Mining Phenotypes and Informative Genes from Gene Expression Data

Chun Tang Aidong Zhang Jian Pei
Department of Computer Science and Engineering
State University of New York at Buffalo, Buffalo, NY 14260
{chuntang, azhang, jianpei}@cse.buffalo.edu

ABSTRACT

Mining microarray gene expression data is an important research topic in bioinformatics with broad applications. While most of the previous studies focus on clustering either genes or samples, it is interesting to ask whether we can partition the complete set of samples into exclusive groups (called *phenotypes*) and find a set of *informative genes* that can manifest the phenotype structure. In this paper, we propose a new problem of *simultaneously mining phenotypes and informative genes* from gene expression data. Some statistics-based metrics are proposed to measure the quality of the mining results. Two interesting algorithms are developed: the heuristic search and the mutual reinforcing adjustment method. We present an extensive performance study on both real-world data sets and synthetic data sets. The mining results from the two proposed methods are clearly better than those from the previous methods. They are ready for the real-world applications. Between the two methods, the mutual reinforcing adjustment method is in general more scalable, more effective and with better quality of the mining results.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms, Experimentation

Keywords

Phenotype, informative genes, array data, bioinformatics

1. INTRODUCTION

The DNA microarray technology enables rapid, large-scale screening for patterns of gene expression. The raw microarray data are transformed into gene expression matrices in which a row represents a gene and a column represents a sample. The numeric value in each cell characterizes the expression level of a specific gene in a particular sample.

SIGKDD '03, August 24-27, 2003, Washington, DC, USA

Effective and efficient analysis techniques are demanding as gene expression data are accumulated rapidly. The gene expression matrix can be analyzed in two ways. On the one hand, genes can be clustered based on similar expression patterns [7]. In such a *gene-based method*, the genes are treated as the objects, while the samples are the attributes. On the other hand, the samples can be partitioned into homogeneous groups. Each group may correspond to some particular macroscopic phenotypes, such as clinical syndromes or cancer types [8]. Such a *sample-based method* regards the samples as the objects and the genes as the attributes.

While a gene expression matrix can be analyzed from orthogonal angles, the gene-based and sample-based methods are facing very different challenges. The number of genes and the number of samples are very different in a typical gene expression matrix. Usually, we may have tens or hundreds of samples but thousands or tens of thousands of genes. Thus, we may have to adopt very different computational strategies in the two situations.

In this paper, we will focus on the *sample-based analysis*. In particular, we are interested in mining *phenotypes* and *informative genes*. Within a gene expression matrix, there are usually several phenotypes of samples related to some diseases or drug effects, such as diseased samples, normal samples or drug treated samples. Figure 1 shows a gene expression matrix containing two phenotypes of samples. Samples $1 \sim 3$ are in one phenotype, while samples $4 \sim 7$ are in the other one. For the sake of simplicity, the gene expression levels in the matrix are discretized into binary values, i.e., either "on" or "off".

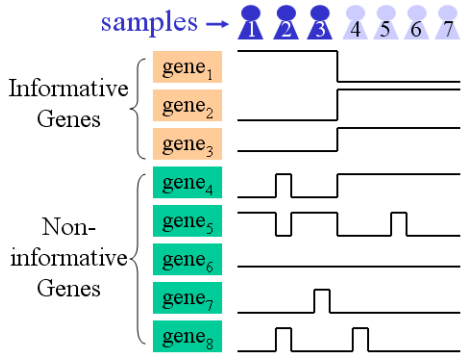

Figure 1: A simplified example of a gene expression matrix.

Biologists are particularly interested in finding the *informative genes* that manifest the phenotype structure of the samples. For example, in Figure 1, $gene_1 \sim gene_3$ are informative genes, since each gene shows all "on" signals for one phenotype of samples and all "off" for the other phenotype. $Gene_4 \sim gene_8$ provide

inconsistent signals for the samples in a phenotype, and thus cannot be used to distinguish phenotypes. They are called *non-informative* genes.

