspace $\varepsilon_l^{C_i}$ of l vectors ($\varepsilon_l^{C_i} = \{e_{i1}$, e_{i2}, ..., $e_{il}\}$) by transforming each data member $x \in D^{C_i}$ to ($x \cdot e_{i1}$, $x \cdot e_{i2}$, ... $x \cdot e_{il}$), where x and e_{ij} ($1 \leq j \leq l$) are d-dimensional vectors. In this way, we obtain a new $l-$dimensional dataset and can carry on another level of clustering. Figure 4.7 shows the overview of our algorithm.

4.2.5 Time Complexity Analysis

In this section, we analyze the time complexity of CURLER. We focus our analysis on the EM-clustering algorithm and the cluster expansion since these two are the most expensive steps among the four.

• **EM Clustering:**

In the EM part, the algorithm runs iteratively to refine the microclusters. The bottleneck is Step 2, where the membership probability of each data object x for each microcluster $M_i \in MCS$ is calculated. The time complexity of matrix inversion, matrix determinant, and matrix decomposition is $O(d^3)$; thus, the time complexity of matrix operation for k_0 microclusters is $O(k_0 \cdot d^3)$. Besides, the time complexity of computing $PR^j(x|M_i)$ is $O(d^2)$ for each pair of x and M_i. For all data objects and all microclusters, the total time complexity of EM clustering is $O(k_0 \cdot n \cdot d^2 + k_0 \cdot d^3)$.

• **Cluster Expansion:**

The time complexity of computing the initial co-sharing level matrix is $O(n * l_{top}^2)$, as explained in Section 4.2.2. As there is no index available for CURLER due to our unique co-sharing level function, all the unprocessed microclusters have to be checked to determine the co-shared $\epsilon - neighborhood$ of the current cluster. So the time complexity of the nearest neighbor search for one cluster is $O(k_0)$ and the time complexity of the total nearest neighbor search is $O(k_0^2)$. Also, as the time complexity of each co-sharing level matrix update during cluster merging is $O(k_0)$, and there is maximum k_0 updates, the time complexity of the entire correlation distance matrix update is $O(k_0^2)$. As a result, the time complexity of cluster expansion is $O(n \cdot l_{top}^2 + k_0^2)$.

4.3 Experimental Studies

We tested CURLER on a 1600 MHz PVI PC with 256M memory to ascertain its effectiveness and efficiency. We evaluated CURLER on a 9D synthetic dataset of three helix clusters with different cluster existence spaces, the iris plant dataset and the image segmentation dataset from the UCI Repository of Machine Learning Databases and Domain Theories [13], and the Iyer time series gene expression data with ten well-known linear clusters [42].

4.3.1 Parameter Setting

As illustrated in Figure 4.7, CURLER generally requires five input parameters: $MaxLoopNum$, log likelihood threshold $\epsilon_{likelihood}$, microcluster number k_0, l_{top} and neighborhood co-sharing level threshold ϵ.

In all our experiments, we set $MaxLoopNum$ between 5 and 20, and $\epsilon_{likelihood}$ as 0.00001. The experiments show that it is quite reasonable to trade off a limited amount of accuracy for efficiency by choosing a smaller $MaxLoopNum$, a larger log likelihood threshold $\epsilon_{likelihood}$ and a smaller l_{top} ranging from 20 to 40.

The number of microclusters k_0 is a core parameter of CURLER. According to our experiments, there is no significant difference in performance when varying

k_0. Of course, the larger value of k_0, the more refined NNCO plots we got. Unlike [2] where each data object is assigned to only one cluster, in CURLER, each data object is assumed to have membership probabilities for l_{top} microclusters. As a result, the performance of CURLER is not affected much by k_0.

The neighborhood co-sharing level threshold ϵ implicitly defines the quality of merged clusters. The larger ϵ indicates more strict requirement on microclusters' similarity in both orientation and spacial distance when expanding clusters; hence, the higher cluster quality we obtained. In our experiments, we set ϵ to 0. To get a rough clustering result for any positive ϵ, we simply moved the horizontal axis up along the vertical axis by a co-sharing level of ϵ in the NNCO plot. This is another advantage of our algorithm.

4.3.2 Efficiency

In this Section, we evaluate the efficiency of our algorithm with a varying database size (n) and a varying number of microclusters (k_0) on the 9-dimensional (d=9) synthetic dataset. In our experiments, we fixed the maximum number of loop time $MaxLoopNum$ at 10, the log likelihood threshold $\epsilon_{likelihood}$ at 0.00001, the neighborhood co-sharing level threshold ϵ as 0, and the number of microcluster memberships for each data object l_{top} at 300. We varied either n or k_0. When n was varied, we fixed k_0 to 300. Likewise, we set n as 3000 when varying k_0. For the output results, we averaged the execution times of five runs under the same parameter setting. In general, CURLER performed approximately linearly with the database size and the number of microclusters, as illustrated in Figure 4.8. The high scalability of our algorithm shows much promise in clustering high-dimensional data.

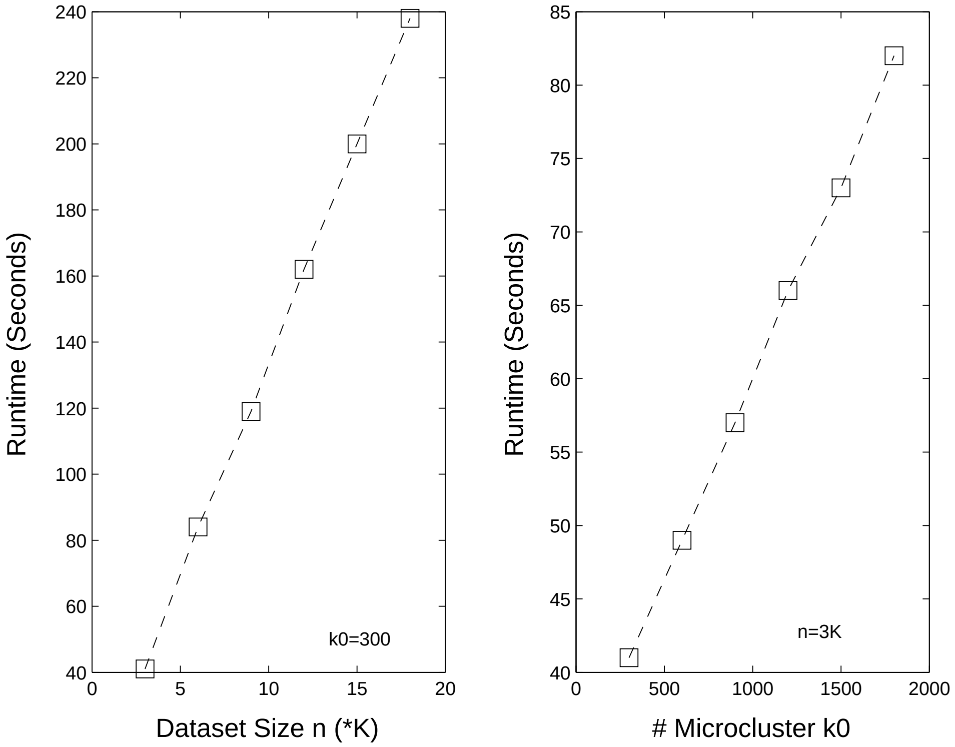


Figure 4.8: Runtime vs Dataset Size n and # Microclusters k_0 on the 9D Synthetic Dataset

4.3.3 Effectiveness

Synthetic Dataset

Because of the difficulty of getting a public high-dimensional dataset of well-known nonlinear cluster structures, we compared the effectiveness of CURLER with 4C on a 9D synthetic dataset of three helix clusters. The three helix clusters existed in

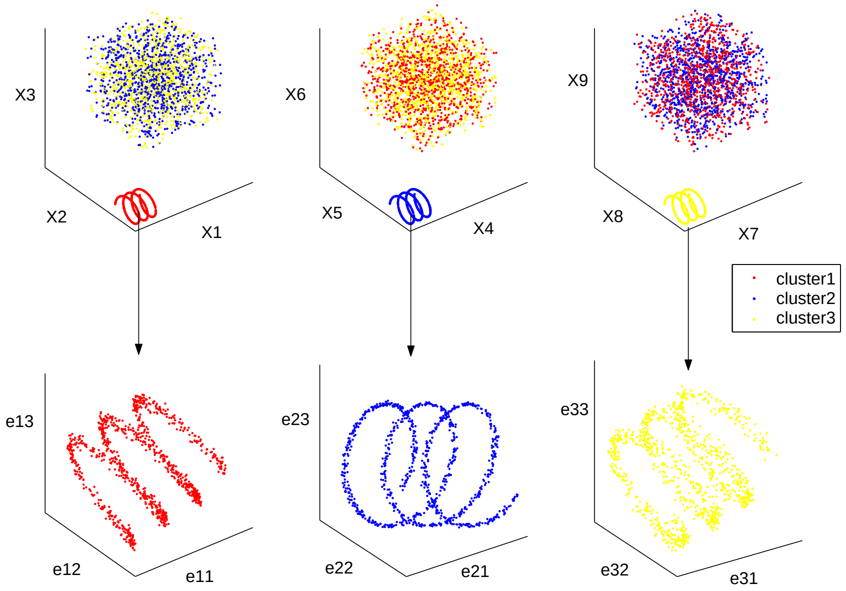


Figure 4.9: Projected Views of Synthetic Data in both Original Space and Transformed Clustering Spaces

dimensions $1-3$ (cluster 1), $4-6$ (cluster 2), and $7-9$ (cluster 3) respectively and the remaining six dimensions of each cluster were occupied with large random noise, approximately five times the data. Each cluster mapped a different color: red for cluster 1, blue for cluster 2, and yellow for cluster 3, as shown in Figure 4.9. Below is the basic generation function of helix, where $t \in [0,\ 6\pi]$,

$$x_1 = c * t,$$

$$x_2 = r * sin(t),$$

$$x_3 = r * cos(t).$$

The top-level NNC plot in Figure 4.10 shows that all the three clusters were identified by CURLER in the sequence of cluster 1, cluster 3 and cluster 2, separated

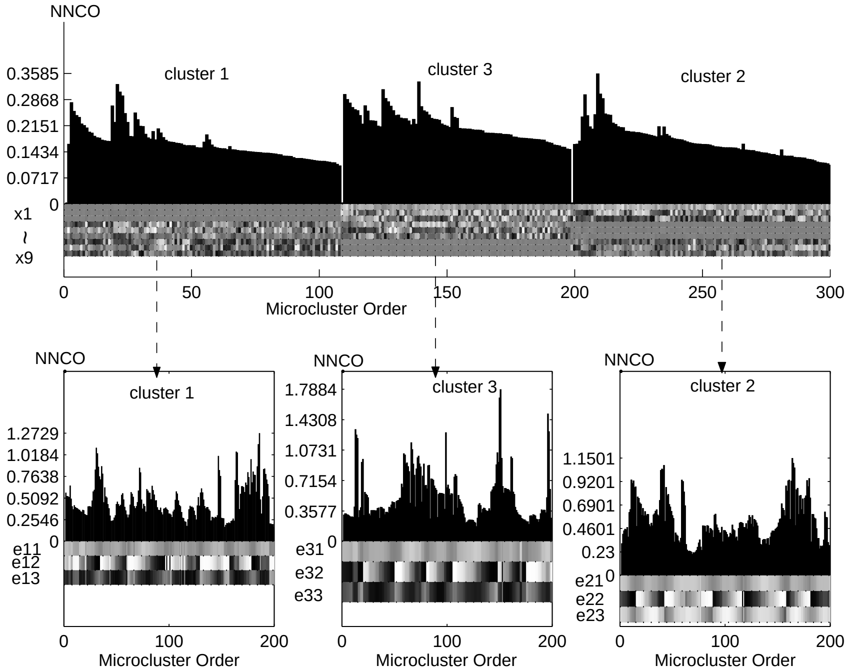


Figure 4.10: Top-level and Sub-level NNCO Plots of Synthetic Data

by two NNC-zero-gaps. The top-level orientation plot further indicates the cluster existence subspace of each cluster, the gray dimensions. The noise dimensions are marked with irregular dazzling darkening and brightening patterns.

For a close look at the nonlinear correlation patterns, we projected the data member of each cluster into its cluster existence subspace of three vectors and performed sub-level clustering. Note that the vectors of cluster existence subspace were NOT subsets of the original vectors. Since $|sin(t)|$ and $|cos(t)|$ had six cycles, when t varied from 0 to 6π, the sub-level NNCO plots show six cycles of shading and brightening orientation patterns in subspace dimensions e_{i1}, e_{i2}, and e_{i3} for each cluster i ($i = 1$, 2, and 3).

