Strategies for Identifying Statistically Significant Dense Regions in Microarray Data

Andy M. Yip Michael K. Ng Edmond H. Wu Tony F. Chan

Abstract

We propose and study the notion of dense regions for the analysis of categorized gene expression data and present some searching algorithms for discovering them. The algorithms can be applied to any categorical data matrices derived from gene expression level matrices. We demonstrate that dense regions are simple but useful and statistically significant patterns that can be used to (i) identify genes and/or samples of interest; (ii) eliminate genes and/or samples corresponding to outliers, noise or abnormalities. Some theoretical studies on the properties of the dense regions are presented which allow us to characterize dense regions into several classes and to derive tailor-made algorithms for different classes of regions. Moreover, an empirical simulation study on the distribution of the size of dense regions is carried out which is then used to assess the significance of dense regions and to derive effective pruning methods to speed up the searching algorithms. Real microarray datasets are employed to test our methods. Comparisons with 6 other well-known clustering algorithms using synthetic and real data are also conducted which confirm the superiority of our methods in discovering dense regions.

Index Terms

Dense region, clustering, categorical data, bicluster, microarray, gene expression, co-expressed genes

I. Introduction

ADVANCES in microarray technologies allow collection of tens of thousands of gene expression data simultaneously within a miniaturized chip. Since the data are of genome-wide scale, they may be used to extract system-level information. Among a variety of data analysis methods, some typical ones include cluster analysis [1], multi-dimensional scaling [2], classification [3], correspondence analysis [4], ANOVA [5] whereas some advanced ones include Bayesian networks [6], differential equations [7], association rule mining [8], gene co-expression networks [9]. These methods may be used to identify subsets of genes responsible for certain specific experimental strains, identify abnormalities, identify functional modules, predict genes' functions, and infer causality relationships when time series data are available.

Gene expression data usually come in the form of quantitative data. They indicate the relative abundance of copies of mRNA in the samples. Depending on the specific type of microarray technology used and the specific task on hand, different transformations and normalization procedures are applied to the raw data before an analysis. The most common ones are perhaps (i) log-transformation of the ratio of the expression levels and a baseline, and (ii) standardization of each gene to zero mean, unit standard deviation across the samples [5].

In addition to the aforementioned transformations, another useful one is the conversion of the continuous expression levels to *categorical data*. Some examples of such a transformation are (i) discretizing the continuous expression levels to a set of discrete values (binning), (ii) grouping the levels into meaningful categories such as over-expressed, under-expressed and unexpressed, and (iii) dichotomizing the levels

A. Yip is with the Department of Mathematics, National University of Singapore, 2 Science Drive 2, Singapore 117543, Singapore. Email: matymha@nus.edu.sg. All correspondence should be addressed to this author.

M. Ng is with the Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong. Email: mng@math.hkbu.edu.hk.

E. Wu is with the Department of Statistics and Actuarial Science, The University of Hong Kong, Pokfulam Road, Hong Kong. Email: hcwu@graduate.hku.hk.

T. Chan is with the Department of Mathematics, University of California, 405 Hilgard Avenue, Los Angeles, CA 90095-1555. Email: chan@math.ucla.edu.

according a binary query, for instance, whether the gene belongs to a certain set of functional categories and the expression level has a certain minimum fold-change. Besides working directly with the expression data matrix, one may also threshold a gene-by-gene correlation matrix to arrive at a matrix of categorical data.