If phenotype information is known, the major task is to select the informative genes that manifest the phenotypes of samples. This can be achieved by *supervised analysis methods* such as the neighborhood analysis [8] and the support vector machine [4].

Although the supervised methods are helpful, the initial identification of phenotypes over samples are usually slow, typically by evolving through years of hypothesis-driven research [8]. Therefore, it is natural to ask "*Can we find both the phenotypes and the informative genes automatically at the same time?*" In other words, we want to discover the phenotypes of the samples as well as to identify a set of genes that manifests the phenotypes of the samples. For example, in Figure 1, if the seven samples' phenotypes are unknown, can we correctly distinguish samples $1 \sim 3$ from $4 \sim 7$ as well as output $gene_1 \sim gene_3$ as informative genes?

Mining both the phenotypes and the informative genes at the same time is challenging. First, the values in data matrices are all real numbers such that there is usually no clear border between informative genes and non-informative ones. Second, there are many genes but only a small number of samples. There is no existing technique to correctly detect class structures from samples. Last, most of the genes collected may not necessarily be of interest. The experience shows that only less than 10% of all the genes involved in a gene expression matrix are informative ones [8]. In other words, the gene expression matrix is very noisy.

In this paper, we tackle the problem of *mining phenotypes and informative genes from gene expression data* by developing novel unsupervised learning methods. We claim the following contributions.

- We identify and formulate the problem of simultaneously mining phenotypes and informative genes. The major differences between this novel problem and the previous studies on clustering or subspace clustering are elaborated.
- A set of statistical measurements of quality of phenotypes and informative genes are proposed. They coordinate and compromise both the sample phenotype discovery and the informative gene selection.
- A heuristic search method and a mutual reinforcing adjustment approach are devised to find phenotypes and informative genes with high quality. Particularly, the mutual reinforcing adjustment method dynamically manipulates the relationship between samples and genes while conducting an iterative adjustment to detect the phenotypes and informative genes.
- An extensive experimental evaluation over some real data sets is presented. It shows that our methods are both effective and efficient and outperform the existing methods. The mutual reinforcing method has the even better performance.

The remainder of this paper is organized as follows. Section 2 reviews the related work. The quality measurements are proposed in Section 3, while the mining algorithms are developed in Section 4. Section 5 reports the experimental results and Section 6 gives the conclusion.

2. RELATED WORK

Recently, some methods have been proposed to find macroscopic phenotypes from samples [6, 14]. In these approaches, samples are partitioned by conventional clustering methods, such as K-means, self-organizing maps (SOM), hierarchical clustering (HC), or graph based clustering. However, these traditional clustering techniques cannot handle the heavy noise well in the gene expression data. Although some approaches [16] filter out genes for partition samples, the gene filtering processes are non-invertible. The deterministic filtering will cause samples to be grouped based on the local decisions.

Sub-space clustering have been studied extensively [1, 5, 17] to find subsets of objects such that the objects appear as a cluster in a sub-space formed by a subset of the attributes. Although the subspace clustering problem may appear similar to the phenotype and informative gene mining problem at the first look, there are two significant and inherent differences between these two. On the one hand, in subspace clustering, the subsets of attributes for various subspace clusters are different. However, in phenotypes and informative gene mining, we want to find a *unique* set of genes to manifest a partition of *all* samples. On the other hand, two subspace clusters can share some common objects and attributes. Some objects may not belong to any subspace cluster. Nevertheless, in phenotype and informative gene mining, a sample must be in a phenotype and the phenotypes are exclusive.

3. QUALITY MEASUREMENTS OF PHENOTYPES AND INFORMATIVE GENES

The phenotypes of samples and informative genes should satisfy two requirements simultaneously. On the one hand, the expression levels of each informative gene should be similar over the samples within each phenotype. On the other hand, the expression levels of each informative gene should display a clear dissimilarity between each pair of phenotypes. To quantize how well the phenotypes and informative genes meet the two requirements, we introduce the *intra-phenotype consistency* and *inter-phenotype divergency* measurements.