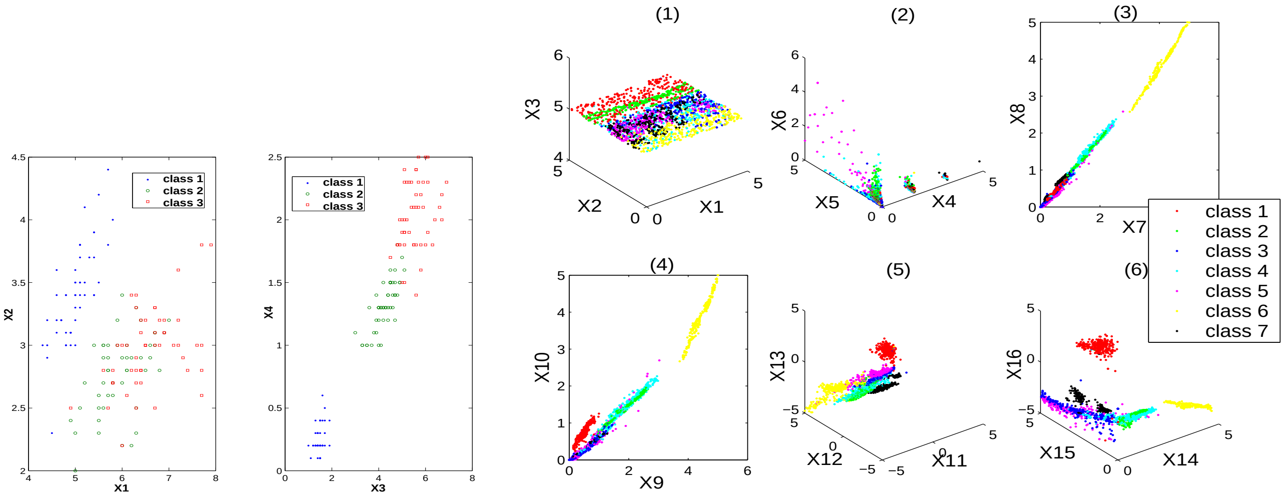
As expected, 4C found no clusters although we set the correlation threshold parameter δ as high as 0.8. The changing orientation in the dataset does not exhibit the linear correlation which 4C is looking for. In contrast, CURLER not only detected the three clusters but also captured their cycling correlation patterns and the subset of correlated features (Figure 4.10).

Real Case Studies

To have a rough idea of the potential of CURLER in practical applications, we applied the algorithm to three real-life datasets in various domains. Our experiments on the iris plant dataset, the image segmentation dataset, and the Iyer time series gene expression dataset show that CURLER is effective for discovering both nonlinear and linear correlation clusters on all the datasets above. As the cluster structures of the first two public datasets have not been described, we will begin our discussion with the examination of their data distributions with the projected views. We will only report the top-level clustering results of CURLER here due to space constraint.

Based on our definition of the data members represented by cluster C in Section 4.2.4, we can infer the class cluster C mainly belongs to. We denote the inferred class label on the top of the cluster or subcluster in the NNCO plot.

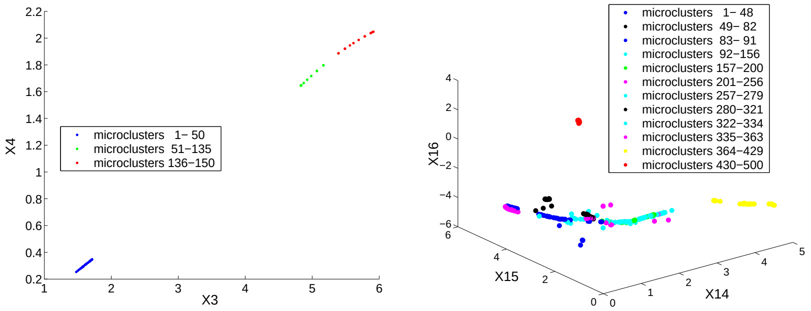
Case 1: Iris Plant The iris plant dataset is one of the most popular datasets in pattern recognition domain. It contains 150 instances from three classes: Iris-virginica (class 1), Iris-versicolor (class 2) and Iris-setosa (class 3), 50 instances each. Each



(a) Iris

(b) Image

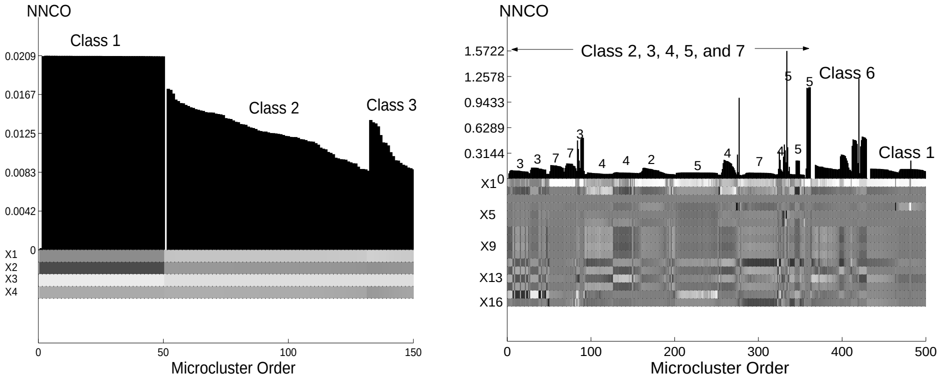
Figure 4.11: Projected Views



(a) Iris

(b) Image

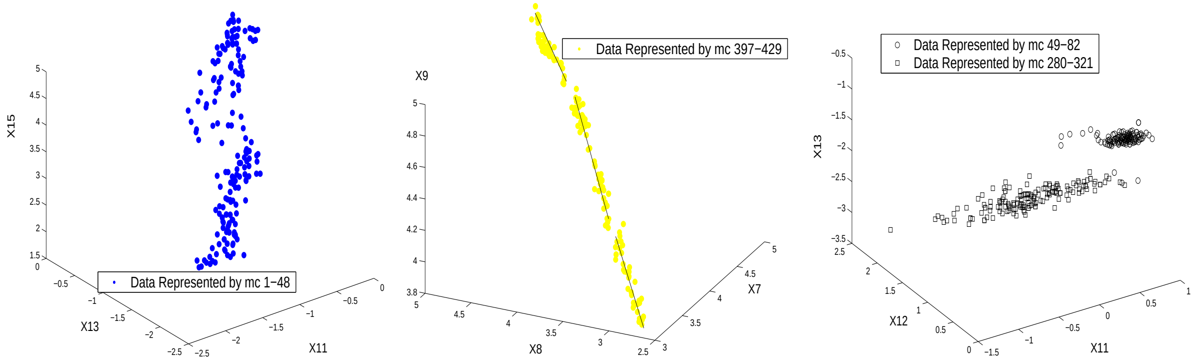
Figure 4.12: Constructed Microclusters



(a) Iris

(b) Image

Figure 4.13: NNCO Plots



(a) Microclusters 1-48

(b) Microclusters 397-429

(c) Microclusters 49-82 and 280-321

Figure 4.14: Cluster Structures Revealed by the NNCO Plots for the Image Dataset

instance has four numeric attributes, denoted as X_1, X_2, X_3 and X_4. Figure 4.11 (a) shows the projected view of this data, where the blue points, green circle and red squares represent instances from class 1, 2 and 3 respectively. We can see that there are two large clusters: one consisting of instances of class 1 and the other consisting of instances from class 2 and class 3. The second cluster can further be divided into two subclusters, one composed of instances from class 2 and the other from class 3.

The microclusters constructed by the EMCluster subroutine are shown in Figure 4.12 (a). As can be seen clearly, the cluster expansion path traverses instances from class 1, class 2 and class 3 in an orderly manner. The NNCO plot of iris (Figure 4.13 (a)) visualizes two large clusters: one composed of 50 microclusters representing instances from class 1 and the second cluster composed of 100 microclusters representing instances from the other two classes. It is also noticeable that the second cluster is further divided into two subclusters (two humps) of 85 and 15 microclusters respectively. As illustrated in Figure 4.12 (a), the two subclusters mainly represent instances from class 2 and class 3 respectively. The different patterns of the clusters in the orientation plot suggest different cluster existence subspaces. It is interesting that the microclusters in the same cluster or the same subcluster are very similar in orientation (very similar color patterns). Thus we can infer that the iris plant dataset has three approximately linear clusters, among which two with very similar orientations are close to each other.

Case 2: Image Segmentation The image segmentation dataset has 2310 instances from seven outdoor images: grass (class 1), path (class 2), window (class 3), cement (class 4), foliage (class 5), sky (class 6), and brickface (class 7). Each instance corresponds to a 3x3 region with 19 attributes. During dataset processing, we removed the three redundant attributes (attributes 5, 7, and 9 were reported to be repetitive with attributes 4, 6, and 8 respectively), and normalized the remaining 16 attributes to the range of [-5, 5]. The 16 attributes contained some statistical measures of the images, denoted as X_1, X_2, ..., X_{16}.

Figure 4.11 (b) shows the projected views on all dimensions. Figure 4.12 (b) is the projected view of our constructed microclusters on dimensions X_{14}, X_{15} and X_{16} in cluster expansion order.

Figure 4.13 (b) is the NNCO plot of the image dataset, which reveals the clustering structure accurately. Note that the image dataset is partitioned into three large clusters separated by NNC-zero-gaps. This is confirmed in our data projection views, Figure 4.11 (b.4) and (b.6), where we can see one large cluster composed of instances from class 1, one composed of instances from class 6, and another large cluster composed of mixed instances from the rest of the classes. The last cluster is nonlinear (Figures 4.11 (b.5) and (b.6)). The NNCO plot indicates that instances from the seven classes are well separated and fairly clustered.

The orientation plot further indicates that the clusters have their own subspaces; this is reflected in the different color patterns. However, some common subspaces also exist. For instance, we observe that the orientation plot on dimensions X_7, X_8, X_9, and X_{10} has synchronous color patterns, indicating synchronous linear correlations of the four attributes. As validated in Figures 4.11 (b.3) and (b.4), the three clusters approximately reside in the diagonal regions of dimensions X_7, X_8, X_9 and X_{10}. Another interesting phenomenon is that line X_1 is strongly highlighted (indicating large variation in X_1), line X_2 is partly highlighted (indicating positive orientation) and partly darkened (indicating negative orientation) while line X_3 is globally gray (indicating no variation at all in dimension X_3). With a closer look at Figure 4.11 (b.1), we see the answer: the three clusters distribute almost parallel with axis X_1 and have little variation in dimension X_3. The approximate gray of lines X_4, X_5, and X_6 also indicates little variation in the three dimensions. As a result of the nonlinear patterns in dimensions X_{11} to X_{16} (Figure 4.11 (b)), there are irregular color patterns in dimensions X_{11} to X_{16}.

Figure 4.14 depicts three interesting cluster structures discovered in the NNCO plot of the image dataset (Figure 4.13 (b)). First, the black-and-white cycling color pattern of microclusters 1-48 in dimensions X_{11}-X_{15} of the orientation plot is a vivid visualization of the nonlinear cluster structure of the corresponding instances of class 3 (Figure 4.14 (a)). Second, the synchronous three-vertical-bar pattern of microcluster 397-429 in both the NNC plot and the orientation plot, especially dimensions X_7-X_{10}, reveals three linear correlation clusters with diagonal orientations (Figure 4.14 (b)). The NNCO plot also indicates that the instances of class 7 can be partitioned into two big subclusters of consecutive microclusters, one represented by microclusters 49-82 and the other represented by microclusters 280-321 respectively. The plot also indicates that the later subcluster has a larger variation in dimensions X_{11}, X_{12}, and X_{13} (microclusters 280-321 have brighter colors in dimensions X_{11} and X_{12} of the orientation plot than microclusters 49-82). Again, this is verified in Figure 4.14 (c).

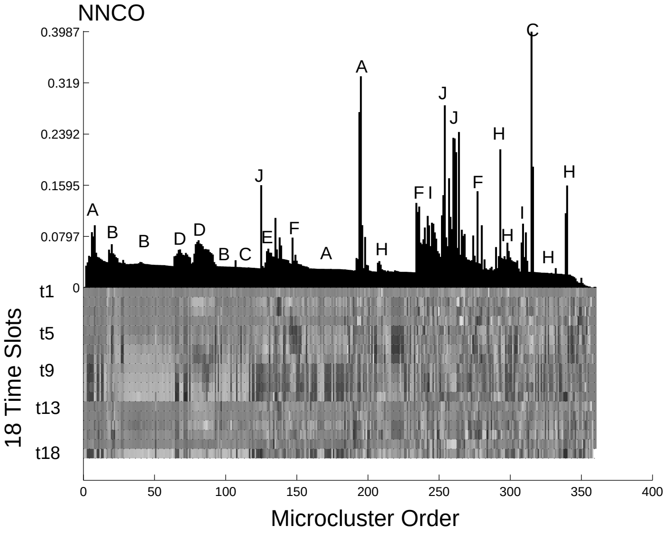


Figure 4.15: NNCO Plot of Iyer

Case 3: Human Serum Data To verify the effectiveness of our algorithm, we also applied CURLER to a benchmark time series gene expression dataset in response of human fibroblasts to serum, the Iyer dataset [42]. The Iyer dataset consists of gene expression patterns of 517 genes across 18 time slots. [42] describes 10 linear correlation clusters of genes, denoted as ‘A’, ‘B’, ..., and ‘J’. CURLER identified nine out of the reported ten clusters successfully among the 517 genes (Figure 4.15); cluster ‘G’, consisting of 13 genes, was the exception. As can be seen, CURLER partitions the reported genes of cluster ‘D’ into two consecutive subclusters, represented by microclusters 63-76 and 77-95 respectively. Likewise, CURLER partitions the genes of cluster ‘H’ into three disjointed big subclusters