In this paper, we focus on the analysis of categorical data arisen from gene expression data. The reasons for transforming the original expression levels to categorical data are of several folds. First, there are situations where a researcher may only be interested in categories such as (i) whether a gene is over-expressed, under-expressed or unexpressed; (ii) whether a gene is ON or OFF. Thus, the use of categorical variables naturally fits into these situations. Second, some analysis methods require or favor the use of categorical variables. For example, in Bayesian networks [6], each node takes one of the finite number of states; in Naïve Bayes classifiers [10], categorical data are preferred because conditional probabilities can be described using a small contingency table. Third, when an analyst executes a binary query over the gene expression data, some two-category binary data are generated. Fourth, discretization has the effect of denoising the data. It has been reported in [11] that several commonly used classifiers showed an improved prediction accuracy after a discretization of the expression levels due to the denoising effect of the discretization. Interestingly, it has been shown in [12] that the cluster structure of some cancer samples is highly preserved after the dichotomization of expression levels. However, an improper use of discretization may lead to a loss of useful information as far as the actual expression levels are concerned. A thorough discussion of when and how to transform gene expression levels to categorical data is beyond the scope of this paper. We assume that the given data are in the form of categorical data.

We propose the notion of *dense regions*, which are sub-matrices of the categorized expression data matrix having mostly constant values. We also develop a theory which allows us to classify dense regions into three distinct classes. Then, some specialized searching algorithms are proposed for each of these three classes. In the worst case, the number of dense regions in a data matrix can be huge — it grows exponentially with the number of genes and the number of samples. Thus the search of all dense regions becomes an infeasible task. In view of this, we study some statistical properties of dense regions and propose a simple method to assess the statistical significance of dense regions. These results are then used to derive a pruning scheme to speed up the searching process. Our algorithms and theory can be applied to any categorical data matrix. However, in case of microarray data where the number of genes is usually much larger than the number of samples, our algorithms become additionally efficient. This is because the complexity of the algorithms mainly depends on the smaller dimension of the data matrix.

Dense regions are of practical interest. A dense region may correspond to a subset of genes which are co-expressed over a subset of samples. A dense region consists of only one single gene but many samples may correspond to an outlier. Some researchers apply some gene filters to remove uninteresting genes before further analysis. For example, for each gene, if the coefficient of variation is smaller than a threshold, then the gene is removed. In this way, the genes are removed independently. Alternatively, one may first threshold the gene expression data matrix to arrive at a binary matrix and then identify all the genes in the dense regions which correspond to low expression levels and which span over most of the samples as candidates for removal. In this way, the decision of whether a set of genes should be removed or kept is done simultaneously and thus provides more flexibility in the design of gene filters. In summary, dense regions may be used in many different ways, depending on how the expression levels are transformed to categorical data. We refer the readers to [13] for a study of discretization methods.

The outline of this paper is as follows. In §II, we review some related work in cluster analysis. In §III, we define the notion of (perfect) dense regions. In §IV, the characterization of dense regions into different classes and their respective searching algorithms are presented. In §V, we present our main algorithm, DRIFT, and its generalization to μ-dense regions. In §VI, we study the statistical significance of dense regions and propose some pruning methods to speed up the algorithms. In §VII, experimental results are reported on several real and synthetic datasets to demonstrate the usefulness of dense regions in microarray data. Finally, some conclusions are given in §VIII.

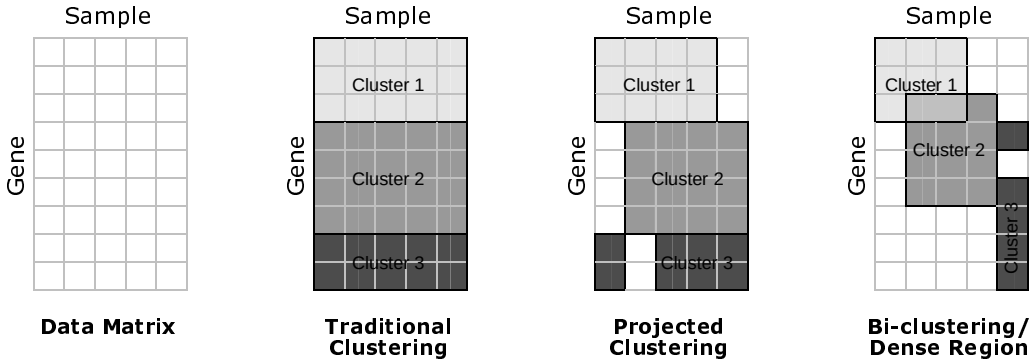

Fig. 1. Schematics of various clustering methods. Traditional clustering of genes: clusters of genes do not overlap, each cluster is defined using all samples. Projected clustering of genes: clusters of genes do not overlap, each cluster is defined using only its own subset of samples. Bi-clustering/dense regions: clusters may overlap, each cluster is defined by a subset of genes and a subset of samples, there is no distinction between clustering of genes and clustering of samples.

