

Lattice based Clustering of Temporal Gene-Expression Matrices

Yang Huang*

Martin Farach-Colton†

Abstract

Individuals show different *cell classes* when they are in the different stages of a disease, have different disease subtypes, or have different response to a treatment or environmental stress. It is important to identify the individuals' cell classes, for example, to decide which disease subtype they have or how they will respond to a certain drug.

In a temporal gene-expression matrix (TGEM) each row represents a time series of expression values of a gene. TGEMs of the same cell class should show similar gene-expression patterns. However, given a set of TGEMs, it can be difficult to classify matrices by cell classes.

In this paper, we develop a tool called LABSTER (LAttice Based cluSTERing) to cluster gene-expression matrices by cell classes. Rather than treating each row or column as a vector, we create a Galois lattice for each matrix, which yields a natural distance function between gene expression matrices. Finally, we cluster based on these distances. A key advantage of our method is that it effectively handles missing values, which is a problem in gene expression data.

We evaluated LABSTER on both simulation data and clinical data. The results show that LABSTER has better clustering performance than several widely used vector-based clustering methods. A bootstrapping procedure is also proposed to further improve the performance of LABSTER. LABSTER has the potential to be used on matrices containing data other than gene expression.

Keywords: gene expression, clustering, Galois lattice, matrix distance.

1 Introduction

Microarray technology, as a powerful tool for monitoring and measuring large-scale gene-transcriptional profiles, leads to the advancement of diagnosis and treatment for numerous diseases. Many microarray experiments are designed to study the temporal patterns in dynamic biological processes by measuring the gene expression of the cells at several distinct time points, thus yielding temporal gene-expression data. Indeed, about a third of microarray studies makes use of time-series experiments [10], for example, for studying the cell cycles of organisms [7, 31], cell development [22], stress response [13] and disease development [25].

In time-series microarray experiments, gene expression data are represented in temporal gene-expression matrices (TGEMs), where rows correspond to genes and columns correspond to time points. Each row represents a time series of expression values of a certain gene.

Individuals often show different *cell classes* when they are on the different development stages of a certain disease, when they have different subtypes of a disease or when they respond differently to a certain stimulus. We are often interested in identifying cell classes by time-series microarray experiments, for example, for deciding the development stage or disease subtype of an individual.

However, due to the experimental cost and time, many previous time-series experiments accumulated limited amount of expression data. They often measured temporal gene expression once for each cell class, resulting one TGEM for every cell class. For example, the common experiment design can be one TGEM for the normal cell and one TGEM for each disease subtype. It can be difficult to distinguish cell classes when only one instance of each class is available.

As the microarray technique becomes more eco-

*Department of Computer Science, Rutgers University, Piscataway, NJ 08854. yahuang@cs.rutgers.edu

†Department of Computer Science, Rutgers University, Piscataway, NJ 08854. farach@cs.rutgers.edu.

nomical and efficient, a new design of time-series microarray experiments is gaining popularity recently. In the new design of experiments, a number of individuals of various cell classes are involved and gene expression is measured for each individual during a time course. In other words, the new experiments measure temporal gene expression multiple times for each cell class, but each time the measurement is performed on different individuals of that cell class. Hence, one TGEM is obtained for one individual and a number of TGEMs are obtained for every cell class.

Many studies have begun to use the new experiment design. For example, to distinguish patients with different responses to the human interferon beta ($rIFN\beta$), Baranzini et al. measured transcriptional profiles of blood cells from many patients at several time points after initiation of therapy [5]. They obtained dozens of TGEMs for patients with good response and patients with poor response. Similarly, a number of TGEMs were obtained for each of several classes of rat liver cells, where cell classes correspond to different doses of a drug [2] or different degrees of environmental pollution [32].

Because the experiments described above provide multiple TGEMs for each cell class, by studying common patterns and dynamics in the matrices we might gain more insights into the complex biological processes behind various cell classes. We expect such experiments will become more important especially in pharmacogenomics, where drug response and disease development are studied in the genomics context.

Ideally, we wish to identify the cell classes from the data accumulated in the above experiments. Thus, given a set of TGEMs, we face the challenge of class discovery, which is to divide those matrices into reproducible classes, each of which represents a cell class.

After reviewing some related work on the analysis of temporal gene-expression data, we will show our method to the problem. Aach and Church developed a time warping algorithm to align temporal gene-expression profiles [1]. Bar-Joseph et al. modeled the time-series data as a cubic spline and performed clustering, aligning and predicting missing values based on spline representation [4]. A func-

tional principle-component analysis was developed to classify individual temporal gene-expression profiles [21]. Bar-Joseph reviewed many more algorithms for temporal gene-expression analysis [3]. However, those algorithms usually focus on mining information from the individual vectors in one TGEM or comparing vectors in several matrices, which makes it difficult to adapt them for our purpose. In [6] the authors modeled the gene-expression profiles with Linear Time Invariant (LTI) system. And then they used a kernel on LTI to classify the gene-expression matrices. Their problem is about class prediction and can be viewed as the complementary to our problem of class discovery, which is often the first step in data analysis.