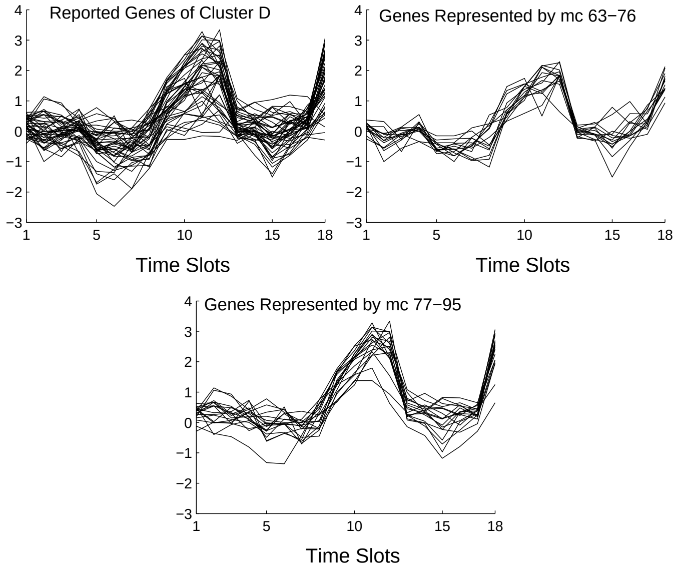


Figure 4.16: Discovered Subclusters for Cluster "D"

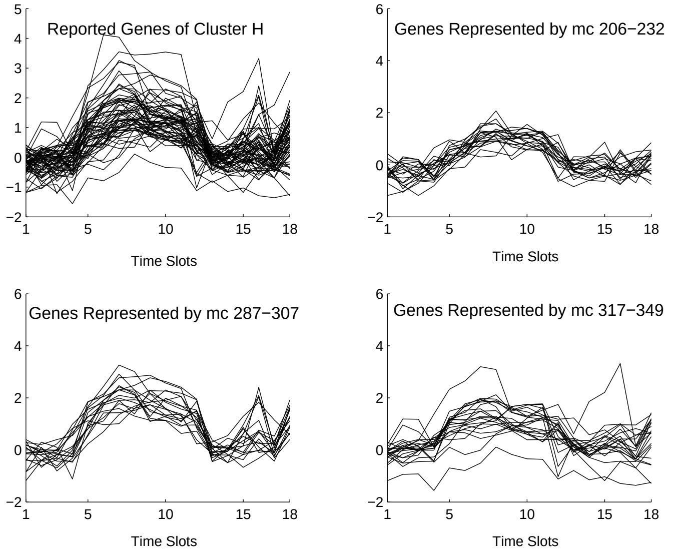


Figure 4.17: Discovered Subclusters for Cluster "H"

of consecutive microclusters: 206-232, 287-307 and 317-349. The latter two big subclusters can be further partitioned at the sub-level as observed in the NNCO plot.

Figures 4.16 and 4.17 illustrate the temporal gene expression patterns across the 18 time slots in the above discovered subclusters. Apparently, the expression patterns in each subcluster are quite cohesive. Note that the expression patterns of genes in the two subclusters of cluster 'D' are different at time slots $t2$ and $t3$: those represented by microclusters 63-76 are negatively expressed while those represented by microclusters 77-95 are positively expressed. Besides, their variation at the two time slots are different, as detected by the NNCO plot. As for genes of the three subclusters of cluster 'H', their expression patterns are delicately different in time slots t9, t10, t11, and t12, as shown in Figure 4.15 and verified in Figure 4.17.

4.4 Summary

In this chapter, we have presented a novel clustering algorithm for identifying and visualizing nonlinear correlation clusters together with the specific subspaces of their existence in high-dimensional space. Almost no work has addressed the issue of nonlinear correlation clusters, let alone the visualization of these clusters. Our work is a first attempt, and it combines the advantage of density-based algorithms represented by OPTICS [7] for arbitrary cluster shape and the advantage of subspace clustering algorithms represented by ORCLUS [2] for subspace detecting.

As shown in our experiments on a wide range of datasets, CURLER successfully captures the subspaces where the clusters exist and the nonlinear cluster structures, even when a large number of noise dimensions are introduced. Moreover, CURLER allows users to interactively select the cluster of their interest, have a close look at its data members in the space where the cluster exists, and perform sub-level clustering when necessary.

We plan to consider other variants to further improve the efficiency of CURLER, i.e., constructing some index structures to accelerate nearest neighbor queries based on the mixture model.

CHAPTER 5

Reg-Cluster

Table 5.1 shows a mini example dataset that we are studying in this chapter. Each row of the table corresponds to a gene (denoted as g_i) while each column corresponds to a certain condition (denoted as c_j) under which gene expression is measured. For example, biologists might in one experiment artificially suppress the expression of a certain gene and look at how other genes are affected under such a condition. A subset of genes showing correlated co-expression patterns across a subset of conditions are expected to be functionally related and involved in the same cellular pathway [40]. By grouping together genes that exhibit similar behaviors, biologists hope to discover new functional groups and ultimately gain more insight into the genetic behavior of life.

One well-known characteristic of high-dimensional data is that data objects (genes) are not correlated in full dimensional space but correlated only in a subset of dimensions (subspace). To handle this problem, density-based *subspace clustering* algorithms [2–4, 14, 39, 41, 68] assume data objects of the same cluster are close with each other in cluster existance subspace. These algorithms also assign each data object to only one cluster. Yet in high-dimensional gene expression data, the situation is much more complex. A gene or a condition may be involved in multiple pathways. To allow overlap between gene clusters, pioneering biclustering algorithms such as [19] have been proposed; these algorithms allow one gene to be assigned to several clusters.

A later advancement, pattern-based biclustering algorithms [80, 83, 85] take into consideration the fact that genes with strong correlation do not have to be spatially close in correlated subspace. More recently, tendency-based biclustering algorithms such as OP-Cluster [58] and TP-Cluster [59] adopt sequence and tendency models respectively for efficient discovery of genes whose expression levels rise and fall synchronously in a subspace. However, such tendency-based biclustering algorithms have no guarantee level of coherency.

In this chapter, we focus on the more general shifting-and-scaling co-regulation patterns with coherence constraint, which have received little attention so far.

5.1 Background

Gene expression clustering algorithms may be classified into two big categories: *full space clustering* algorithms which evaluate the expression profile similarity of genes in all conditions, and *subspace clustering* algorithms which evaluate similarity in a subset of conditions.

The most commonly applied full space clustering algorithms on gene expression profiles are hierarchical clustering algorithms [27], self-organizing maps [75], and K-means clustering algorithms [76]. Hierarchical algorithms merge genes with the most similar expression profiles iteratively in a bottom-up manner. Self-organizing maps and K-means algorithms partition genes into user-specified k optimal clusters. Other full space clustering algorithms applied on gene expression data include Bayesian network [31] and neural network.

Density-based subspace clustering algorithms, [2–4, 14, 39, 41, 68] and our CURLER algorithm too, would assign each data object (gene) to just one cluster. Biclustering algorithms such as [19] provide an answer to this problem which allow overlapping clusters. These algorithms require genes of the same cluster to be dense and close to each other in correlated subspace.

The more recent pattern-based and tendency-based biclustering algorithms, [12, 58, 59, 80, 83, 85] overcome the conventional constraint of spatial proximity and are able to identify pure shifting patterns, pure scaling patterns and synchronous-tendency patterns.

Non existing pattern-based algorithms are able to discover the more complicated shifting-and-scaling patterns. Another unaddressed issue of previous work is negative correlation, which is still confined to full space clustering at present [24, 44, 69].

5.1.1 Motivation

Existing pattern-based biclustering algorithms are only able to address shifting patterns and scaling patterns separately: as shown in Figure 5.1. After a single shifting or scaling, a pattern may coincide with another pattern. In Figure 5.1, the six patterns are of the relationships: $P1 = P2 - 5 = P3 - 15 = P4 = P5/1.5 = P6/3$. PCluster [80] and δ-cluster [83] assume that scaling patterns can be transformed to shifting patterns after a logarithm transformation on the whole dataset D, and focuses on shifting patterns only. Tricluster [85] focuses on scaling patterns only, assuming that after a global exponential transformation of D, shifting patterns will all be transformed into scaling patterns. Assume d_{ic} and d_{jc} are expression levels of gene g_i and g_j on condition c, s_1 and s_2 are the scaling and shifting factors respectively; their mathematical relationships are given as follows:

$$d_{ic} = s_1 * d_{jc} \Rightarrow logd_{ic} = logd_{jc} + logs_1 \text{ [80, 83]} \tag{5.1}$$

$$d_{ic} = d_{jc} + s_2 \Rightarrow e^{d_{ic}} = e^{d_{jc}} \cdot e^{s_2} \text{ [85].} \tag{5.2}$$

No existing pattern-based algorithms can handle dataset with shifting-and-scaling patterns of the form $d_{ic} = s_1 * d_{jc} + s_2$, by which the six cohesive patterns in Figure 5.1 can be grouped together with ease.

gene	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
g_1	10	-14.5	15	10.5	0	14.5	-15	0	-5	-5
g_2	20	15	15	43.5	30	44	45	43	35	20
g_3	6	-3.8	8	6.2	2	7.8	-4	2	0	0

Table 5.1: Running Dataset

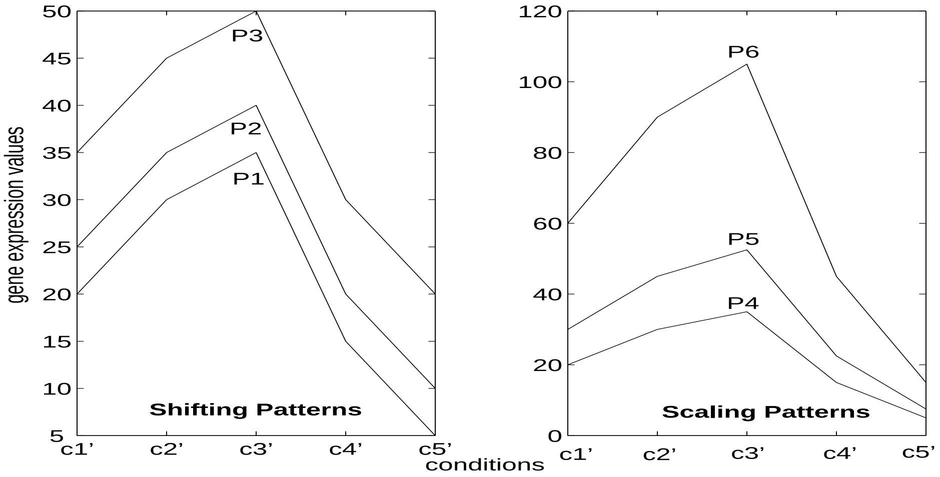


Figure 5.1: Previous Patterns

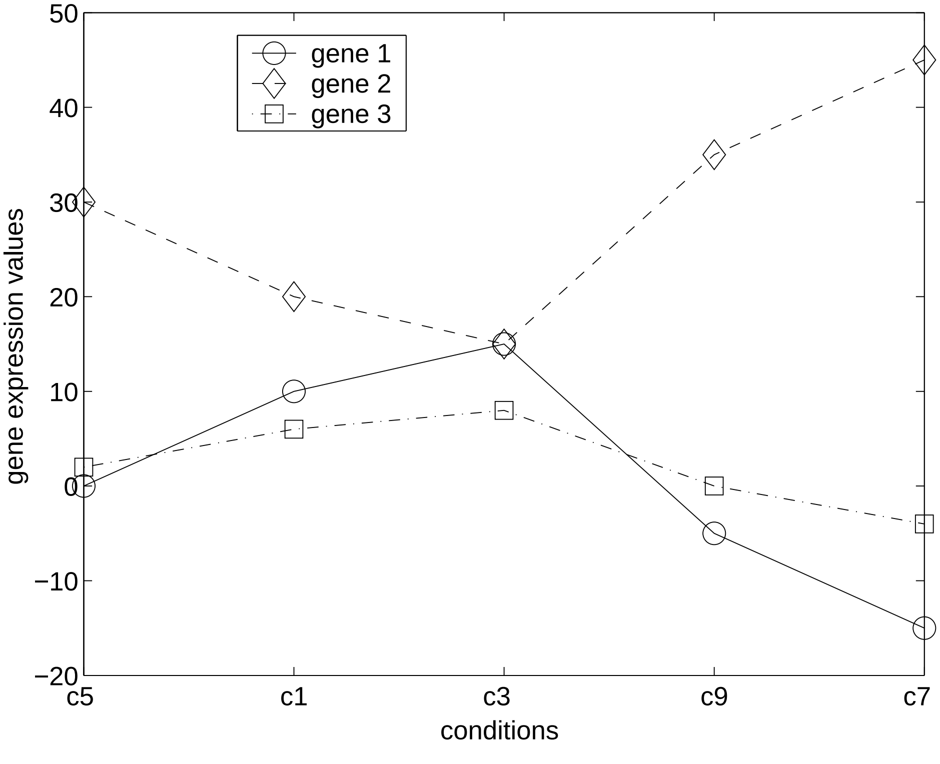


Figure 5.2: Our Shifting-and-Scaling Patterns

There are three problems the pattern-based algorithms and other existing biclustering algorithms have ignored:

• **Regulation Test**: Cheng and Church [19] state that the utmost important goal of gene expression data analysis is to find a set of genes showing strikingly similar *up-regulation* and *down-regulation* under a set of conditions, rather than simply to find a bicluster to perfectly cover the data. The pattern-based and tendency-based algorithms blindly assume any positive increase in expression levels as valid up-regulation and any positive decrease from one condition to the other as valid down-regulation. In fact, those patterns with smaller variations in expression values are probably of little biological meaning. One extreme case is a horizontal-line bicluster in which each gene has a fixed expression level value in the subset of conditions. Methods in [80, 85] view it as a perfect bicluster (pScore =0 and ratio range = 0), although no regulation occurs.