II. Related Work in Cluster Analysis

Among the many analysis methods, cluster analysis is closely related to the dense region analysis. It is therefore worthwhile to point out the major differences between the two approaches.

First of all, we use the term "traditional clustering methods" to refer to algorithms which partition the set of genes and the set of samples into subsets independently. For example, k-means and hierarchical methods fall into this category. For other types of methods such as projected clustering [14] and bi-clustering [15], the clustering of the genes and the clustering of the samples are coupled together. See Fig. 1 for a pictorial illustration of different clustering methods.

Most well-known traditional clustering algorithms are designed for quantitative data but there also exists a variety of algorithms for categorical data, for instance, k-modes [16], ROCK [17], STIRR [18] and COOLCAT [19]. The major difference between a traditional cluster and a dense region is that the latter is defined only over a subset of samples when cluster genes and is defined only over a subset of genes when cluster samples. In practice, a subset of genes may co-express only over a subset of samples but behave differently across other samples. In this case, traditional cluster analysis cannot identify such subsets of genes whereas the corresponding subset of genes and subset of samples may constitute a dense region. Another difference is that clusters are not allowed to overlap whereas dense regions do not have such a restriction. In practice, a gene belongs to more than one dense region may involve in multiple pathways.

Projected clustering methods [14] generate clusters of genes where each cluster is defined on its own subspace (subset of samples), c.f. Fig. 1. Thus they allow us to identify genes that are co-expressed only across a subset of samples. These methods are useful in certain applications but they do not allow the clusters of genes to overlap. Dense regions, however, are allowed to overlap which are useful for discovering genes engaged in multiple pathways.

A notion that is very close to dense region is *bi-clustering* [15] which also aims at identifying subsets of rows and subsets of columns simultaneously. A δ-bi-cluster is a sub-matrix defined by a subset of genes R and a subset samples C such that the mean squared residual (MSR)

$$\frac{1}{|R||C|}\sum_{i\in R, j\in C}(x_{ij}-x_{iC}-x_{Rj}+x_{RC})^2$$

is smaller than or equal to δ. Here,

$$\begin{aligned} x_{ij} &= \text{the expression level of the } i\text{-th gene in the } j\text{-th sample} \\ x_{Rj} &= \frac{1}{|R|}\sum_{i\in R} x_{ij} \qquad x_{iC}=\frac{1}{|C|}\sum_{j\in C} x_{ij} \qquad x_{RC}=\frac{1}{|R||C|}\sum_{i\in R, j\in C} x_{ij}. \end{aligned}$$

The MSR is zero if each column is a constant vector or if each row is a constant vector.

Bi-clusters are primarily defined for quantitative data whereas dense regions are defined for categorical data. One cannot use bi-clusters directly for categorical data since the notions of mean (appeared as x_{Rj}, x_{iC} and x_{RC} in the MSR) and distance (appeared as the squared residual $(x_{ij} - x_{iC} - x_{Rj} + x_{RC})^2$ in the MSR) do not make sense anymore. However, we can connect dense regions and bi-clusters in the following way. First, a dense region is a sub-matrix with constant entries (as opposed to constant rows or constant columns). So, the MSR should be replaced by

$$\frac{1}{|R||C|} \sum_{i \in R, j \in C} (x_{ij} - x_{RC})^2. \tag{1}$$

Second, we need to use a metric that is specifically designed for categorical data. To this end, we define a metric $d(x, y)$ by $d(x, y) = 0$ if $x = y$ and $d(x, y) = 1$ if $x \neq y$ where x, y are scalars. Third, the mean x_{RC} in (1) should be replaced by the target category v that the user specifies. As a result, the MSR adapted for categorical data becomes

$$\frac{1}{|R||C|} \sum_{i \in R, j \in C} d(x_{ij}, v)^2. \tag{2}$$

The above formula is nothing but the proportion of the entries in the sub-matrix defined by R and C having their value equal to v. This shows that dense regions can be treated as the analogy of bi-clusters for categorical data.