Let $S = \{s_1, \ldots, s_m\}$ be a set of samples and $G = \{g_1, \ldots, g_n\}$ be a set of genes. The corresponding gene expression matrix can be represented as $M = \{w_{i,j} | 1 \leq i \leq n, 1 \leq j \leq m\}$, where $w_{i,j}$ is the expression level value of sample s_j on gene g_i. Given a subset of samples $S' \subseteq S$ and a subset of genes $G' \subseteq G$, $M_{S',G'} = \{w_{i,j} | g_i \in G', s_j \in S'\}$ is the corresponding submatrix w.r.t. S' and G'.

Intra-phenotype consistency: The variance of each row measures whether a given gene has consistent expression level values over all samples within the sub-matrix. A small variance value indicates that the gene has consistent values on all the samples. Thus we can measure whether every gene has good consistency on a set of samples by the average of variance in the subset of genes. That is, we define the *intra-phenotype consistency* as:

$$Con(G', S') = \frac{1}{|G'| \cdot (|S'| - 1)} \sum_{g_i \in G'} \sum_{s_j \in S'} (w_{i,j} - \overline{w}_{i,S'})^2.$$

Inter-phenotype divergency: The *inter-phenotype divergency* quantizes how a subset of genes can distinguish two phenotypes of samples. The *inter-phenotype divergency* of a set of genes G' on two groups of samples (denoted as S_1 and S_2) such that $S_1 \subset S$, $S_2 \subset S$, and $S_1 \cap S_2 = \emptyset$ is defined as

$$Div(G', S_1, S_2) = \frac{\sum_{g_i \in G'} |\overline{w}_{i,S_1} - \overline{w}_{i,S_2}|}{|G'|}. \quad (1)$$

The greater the inter-phenotype divergency, the better the genes differentiate the phenotypes.

The quality of phenotypes and informative genes: Suppose a set of samples S is partitioned into k exclusive groups, $S_1, \ldots, S_k$. Given a set of genes G', the quality measure Ω quantizes how pure

the phenotypes are w.r.t. the genes and how well the genes differentiate the phenotypes.

$$\Omega = \frac{1}{\sum_{S_i,S_j(1\leq i,j\leq k;i\neq j)} \frac{\sqrt{Con(G',S_i)+Con(G',S_j)}}{Div(G',S_i,S_j)}}. \quad (2)$$

The greater the quality value, the better the phenotypes and the more informative the genes are.

Problem statement. Given a gene expression matrix M with m samples and n genes, and the number of phenotypes k, the **problem of mining phenotypes and informative genes** is to find a k partition of the samples as the phenotypes and a subset of genes as informative genes such that the quality measure (Ω) is maximized.

4. ALGORITHMS

In this section, we will develop two methods. The first one is a heuristic searching algorithm adopting the *simulated annealing technique* [10]. Moreover, we will propose a novel mutual reinforcing adjustment algorithm to approximate the best solution.

Both algorithms maintain a set of genes as the candidates of informative genes and a partition of samples as the candidates of phenotypes. The best quality will be approached by iteratively adjusting the candidate sets.

Both algorithms maintain two basic elements, a *state* and the corresponding *adjustments*. The **state** of the algorithm describes the following items:

- A partition of samples $\{S_1, S_2, \ldots, S_k\}$.
- A set of genes $G' \subseteq G$.
- The quality Ω of the state calculated based on the partition $\{S_1, S_2, \ldots, S_k\}$ on G'.

An **adjustment** of a state is one of the following.

- For a gene $g_i \notin G'$, *insert* g_i into G';
- For a gene $g_i \in G'$, *remove* g_i from G';
- For a sample s in S', *move* s_i to S'' where $S' \neq S''$.

To measure the effect of an adjustment to a state, we calculate the **quality gain** of the adjustment as the change of the quality, i.e., $\Delta\Omega = \Omega' - \Omega$, where Ω and Ω' are the quality of the states before and after the adjustment, respectively.

Now, the problem becomes, given a starting state, we try to apply a series of adjustments to reach a state such that the accumulated quality gain is maximized. Both algorithms record the **best state**, in which the highest quality so far is achieved.