Riout et al. built a Galois lattice for a gene-expression matrix to take all genes in the same *concept* as the candidates for co-regulated genes [29, 30]. Though we also use the lattice in our method, our goal is different from theirs. We do not make any assumption about co-regulation and are interested in comparing gene-expression matrices. Murali and Kasif searched for the large conserved gene-expression motif in gene-expression data [24]. Though it was not stated in the paper, a motif indeed corresponds to a size-constrained *maximal submatrix*, which we will define later. It is possible that essential biological information is lost when only the largest submatrices are considered in their method. Contrast to their method, we develop a systematic way to consider all maximal submatrices in the gene-expression matrix.

To tackle the problem of class discovery in a set of TGEMs, we propose to cluster the TGEMs and develop a tool called LABSTER (LAttice Based cluSTERing). In our method we develop a distance metric to measure the dissimilarity among TGEMs based on *pattern similarity coefficient*, which we introduce to measure the similarity between two genes' expression. We use the properties of maximal submatrices, in which the two genes coexist, to define the coefficient. To calculate the distance, we construct a Galois lattice for each TGEM efficiently. Once obtaining the pairwise distance among TGEMs, we apply a clustering method to infer the clusters of TGEMs.

Clustering is one of the most widely used methods for analyzing gene-expression data. For a recent review, please refer to [15]. However, as the model in [15] suggests, the objects to be clustered in most previous methods are represented by vectors. In the time-series experiments, clustering vectors only considers one time point at a time. To take the advantage of temporal data, we cluster TGEMs to consider all the time points at the same time. Furthermore, our method has the following advantages: First, for the missing values, a common problem in gene-expression data, our method can handle it in a natural way. Secondly, our method can deal with the noise problem, another problem in gene-expression data, which becomes more serious when a large number of microarrays are used. Thirdly, an effective and efficient distance function is introduced to measure the difference between two TGEMs. The experimental results have shown that LABSTER is able to cluster TGEMs with low error rate and performs better than applying traditional clustering methods on vectors in the matrices.

The outline of our paper is as follows: We introduce the formal problem and Galois lattice in section 2. In section 3, we describe the complete method in detail. In section 4 we show the performance of LABSTER on both simulation data set and clinical data set. We conclude in section 5.

2 Preliminary

2.1 Problem Given N $n \times m$ ($m \ll n$) TGEMs, we want to cluster them so that matrices with high similarity can be put into the same cluster, while matrices in different clusters have more dissimilarity. The TGEMs are obtained by measuring gene expression of n genes at m time points for each of N individuals. It is assumed that the measurement timing is the same for all individuals, the experiment protocol is the same in each measurement and the corresponding rows and columns of different matrices carry the same experiment notation. However, due to noise and experimental handling, some values in some matrices might be missing.

2.2 Introduction to Galois lattices Given a set of *objects* $\mathcal{O} = (g_1, g_2, \dots, g_n)$, and a set of *attributes* $\mathcal{T} = (t_1, t_2, \dots, t_m)$, we use a $n \times m$ binary matrix R to represent a binary relation $\mathcal{I} \subseteq \mathcal{O} \times \mathcal{T}$, i.e. $R_{i,j} = 1$ if $(g_i, t_j) \in \mathcal{I}$ and $R_{i,j} = 0$ otherwise. For $g_i \in \mathcal{O}$, we define a function $f_{\mathcal{O}}(g_i) = \{t_j | R_{i,j} = 1\}$. Furthermore, for an object set $A \subseteq \mathcal{O}$, we denote $f_{\mathcal{O}}(A) = \bigcap_{g_i \in A} f_{\mathcal{O}}(g_i)$. Similarly, we define $f_{\mathcal{T}}(t_j) = \{g_i | R_{i,j} = 1\}$ for $t_j \in \mathcal{T}$ and $f_{\mathcal{T}}(B) = \bigcap_{t_j \in B} f_{\mathcal{T}}(t_j)$ for an attribute set $B \subseteq \mathcal{T}$. With the above notation we are ready to define the concept.

DEFINITION 2.1. *The concept $C \in 2^{\mathcal{O}} \times 2^{\mathcal{T}}$ is a pair (A, B) where $A = f_{\mathcal{T}}(B)$ and $B = f_{\mathcal{O}}(A)$. $A = f_{\mathcal{T}}(f_{\mathcal{O}}(A))$ and $B = f_{\mathcal{O}}(f_{\mathcal{T}}(B))$ are called closed.*

Denoting the set of all concepts to be $\mathcal{B} \subseteq 2^{\mathcal{O}} \times 2^{\mathcal{T}}$, we define a partial order \prec on \mathcal{B} :

$$(A_1, B_1) \prec (A_2, B_2) \iff A_1 \subseteq A_2 (B_2 \subseteq B_1)$$

where $(A_1, B_1), (A_2, B_2) \in \mathcal{B}$.

DEFINITION 2.2. *The ordered set $\mathcal{L} = \langle \mathcal{B}, \prec \rangle$ has the mathematical structure of a complete lattice and is called the Galois lattice or concept lattice.*

When the context is clear, we will refer to a Galois lattice as lattice. The diagram representing an ordered set is called a *Hasse diagram*, where an edge connects two adjacent concepts in the ordered set. We show an example of binary matrix and the Hasse diagram of its corresponding lattice in Figure 1. For more details about the lattice, please refer to [12].