One advantage of using dense regions in lieu of bi-clusters is that missing values and outlying entries can be handled naturally — they are simply one of the categories. However, bi-clusters are sensitive to outlying entries and cannot handle missing values in a straightforward way.

As we illustrated above, there is a close relationship between bi-clusters and dense regions. A natural question is whether we can use a suitably modified version of the bi-clustering *algorithm* proposed by Cheng and Church in [15] for dense region discovery. We found that if we replace the definition of MSR by the one in (2), then Cheng and Church's algorithm can be used to identify some dense regions in categorical data matrices. However, we found that the resulting algorithm often finds many small-sized dense regions while many large-sized ones are missed (see Experiments 1–3 in §VII). The problem is that during the node deletion phase, many rows and columns which are part of large-sized dense regions are deleted. This is because even if a row or a column is part of a dense region, if its overall percentage of target values is smaller than δ, then it will be deleted. We propose several algorithms which search the data matrix in a more systematic and effective way. These algorithms are built on the top of the theory we develop.

Another related work is given in [20]. It aims at computing a bi-partitioning of a categorical data matrix from a given set of bi-sets (e.g. dense regions). Thus our algorithm can be applied to obtain a set of bi-sets which are then used as an input to the bi-partitioning algorithm.

III. Definition of Dense Regions

We first introduce some notations and give the definition of a dense region. Let $X = [x_{ij}]$ be an n-by-p data matrix where n is the number of rows (usually genes) and p is the number of columns (usually conditions). Let R and C be the index sets of a subset of rows and columns of X respectively. Since we do not distinguish between a matrix and its permuted versions, we assume that R and C are sorted in the ascending manner. A sub-matrix of X formed by the set of rows R and the set of columns C is denoted by $X(R, C)$. We also identify $X(R, C)$ by the combined index set $D = R \times C$. For example, let $R = \{3\}$ and $C = \{1, 2\}$, then $R \times C = X(R, C) = (x_{31} x_{32})$. In the sequel, we assume that all dense regions have a common target value v unless specified otherwise.

Definition 1 (Dense Regions (DRs)): A sub-matrix $X(R, C)$ is called a dense region with respect to v if

(a) $X(R, C)$ is a constant matrix whose entries are v (density), and,
(b) any proper superset of $X(R, C)$ is a non-constant matrix (maximality).

Example 1: Let X be a data matrix given by the left-most matrix below. The DRs with respect to 1 are depicted in the next four matrices.

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 2 & 0 \end{pmatrix} \begin{pmatrix} 1 & * & * & * \\ 1 & * & * & * \\ 1 & * & * & * \\ 1 & * & * & * \end{pmatrix} \begin{pmatrix} 1 & * & * & 1 \\ 1 & * & * & 1 \\ 1 & * & * & 1 \\ * & * & * & * \end{pmatrix} \begin{pmatrix} * & * & * & * \\ 1 & 1 & * & 1 \\ 1 & 1 & * & 1 \\ * & * & * & * \end{pmatrix} \begin{pmatrix} * & * & * & * \\ 1 & 1 & * & * \\ 1 & 1 & * & * \\ 1 & 1 & * & * \end{pmatrix}.$$

Alternatively, we may denote the above dense regions by $\{1, 2, 3, 4\} \times \{1\}$, $\{1, 2, 3\} \times \{1, 4\}$, $\{2, 3\} \times \{1, 2, 4\}$ and $\{2, 3, 4\} \times \{1, 2\}$ respectively. □