4.1 A Heuristic Searching Algorithm

An immediate solution (shown in Figure 2) to the problem is to start from a random state and iteratively conduct adjustments to approach the optimal state.

The algorithm has two phases: *initialization phase* and *iterative adjusting phase*. In the initialization phase, an initial state st_0 is generated randomly and the corresponding quality value, Ω_0, is computed.

In the iterative adjusting phase, during each iteration, genes and samples are examined one by one. The adjustment to a gene or sample will be conducted if the quality gain $\Delta\Omega$ is positive. Otherwise, the adjustment will be conducted with a probability $p = e^{\frac{\Delta\Omega}{\Omega \times T(i)}}$, where $T(i)$ is a decreasing simulated annealing function [10] and i is the iteration number.

The probability function p has two components. The first part, $\frac{\Delta\Omega}{\Omega}$, considers the quality gain in proportion. The more Ω reduces, the less probability the adjustment will be performed. The second part, $T(i)$, is a decreasing simulated annealing function where i is the iteration number. In our implementation, we set $T(0) = 1$, and $T(i) = \frac{1}{1+i}$.

Algorithm 1 (Heuristic Searching)
Initialization phase:
adopt a random initialization and calculate the quality Ω_0
Iterative adjusting phase:
1) list a sequence of genes and samples randomly;
for each gene or sample along the sequence, do
1.1) if the entity is a gene,
compute $\Delta\Omega$ for the possible insert/remove;
else if the entity is a sample,
compute $\Delta\Omega$ for the largest quality gain move;
1.2) if $\Delta\Omega \geq 0$, then conduct the adjustment;
else if $\Delta\Omega < 0$, then conduct the adjustment
with probability $p = \exp(\frac{\Delta\Omega}{\Omega \times T(i)})$;
2) goto 1), until no positive adjustment can be conducted;
3) output the best state;

Figure 2: The heuristic search algorithm.

The heuristic algorithm is sensitive to the order of genes and sample adjustments considered in each iteration. To give every gene or sample a fair chance, all possible adjustments are sorted randomly at the beginning of each iteration.

We set the termination criterion as *whenever in an iteration, no positive adjustment is conducted*. Once the iteration stops, the partition of samples and the candidate gene set in the best state will be output.

4.2 The Mutual Reinforcing Adjustment Algorithm

In the iteration phase of the heuristic searching algorithm, samples and genes are examined and adjusted with equal chances. However, since the number of samples is far less than the number of genes, each sample should play a crucial role during the adjustment process. As they are treated equally with all genes, the samples thus have less chances to be adjusted. In addition, because the number of samples is quite small, even one or two noise or outlier samples may highly interfere the quality and the adjustment decisions. The heuristic approach cannot detect or eliminate the influence of noise or outliers in the samples effectively. Thus, we propose here a more robust approach called *mutual reinforcing adjustment algorithm*. The general idea is that we adopt a deterministic, noise-insensitive method to adjust samples.

The algorithm is shown in Figure 3. Details of the algorithm will be discussed in the following subsections.

4.2.1 Partitioning the matrix

The first step is to divide the complete set of samples and the set of candidate informative genes into some smaller groups. At the beginning of the first iteration, the set of candidate informative genes contains all the genes.

The algorithm CAST (for cluster affinity search technique) [3] is applied to group genes and samples and the *Pearson's Correlation Coefficient* is chosen to calculate the similarity matrix.[1] CAST is a method specially designed for grouping gene expression data based on their pattern similarities. Thus the entities belonging to the same group should have similar expression patterns while the

[1]The correlation coefficient between two vectors $X = (x_1, x_2, \ldots x_n)$ and $Y = (y_1, y_2, \ldots y_n)$ is $\rho_{X,Y} = \frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\overline{y})^2}}$.