The problem of constructing a lattice from a binary matrix is closely related to enumerating maximal bipartite cliques in a graph and listing closed frequent itemsets in a transaction database [37]. Many algorithms have been developed to solve those three problems. Nourine and Raynaud's algorithm for computing a lattice takes the total time $O(|\mathcal{O}||\mathcal{T}||\mathcal{B}|)$ [26], which is the fastest in theory. The performance comparison among many lattice construction algorithms was studied in [19].

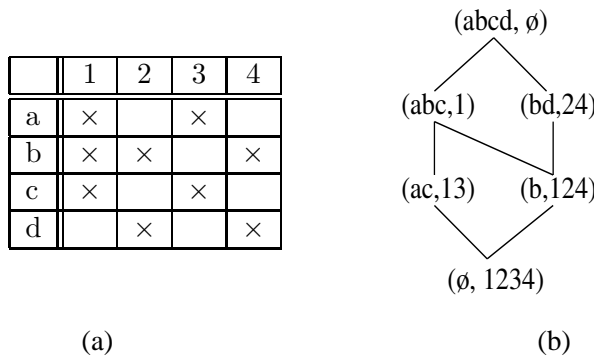


Figure 1: (a) A binary matrix where x represents 1 and space represents 0. (b) The corresponding lattice.

Makino and Uno proposed an algorithm to enumerate all maximal bipartite cliques in $O(\delta^2)$ polynomial delay time where δ is the maximum degree in the graph [23]. Their algorithm is the best known algorithm in enumerating all maximal bipartite cliques. CHARM [36] and CLOSET+ [34] are two state-of-the-art algorithms for mining closed frequent itemsets.

3 Method

We will first define the distance between gene-expression matrices and show how to compute it efficiently. Then we will give an overview of LABSTER. Finally, we propose a bootstrapping procedure for LABSTER.

3.1 Distance Between Gene-Expression Matrices

Traditional clustering methods cluster vectors. In the vector space, the distance metric and other distance functions are well defined [14]. Euclidean distance and Pearson coefficient are two distance functions that are often used in clustering gene-expression data. The Euclidean distance between vectors x_1 and x_2 is $|x_1 - x_2|_2$, the 2-norm of $x_1 - x_2$. Analogously, to compare two gene-expression matrices M_1 and M_2 , we may want to compute $\|M_1 - M_2\|_p$, the p -norm of $M_1 - M_2$. However, such a simple solution often does not work due to high level of noise and complex properties of gene-expression data. We need to define a new matrix distance. The general requirement for distance between TGEMs is that the distance

between the ones of the same cell class should be small and the distance between the ones of different cell classes should be large. In our method, we consider discretized gene-expression matrices. We define a maximal submatrix as follows:

DEFINITION 3.1. A matrix is called constant if each of its entry has the same value. A matrix is called positive if each of its entry contains a positive value. In a matrix of discrete values, the maximal submatrix is a constant submatrix for which no other constant submatrices contain all of its rows and columns. The set of rows (columns) of a maximal submatrix is called maximal as well.

One way to compare two TGEMs is to compare their maximal submatrices, which characterize the matrices. However, in our application, each TGEM contains various levels of noise. A direct comparison of maximal submatrices might fail to yield meaningful distance. We use an alternative approach. Given a discretized TGEM M , we say a set of gene (time points) is maximal if the corresponding set of rows (columns) is maximal in M . Intuitively, two genes show similar expression pattern if they appear in the same maximal gene set. To measure how similar two genes' expression is in M , we define a pattern similarity coefficient for the two genes by looking at how they appear in maximal gene sets. For the gene i and j , let $c_{i,j}$ be their pattern similarity coefficient, let $n_{i,j}$ be the number of maximal gene sets of M which they both belong to, and let s_k be the size of the k th such maximal gene set.

DEFINITION 3.2. The pattern similarity coefficient $c_{i,j}$ for the gene i and j in M is defined to be

$$c_{i,j} = \sum_{k=1}^{n_{i,j}} \sqrt{s_k}.$$

If the gene i and j never appear in the same maximal gene set, $c_{i,j} = 0$. In addition, we define $c_{i,i} = 0$. The pattern similarity coefficient considers not only the number of maximal gene sets to which two genes both belong, but also the size of those gene sets. This is because the significance of a maximal gene set can roughly be assayed by its

size. If two genes are in a large maximal gene set, it means that the similarity between their gene expression is more significant, thus worth more weight. The reason to use $\sqrt{s_k}$ instead of s_k is that a large s_k term may easily dominate the summation. In addition, we find this definition works better than defining $c_{i,j}$ by just counting $n_{i,j}$.

After we have defined pattern similarity coefficients for pairs of genes, a natural definition for the distance between discrete matrices M_1 and M_2 seems to be:

$$d(M_1, M_2) = \sum_{i=1}^n \sum_{j>i}^n |c_{i,j}^1 - c_{i,j}^2|$$

where $c_{i,j}^h$ refers to the pattern similarity coefficient in M_h ($h = 1, 2$) and n is the number of genes. When we compute the distance between two TGEMs of the same cell class, the fact that they should have more gene pairs showing similar patterns, results in more terms in the above distance tend to be zero. Hence, the distance tends to be small. In addition, it can be shown that such distance defines a semi-metric space for discrete matrices. However, with this definition we notice that two matrices with more maximal submatrices tend to have larger distance compared to matrices with less maximal submatrices. So we add a normalization term to reduce that effect. Note that the similarity measurement among databases proposed in [28], which was computed by using association rules, may also be modified and applied to compute the distance between discrete matrices. However, the similarity in [28] can not be used for a set of databases, each of which shares few association rules with others.