• **Pattern Universality**: Co-regulated genes may respond to environmental stimuli or conditions coherently, forming certain shifting-and-scaling patterns due to varying individual sensitivities. For instance, expression profiles of g_1 and g_3 of the running example (Table 5.1) in Figure 5.2 are shifting-and-scaling patterns: $d_{1,\{5,1,3,9,7\}} = 2.5 * d_{3,\{5,1,3,9,7\}} - 5$. Current pattern-based models [80, 83, 85] only validate a partial correlation, either a pure shifting pattern or a pure scaling pattern, which are just two special cases of the shifting-and-scaling pattern. Also, they may fail to detect biclusters composed of a mixture of shifting patterns and scaling patterns, such as the ones in Figure 5.1. Note that the assumptions of existing pattern-based biclustering algorithms [80, 83, 85] of either shifting-to-scaling transformation (Equation 5.2) or scaling-to-shifting transformation (Equation 5.1) do not hold for the more general shifting-and-scaling patterns. Therefore, many co-regulation patterns would be missed by existing pattern-based algorithms.

- **Negative Correlation**: The complex biological system exhibits an even greater diversity in gene correlations than any existing subspace clustering and biclustering algorithms can capture. One is negative-correlation, i.e., when one gene has a high expression level, the expression level of the other gene is low and vice versa. Both positive-correlated genes and negative-correlated genes should be grouped together because genes that are functionally related may demonstrate strong anti-correlation in their expression levels, i.e., a gene may be strongly suppressed to allow another to be expressed [74], and both positive-correlated genes and negative-correlated genes could be involved in the same biological pathway [24]. It is therefore desirable to group together genes whose expression profiles are either positively correlated or negatively correlated on a subset of conditions. Although a wealth of work in subspace clustering and biclustering has been done on expression data, none has addressed the issue of negative correlation in a systematic way. The existing work on clustering negative-correlated genes is still confined to full dimensional space [24, 44, 69]. Moreover, from a broader view, negative correlation in subspace also pertains to the shifting-and-scaling pattern with a negative scaling factor, such as the relationship between g_2 and the other two genes in Figure 5.2, $d_{2,\{5,1,3,9,7\}} = -2.5 * d_{3,\{5,1,3,9,7\}} + 35 = -d_{1,\{5,1,3,9,7\}} + 30$.

5.1.2 Goal

To address the various problems that we have just discussed, we propose a new model called **reg-cluster**. The proposed model can better accommodate the regulation constraint and various correlation measures on gene expression profiles employed previously, including both positive and negative co-regulations. The proposed model also allows for shifting-and-scaling co-regulation as well as pure shifting and scaling one. Table 5.1 illustrates the expression levels of three genes under 10 conditions. As Figure 5.2 shows, g_1 and g_3 are strongly positively co-regulated, but g_2 is strongly negatively co-regulated with g_1 and g_3 on conditions c_5, c_1, c_3, c_9 and c_7. The three genes form a candidate 3×5 reg-cluster before the regulation constraint is applied. A reg-cluster exhibits the following characteristics which are suitable for expression data analysis:

- increase or decrease of gene expression levels across any two conditions of a reg-cluster is significant with regard to the regulation threshold γ.
- increase or decrease of gene expression levels across any two conditions of a reg-cluster is in proportion, allowing small variations defined by the coherence threshold ϵ.
- genes of a reg-cluster can be either positively correlated or negatively correlated.

5.1.3 Challenges

In correlated subspace, positive-correlated genes and negative-correlated genes exhibit no spatial proximity at all. This makes it impractical to apply density-based subspace clustering algorithms [2–4, 14, 39, 41, 68] and the mean-squared-residue-score based biclustering algorithm [19].

For pattern-based and tendency-based biclustering algorithms, there are three main challenges for reg-cluster discovery.

Naturally, the biggest challenge is the need of a novel coherent cluster model that can capture the more general shifting-and-scaling co-regulation patterns. For instance, the shifting-and-scaling patterns in Figure 5.2 are coherent, but they satisfy neither the pattern coherence measure of pScore [80] nor that of the valid expression level ratio range [85].

Another challenge is how to apply a non-negative regulation threshold. Tendency-based models of [12, 58, 59] are not suitable for adopting a regulation threshold γ. For example, [58] adopts a sequence model to translate the expression profiles of each gene g_i into a sequence by first sorting the conditions in non-descending order and later grouping the conditions whose expression values are equivalent according to γ. Assuming the user-specified regulation threshold for g_2 is 0.8, we are unable to find an appropriate sequence model for g_2 in the running example on conditions c_2, c_{10}, c_8, c_4 and c_6 with expression levels $\{15, 20, 43, 43.5, 44\}$ such that both non-regulated condition-pairs $c_8 - c_4$ and $c_4 - c_6$ are grouped together but the regulated condition-pair $c_6 - c_8$ is not.

The third challenge is negative co-regulation. Note that our scaling coefficient can be a negative real number. One approach to handle negative correlation is the PearsonR model [23]. A large positive PearsonR value indicates a strong positive correlation while a large negative one indicates a strong negative correlation. However, without knowing the subset of correlated conditions in advance, we are unable to apply the PearsonR approach appropriately. Nor can existing pattern-based biclustering algorithms efficiently handle the negative co-regulation problem. Coexistence of positively and negatively correlated genes would lead to a rather large pScore [80] or expression ratio range [85].

5.2 Reg-Cluster Model

5.2.1 Regulation Measurement

Suppose d_{ic_a} and d_{ic_b} are the expression levels of gene g_i under conditions c_a and c_b respectively. We could then say g_i is **up-regulated** from condition c_b to condition c_a, denoted as $\mathcal{R}eg(i, c_a, c_b) = Up$, if the increase in expression level exceeds its regulation threshold γ_i, as described in Equation 5.3. Alternatively, we say g_i is **down-regulated** from condition c_a to c_b, denoted as $\mathcal{R}eg(i, c_b, c_a) = Down$. In this case, we call c_b the **regulation predecessor** of c_a, denoted as $c_b \curvearrowleft c_a$, and c_a as the **regulation successor** of c_b for g_i, denoted as $c_a \curvearrowright c_b$ (the arrow always points from bigger value to smaller value). Otherwise there is no regulation between c_a and c_b for g_i.

$$\mathcal{R}eg(i, c_a, c_b) = \begin{cases} Up & \text{if } d_{ic_a} - d_{ic_b} > \gamma_i \\ Down & \text{if } d_{ic_a} - d_{ic_b} < \gamma_i \end{cases} \tag{5.3}$$

In this chapter, for ease of understanding, we assume the regulation threshold of g_i, γ_i, as a pre-defined percentage of the expression range of g_i in Equation 5.4, where n is the dimensionality of the expression dataset and γ is a user-defined parameter ranging from 0 to 1.0. We consider imposing a regulation threshold important for pattern validation, as it will help to distinguish useful patterns from noise. In practice, other regulation thresholds, such as the average difference between every pair of conditions whose values are closest [58], normalized threshold [44], average expression value [18], etc., can be used where appropriate.

$$\gamma_i = \gamma \times (MAX_{1 \leq j \leq n}(d_{ic_j}) - MIN_{1 \leq j \leq n}(d_{ic_j})), \tag{5.4}$$

The intuition behind using a local regulation threshold for different genes instead of a global one is that individual genes have different sensitivities to environmental stimulations. For instance, studies in [21] reveal that the magnitudes of the rise or fall in the expression levels of a group of genes inducible or repressible by hormone E2 can differ by several orders of magnitude.

Current pattern-based and tendency-based models [12, 58, 59] can only cope with the extreme and probably biased case where $\gamma = 0$, and is constrained to the positive correlation. If $\gamma > 0$, these models become problematic.

To support this general concept of regulation, a naive approach is to record the regulation relationships between all possible pairs of C_n^2 conditions. Instead, we propose a new model, called $RWave^{\gamma}$ [1], which only keeps the regulation information of *bordering condition-pairs* for the genes in a wave-boosting manner with respect to γ. Figure 5.3 illustrates the $RWave^{0.15}$ model ($\gamma_1 = \gamma_2 = 4.5$ and $\gamma_3 = 1.8$) for the running example (Table 5.1). $c_5 - c_1$ is one bordering condition-pair for g_1 since it represents the smallest interval above $\gamma_1 = 4.5$. Consequently, any condition c_i that lies on the left hand side of c_5 will guarantee to have a bigger difference than γ_1 when compared to any condition c_j that lies on the right hand side of c_1. As can be seen, there is no need to keep the regulation information of non-bordering pairs. The formal definition of the $RWave^{\gamma}$ model is given below.

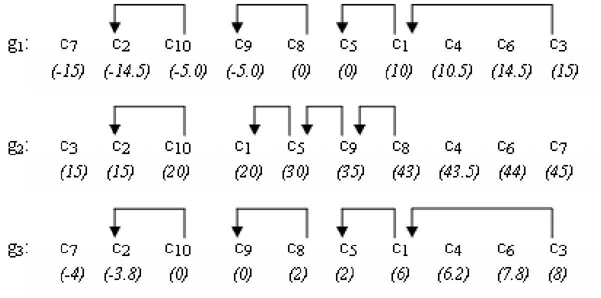


Figure 5.3: $RWave^{0.15}$ Models

Definition 5.2.1 $RWave^{\gamma}$

Given the regulation threshold γ, the $RWave^{\gamma}$ model of gene g_i on the set of conditions c_1, c_2, ..., and c_n is a non-descending ordering ($\preceq$) of the set according to their expression values with regulation pointers marking all the bordering regulation relationships such that for each regulation pointer pointing from c_b to c_a, we have, (1) $\forall c_p \succeq c_b$ and $\forall c_q \preceq c_a$, $\mathcal{R}eg(i, c_p, c_q) = Up$, denoted as $c_q \curvearrowleft c_p$; and (2) there is

[1] $RWave$ stands for regulation wave

no other embedded pointer pointing from $c_{b'}$ to $c_{a'}$, such that $c_{a'} \succeq c_a$ and $c_{b'} \preceq c_b$, $\forall c_{p'} \succeq c_{b'}$ and $\forall c_{q'} \preceq c_{a'}$, we have $\mathcal{R}eg(i, c'_p, c'_q) = Up$, denoted as $c_{q'} \curvearrowleft c_{p'}$. □

Note that if $c_q \preceq c_p$ in g_i's $RWave^{\gamma}$ model, indicating $d_{iq} \leq d_{ip}$, then c_q may not be c_p's regulation predecessor. Here, $\preceq$ and $\succeq$ indicate the ordering of the conditions while $\curvearrowleft$ and $\curvearrowright$ indicate the upward and downward regulation relationships of a condition-pair with respect to γ. Given the regulation threshold γ, the regulation relationship of any condition-pair of g_i can be easily inferred from its $RWave^{\gamma}$ model by simply checking whether there is a regulation pointer between the two conditions and what the pointer direction is. The conditions of a reg-cluster whose pairwise differences in expression levels are either upward or downward defined by γ must be separated by at least ONE regulation pointer in the $RWave^{\gamma}$ model of its genes, thus forming a “$\curvearrowleft$” or “$\curvearrowright$” linked regulation chain.

Besides, Lemma 5.2.1 ensures that $\forall c_k$ of a gene g_i, we can locate all the regulation predecessors and regulation successors of c_k for g_i efficiently by using the $RWave^{\gamma}$ model.

Lemma 5.2.1 Given the regulation threshold γ, a gene g_i and a condition c_a, let $c_p \curvearrowleft c_q$ be the nearest regulation pointer that is **before** c_a with respect to g_i. All conditions c_b such that $c_b \preceq c_p$ are all regulation predecessors of c_a with respect to g_i. Likewise, if $c_p \curvearrowleft c_q$ is the nearest regulation pointer that is **after** c_a, then all conditions c_b such that $c_q \preceq c_b$ are definitely the regulation successors of c_a with respect to g_i.

Proof: Since the conditions are sorted in non-descending order of their expression levels, $c_b \preceq c_p \prec c_q \preceq c_a$ if $c_p \curvearrowleft c_q$ represents the nearest regulation pointer before c_a. Since the difference between the expression levels of c_p and c_q is greater than γ based on the definition of regulation pointer, we can also see that the difference between the expression levels of c_b and c_a is greater than γ. Thus c_b is considered to be the regulation predecessor of c_a. For the case in which $c_p \curvearrowleft c_q$ is the nearest regulation pointer after c_a, the same argument applies. □

Given the $RWave^{0.15}$ models in Figure 5.3, assume we want to find the regulation predecessors of c_6 for g_1, we simply follow the closest regulation pointer before it, which points from c_1 to c_5. c_7, c_2, c_{10}, c_9, c_8 and c_5 are exactly the regulation predecessors of c_6. We can also infer that there are no regulation successors of c_6 as no regulation pointer exists after c_6. Interested readers may refer to Table 5.1 for a more detailed analysis.