Algorithm 2 (Mutual Reinforcing Adjustment)
start from $G' = G$, do the following iteration:
Iteration phase:
1) partitioning the matrix
1.1) group samples S into m_s ($m_s > k$) groups;
1.2) group genes in G' into n_g groups.
2) identifying the reference partition
2.1) compute *reference degree* for each sample groups;
2.2) select k groups from m_s groups of samples;
2.3) do partition adjustment.
3) adjusting the genes
3.1) compute Ω for *reference partition* on G';
3.2) perform possible adjustment of each genes
3.3) update G', go to step 1.
Until no positive adjustment can be conducted.
Refinement phase:
find the highest *pattern quality state*.
do *state* refinement.

Figure 3: The mutual reinforcing adjustment algorithm.

different groups should have different, well separated patterns. A small amount of outliers will be filtered out without being assigned to any groups. One advantage of CAST is that the number of groups does not need to be pre-specified. Instead, a threshold has to be set to approximate the size of each group. In biological applications, the experiences show that genes associated with similar functions always involve from several dozen to several hundred entities [8]. Thus, when grouping genes, the threshold should be set so that a majority of groups will contain several dozen to several hundred genes. For samples, the threshold can be chosen so that the number of groups ranges within $k \sim 2k$ to maintain a good compromise between the number of clusters and the separation among them.

4.2.2 Reference partition detection

Suppose that, after partitioning the matrix, we have m_s ($m_s > k$) exclusive groups of samples and n_g groups of genes in set G' of candidate informative genes.

Among the m_s sample groups, k groups of them will be selected to "represent" the phenotypes of the samples. The set of representatives is called a reference partition. The reference partition is selected among the sample groups which have small intra-phenotype consistency value (i.e., being highly consistent) and large inter-phenotype divergency among each other. The purpose of selecting such a reference partition is to approximate the phenotypes as closely as possible and to eliminate noise interference caused by outlier samples.

A **reference degree** is defined for each sample group S_j by accumulating its intra-phenotype consistency over the n_g gene groups generated from the last step. This reference degree measures the likelihood of a given sample group to be included into the reference partition. That is,

$$Ref(S_j) = \log|S_j| \sum_{G_i \subset G'} \frac{1}{Con(G_i, S_j)}. \quad (3)$$

A high reference degree indicates that the group of the samples are consistent on most of the genes. Such groups should have a high probability to represent a phenotype of samples.

We first choose a sample group having the highest reference degree, denoted by S_{p_0}. Then, we choose the second sample group S_{p_1} that has the lowest intra-phenotype consistency value and the highest inter-phenotype divergency with respect to S_{p_0} among the remaining groups. When it comes to the third group, the inter-phenotype divergency with respect to both S_{p_0} and S_{p_1} will be considered. Here we define a *selection criterion* for the x-th sample group S_{p_x} by combining its intra-phenotype consistency and inter-phenotype divergency with respect to the sample groups already selected.

$$Ran(S_{p_x}) = \log|S_{p_x}| \sum_{G_i \in G'} \frac{\sum_{t=0}^{x-1} Div(G_i, S_{p_x}, S_{p_t})}{Con(G_i, S_{p_x})}. \quad (4)$$

We will calculate Ran values for the groups which have not been selected. The group having the highest Ran value will be selected as the x-th sample group. When there is a tie, we will choose the group in which the number of samples is the largest. Totally $(k-1)$ sample groups, along with the fist group S_{p_0}, are selected to form the reference partition.

The reference partition selected above is based on the phase partitioning the matrix. Some samples in other groups that may also be good candidates to form the representative partition may be missed. Thus, we need to conduct a "*partition adjustment*" step by adding some other samples that can improve the quality of the reference partition. For each sample s which is not yet included in the reference partition, its probability to be added into the reference partition is determined as follows. We calculate the quality gains for inserting s groups in the reference partition. The group with the highest quality gain is called the *matching group* of s. If the quality gain is positive, then s will be inserted into its matching group; otherwise, s will be inserted into its matching group with a probability $p = e^{(\frac{\Delta\Omega}{\Omega \times T(i)})}$.

4.2.3 Gene adjustment

In this step, the reference partition derived from the last step is used to guide the gene adjustment. Notice that the quality in the mutual reinforcing method is not computed based on the full partition of samples, but on the reference partition and the current candidate gene set G'. Thus the state and the best state maintained by the algorithm are also based on the reference partition.