DEFINITION 3.3. *Let M_1 and M_2 be two matrices with n genes. The distance between two matrices is*

$$d(M_1, M_2) = \frac{\sum_{i=1}^n \sum_{j>i}^n |c_{i,j}^1 - c_{i,j}^2|}{\sqrt{(\sum_{i=1}^n \sum_{j>i}^n c_{i,j}^1) \times (\sum_{i=1}^n \sum_{j>i}^n c_{i,j}^2)}}$$

where $c_{i,j}^h$ is the pattern similarity coefficient in M_h ($h = 1, 2$).

The main advantages for using the above distance to represent the difference between TGEMs

is that the distance is less affected by noise. First, the pattern similarity coefficient of two genes is defined based on the maximal gene sets, while maximal gene sets are determined by genes' discretized expression values, which are less affected by the noise specific to each microarray. Secondly, instead directly comparing the expression values of two matrices in some way, we use pattern similarity coefficients to compute the matrix distance, which alleviates the problem caused by various noise levels across various microarrays.

3.2 Distance Computation Once we have the families of maximal gene sets of two TGEMs, it is easy to calculate the distance between them. We just need to go through each maximal gene set to compute pattern similarity coefficient for gene pairs and then the distance follows.

It is the lattice that can help us to obtain all maximal gene sets since it has the following property:

PROPERTY 3.1. *Suppose a lattice \mathcal{L} is constructed from a binary matrix R . The set of concepts of \mathcal{L} and the set of maximal positive submatrices of R are in one-to-one correspondence. For each concept (A, B) , A (B) corresponds to a maximal row (column) set in R .*

We then need to transform a discretized matrix to a binary matrix. Given a discretized $n \times m$ TGEM M with discretized levels $[0..d - 1]$, we create a $n \times md$ binary matrix R as follows: If $M_{i,j} = l$, we let $R_{i,d(j-1)+l+1} = 1$ and $R_{i,k} = 0$ for $d(j-1)+1 \leq k \leq dj$ and $k \neq d(j-1)+l+1$. If $M_{i,j}$ is missing, then we let $R_{i,k} = 0$ for $d(j-1)+1 \leq k \leq dj$. Such transformation guarantees that maximal submatrices of M are in one-to-one correspondence with the positive ones of R . Hence, the lattice \mathcal{L} built from such R will have the following property:

PROPERTY 3.2. *The set of concepts of \mathcal{L} and the set of maximal submatrices of M are in one-to-one correspondence. For each concept (A, B) , A corresponds to a maximal gene set.*

As a result, after constructing one lattice for each of two TGEMs, we go through every object

set of each lattice to compute pattern similarity coefficients in each matrix. Then we calculate the distance between the two TGEMs using the coefficients.

Note that the lattice provides a nice way to organize genes and time points for TGEMs. Since a gene can appear in multiple maximal gene sets, hence the maximal gene sets can be overlapped. Conceptually, each concept, containing a maximal gene set and a maximal set of time points, is like a cluster identified in subspace clustering. Subspace clustering is able to identify clusters that exist in multiple, possibly overlapping subspaces [27]. It is well known that in biological processes a certain number of genes perform certain functions only during a certain period, which suggests subspace clustering might provide biological insights into the gene regulation and transcription. Similarly, the lattice might prove useful in studying the TGEM. In addition, a single gene may have multiple biological functions and participate in multiple pathways. So it is important to allow one gene to appear in more than one concept with different sets of genes, as a lattice does.

3.3 LABSTER Overview LABSTER performs the following steps to cluster TGEMs:

1. discretize the set of TGEMs;
2. transform discretized TGEMs into binary matrices;
3. construct a lattice for each binary matrix;
4. build a distance matrix by computing pairwise distance among TGEMs with lattices;
5. cluster TGEMs based on the distance matrix.

For step 1, any discretization method that is suitable for gene-expression data can be applied. For step 2, it can be seen that one advantage of transforming a discretized TGEM into a binary matrix is that it can handle missing values without special efforts. Let us recall the transformation. If a value is missing, 0 will be put into all d entries, which naturally makes the lattice construction algorithm aware that there is a missing value, and hence the algorithm will not consider the gene's behavior at that time point. Neither do we give up certain genes or time points because of missing

values nor we use some complicated method to predict missing values, such as [35].

Next, we apply an improved implementation of the algorithm presented in [8] to the binary matrix to construct a lattice. The implementation employs depth first search (DFS) on the lattice, diffset for intersection [36], fast grouping for union and other techniques to improve the running speed.

The (i, j) entry of the distance matrix in the 4th step is the distance between the TGEM i and j . As for the clustering method used in the last step, any method that works on the distance matrix can be applied, such as most hierarchical clustering methods and some partitional clustering methods.