5.2.2 Coherence Measurement

Besides the regulation threshold γ, reg-cluster should be validated with the shifting-and-scaling coherency constraint ϵ. Assume d_{iY} and d_{jY} are two perfect shifting-and-scaling co-regulation patterns of g_i and g_j on condition set Y, then we there should exists s_1 and s_2 such that,

$$d_{iY} = s_1 * d_{jY} + s_2, \tag{5.5}$$

where s_1 and s_2 are the scaling and shifting factors respectively. The value of s_1 can be either positive ($s_1 > 0$), indicating d_{iY} and d_{jY} are **positively correlated** on Y, or negative ($s_1 < 0$), indicating d_{iY} and d_{jY} are **negatively correlated** on Y. Note that any subsequent shifting or scaling transformations on d_{iY} will not affect the general form given in Equation 5.5. Formally speaking, if $d'_{iY} = d_{iY} * s_3$ (further scaling), then $s'_1 = s_1 * s_3$ and $s'_2 = s_2 * s_3$; if $d'_{iY} = d_{iY} + s_4$ (further shifting), then $s'_1 = s_1$ and $s'_2 = s_2 + s_4$. Likewise, subsequent shifting and scaling on d_{jY} will not change the general form. Only the scaling and shifting factors may change values. As we can observe, the shifting patterns and scaling patterns addressed in [80, 83, 85] correspond to the two special cases of $d_{iY} = d_{jY} + s_2$ and $d_{iY} = s_1 * d_{jY}$ respectively.

Based on Equation 5.5, we can further infer the necessary and sufficient condition for the existence of shifting-and-scaling pattern, where the scaling factor s_1 can be either positive or negative, as proposed in Lemma 5.2.2.

Lemma 5.2.2 *Suppose d_{iY} and d_{jY} are the expression profiles of genes g_i and g_j on subspace Y, $Y = \{c_1, c_2, ..., c_n\}$, $d_{ic_1} < d_{ic_2} < ...... < d_{ic_n}$, and assume we choose c_1 and c_2 as the baseline condition-pair, then d_{iY} and d_{jY} are shifting-and-scaling patterns, either shifting-and-positive scaling or shifting-and-negative scaling, in subspace Y if and only if $\forall c_k$, $c_{(k+1)}$, $1 \le k < n$,*

$$\frac{d_{ic_{k+1}} - d_{ic_k}}{d_{ic_2} - d_{ic_1}} = \frac{d_{jc_{k+1}} - d_{jc_k}}{d_{jc_2} - d_{jc_1}}. \tag{5.6}$$

Proof:

(1) If d_{iY} and d_{jY} are two shifting-and-scaling patterns, then $\exists s_1$ and s_2, $d_{iY} = s_1 * d_{jY} + s_2$. Furthermore, $\forall c_{(k+1)}$ and c_k, $1 \le k < n$, we have $d_{ic_{k+1}} = s_1 * d_{jc_{k+1}} + s_2$ and $d_{ic_k} = s_1 * d_{jc_k} + s_2$, so $\frac{d_{ic_{k+1}} - d_{ic_k}}{d_{ic_2} - d_{ic_1}} = \frac{d_{jc_{k+1}} - d_{jc_k}}{d_{jc_2} - d_{jc_1}}$.

(2) On the other hand, if $\forall c_k$, $c_{(k+1)}$, $1 \leq k < n$ *such that* $\frac{d_{ic_{k+1}}-d_{ic_k}}{d_{ic_2}-d_{ic_1}} = \frac{d_{jc_{k+1}}-d_{jc_k}}{d_{jc_2}-d_{jc_1}}$, *then* $\forall c_p$, $c_q \in Y$, $p \neq q$, *we have*

$$\begin{aligned}
&\frac{d_{ic_p} - d_{ic_q}}{d_{ic_2} - d_{ic_1}} \\
=&\frac{(d_{ic_p} - d_{ic_{p-1}}) + (d_{ic_{p-1}} - d_{ic_{p-2}}) + ... + (d_{ic_{q+1}} - d_{ic_q})}{d_{ic_2} - d_{ic_1}} \\
=&\frac{(d_{jc_p} - d_{jc_{p-1}}) + (d_{jc_{p-1}} - d_{jc_{p-2}}) + ... + (d_{jc_{q+1}} - d_{jc_q})}{d_{jc_2} - d_{jc_1}} \\
=&\frac{d_{jc_p} - d_{jc_q}}{d_{jc_2} - d_{jc_1}}.
\end{aligned}$$

Therefore, $\frac{d_{ic_p}-d_{ic_q}}{d_{jc_p}-d_{jc_q}}$ *is a constant for* g_i *and* g_j, *say* s_1. *Then* $\forall c_p$, $c_q \in Y$, $p \neq q$, *we have* $d_{ic_p} = s_1 * d_{jc_p} - s_1 * d_{jc_q} + d_{ic_q}$, *suggesting* $d_{ic_p} - s_1 * d_{jc_p}$ *being a constant as well, say* s_2. *So we can conclude that* $d_{iY} = d_{jY} * s_1 + s_2$. □

Given Lemma 5.2.2, we need not check the coherence of reg-cluster on all combinations of pair-wise conditions, which was necessary in previous work. Instead, we simply check all adjacent condition-pairs c_k and c_{k+1} with regard to the baseline condition-pair, c_1 and c_2, according to a coherence threshold ϵ.

$$H(i, c_1, c_2, c_k, c_{k+1}) = \frac{d_{ic_{k+1}} - d_{ic_k}}{d_{ic_2} - d_{ic_1}}. \tag{5.7}$$

We can conclude that the expression profiles of the three genes in Figure 5.2 are shifting-and-scaling patterns on conditions c_7, c_9, c_5, c_1 and c_3 with each other because these three genes share exactly the same coherence scores: $\forall g_i \in \{g_1$, g_2, $g_3\}$, $H(i, c_7, c_9, c_7, c_9) = 1.0$, $H(i, c_7, c_9, c_9, c_5) = 0.5$, $H(i, c_7, c_9, c_5, c_1) = 1.0$

and $H(i, c_7, c_9, c_1, c_3) = 0.5$, with an order of either $c_7 \prec c_9 \prec c_5 \prec c_1 \prec c_3$ (g_1 and g_3) or $c_7 \succ c_9 \succ c_5 \succ c_1 \succ c_3$ (g_2).

We impose the coherence threshold ϵ to flexibly control the coherence of the clusters. In this way, we can ensure the variations in coherence scores, given in Equation 5.7, are within ϵ for genes in the same cluster. Perfect shifting-and-scaling patterns correspond to the case where $\epsilon = 0$.

5.2.3 Model Definition and Comparison

By combining both the regulation constraint and the shifting-and-scaling coherence constraint, we now propose the definition of a reg-cluster.

Definition 5.2.2 *Reg-Cluster*

Given the regulation threshold γ and coherence threshold ϵ, a bicluster $C_{X \times Y}$, where X is a subset of genes and $Y = \{c_1, c_2, ..., c_n\}$ is the subset of correlated conditions such that $\forall g_i \in X$, either $d_{ic_1} < d_{ic_2} < ... < d_{ic_n}$ or $d_{ic_1} > d_{ic_2} > ... > d_{ic_n}$, is a reg-cluster if and only if:

(1) $\forall g_i \in X$, based on its $RWave^\gamma$ model, we have either

$$c_1 \curvearrowleft c_2 \curvearrowleft \ldots\ldots \curvearrowleft c_n,$$

$$or \;\; c_1 \curvearrowright c_2 \curvearrowright \ldots\ldots \curvearrowright c_n,$$

and (2) $\forall g_i, g_j \in X$, $\forall k$, $1 \leq k < n$,

$$|H(i, c_1, c_2, c_k, c_{k+1}) - H(j, c_1, c_2, c_k, c_{k+1})| < \epsilon \tag{5.8}$$

□

In this way, with the reg-cluster model, we are able to identify all the significant shifting-and-scaling co-regulation patterns with regard to γ and ϵ. Two genes of a reg-cluster can be positively co-regulated if complying with the same regulation chain and negatively co-regulated if complying with inverted regulation chains.

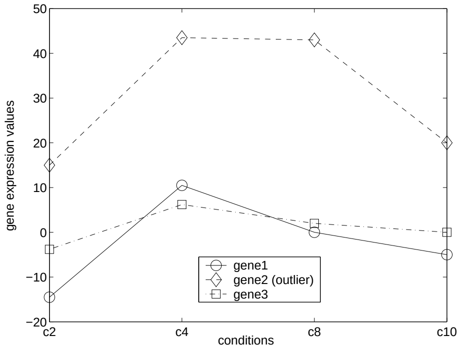


Figure 5.4: An Outlier

For a brief comparison between our reg-cluster model and previous models, we shall consider the projection of the three genes in the running example on conditions c_2, c_4, c_8 and c_{10} as shown in Figure 5.4, where $d_{3,\{2,4,8,10\}} = 0.4 * d_{1,\{2,4,8,10\}} + 2$ and there is no shifting-and-scaling relationship between g_2 and the other two genes. Given the regulation threshold $\gamma = 0.15$ and coherence threshold $\epsilon = 0.1$, our reg-cluster model can easily identify the outlier gene g_2 because (1) the $RWave^{0.15}$ model of g_2 indicates there are no regulation between c_4 and c_8; and (2) g_1 and g_3 have exactly the same coherence score along the four conditions while g_2 does not, i.e., $H(1, c_2, c_{10}, c_{10}, c_8) = H(3, c_2, c_{10}, c_{10}, c_8) = 0.5263$ but $H(2, c_2, c_{10}, c_{10}, c_8) = 4.6$, far beyond the allowed variation ϵ in coherence measure. In contrast, the pattern-based models discover no patterns, as there are no pure shifting or pure scaling relationships while the tendency-based models always cluster the three genes together because the three genes have exactly the same subsequence and tendency on the four conditions.

5.3 Algorithm

The essential idea of our algorithm is to systematically identify the representative regulation chain for each validated reg-cluster. A **representative regulation chain** $C.Y = c_{k1} \curvearrowleft c_{k2} \curvearrowleft ... \curvearrowleft c_{km}$ (a series of conditions connected by regulation pointers) represents genes that are correlated or anti-correlated with the chain. We refer to them as the **p-members** $C.pX$ (gene complying with $C.Y$) and **n-members** $C.nX$ of the reg-cluster, respectively. We can conveniently obtain $C.pX$ by searching **along** the $RWave^{\gamma}$ model and $C.nX$ by searching in the **opposite** direction. Note that there are two regulation chains that a reg-cluster may satisfy: $C.Y$ and $invert(C.Y) = \{c_{k1} \curvearrowright c_{k2} \curvearrowright ... \curvearrowright c_{km}\}$.

To avoid redundancy and overlap of the output clusters, we assume that the *representative* regulation chain always captures the pattern of the majority of genes in a reg-cluster: the number of p-members is greater than or equal to the number of n-members. If the number of p-members is equal to that of the n-members, we assume the regulation chain starting with a predecessor of larger condition ID as the "representative". For instance, the representative regulation chain for the reg-cluster in Figure 5.2 is $c_7 \curvearrowleft c_9 \curvearrowleft c_5 \curvearrowleft c_1 \curvearrowleft c_3$ with its p-members $\{g_1, g_3\}$ and n-members $\{g_2\}$. The inverted $c_7 \curvearrowright c_9 \curvearrowright c_5 \curvearrowright c_1 \curvearrowright c_3$ is not a representative

Input: $D = G \times C$: 2D dataset, $MinG$: minimum number of genes, $MinC$: minimum number of conditions, γ: regulation threshold and ϵ: coherence threshold.
Output: all validated reg-clusters w.r.t. γ, ϵ, $MinG$ and $MinC$: $\{C|C = X \times Y\}$ such that $C.X$ is the maximal gene set for the representative regulation chain $C.Y$.

```
$\backslash * RWave^{\gamma}$ model construction $* \backslash$
for each gene $g_i \in G$ do
    sort the conditions $c_j \in C$ in non-descending
    order of $d_{ij}$.
    for each $c_j$ in sorted order do
        find $c_j$'s closest regulation predecessor $c_k$ w.r.t. $\gamma$.
        if no regulation pointer exists between $c_j$ and $c_k$ then
            insert a new pointer $c_k \curvearrowleft c_j$ in $g_i$'s $RWave^{\gamma}$ model.

$\backslash *$ reg-cluster mining $* \backslash$
$C.pX = C.nX = G$.
$C.Y = \emptyset$.
$C^2Set = \emptyset$.
$MineC^2(C, C^2Set)$.
```

Subroutine: $MineC^2(C, C^2Set)$.
Parameters:

- $C.Y$: the current representative regulation chain;
- $C.X$: the corresponding genes for $C.Y$;
- C^2Set: the set of discovered validated reg-clusters.