IV. Characterizations of Dense Regions

A. The Basic DRIFT Algorithm

The Basic DRIFT algorithm (an abbreviation for **D**ense **R**egion **I**denti**F**ica**T**ion) starts from a given point (s, t) (which contains the target value v) and returns two DRs containing (s, t) where one of them is obtained by a vertical-first-search; the other is by a horizontal-first-search. The algorithm is outlined as follows:

Algorithm: BasicDRIFT(X, s, t)

/* Horizontal-first-search */
$R_v \leftarrow$ the set of row indexes whose corresponding entries in the t-th column equal to $X(s, t)$
$C_v \leftarrow$ the set of column indexes whose corresponding entries in the rows R_v equal to $X(s, t)$

/* Vertical-first-search */
$C_h \leftarrow$ the set of column indexes whose corresponding entries in the s-th row equal to $X(s, t)$
$R_h \leftarrow$ the set of row indexes whose corresponding entries in the columns C_h equal to $X(s, t)$
Return $\{R_v \times C_v, R_h \times C_h\}$

Suppose the two returned DRs have dimensions n_v-by-p_v and n_h-by-p_h respectively. The number of computations required by the algorithm is $n + n_v p + p + p_h n$. Thus, the worst-case complexity of the algorithm is $O(np)$. In applications to microarray data, n_v and n_h are often much smaller than n, whereas p_v and p_h often are much smaller than p. In this case, the complexity is of $O(n + p)$ essentially.

A property of the Basic DRIFT is that the two returned regions are in fact DRs. This justifies the correctness of the algorithm. This fact is stated in the following theorem which is a direct consequence of the construction of the algorithm. The proof is trivial and is omitted.

Theorem 1: The sub-matrices $R_v \times C_v$ and $R_h \times C_h$ obtained from the Basic DRIFT algorithm are dense regions containing the starting point.

B. The Classes $\mathcal{NC}$ and $\mathcal{C}$

In this subsection, we prove that the Basic DRIFT algorithm can discover a class of DRs, $\mathcal{NC}$ (defined as below). We first introduce the notion of an *isolated point*. The basic rationale is to differentiate points that belong to a single region from points that belong to multiple regions. In this way, a DR can be identified by its isolated points (if exist).

Definition 2 (Isolated Points): A point (i, j) in a dense region D is isolated if it is not contained in any other dense regions.

Example 2: The point $(1, 4)$ in Example 1 is an isolated point as it only belongs to the dense region $\{1, 2, 3\} \times \{1, 4\}$. □

Next, we prove an important property which says that an isolated point can be completely characterized by the two regions obtained from the Basic DRIFT algorithm.

Theorem 2: The Basic DRIFT algorithm starting at (s,t) returns two identical dense regions if and only if (s,t) is an isolated point.

Proof: For the "if" part, if the two regions returned by the algorithm are not identical, then the starting point belongs to at least two distinct DRs and thus is not an isolated point. For the "only if" part, suppose that D' is any DR containing (s,t). We would like to show that D' equals to the region $D = R_v \times C_v = R_h \times C_h$ returned by the Basic DRIFT algorithm so that (s,t) belongs to only one DR. Let R' and C' be row and column indices of D' respectively. We note that R_v contains all the entries in the t-th column having the desired value. Thus, $R' \subset R_v$. Similarly, we have $C' \subset C_h$. Hence, we have $D' \subset R_v \times C_h = D$. By maximality of DRs, $D' = D$. □

In the following, we introduce the notions of a *covered dense region* and a *non-covered dense region*. Intuitively, a non-covered DR stands out on its own and is easier to be discovered while a covered one is hidden and is harder to be discovered.

Definition 3 (Covered DRs): A dense region D with respect to v is said to be a covered dense region, if there exists a set of dense regions $\{D_i\}$ with respect to v where $D_i \neq D$ for all i such that $D \subset \cup_i D_i$. Note that $D \neq \cup_i D_i$, otherwise some D_i are not maximal.

Definition 4 (Non-Covered DRs): A dense region that is not covered is called non-covered.

Definition 5 (The Classes $\mathcal{C}$ and $\mathcal{NC}$): For a given X, the class of all covered DRs in X is denoted by $\mathcal{C}$ while the class of all non-covered DRs is denoted by $\mathcal{NC}$.