3.4 Bootstrapping for LABSTER To fit LABSTER in the general framework of combining multiple clustering results and to improve its clustering performance, we develop a bootstrapping procedure based on the one proposed in [9]. We randomly sample r (c) rows (columns) without replacement from each matrix to form a new matrix. Note that it is necessary to perform resampling without replacement because of the following property:

PROPERTY 3.3. *Suppose there are two same rows in the matrix M . Let the matrix be M' after removing one of such row from M . The two lattices, built from M and M' respectively, have the same structure and the same family of maximal column sets.*

In this case, we say M and M' are *equivalent*. If we resample with replacement it is possible that one of the sampled matrix of r rows is equivalent to a matrix of less rows, which makes it contain less than r genes. Thus the problem formulation mentioned in section 2 that all TGEMs should have the same number of genes is not satisfied. Similar argument applies to sampling of columns.

After the resampling, LABSTER is applied to cluster the new set of gene-expression matrices. The procedure is repeated for k times. The dissimilarity between the TGEM i and j is defined to be $1 - u_{i,j}/k$ where $u_{i,j}$ is the number of times the two matrices are clustered together in k clustering results. Thus, the more often two matrices

are clustered together, the smaller their dissimilarity is. We then feed the dissimilarity matrix to a clustering method with the appropriate cluster number. This way we get a new clustering of gene-expression matrices through bootstrapping.

4 Experimental Results

The first step in LABSTER is to discretize the input TGEMs containing real values. The discretization method used in all our experiments is a simple percentile based method: Suppose d discrete levels are preferred. At each time point, i.e. at each column of the matrix, the expression values are sorted. The smallest $1/d$ of them are discretized as 0, the next $1/d$ of them are discretized as 1 and so on. To be able to compute the clustering accuracy easily, we choose to use a partitional clustering method called partitioning around medoids (PAM) [18] in LABSTER. Compared to other partitional clustering methods, PAM accepts a distance matrix and provides silhouette plot to help to select number of clusters. PAM is based on the search of medoids among all data points, to minimize the sum of distance of data points to their closest medoids. In PAM, a medoid is a representative point for a cluster.

We tested LABSTER on two data sets. One is a simulation data set generated according to an extended patient-gene model. The other is a clinical data set where class labels are assigned by doctors. To measure the performance of our clustering result, we define the error rate e , which is derived from the cluster disagreement used in [20]. Suppose we have N data points in K clusters. Let us define a 0-1 membership function $x_{i,v}$. $x_{i,v} = 1$ if the i th data point is in the v th cluster or else it is 0. With such function, e is defined as

$$e = \min_{\pi \in S_K} \left(1 - \frac{1}{N} \sum_{i=1}^N \sum_{v=1}^K x_{i,\pi(v)} t_{i,v} \right),$$

where $t_{i,v}$ is the membership function for the ground truth and $x_{i,v}$ is the membership function for the clustering result obtained by LABSTER. $\pi \in S_K$ denotes a permutation and S_K is the set of all permutations on $[1, \dots, K]$

Though the worst-case running time of constructing a lattice and computing the matrix distance is $O(nmd2^{md})$, where n is the number of

genes, m is the number of time points and d is the number of discretized levels, LABSTER performs well in practice due to certain heuristics. In most cases, it is able to complete the job within minutes. In the real world applications, most of time-series data are short, which means m is small. For example, in Stanford Microarray Database (SMD) $> 80\%$ of time-series datasets contains ≤ 8 time points [11]. We believe that the running time should not be a concern for LABSTER.

4.1 Simulation data set In [17] authors suggested a patient-gene model for modeling temporal gene-expression data from individual patients, each of whom is assumed to belong to one of several classes. There are two kinds of genes: patient-specific genes and class-specific genes. The patient-specific genes are genes each of whose expression forms a unique patient-specific pattern. And class-specific genes are genes whose expression follows a pattern common for patients of that class.

Based on the model, we further assume that genes specific to a certain class can be grouped according to their temporal expression profiles. The expression profile of each patient-specific gene is different from each other, while the profile of the class-specific gene of the same group in patients of the same class is similar to each other. However, due to the noisy nature of gene-expression data and individual genetic variance, even in patients of the same class, the grouping of class-specific genes might be slightly different. Later it will be shown how this can be done.

Consider a class-specific gene of group G in a patient P of class C . We assume the temporal profile of that gene in P should look more like profiles of other genes of G in the same patient than profiles of any gene of G in other patients of class C . To achieve this goal, we introduce in-array noise and across-array noise, where in-array noise has smaller variance and is used to model the difference between profiles of genes of the same group in the same patient, and across-array noise has larger variance and is used to model the difference between profiles of genes of the same group in different patients of the same class.

Summarizing the above ideas, we develop the

following method to generate one simulated TGEM for each of 100 patients. The 100 patients are evenly divided into 5 classes. The expression matrix is set for 100 genes and 8 time points, hence its size is 100×8 .

We first generate a template for each class. Each class template consists of 5 gene groups of size 10, 50 remaining non-grouped genes and one expression profile for each group. The 10 genes of the same group show similar profiles in patients of the same class. To generate a template, we randomly take 50 genes and assign them into 5 groups. For the i th group, a profile of 8 expression values is sampled from normal distribution $N(m_i, 1)$, where $m_1 = -3$, $m_2 = -2$, $m_3 = -1$, $m_4 = 1$ and $m_5 = 2$.