Method:

```
1. apply pruning (1): if $|C.X| < MinG$, then return.

2. apply pruning (3).(a): if $|C.pX| < MinG/2$, then return.

3. assume $C.Y = c_{k1} \curvearrowleft c_{k2} ... \curvearrowleft c_{km}$,
   if $|C.Y| \geq MinC$ and $|C.X| \geq MinG$ and ($|C.pX| > |C.nX|$ or ($|C.pX| == |C.nX|$ and $k1 < k2$)) then
   apply pruning (3).(b): if $C$ is already in $C^2Set$ then return else output $C$ to $C^2Set$.

4. Scan the $RWave^{\gamma}$ models of $C.pX$ when applying pruning (2) and store the condition candidates to $CandiSet$.

5. for each candidate condition $c_i \in CandiSet$ do
   find the subset of genes $X^{c_i} \subseteq C.X$ which match
   either $C.Y$+"$\curvearrowleft c_i$" or invert($C.Y$+"$\curvearrowleft c_i$")
   when applying pruning (2);
   sort $X^{c_i}$ on coherence score discrepancy
   $H(j, c_{k1}, c_{k2}, c_{km}, c_i)$ where $g_j \in X^{c_i}$;
   apply sliding window with minimum length
   $MinG$ and threshold $\epsilon$ on sorted $X^{c_i}$;
   apply pruning (4): if no validated gene interval $X''$ then continue;
   for each validated $X''$ after sliding do
       $C'.Y = C.Y$+"$\curvearrowleft c_i$"; $C'.X = X''$;
       $MineC^2(C', C^2Set)$
```

Figure 5.5: reg-cluster Mining Algorithm

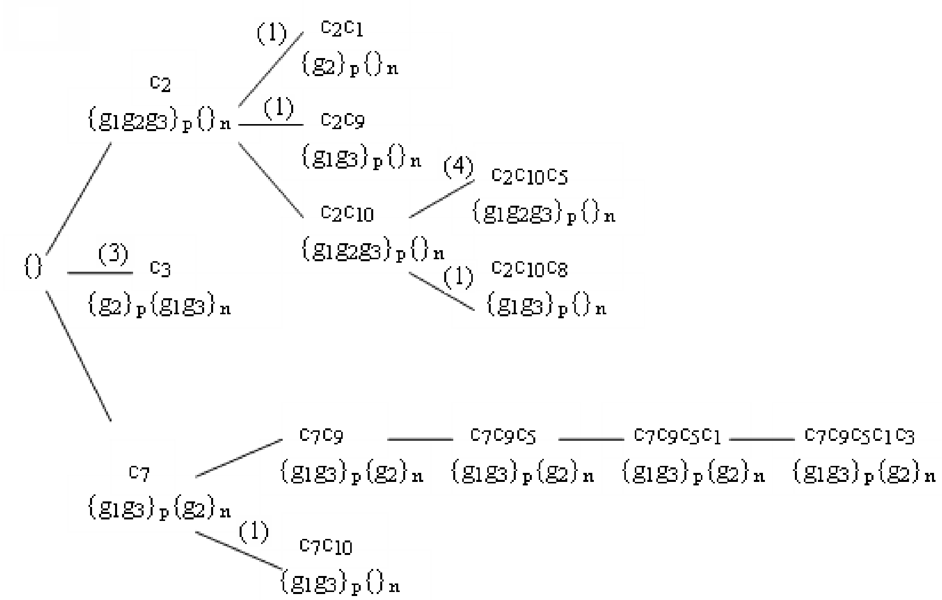


Figure 5.6: Enumeration Tree of Representative Regulation Chains w.r.t. $\gamma = 0.15$, $\epsilon = 0.1$, $MinG = 3$ and $MinC = 5$

regulation chain.

In summary, our reg-cluster algorithm illustrated in Figure 5.5 performs a *bi-directional depth-first* search on the $RWave^{\gamma}$ models for representative regulation chains ($C.Y$) satisfying the user specified minimum number of genes $MinG$, minimum number of conditions $MinC$, regulation threshold γ, and coherence threshold ϵ. At any step, the candidate regulation successors for the partially enumerated representative regulation chain $C.Y$ are held in $CandiSet$. For each candidate $c_i \in CandiSet$, we locate the subset of genes $X^{c_i} \subseteq C.X$ which satisfy $C.Y \curvearrowleft c_i$ and sort them in non-descending order of the coherence score ($H(j$, c_{k1}, c_{k2}, c_{km}, $c_i)$, $g_j \in X^{c_i}$). Then we use a sliding window of the minimum length $MinG$ and coherence threshold ϵ to partition X^{c_i} into a set of validated maximal subset of genes X'', which may overlap. The same process $MineC^2()$ is applied to each partition C' ($C'.Y = C.Y \curvearrowleft c_i$ and $C'.X = X''$) recursively.

Figure 5.6 shows an example of representative regulation chain enumeration process. We apply the following pruning strategies:

(1) $MinG$ pruning: Whenever the total number of p-members and n-members of the current enumerated representative regulation chain is below $MinG$, we prune the search after this node, as further extension of the representative regulation chain will only reduce the number of genes.

(2) $MinC$ pruning: Whenever the estimated maximal length of the current enumerated representative regulation chain of a gene falls below $MinC$, we remove the gene from further consideration.

(3) Redundant pruning: (a) Whenever the number of p-members is below $MinG/2$ ($|C.pX| < MinG/2$), we prune the candidate reg-cluster because the number of p-members would be smaller than the number of n-members. (Any validated reg-cluster contains at least $MinG$ members.) (b) Whenever a validated reg-cluster is found to be repetitive (as a result of overlapping gene sets after applying the sliding window techniques), we prune the search because the search space rooted at this node is redundant.

(4) Coherence pruning: Whenever less than $MinG$ genes are coherent (defined by ϵ) at a node, we prune the search.

Note that with pruning strategies (2) and (3).(a), we only need to look at p-members of the current enumerated representative regulation chain $C.Y$ when searching for extending condition candidates.

Figure 5.6 is the representative regulation chain enumeration tree for the running example (Table 5.1) when $\gamma = 0.15$, $\epsilon = 0.1$, $MinG = 3$ and $MinC = 5$, which consists of six levels, 0, 1, ..., 5. The number on the tree edge indicates the pruning strategies applied. At the ith level, the bicluster subroutine tests all possible representative regulation chains of length i. The depth-first search starts from the root node initialized with an empty chain. At level 1, the only possible candidate conditions are c_2, c_3 and c_7. The rest conditions cannot grow any regulation chain of length 5 along the $RWave^{0.15}$ models (Figure 5.3). So we can prune the search on c_1, c_4, c_5, c_6, c_8, c_9 and c_{10} according to pruning strategies (2) and (3).(a). Moreover, we can prune the search following node c_3 using pruning strategy (3).(a), because the number of p-members of the regulation chain c_3 is 1, which is smaller than $MinG/2$. Then, we grow the subtree of node c_2 with candidates c_1, c_9 and c_{10}, which are all possible conditions for extending a regulation chain of minimum length 5. With pruning strategy (1), we can prune the search after nodes c_2c_1 and c_2c_9. The only extensible child of node c_2 is c_2c_{10}, whose candidates are c_5 and c_8 with pruning strategy (3).(a). Node $c_2c_{10}c_5$ is pruned during coherence test with pruning strategy (4), since $H(1, c_2, c_{10}, c_{10}, c_5) = H(3, c_2, c_{10}, c_{10}, c_5) = 0.5263$ while $H(2, c_2, c_{10}, c_{10}, c_5) = 2$ and, therefore, no validated gene subset is discovered when sliding the window of minimum length 3 and $\epsilon = 0.1$. Node $c_2c_{10}c_8$ is pruned with pruning strategy (1). Again, we examine the p-members of node c_7 and find the candidates for further extension are c_9 and c_{10}. c_7c_{10} is pruned with strategy (1) and the only validated representative regulation chain discovered is $c_7 \curvearrowleft c_9 \curvearrowleft c_5 \curvearrowleft c_1 \curvearrowleft c_3$.

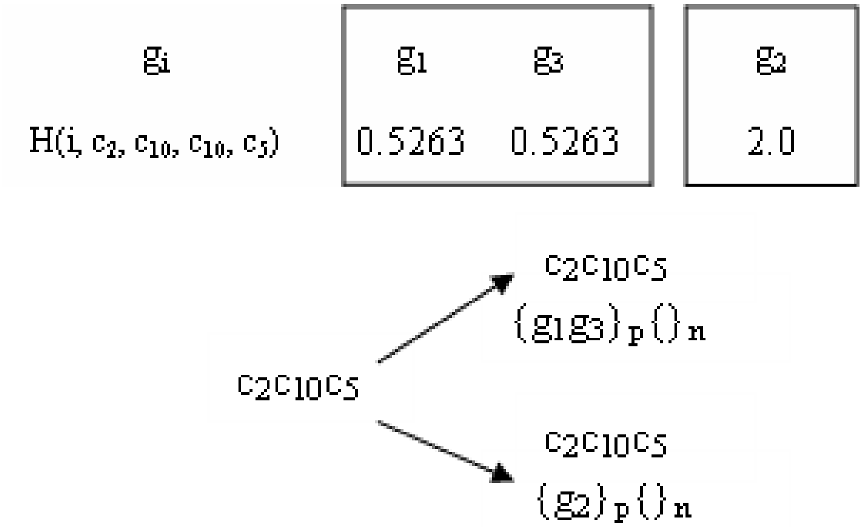
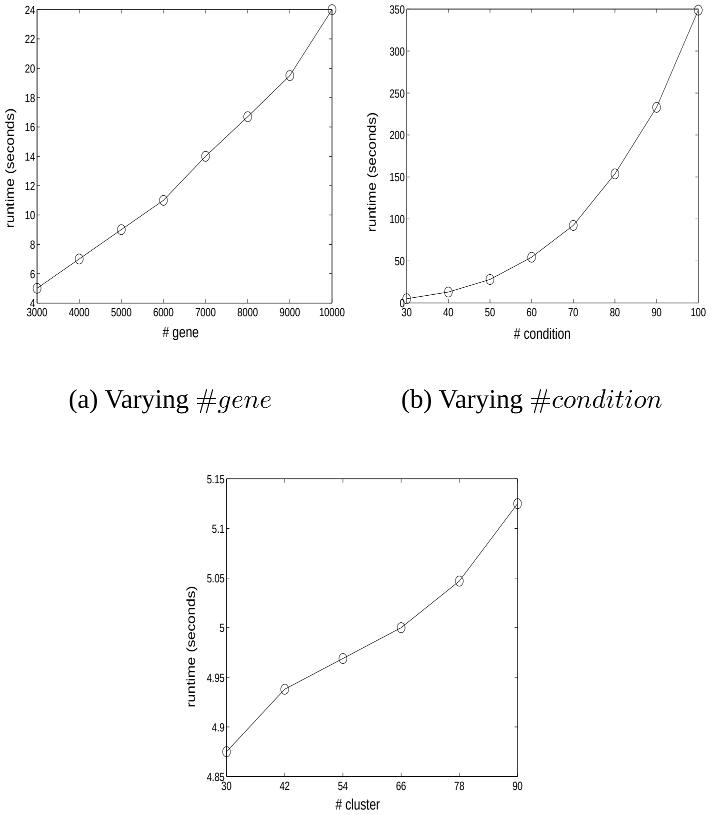


Figure 5.7: A Simple Node Split Case when $MinG = 1$ and $\epsilon = 0.1$

A more complicated case than Figure 5.6 is node split. When a sliding window with minimum length of $MinG$ and coherence threshold ϵ is applied on the genes sorted by coherence scores, the genes will be assigned to several possibly overlapping maximal gene subsets. In our running example, node split occurs at node $c_2c_{10}c_5$ when $\epsilon = 0.1$ and $MinG = 1$, as illustrated in Figure 5.7.

5.4 Experimental Studies

To evaluate the performance of our reg-cluster algorithm, we performed experiments on a series of synthetic datasets and two real-life gene expression datasets, 2D and 3D respectively, on a 3.0-GHz Dell PC with 1G memory running Window XP.



(a) Varying $\#gene$

(b) Varying $\#condition$

(c) Varying $\#cluster$

Figure 5.8: Evaluation of Efficiency on Synthetic Datasets

As the runtime of reg-cluster algorithm on real datasets are too short for in-depth analysis, we evaluate the efficiency of our algorithm on synthetic datasets, which are obtained with a data generator with three input parameters: number of genes ($\#gene$), number of conditions ($\#condition$), and number of embedded clusters ($\#cluster$). We set the default parameters of the data generator algorithm as $\#gene = 3000$, $\#condition = 30$ and $\#cluster = 30$. The synthetic dataset is initialized with random values ranging from 0 to 10. Then a number of $\#cluster$ perfect shifting-and-scaling clusters of average dimensionality 6 and average number of genes (including both p-member genes and n-member genes) equal to $0.01 * \#gene$ are embedded into the data, which are reg-clusters with parameter settings $\epsilon = 0$ and $\gamma = 0.15$.

We evaluate the effectiveness and extensibility of our reg-cluster algorithm on a benchmark 2D yeast gene expression data [76], available at `http://arep.med.harvard.edu/biclustering/`, and the 3D $gene \times sample \times time$ in [85] respectively. The 2D dataset contains the expression levels of 2884 genes under 17 conditions while the 3D dataset contains the expression values of 7679 genes from 13 samples under 14 time points.

Cluster	Process	Function	Cellular Component
c_1^2	DNA replication (p=3.64e-07)	DNA-directed DNA polymerase activity (p=0.01586)	replication fork (p=0.00019)
c_3^2	protein biosynthesis (p=0.00016)	structural constituent of ribosome (p=1.45e-07)	cytosolic ribosome (p=1.44e-08)
c_{13}^2	cytoplasm organization and biogenesis (p=5.72e-05)	helicase activity (p=0.00175)	ribonucleoprotein complex (p=0.0002)
c^3	mRNA transport (p=0.00057)	ATP binding (p=0.00313)	integral to membrane (p=0.00329)

Table 5.2: Top GO Terms of the Discovered Biclusters and Tricluster

5.4.1 Efficiency

Given the default parameter setting of the data generator algorithm above, we test the scalability of reg-cluster by varying only one input parameter while keeping the other two as default. The average runtime of reg-cluster when we vary the parameters invoked with $MinG = 0.01 * \#gene$, $MinC = 6$, $\gamma = 0.1$ and $\epsilon = 0.01$ is illustrated in Figure 5.8. As we can observe, the runtime of the reg-cluster algorithm is slightly more than linear in terms of the number of genes ($\#gene$). It shows worse scalability with respect to the number of conditions ($\#condition$). This is because the reg-cluster algorithm may examine all possible permutations of conditions when looking for the representative regulation chains, but it only searches for the maximal sets of genes that are projected onto the enumerated (inverted) representative regulation chains. Typically, the number of conditions is much smaller than the number of genes. Figure 5.8 shows an approximately linear relationship between the runtime of the reg-cluster algorithm and the number of clusters ($\#cluster$).