Example 3: The DR $\{1,2,3,4\}\times\{1\}$ in Example 1 is covered by the union of the DRs $\{1,2,3\}\times\{1,4\}$ and $\{2,3,4\}\times\{1,2\}$. The DR $\{1,2,3\}\times\{1,4\}$ is non-covered for it is not contained in the union of the other 3 DRs. □

The following theorem says that whether a DR is in the class $\mathcal{C}$ or in the class $\mathcal{NC}$ can be exactly characterized by its number of isolated points.

Theorem 3: A dense region is in $\mathcal{NC}$ if and only if it has at least one isolated point.

Proof: If a dense region D has no isolated point, then each point $(i,j) \in D$ is contained in some other dense regions $D_{ij} \neq D$. Consider the set of DRs $\{D_{ij}\}$. We have $D \subset \cup D_{ij}$ and thus D is covered. On the other hand, if D has at least one isolated point, then these isolated points cannot belong to any other dense region. Hence, D is non-covered. □

The impact of the above theorem is that it guarantees that *all* DRs in $\mathcal{NC}$ can be found by starting the Basic DRIFT at every point in X having the target value v. More precisely, a non-covered DR must possess at least one isolated point and can be discovered by starting the Basic DRIFT at any of its isolated points.

C. The Classes $\mathcal{C}_{\mathcal{NR}}$ and $\mathcal{C}_{\mathcal{R}}$

We further divide the class $\mathcal{C}$ into two subclasses $\mathcal{C}_{\mathcal{NR}}$ and $\mathcal{C}_{\mathcal{R}}$ according to their difficulty to be discovered. First, we show a DR which is unidentifiable by the Basic DRIFT algorithm.

Example 4: The data matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & \mathbf{1} & \mathbf{1} & 1 \\ 1 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

contains a DR (in bold face) in $\mathcal{C}$ that cannot be identified by the Basic DRIFT. □

We observe that in Example 4, the DR is surrounded by four isolated points which are identifiable by the Basic DRIFT. If we remove these isolated points (by resetting their values to be any value other than 1), then the DR in bold face becomes non-covered (with respect to the modified data matrix) and is identifiable by another sweep of the Basic DRIFT. Thus the removal of isolated points enlarges the set of identifiable DRs by the Basic DRIFT. Moreover, such a procedure can be done repeatedly which allows

us to find more DRs. The set of DRs in $\mathcal{C}$ that are identifiable by such a repeating process is denoted by $\mathcal{C}_{\mathcal{NR}}$ (*non-recursively covered*).

Definition 6 (The Class $\mathcal{C}_{\mathcal{NR}}$): A covered DR is in $\mathcal{C}_{\mathcal{NR}}$ (the class of non-recursively covered DRs) if it becomes non-covered after a finite number of rounds of removal of isolated points.

Next we show an example where some covered DRs are still unidentifiable by the Basic DRIFT after any finite number of rounds of removal of isolated points.

Example 5: The data matrix

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & \mathbf{1} & \mathbf{1} & 1 \\ 1 & \mathbf{1} & \mathbf{1} & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

contains no isolated point and the DR in bold face cannot be identified by repeatedly applying the Basic DRIFT and deleting isolated points. □

Definition 7 (The Class $\mathcal{C}_{\mathcal{R}}$): A covered DR is in $\mathcal{C}_{\mathcal{R}}$ (the class of recursively covered DRs) if it remains covered after any finite number of rounds of removal of isolated points.

Roughly speaking, a region in $\mathcal{C}_{\mathcal{NR}}$ is covered by a set of regions in $\mathcal{NC}$. Thus it is easy to remove points from these covering regions to make the region non-covered. However, a region in $\mathcal{C}_{\mathcal{R}}$ and its covering regions are entangled with each other which makes it difficult to isolate such a region.