To generate one TGEM for one patient P from the class template, we then perform the following steps: First, we scan each group of the template once. Suppose we are scanning the group G . Every member in G has a probability 0.85 to be selected as a class-specific gene of the corresponding group in the patient P . If after we scan G , j members are selected, then $10 - j$ genes will be randomly picked to fill the group in P from the template's 50 non-grouped genes. This way we make sure that for each group in P , its class-specific gene membership is slightly different from the corresponding group in other patients of the same class. The unselected genes of the template's group will be considered as non-grouped genes. Then, after deciding the membership of 5 groups for P , for the i th group we generate 8 random values as the across-array noise from normal distribution $N(0, 0.25)$, and add the noise to the template's i th expression profile to be the profile of a randomly picked gene g in the group. To generate the profile for each of the other 9 genes in the same group with g , an in-array noise series of 8 values is sampled from normal distribution $N(0, 0.09)$, and added to the profile of g . Finally, we take the remaining 50 non-grouped genes as patient-specific. 8 expression values are sampled from $N(0, 1)$ for each of them. Following the above steps, we generate 20 TGEMs from each of 5 class templates.

Then we discretize the 100 TGEMs with 4 levels and apply LABSTER on them. Interestingly,

when we set the number of clusters to be 5 (corresponding to 5 classes of patients), the error rate $e = 0$, which indicates that LABSTER correctly puts each 20 matrices of patients of the same class in a separate cluster.

4.2 Clinical Data Set Multiple sclerosis (MS) is an inflammatory disease of the central nervous system, which affects more than 1 million people around the world [16]. The exact cause of the disease has not been understood. And there is yet no cure for MS. It has become important in pharmacogenomics to carry out analysis with microarrays to study the disease like MS. Researchers studied the drug response of relapsing-remitting MS patients to recombinant human interferon beta (rIFN β) with the time-series microarray experiment [5]. Since the experiment was performed without housekeeping gene normalization, the gene expression data were obtained by counting RNA content.

The whole data set contains 52 expression matrices, each for one patient. The gene expression is measured for 70 genes at 7 or less different time points after IFN β treatment, which are 0, 3, 6, 9, 12, 18, 24 months. However, only 27 patients' gene expression is measured 7 times. 17 patients missed one test and 8 patients missed two tests. We consider the 70×7 TGEMs of those 27 patients. Among those 27 patients, there are 19 patients with good response and 8 patients with poor response, which corresponds to 2 cell classes. Note that there are missing values in some of the matrices. We do not compute the fold change between time points, which makes one column of data become all 1s, because we want to take the full use of all data. The discretization level is 3 and the number of clusters used in PAM is 2. The error rate of clustering TGEMs by LABSTER is 0.074.

In [5] authors reported clustering samples at each time point did not give satisfactory result according to the response status. But they did not show the error rate of the clustering. To compare with LABSTER, we use PAM and K-means implemented in R [33] to cluster samples at each time point. That is, for each time point we try to cluster the corresponding 27 column vectors

in the matrices.

We use both Euclidean distance and Pearson coefficient to measure the difference among vectors. Besides clustering original column vectors directly, we cluster normalized column vectors as well. The normalization is performed as follows: For V_i^j , the i th element of vector V^j , $j \in [1..27]$, we normalize it as $V_i^j / \max(V_i^1, \dots, V_i^{27})$. Furthermore, to let PAM and K-means make use of all data in a matrix, we transform a gene-expression matrix to a vector by concatenating its column vectors of one particular time point and all genes into one higher dimensional vector. We call the obtained vectors “mega-vectors”. Then we cluster the mega-vectors.

If there is a missing value for a gene, that gene will be discarded from that time point for all 27 vectors. The number of clusters used in all methods is 2. For K-means method, we set the maximum number of iterations to be 100 and choose the lowest error rate as we change the number of initial random centers from 1 to 5. The error rate of all methods is shown in Figure 2. Note that method 3 to 6 and method 11 to 12 are applied at every time point and we have obtained 7 error rates for each of those methods. But in the figure we only present the lowest error rate for each method.

The comparison clearly shows that LABSTER outperforms other vector-based clustering methods with various distance measurements and vector representation. This indicates the advantage of clustering matrices over clustering vectors since we consider all time points together and avoid the curse of dimensionality when we cluster the matrices.

Among patients who missed one test, most of them missed the test at 18 months or 24 months. 4 patients missed the test at 18 months and 6 missed the test at 24 months. We cluster 31 (33) TGEMs of size 70×6 without time point 18 (24) months. The error rate increases a little when only 6 time points are available. Without time point 18 months the error rate of LABSTER is 0.161, and without time point 24 months, the error rate is 0.152.