5.4.2 Effectiveness

We ran the reg-cluster algorithm on the 2D 2884×17 yeast dataset with $MinG = 20$, $MinC = 6$, $\gamma = 0.05$ and $\epsilon = 1.0$; 21 bi-reg-clusters are output in 2.5 seconds, where the overlapping percentage a bi-reg-cluster with another one generally ranges from 0% to 85%. Note that we did not perform any splitting and merging of clusters. Due to space limit, we only report the details of three non-overlapping bi-reg-clusters with 21 genes and six conditions each.

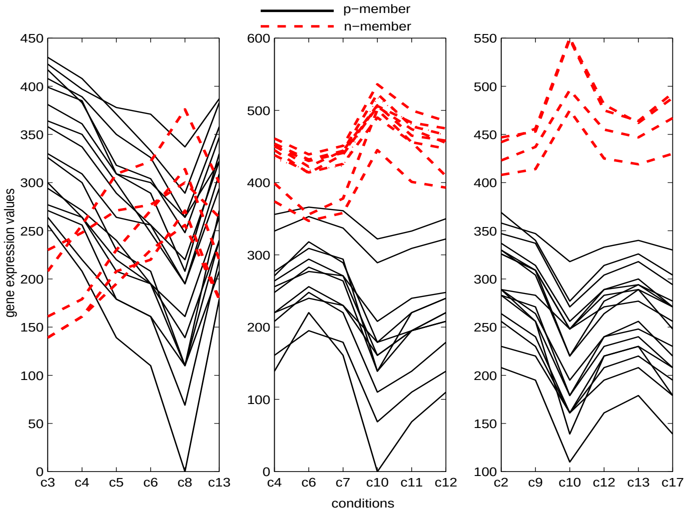


Figure 5.9: Three biclusters

Figure 5.9 illustrates the gene expression profiles for each of the three bi-reg-clusters. Our reg-cluster algorithm can successfully identify shifting-and-scaling patterns satisfying the regulation and coherence thresholds, where the scaling factor can be either positive or negative. For each bi-reg-cluster, we represent its p-members with black solid lines and its n-members with red dashed lines. Obviously, the relationship between any two p-member genes or between any two n-member genes of the same cluster is shifting-and-positive-scaling while that between a p-member gene and a n-member gene is shifting-and-negative-scaling. As a remarkable characteristic of reg-clusters, crossovers can be observed frequently in the gene expression profiles of a pair of genes, resulting from the combination effects of shifting and scaling. In contrast, previous pattern-based biclustering algorithms [80, 83, 85] only allow pure shifting or pure positive-scaling patterns (but not a mixture of both) and hence fail to identify the three bi-reg-clusters.

We apply the yeast genome gene ontology term finder (`http://db.yeastgenome.org/cgi-bin/GO/goTermFinder`) on each discovered clusters to evaluate their biological significance in terms of associated biological processes, cellular components and gene function respectively. Table 5.2 shows the top GO terms of the three categories and the GO terms with the lowest p-values for the 3 bi-reg-clusters in Figure 5.9, which have been overlooked by previous work. Despite the relatively smaller number of genes with our regulation threshold $\gamma = 0.05$, the extremely low p-values suggest that the three bi-reg-clusters are of significant biological meaning in terms of biological process, cellular component and gene function.

Further experimental results show that our reg-cluster algorithm can identify a much broader range of biologically significant gene clusters. Each group of genes in these clusters show strikingly similar regulation under a subset of conditions.

5.4.3 Extension to 3D Dataset

Our reg-cluster mining algorithm can be easily extended for mining the 3D $gene \times sample \times time$ expression dataset in [85]. All we need is to replace the biclustering subroutine of the tricluster algorithm in [85] with our reg-cluster algorithm in Figure 5.5, and replace its coherence cluster model with our reg-cluster model. We build the 3D $gene \times sample \times time = 7679 \times 13 \times 14$ expression dataset as that used in [85] by choosing 13 attributes as samples of the raw data taken at each of the 14 time points (0min, 30min, ..., 390min) for 7679 genes during the elutriation experiments. The raw data is available at `http://genome-www.stanford.`

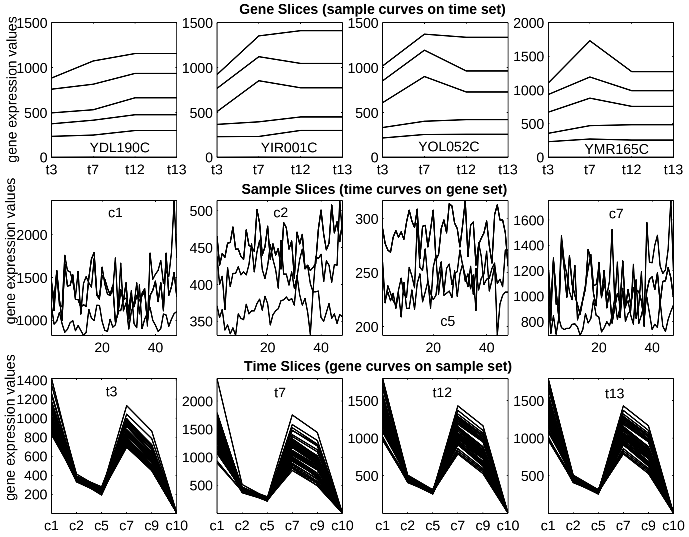


Figure 5.10: One Tricluster

`edu/cellcycle/data/rawdata/individual.html`. Here, the sample dimension corresponds to our condition dimension.

We first mine the biclusters in the 2D time slice $|gene| \times |sample| = 7679 \times 13$ for each of the 14 time points with $\gamma_{gene \times sample} = 0.1$. Then we search the intersections of the time slices with $MinTime = 4$ and relax the regulation thresholds $\gamma_{gene \times time}$ and $\gamma_{sample \times time}$ to zeroes, considering the gene expression levels across different time points need not regulate rigidly. In the whole process, we set the coherence thresholds as $\epsilon_{gene \times sample} = \epsilon_{sample \times time} = \epsilon_{gene \times time} = 5$, accommodating the noise in the raw data. With $MinG = 40$, $MinC = 6$ and $MinTime = 4$, our reg-cluster algorithm identified one $48 \times 6 \times 4$ tricluster on sample set $\{c_1, c_2, c_5, c_7, c_9, c_{10}\}$ and time set $\{t_3, t_7, t_{12}, t_{13}\}$ after cluster merging. The gene slice views (projected on the 12th, 24th, 36th and 48th genes), the sample slice views (projected on the first four samples) and the four time slice views of this tricluster is shown in Figure 5.10. Again, the GO term finder reports significant biological meaning of this cluster, denoted as c^3 (Table 5.2).

5.5 Summary

To overcome the limitations of previous pattern-based biclustering algorithms which can only find either pure shifting or pure positive scaling patterns, we have proposed a general reg-cluster model for identifying arbitrary shifting-and-scaling co-regulation patterns, where the scaling can be either positive or negative. Unlike previous work, our algorithm also allows a flexible regulation threshold to quantify up or down regulation. The shifting-and-scaling patterns manifest a synchronous and proportional change of expression values in a subspace, and are able to capture both positive correlations and negative correlations among the genes in the subspace. We have developed a bi-directional depth-first algorithm which effectively and efficiently mine the reg-clusters using a novel $RWave^{\gamma}$ model. Our experimental results prove that our reg-cluster algorithm is able to: (1) discover a significantly number of biologically meaningful reg-clusters missed by previous work; and (2) be easily extended to 3D $gene \times sample \times time$ dataset.

CHAPTER 6

Conclusion

In recent years, large amounts of high-dimensional data, such as images, handwriting and gene expression profiles, have been generated. Analyzing and handling such kinds of data have become an issue of keen interest. Elucidating the patterns hidden in high-dimensional data imposes an even greater challenge on cluster analysis. In this thesis, we have proposed effective and efficient data mining methods for gene expression analysis in capturing the correlation between gene expression profiles and environmental conditions, and also the correlation among genes themselves. While we focus on gene expression data, our data mining techniques can be applied to other kinds of high-dimensional data with homologous correlations as well. We summarize our work as follows.

- The high-dimensionality of gene expression data renders traditional item-wise association rule mining algorithms impractical due to exponential explosion of item combinations. Although a recent row-wise rule mining algorithm FARMER is much more efficient than traditional item-wise algorithms by identifying interesting rule groups instead of searching individual rules one by one, the number of interesting rule groups can still be very large. We proposed the concept of top k covering rule groups, TopKRGs, and developed an efficient algorithm for TopKRGs discovery. In this way, we not only solved the problems of inefficiency and huge rule number, but also helped users concentrate on the most significant information and minimized the information loss. Experimental studies on four benchmark gene expression datasets demonstrate that our TopKRGs algorithm is significantly faster than FARMER.

- Based on TopKRGs, we designed a novel associative classifier RCBT composed of a committee of k sub-classifiers. Each test sample is classified by the highest ranked sub-classifier and will be assigned the default class only when no sub-classifiers matches the test sample. Compared with previous associative classifiers [20, 56], RCBT greatly reduces the chance of default class judgement as well as successfully locating globally significant rules. Moreover, by combining the discriminating powers of the delicately selected rules from TopKRGs, RCBT achieves a rather high classification accuracy on four benchmark gene expression datasets. To give users some hints on Top-

KRGs criteria, effective visualization techniques are also introduced, which provides an interactive graphic interface for users to observe, compare and explore rule groups.

- To address nonlinear correlation, we proposed a novel algorithm CURLER which adopts a fuzzy EM clustering subroutine to estimate the nonlinear orientations of the data in a trade off for efficiency and accuracy. Inspired by the reachability plot of OPTICS, we also proposed NNCO plot which visualizes the clusters embedded in subspace as well as their orientations. As another contribution, CURLER works in top-down manner so that users are able to further explore the sub-structure of any cluster of their interest. Experimental studies were carried out on synthetic helix datasets, UCI machine learning repository and real-life gene expression data to show the efficiency and effectiveness.

- Correlated genes can demonstrate pure shifting or pure scaling expression patterns across a subset of samples. Such correlation is pattern-based, which is neither linear nor nonlinear. We successfully improved existing pattern-based subspace clustering algorithms which ignore the general shifting-and-scaling pattern by proposing reg-cluster to cluster genes exhibiting shifting-and-scaling patterns w.r.t. coherence threshold ϵ and regulation threshold γ. Experimental studies on real-life gene expression data show that these shifting-and-scaling patterns ignored by previous work have rather high biological significance. Experimental results also indicate that our reg-cluster discovery algorithm is efficient and scalable on high-dimensional data.

In our future studies, we would like to further explore the following related problems.

- Association rule mining algorithms run on discretized data. One interesting question is whether the discretization subroutine and the association mining subroutine can be integrated simultaneously. For classification purpose, entropy discretization method is usually adopted to partition the data first. However, the resulting genes may still contain duplicate information. And the discovered rules may have such redundant information as well. The performance of the associative classifier may be increased by combining discretization and rule ming together to filter out most important information directly.

- Another problem related with class association rule mining on gene expression data is the disregard of time factor. The gene expression profiles of patients could be rather different at distinct disease phases, while current associative rules just reflect the correlation at a single one phase. When applied to cancer diagnosis in clinical practice, these rules may be problematic. A better way may be to discover class association rule whose item corresponds to an expression interval of a gene at a certain phase or a tendency change of individual genes rather than a fixed expression interval.

- Our CURLER algorithm is able to identify nonlinear as well as linear correlation gene clusters in subspace and our reg-cluster algorithm is capable of finding the general shifting-and-scaling clusters in subspace. However, neither of them has considered the case where the density-based, no matter linear or nonlinear, and the pattern-based clusters coexist together. It will be interesting if we can combine density measurement and pattern similarity measurement together.

Although biological technology will continue growing and evolving, data mining will remain a powerful tool to effective and efficiently discover the most important information from the vast and complex data. It is admitted that the ultimate impact of the studies on biology will depend heavily on data mining and statistical analysis. Data miners definitely undertake great responsibility with the advance of new biology era.

BIBLIOGRAPHY

[1] Hinneburg A. and Keim D.A. An efficient approach to clustering in large multimedia databases with noise. In *Proc. Int. Conf. Knowledge Discovery and Data Mining*, pages 58–65, Aug. 1998.

[2] Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proc. of ACM SIGMOD Conf. Proceedings*, volume 29, 2000.

[3] Charu C. Agrawal, Cecilia Procopiuc, Joel L. Wolf, Philip S. Yu, and Jong Soo Park. Fast algorithms for projected clustering. In *Proc. of ACM SIGMOD Int. conf. on Management of Data*, pages 61–72, 1999.

[4] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data

mining applications. In *Proc. of ACM-SIGMOD Int. Conf. on Management of Data*, pages 94–105, June 1998.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 487–499, Sept. 1994.

[6] T. R. Anderson and T. A. Slotkin. Maturation of the adrenal medulla–iv. effects of morphine. *Biochem Pharmacol,* August 1975.