The gene adjustment process is similar to that in the heuristic searching algorithm. All the genes will be examined one by one in a random order. The possible adjustment is to "remove" genes from G' or to "insert" genes into G' for genes not in the candidate set. The adjustment of each gene will be conducted if the quality gain is positive, or with a probability $p = e^{(\frac{\Delta\Omega}{\Omega \times T(i)})}$ if the quality gain is negative.

After each iteration, the gene candidate set G' is changed. The next iteration starts with the new gene candidate set G' and the complete set of samples. In each iteration, we use the gene candidate set to improve the reference partition, and use the reference partition to improve the gene candidate set. Therefore, it is a mutually reinforcing process.

4.2.4 Refinement phase

The iteration phase terminates when no positive adjustment is conducted in the last iteration. The reference partition corresponding to the best state may not cover all the samples. Therefore, a *refinement phase* is conducted. We first add every sample not covered by the reference partition into its matching group. Thus, the refined reference partition becomes a full partition. It will be output as the phenotypes of the samples. Then, a gene adjustment phase is conducted. We execute all adjustments with positive quality gain. Then the genes in the candidate set of genes G' are output as informative genes.

It can be shown that time complexity of both the heuristic searching and the mutual reinforcing adjustment algorithm is $O(n \cdot m^2 \cdot l)$,

where l is the number of iterations. Comparing to the random adjustments of samples in the heuristic searching approach, the mutual reinforcing method has a more deterministic, more robust and more noise-insensitive adjustment method for samples by improving the reference partition.

5. PERFORMANCE EVALUATION

In this section, we will report an extensive performance evaluation on the effectiveness and efficiency of the proposed two methods using various real-world gene expression data sets. The ground-truths of the partition, which includes the information such as how many samples belong to each class and the class label for each sample, is used only to evaluate the experimental results.

Rand index [12], the measurement of "agreement" between the ground-truths of macroscopic phenotypes of the samples and the partition results, is adopted to evaluate the effectiveness of the algorithm. The Rand index is between 0 and 1. The higher the index value, the better the algorithm performs.

Table 1 shows the data sets and the results obtained by applying our two algorithms and some unsupervised sample clustering algorithms proposed previously. A recently developed effective subspace clustering algorithm, δ-cluster [17], is also included. As shown, the two methods proposed in this paper consistently achieve clearly better mining results than the previously proposed methods. In J-Express[13], CLUTO and SOTA, samples are partitioned based on the complete set of genes. The mining results using these methods suffer from the generally heavy noise in the gene expression data sets. Other approaches adopt some dimensionality reduction techniques, such as the principal component analysis (PCA). However, the principal components in PCA do not necessarily capture the class structure of the data . Therefore the subspace clustering methods may not find the phenotypes and informative genes precisely.

Figure 4 shows the informative genes of the Leukemia-G1 data set detected by the mutual reinforcing adjustment approach. 49 genes are output as the informative genes. In Figure 4, each column represents a sample, while each row corresponds to an informative gene. The description and probe for each gene are also listed. Different colors (grey degree in a black and white printout) in the matrix indicates the different expression levels. Figure 4 shows that the top 22 genes distinguish ALL-AML phenotypes according to "on-off" pattern while the rest 27 genes follow "off-on" pattern.

We also apply two extensively accepted supervised methods, the *neighborhood analysis* [8] and the *statistical regression modeling approach* [15] to mine this data set. Both methods selected top 50 genes to distinguish ALL-AML classes. Among the two top 50-gene sets, 29 genes are overlapped.

The number in the column "match" in Figure 4 shows whether the corresponding gene matches either of these two supervised methods. That is, a 2 here means that this gene is in the top 50 genes selected by both supervised methods, while a 1 means the gene is selected by one of the above two methods. Interestingly, as shown in Figure 4, 41 out of the 49 informative genes identified by the mutual reinforcing adjustment method are either selected by the neighborhood analysis or by the statistical regression modeling approach. This strongly indicates that, even without supervision, the mutual reinforcing method learns well from the real-world data sets.