To test the stability of our method regarding to the number of genes in the matrix, we carry out the following test: We randomly select r genes from 70 genes and extract their corresponding rows

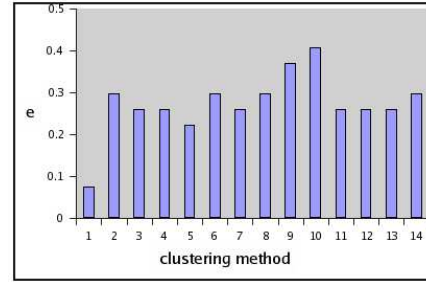


Figure 2: Comparison of error rate of LABSTER and other vector-based clustering methods. Method 1: LABSTER; 2: trivial method which puts all vectors into one cluster; 3(4): PAM with Euclidean distance (Pearson coefficient) on vectors; 5(6): PAM with Euclidean distance (Pearson coefficient) on normalized vectors; 7(8): PAM with Euclidean distance (Pearson coefficient) on mega-vector; 9(10): PAM with Euclidean distance (Pearson coefficient) on normalized mega-vector; 11(12): K-means on (normalized) vectors; 13(14): K-means on (normalized) mega-vectors.

from each matrix to form a new matrix. And then we cluster the new set of smaller matrices by LABSTER. The procedure is repeated for k times. We let r be 30, 40, 50 and 60 and fix k to be 50. The lowest error rates obtained in k runnings for each r are shown with blue bars in Figure 3(a).

Similarly, we test how the number of time points in the matrix affects the performance of LABSTER. Since the number of time points is quite small compared to the number of genes, we are able to generate all possible $\binom{7}{c}$ combinations for c time points. For each combination we extract their corresponding columns from each matrix to form a new matrix and then apply LABSTER on the new set of matrices. We let c be 3, 4, 5, 6. The lowest error rates obtained for each c are shown in Figure 3(b). From the figure, we can see that the error rate does not vary much, which indicates LABSTER is stable and not sensitive to the size of the matrix.

Finally, we test the bootstrapping procedure for LABSTER. We set the number of resampled genes r to be 30, 40, 50 60 and set k to be 50. The error rates for each r are displayed by brown

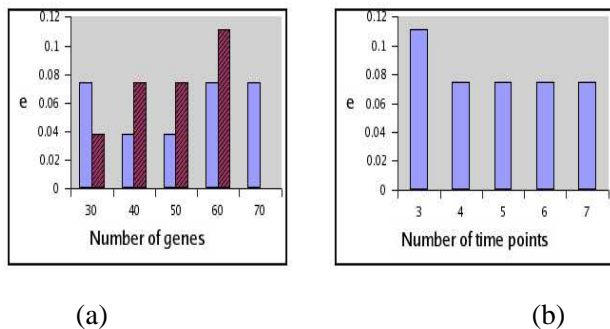


Figure 3: (a) Error rate of LABSTER for various number of genes. At 30, 40, 50, 60 genes, the blue bar on the left indicates the lowest error rate in 50 runnings, and the brown bar on the right indicates the error rate of the bootstrapping procedure. At 70 genes the bar indicates the error rate of LABSTER on the original data; (b) Error rate of LABSTER for various number of time points.

bars in Figure 3(a). It can be seen from the figure that it is possible to further improve the clustering performance of LABSTER by the bootstrapping procedure.

5 Conclusion and future work

As temporal gene-expression data become more widely used, we not only need to cluster gene-expression vectors but also need to cluster gene-expression matrices to identify the cell classes they represent on many occasions, such as in the study of drug response and disease development. In this paper we formulate the problem of clustering TGEMs and develop an efficient tool LABSTER to solve the problem. It first constructs a lattice out of a TGEM, then computes the distance between any two TGEMs based on the pattern similarity coefficient. Finally, it performs clustering on the distance matrix. We test the performance of LABSTER on both simulation and clinical data set. The experimental result is promising, showing that LABSTER has the better performance compared to other popular vector-based clustering methods. In addition, the performance is stable regarding to the size of the gene-expression matrix. Our method can process not only TGEMs, but also

other types of gene-expression matrices. The samples in a matrix can be obtained from different time points, under different conditions or from different tissues.

The future direction of our work includes searching for better matrix distance that is a metric, developing faster lattice construction algorithm, testing other possible representation for the gene-expression matrix and making the full use of temporal correlation among time points. In addition, we will apply our method to matrices containing data other than gene expression.

References

- [1] J. Aach and G. M. Church. Aligning gene expression time series series with time warping algorithms. *Bioinformatics*, 17:495–508, 2001.
- [2] Richard R. Almon, Josephine Chen, Grayson Snyder, Debra C. DuBois, William J. Jusko, and Eric P. Hoffman. In vivo multi-tissue corticosteroid microarray time series available online at public expression profile resource (PEPR). *Pharmacogenomics*, 4:791–799, 2003.
- [3] Ziv Bar-Joseph. Analyzing time series gene expression data. *Bioinformatics*, 20:2493–2503, 2004.
- [4] Ziv Bar-Joseph, Georg K. Gerber, David K. Gifford, Tommi S. Jaakkola, and Itamar Simon. Continuous representations of time series gene expression data. *Journal of Computational Biology*, 10:341–356, 2003.
- [5] S. E. Baranzini, P. Mousavi, J. Rio, S. J. Cailier, and A. Stillman et al. Transcription-based prediction of response to IFN β using supervised computational methods. *PLoS Biology*, 3, 2005.
- [6] Karsten M. Borgwardt, S. V. N. Vishwanathan, and Hans-Peter Kriegel. Class prediction from time series gene expression profiles using dynamical systems kernels. In *Proceedings of the 11th Pacific Symposium on Biocomputing*, pages 547–558, 2006.
- [7] Raymond J. Cho, Mingxia Huang, Michael J. Campbell, Helin Dong, and Lars Steinmetz et al. Transcriptional regulation and function during the human cell cycle. *Nature Genetics*, 27:48–54, 2001.
- [8] Vicky Choi and Yang Huang. Faster algorithms for constructing a Galois lattice, enumerating all maximal bipartite cliques and closed frequent sets. *SIAM Conference on Discrete Mathematics*, 2006.