[7] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jorg Sander. Optics: Ordering points to identify the clustering structure. In *Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data*, pages 49–60, 1999.

[8] Chinatsu Arima, Taizo Hanai, and Masahiro Okamoto. Gene expression analysis using fuzzy k-means clustering. *Genome Informatics*, 14, 2003.

[9] Pierre Baldi and S. Brunak. *Bioinformatics: the Machine Learning Approach*. MIT Press, 1998.

[10] Roberto J. Bayardo and Rakesh Agrawal. Mining the most intersting rules. In *Proc. of ACM SIGKDD*, 1999.

[11] Roberto J. Bayardo, Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense databases. In *Proc. 15th Int. Conf. on Data Engineering*, 1999.

[12] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: the ordering-preserving submatrix problem. In *Recomb*, 2002.

[13] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[14] Christian Bohm, Karin Kailing, Peer Kroger, and Arthur Zimek. Computing clusters of correlation connected objects. In *Proc. of the 2003 ACM SIGMOD Int. Conf. on Management of data*, 2003.

[15] K. S. Bose and R. H. Sarma. Delineation of the intimate details of the backbone conformation of pyridine nucleotide coenzymes in aqueous solution. *Biochem Biophys Res Commun*, October 1975.

[16] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[17] Kaushik Chakrabarti and Sharad Mehrotra. Local dimensionality reduction: a new approach to indexing high dimensional spaces. *Proc. of the 26th VLDB Conference*, 2000.

[18] Ting Chen, Vladimir Filkov, and Steven S. Skiena. Identifying gene regulatory networks from experimental data. In *Recomb*, 1999.

[19] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proc. of the Eighth Int. Conf. on Intelligent Systems for Molecular Biology*, 2000.

[20] Gao Cong, Anthony K. H. Tung, Xin Xu, Feng Pan, and Jiong Yang. Farmer: Finding interesting rule groups in microarray datasets. *In the 23rd ACM SIGMOD International Conference on Management of Data*, 2004.

[21] Kathryn R. Coser, Jessica Chesnes, Jingyung Hur, Sandip Ray, Kurt J. Isselbacher, and Toshi Shioda. Global analysis of ligand sensitivity of estrogen inducible and suppressible genes in mcf7/bus breast cancer cells by dna microarray. In *Proc. of the National Academy of Sciences of the United States of America*, 2003.

[22] Chad Creighton and Samir Hanash. Mining gene expression databases for association rules. *Bioinformatics*, 19, 2003.

[23] P. D'haeseleer, S. Liang, and R. Somogyi. Gene expression analysis and genetic network modeling. In *Pacific Symposium on Biocomputing*, 1999.

[24] Inderjit S. Dhillon, Edward M. Marcotte, and Usman Roshan. Diametrical clustering for identifying anti-correlated gene clusters. *Bioinformatics*, 19:1612–1619, 2003.

[25] S. Doddi, A. Marathe, S.S. Ravi, and D.C. Torney. Discovery of association rules in medical data. *Med. Inform. Internet. Med.*, 26:25–33, 2001.

[26] Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. Caep: Classification by aggregating emerging patterns. *Discovery Science*, 1999.

[27] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein.

Cluster analisis and display of genome-wide expression patterns. In *Proc. Natl. Acad. Sci. USA*, volume 95, pages 14863–14868, 1998.

[28] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[29] Pan F., Wang B., Hu X., and Perrizo W. Comprehensive vertical sample-based knn/lsvm classification for gene expression analysis. In *J Biomed Inform*, 2004.

[30] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[31] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using bayesian network to analyze expression data. In *Proc. 4th Annual International Conference on Computational Molecular Biology (Recomb 2000)*, pages 127–135, 2000.

[32] Denis J. Glenn and Richard A. Maurer. Mrg1 binds to the lim domain of lhx2 and may function as a coactivator to stimulate glycoprotein hormone α-subunit gene expression. *J Biol Chem*, 1999.

[33] Benjamin Good, Jeremy Peay, Satish Pillai, and Jacques Corbeil. Class prediction based on gene expression: Applying neural networks via a genetic al-

gorithm wrapper. In *2001 Genetic and Evolutionary Computation Conference Late Breaking Papers*, pages 122–130, July 2001.

[34] Patrik D Haeseleer, Xiling Wen, Stefanie Fuhrman, and Roland Somogyi. Mining the gene expression matrix: Inferring gene relationships from large scale gene expression data. *Information Processing in Cells and Tissues,* pages 203–212, 1998.

[35] Jiawei Han and Micheline Kamber. *Data mining concepts and techniques*. Morgan Kaufmann Publishers,San Francisco, CA, 2001.

[36] Jiawei Han and Jian Pei. Mining frequent patterns by pattern growth: methodology and implications. *KDD Exploration*, 2, 2000.

[37] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM-SIGMOD International Conference Management of Data*, 2000.

[38] Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(200):175–181, 2000.

[39] Alexander Hinneburg and Daniel A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proc. 1999 Int. Conf. Very Large Data Bases,* pages 58–65, Edinburgh, UK, Sept. 1999.

[40] Timothy R. Hughes, Matthew J. Marton, Allan R. Jones, Christopher J.

Roberts, Roland Stoughton, Christopher D. Armour, Holly A. Bennett, Ernest Coffey, Hongyue Dai, Yudong D. He, Matthew J. Kidd, Amy M. King, Michael R. Meyer, David Slade, Pek Y. Lum, Sergey B. Stepaniants, Daniel D. Shoemaker, Daniel Gachotte, Kalpana Chakraburtty, Julian Simon, Martin Bard, and Stephen H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102:109–126, 2000.

[41] Chun hung Cheng, Ada Wai chee Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*, 1996.

[42] V.R. Iyer, M.B. Eisen, D.T. Ross, G. Schuler, T. Moore, J.C.F. Lee, J.M. Trent, L.M. Staudt, J.Jr Hudson, M.S. Boguski, D. Lashkari, D. Shalon, D. Botstein, and P.O. Brown. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283:83–87, 1999.

[43] Banfield J.D. and Raftery A.E. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49:803–821, September, 1993.

[44] Liping Ji and Kian-Lee Tan. Mining gene expression data for positive and negative co-regulated gene clusters. *Bioinformatics*, 20:2711–2718, 2004.

[45] Thorsten Joachims. Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Learning*, 1999. http://svmlight.joachims.org/.

[46] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 2002.

[47] Masaaki Kasai, Jennifer Guerrero-Santoro, Robert Friedman, Eddy S. Leman, Robert H. Getzenberg, and Donald B. DeFranco. The group 3 lim domain protein paxillin potentiates androgen receptor transactivation in prostate cancer cell lines. *Cancer Research*, 2003.

[48] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, 1990.

[49] T. Kohonen. Self-organizing maps. *Springer*, 1997.

[50] Balaji Krishnapuram, Lawrence Carin, and Alexander J. Hartemink. Joint classifier and feature optimization for cancer diagnosis using gene expression data. In *Recomb*, 2003.

[51] S. Kurimoto, N. Moriyama, K. Takata, S. A. Nozaw, Y. Aso, and H. Hirano. Detection of a glycosphingolipid antigen in bladder cancer cells with monoclonal antibody mrg-1. *Histochem J.*, 1995.

[52] Jinyan Li, Huiqing Liu, James R. Downing, Allen Eng-Juh Yeoh, and Limsoon Wong. Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (all) patients. *Bioinformatics*, 19:71–78, 2003.

[53] Jinyan Li and Limsoon Wong. Identifying good diagnostic genes or genes groups from gene expression data by using the concept of emerging patterns. *Bioinformatics*, 18:725–734, 2002.

[54] Jinyan Li and Limsoon Wong. Using rules to analyse bio-medical data: A comparison between c4.5 and pcl. *Proc. of 4th Int. Conf. on Web-Age Information Management*, 2003.

[55] Wenmin Li, Jiawei Han, and Jian Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *Proc. of 2001 IEEE Int. Conf. on Data Mining*, pages 369–376, 2001. http://citeseer.nj.nec.com/li01cmar.html.

[56] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining*, 1998.

[57] Bing Liu, Wynne Hsu, and Yiming Ma. Pruning and summarizing the discovered associations. In *ACM KDD*, 1999.

[58] Jinze Liu and Wei Wang. Op-cluster: Clustering by tendency in high dimensional space. In *Proc. of the Third IEEE Int. Conf. on Data Mining*, 2003.

[59] Jinze Liu, Wei Wang, and Jiong Yang. Gene ontology friendly biclustering of expression profiles. In *Computational Systems Bioinformatics*, 2004.

[60] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In *Proc. AAAI'94 Workshop Knowledge Discovery in Databases*, 1994.

[61] Masaki Nagata, Hajime Fujita, Hiroko Ida, Hideyuki Hoshina, Tatsuo Inoue,

Yukie Seki, Makoto Ohnishi, Tokio Ohyama, Susumu Shingaki, Masataka Kaji, Takashi Saku, and Ritsuo Takagi. Identification of potential biomarkers of lymph node metastasis in oral squamous cell carcinoma by cdna microarray analysis. *International Journal of Cancer*, 106:683–689, June 2003.

[62] Raymond T. Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data*, 1998.

[63] Giulia Pagallo and David Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5:71–99, 1990.

[64] Feng Pan, Gao Cong, Anthony K. H. Tung, Jiong Yang, and Mohammed J. Zaki. Carpenter: Finding closed patterns in long biological datasets. In *Proc. 2003 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[65] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. 7th Int. Conf. Database Theory*, Jan. 1999.

[66] Jian Pei, Jiawei Han, Hongjun Lu, Shojiro Nishio, Shiwei Tang, and Dongqing Yang. H-mine: Hyper-structure mining of frequent patterns in large databases. In *Proc. IEEE 2001 Int. Conf. Data Mining*, Novermber 2001.

[67] John L. Pfaltz and Christopher M. Taylor. Closed set mining of biological data. *Workshop on Data Mining in Bioinformatics*, pages 43–48, 2002.

[68] Cecilia M. Procopiuc, Michael Jones, Pankaj K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2002.

[69] Jiang Qian, Marisa Dolled-Filhart, Jimmy Lin, Haiyuan Yu, and Mark Gerstein. Beyond synexpression relationships: Local clustering of time-shifted and inverted gene expression profiles indentifies new, biologically relevant interactions. In *Journal of Molecular Biology*, 2001.

[70] J. R. Quinlan. Bagging, boosting, and C4.5. In *Proc. 1996 Nat. Conf. Artificial Intelligence (AAAI'96)*, volume 1, pages 725–730, Portland, OR, Aug. 1996.

[71] J.R. Quinlan. C4.5: Programs for machine learning. In *Morgan Kaufmann, San Mateo, CA*, 1993.

[72] Rajeev Rastogi and Kyuseok Shim. Mining optimized association rules with categorical and numeric attributes. In *Int. Conf. on Data Engineering*, 1998.

[73] J. Roy. A fast improvement to the em algorithm on its own terms. *JRSS(B)*, 51:127–138, 1989.

[74] H. Shatkay, S. Edwards, W. J. Wilbur, and M. Boguski. Genes, themes, and microarray: Using information retrieval for large-scale gene analysis. In *Proc. of the Eighth Int. Conf. on Intelligent Systems for Molecular Biology*, 2000.

[75] Pablo Tamayo, Donna Slnim, Jill Mesirov, Qing Zhu, Sutisak Kitareewan, Ethan Dmitrovsky, Eric S. Lander, and Todd R. Golub. Interpreting patterns

of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. In *Proc. Natl. Acad. Sci. USA*, volume 96, pages 2907–2912, 1999.

[76] Saeed Tavazoie, Jason D. Hughes, Michael J. Campbell, Raymond J. Cho, and George M. Church. Systematic determination of genetic network architecture. In *Nature Genetics*, volume 22, pages 281–285, 1999.

[77] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2323–2326, 2000.

[78] Anthony K. H. Tung, Jiawei Han, Laks V. S. Lakshmanan, and Raymond T. Ng. Constraint-based clustering in large databases. In *Proc. 2001 Int. Conf. Database Theory*, Jan. 2001.

[79] Anthony K. H. Tung, Jean Hou, and Jiawei Han. Spatial clustering in the presence of obstacles. In *Proc. 2001 Int. Conf. Data Engineering*, Heidelberg, Germany, April 2001.

[80] Haixun Wang, Wei Wang, Jiong Yang, and Philip S. Yu. Clustering by pattern similarity in large data sets. In *Proc. of the 2002 ACM SIGMOD Int Conf. on Management of data*, 2002.

[81] Jianyong Wang, Jiawei Han, and Jian Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proc. 2003 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2003.

[82] Xiaowei Xu, Martin Ester, H.P. Kriegel, and J. Sander. A distributed-based clustering algorithm for mining in large spatial databases. In *Proc. Int. Conf. Data Engineering*, 1998.

[83] Jiong Yang, Wei Wang, Haixun Wang, and Philip Yu. δ-clusters: Capturing subspace correlation in a large data set. In *Proc. of the 18th Int. Conf. on Data Engineering*, 2002.

[84] Mohammed J. Zaki and Ching-Jui Hsiao. Charm: An efficient algorithm for closed association rule mining. In *Proc. 2nd SIAM Int. Conf. on Data Mining*, Apr. 2002.

[85] Lizhuang Zhao and Mohammed J. Zaki. Tricluster: An effective algorithm for mining coherent clusters in 3d microarray data. In *Proc.of the 2005 ACM SIGMOD Int. Conf. on Management of data*, 2005.