D. The Extended Basic DRIFT Algorithm

Since DRs in the class $\mathcal{C}_{\mathcal{R}}$ are not guaranteed to be found by the Basic DRIFT algorithm, we now introduce an extended version called Extended Basic DRIFT (depicted below) to discover DRs in $\mathcal{C}_{\mathcal{R}}$. The idea is that after finding the sets R_v and C_h as in the original Basic DRIFT algorithm starting at (s,t), we perform a horizontal search over all possible subsets of R_v *or* a vertical search over all possible subsets of C_h, depending the size of R_v and C_h. We remark that some regions found in this way may be a proper subset of a DR and hence violate the maximality condition in Definition 1 — we regard such regions as illegitimate. We allow the user to provide two parameters $MinRow$ and $MinCol$ which specify the minimum number of rows and columns of the output DRs.

Algorithm: ExtendedBasicDRIFT(X, s, t)

Initialize D as an empty list
$R_v \leftarrow$ the set of row indexes whose corresponding entries in the t-th column equal to $X(s,t)$
$C_h \leftarrow$ the set of column indexes whose corresponding entries in the s-th row equal to $X(s,t)$
If $|R_v| < MinRow$ or $|C_h| < MinCol$
  Return D
ElseIf $|R_v| < |C_h|$
  For each subset R'_v of R_v containing $\{s\}$ and satisfying $|R'_v| \geq MinRow$ **do**
    $C'_v \leftarrow$ the set of column indexes whose corresponding entries in R'_v equal to $X(s,t)$
    Insert $R'_v \times C'_v$ into D if it is a legitimate dense region
  EndFor
Else
  For each subset C'_h of C_h containing $\{t\}$ and satisfying $|C'_h| \geq MinCol$ **do**
    $R'_h \leftarrow$ the set of row indexes whose corresponding entries in C'_h equal to $X(s,t)$
    Insert $R'_h \times C'_h$ into D if it is a legitimate dense region
  EndFor
EndIf
Return D

Theoretically, the Extended Basic DRIFT algorithm is capable of finding all DRs. However, it requires more computations than the Basic DRIFT does. Thus, we want to restrict its use as much as possible. The question now becomes how to combine the two algorithms in an effective way.

We remark that in the *worst case*, the number of DRs can grow exponentially with the matrix size. Indeed, the n-by-n matrix with all ones except for its main diagonal possesses $(2^n - 2)$ DRs. Hence the Extended Basic DRIFT may have an exponential time complexity in such a worst case. However, by using suitable pruning schemes (presented in Section VI) to reduce the search space, the complexity can be reduced significantly for practical use.

V. The Main Algorithms

A. The DRIFT Algorithm for Perfect Dense Regions

We now present the DRIFT algorithm which combines the Basic DRIFT and the Extended Basic DRIFT algorithms to find all DRs (with size larger than a user-specified threshold).

Algorithm: DRIFT(X, v)

Initialize D as an empty list
Repeat
  Initialize I as an empty list
  For each point (s, t) such that $X(s, t) = v$
    Run BasicDRIFT(X, s, t) to obtain two DRs D_v and D_h
    Insert D_v into D if it is legitimate
    Insert D_h into D if it is legitimate
    Insert (s, t) into I if it is an isolated point (i.e. $D_v = D_h$)
  EndFor
  Set the entries in X corresponding to the list I to be ∞
Until no further isolated point is found (i.e. I is empty)
For each point (s, t) such that $X(s, t) = v$
  Run ExtendedBasicDRIFT(X, s, t)
  Insert the legitimate DRs into D
EndFor

The DRIFT algorithm works as follows. The first pass of the repeat-until-loop invokes the Basic DRIFT to find out all the DRs in the class $\mathcal{NC}$. After each pass, all identified isolated points are removed to allow the discovery of the DRs in the class $\mathcal{C}_{\mathcal{NR}}$ in the next pass. When no further isolated point is found, it is guaranteed that all DRs in $\mathcal{NC}$ and $\mathcal{C}_{\mathcal{NR}}$ are discovered. Beginning from the second pass of the repeat-until-loop, where the original matrix X has been modified, a region returned by the Basic DRIFT may be a proper subset of a previously found DR. In this case, such a subset is regarded as "illegitimate" and is not inserted into D. When we check whether a region is legitimate, we need to check it against the original data matrix instead of the modified one.