- [9] Sandrine Dudoit and Jane Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19:1090–1099, 2003.
- [10] Jason Ernst and Ziv Bar-Joseph. STEM: a tool for the analysis of short time series gene expression data. *BMC Bioinformatics*, 7, 2006.
- [11] Jason Ernst, J. Nau, and Ziv Bar-Joseph. Clustering short time series gene expression data. *Bioinformatics*, 21(Suppl. 1):i159–i168, 2005.
- [12] B. Ganter and R. Wille. *Formal concept analysis: Mathematical Foundations*. Springer, Heidelberg, 1999.
- [13] Audrey P. Gasch, Paul T. Spellman, Camilla M. Kao, Orna Carmel-Harel, and Michael B. Eisen et al. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11:4241–4257, 2000.
- [14] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, England, 1990.
- [15] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16:1370–1386, 2004.
- [16] Janet E. Joy and Richard B. Johnston Jr. *Multiple Sclerosis: Current Status and Strategies for the Future*. National Academy Press, Washington, DC, 2001.
- [17] Naftali Kaminski and Ziv Bar-Joseph. A patient-gene model for temporal expression profiles in clinical studies. In *Proceedings of the 10th Conference on Research in Computational Molecular Biology*, pages 69–82, 2006.
- [18] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990.
- [19] S. O. Kuznetsov and S. A. Obedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14:189–216, 2002.
- [20] Timan Lange and Joachim M. Buhmann. Combining partitions by probabilistic label aggregation. In *Proceeding of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 147–156, 2005.
- [21] Xiaoyan Leng and Hans-Georgy Muller. Classification using functional data analysis for temporal gene expression data. *Bioinformatics*, 22:68–76, 2006.
- [22] Susan Magdaleno, Patricia Jensen, Craig L. Brumwell, Anna Seal, and Karen Lehman et al. BGEM: An in situ hybridization database of gene expression in the embryonic and adult mouse nervous system. *PLoS Biology*, 4, 2006.
- [23] K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In *Proceedings of 9th Scand. Workshop on Algorithm Theory*, pages 260–272, 2004.
- [24] T. M. Murali and Simon Kasif. Extracting conserved gene expression motifs from gene expression data. In *Proceedings of the 8th Pacific Symposium of Biocomputing*, pages 77–88, 2003.
- [25] Gerard J. Nau, Joan F. L. Richmond, Ann Schlesinger, Ezra G. Jennings, Eric S. Lander, and Richard A. Young. Human macrophage activation programs induced by bacterial pathogens. *Proceedings of the National Academy of Sciences of USA*, 99:1503–1508, 2002.
- [26] L. Nourine and O. Raynaud. A fast algorithm for building lattices. *Information Processing Letters*, 71:199–204, 1999.
- [27] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: A review. *ACM SIGKDD Explorations Newsletter*, pages 90–105, 2004.
- [28] Srinivasan Parthasarathy and Mitsunori Ogihara. Clustering distributed homogeneous datasets. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 566–574, 2000.
- [29] F. Rioult, J. F. Boulicaut, B. Crémilleux, and J. Besson. Using transposition for pattern discovery from microarray data. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 73–79, 2003.
- [30] F. Rioult, C. Robardet, S. Blachon, B. Crémilleux, O. Gandrillon, and J. F. Boulicaut. Mining concepts from large sage gene expression matrices. In *Proceedings of the 2nd Workshop on Knowledge Discovery in Inductive Databases*, pages 107–118, 2003.
- [31] Paul T. Spellman, Gavin Sherlock, Michael Q. Zhang, Vishwanath R. Iyer, and Kirk Anders et al. Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.
- [32] Yongxi Tan, Leming Shi, Saber M. Hussain, Jun Xu, and Weida Tong et al. Integrating time-course microarray gene expression profiles with cytotoxicity for identification of biomarkers in primary rat hepatocytes exposed to cadmium. *Bioinformatics*, 22:77–87, 2006.
- [33] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R

- Foundation for Statistical Computing, Vienna, Austria, 2006.
- [34] J. Wang, J. Han, and J. Pei. Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 344–353, 2004.
 - [35] Xian Wang, Ao Li, Zhaohui Jiang, and Huanqing Feng. Missing value estimation for DNA microarray gene expression data by support vector regression imputation and orthogonal coding scheme. *BMC Bioinformatics*, 7, 2006.
 - [36] M. J. Zaki and C.-J. Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transaction on Knowledge and Data Engineering*, 17:462–478, 2005.
 - [37] M. J. Zaki and M. Ogihara. Theoretical foundations of association rules. In *Proceedings of 3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 1–7, 1998.