Table 2 reports the average number of iterations and the response time (in second) of the above gene expression data sets. Each algorithm is executed 50 times with different parameters. The algorithms are implemented with MATLAB package and are executed on SUN Ultra 80 workstation with 450 MHz CPU and 256 MB main memory. The number of iterations are dominated by the simulate annealing function we used. We used a slow simulate annealing function for effectiveness of the approaches. Since in reality, the number of genes in the human genome is about $30,000 \sim 50,000$, efficiency is not a major concern.

	Heuristic Searching		Mutual Reinforcing	
Data Size	# of iterations	runtime	# of iterations	runtime
4132×28	96	63	27	58
4132×30	102	68	27	59
7129×38	116	158	29	125
7129×34	117	155	29	120
2000×62	81	168	23	156
3226×22	94	57	26	52

Table 2: Number of iterations and response time (in second) with respect to the matrix size.

6. CONCLUSIONS

In this paper, we identified the novel problem of mining phenotypes and informative genes simultaneously from gene expression data. A set of statistics-based metrics are proposed to coordinate and compromise both the sample phenotype discovery and the informative genes mining. We proposed two interesting mining methods: the heuristic search and the mutual reinforcing adjustment approach. In particular, the mutual reinforcing adjustment approach incorporates deterministic, robust techniques to dynamically manipulates the relationship between samples and genes while conducting an iterative adjustment to detect the phenotypes and informative genes. We demonstrated the performance of the proposed approaches by extensive experiments on various real-world gene expression data sets. The empirical evaluation shows that our approaches are effective and scalable on mining large real-world data sets. The mining results are consistently with good quality.

7. REFERENCES

[1] Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

[2] Alon U., Barkai N., Notterman D.A., Gish K., Ybarra S., Mack D. and Levine A.J. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide array. *Proc. Natl. Acad. Sci. USA*, Vol. 96(12):6745–6750, June 1999.

[3] Ben-Dor A., Shamir R. and Yakhini Z. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

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

[5] Cheng Y., Church GM. Biclustering of expression data. *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 8:93–103, 2000.

[6] Dysvik B. and Jonassen I. J-Express: exploring gene expression data using Java. *Bioinfomatics*, 17(4):369–370, 2001. Applications Note.

[2]Results are based on hierarchical clustering method.

Data Set	MS_IFN [11]	MS_CON [11]	Leukemia-G1 [8]	Leukemia-G2 [8]	Colon [2]	Breast [9]
Data size	4132×28	4132×30	7129×38	7129×34	2000×62	3226×22
k	2	2	2	2	2	3
J-Express [2]	0.4815	0.4851	0.5092	0.4965	0.4939	0.4112
SOTA	0.4815	0.4920	0.6017	0.4920	0.4939	0.4112
CLUTO	0.4815	0.4828	0.5775	0.4866	0.4966	0.6364
SOM with PCA	0.5238	0.5402	0.5092	0.4920	0.4939	0.5844
Kmeans with PCA	0.4841	0.4851	0.6586	0.4920	0.4966	0.5844
δ-cluster	0.4894	0.4851	0.5007	0.4538	0.4796	0.4719
Heuristic Searching	0.8052	0.6230	0.9761	0.7086	0.6293	0.8638
Mutual Reinforcing	0.8387	0.6513	0.9778	0.7558	0.6827	0.8479

Table 1: Rand Index value reached by applying different methods.

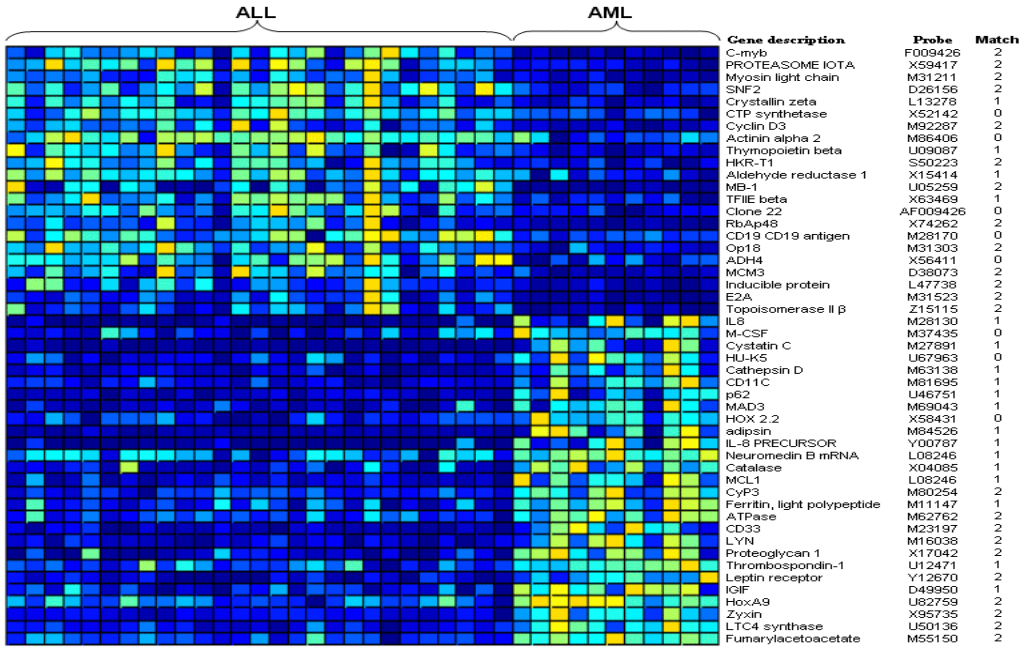

Figure 4: Informative genes detected from Leukemia-G1 (49 genes selected).

[7] Eisen M.B., Spellman P.T., Brown P.O. and Botstein D. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, Vol. 95:14863–14868, 1998.

[8] Golub T.R., Slonim D.K. *et al.* Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, Vol. 286(15):531–537, October 1999.

[9] Hedenfalk, I., Duggan, D., Chen, Y. D., Radmacher, M., Bittner, M., Simon, R., Meltzer, P., Gusterson, B., Esteller, M., Kallioniemi, O. P., Wilfond, B., Borg, A., and Trent, J. Gene-expression profiles in hereditary breast cancer. *The New England Journal of Medicine*, 344(8):539–548, February 2001.

[10] Kirkpatrick, S., Gelatt, C. D. Jr., andVecchi, M. P. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[11] Nguyen LT., Ramanathan M., Munschauer F., Brownscheidle C., Krantz S., Umhauer M., et al. Flow cytometric analysis of in vitro proinflammatory cytokine secretion in peripheral blood from multiple sclerosis patients. *J Clin Immunol*, 19(3):179–185, 1999.

[12] Rand, W.M. Objective criteria for evaluation of clustering methods. *Journal of the American Statistical Association*, 1971.

[13] Rhodes, D.R., Miller, J.C., Haab, B.B., Furge, K.A. CIT: Identification of Differentially Expressed Clusters of Genes from Microarray Data. *Bioinformatics*, 18:205–206, 2001.

[14] Schloegel, Kirk, Karypis, George. *CRPC Parallel Computing Handbook*, chapter Graph Partitioning For High Performance Scientific Simulations. Morgan Kaufmann, 2000.

[15] Thomas J.G., Olson J.M., Tapscott S.J. and Zhao L.P. An Efficient and Robust Statistical Modeling Approach to Discover Differentially Expressed Genes Using Genomic Expression Profiles. *Genome Research*, 11(7):1227–1236, 2001.

[16] Xing E.P. and Karp R.M. Cliff: Clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. *Bioinformatics*, Vol. 17(1):306–315, 2001.

[17] Yang, Jiong, Wang, Wei, Wang, Haixun and Yu, Philip S. δ-cluster: Capturing Subspace Correlation in a Large Data Set. In *Proceedings of 18th International Conference on Data Engineering (ICDE 2002)*, pages 517–528, 2002.