After the repeat-until-loop, we start the Extended Basic DRIFT at every point in the reduced matrix X, which contains no more isolated point, to find the DRs in $\mathcal{C}_{\mathcal{R}}$.

In the repeat-until-loop, each iteration deletes at least one isolated point or otherwise the iteration terminates. Thus, the number of iterations is at most np. In our experience with real data, the algorithm usually terminates in less than 5 iterations.

In practice, one might want to discard DRs with small size. To do so, one may simply define a DR to be "illegitimate" if its size is below a user-specified threshold $(MinRow, MinCol)$ so that it will not be inserted into the output sequence D. We will discuss in the next section on choosing such a threshold according to the statistical significance of the resulting DRs.

To further reduce the complexity, we may start the Extended Basic DRIFT only at a random sample of points. We found that this strategy is quite effective for finding moderate- and large-sized DRs. This is due to the facts that (i) the probability of sampling a point from a larger DR is higher than that from a smaller DR; (ii) the Extended Basic DRIFT can find all DRs containing the starting point. However, the less significant small-sized regions may be missed. In any case, users are often interested in a small number of "significant" DRs only.

B. Error Tolerant Dense Regions

So far we consider regions with a constant value. In practice, regions with only a majority of constant value may also be interesting. In this subsection, we discuss the notion of μ *dense regions* (imperfect dense regions) where the density property of DRs is relaxed.

Definition 8 (μDRs): A region $R \times C$ is called a μDR with value v if at least a percentage μ of the entries of $R \times C$ take the value v.

Example 6: The vector $(0, 1, 1, 1)$ is a 75%-DR. In fact it is a μDR for any $0 \leq \mu \leq 75\%$. □

We now present an algorithm, called μDRIFT, to deal with μDRs. It is used on the top of DRIFT. Starting with a (perfect) DR $R \times C$ found by the DRIFT, the μDRIFT algorithm repeatedly enlarges the region in a greedy way while the density of the enlarged region is maintained to be at least μ.

Algorithm: μDRIFT(X, v, R, C, μ)

Set $R_\mu \leftarrow R$ and $C_\mu \leftarrow C$
Set $R_\mu^{\text{new}} \leftarrow R$ and $C_\mu^{\text{new}} \leftarrow C$
Repeat
  /* Insert columns whose percentage of v's exceeds μ */
  For each column j not in C_μ
    $Count \leftarrow$ number of v's in $X(R_\mu, \{j\})$
    If $Count \geq \mu |R_\mu|$
      $C_\mu^{\text{new}} \leftarrow \{j\} \cup C_\mu^{\text{new}}$
    EndIf
  EndFor
  $C_\mu \leftarrow C_\mu^{\text{new}}$
  /* Insert rows whose percentage of v's exceeds μ */
  For each row i not in R_μ
    $Count \leftarrow$ number of v's in $X(\{i\}, C_\mu)$
    If $Count \geq \mu |C_\mu|$
      $R_\mu^{\text{new}} \leftarrow \{i\} \cup R_\mu^{\text{new}}$
    EndIf
  EndFor
  $R_\mu \leftarrow R_\mu^{\text{new}}$
Until no further rows and columns are added
Return $R_\mu \times C_\mu$

The value μ signifies the density of the DRs. The larger the μ is set, the closer the resulting DRs to perfect DRs. If it is set to be too large, then a large region with a few non-target values may be broken into pieces. If it is set to be too small, then the resulting DRs will contain many non-target values. In practice, we found that setting the μ between 70% and 90% is a good comprise between the size and the density of the DRs.

VI. Statistical Properties of DRs

In this section, we discuss how to assess statistical significance of the DRs using the distribution of the size of DRs. We also demonstrate how to make use of the distribution to derive effective pruning schemes

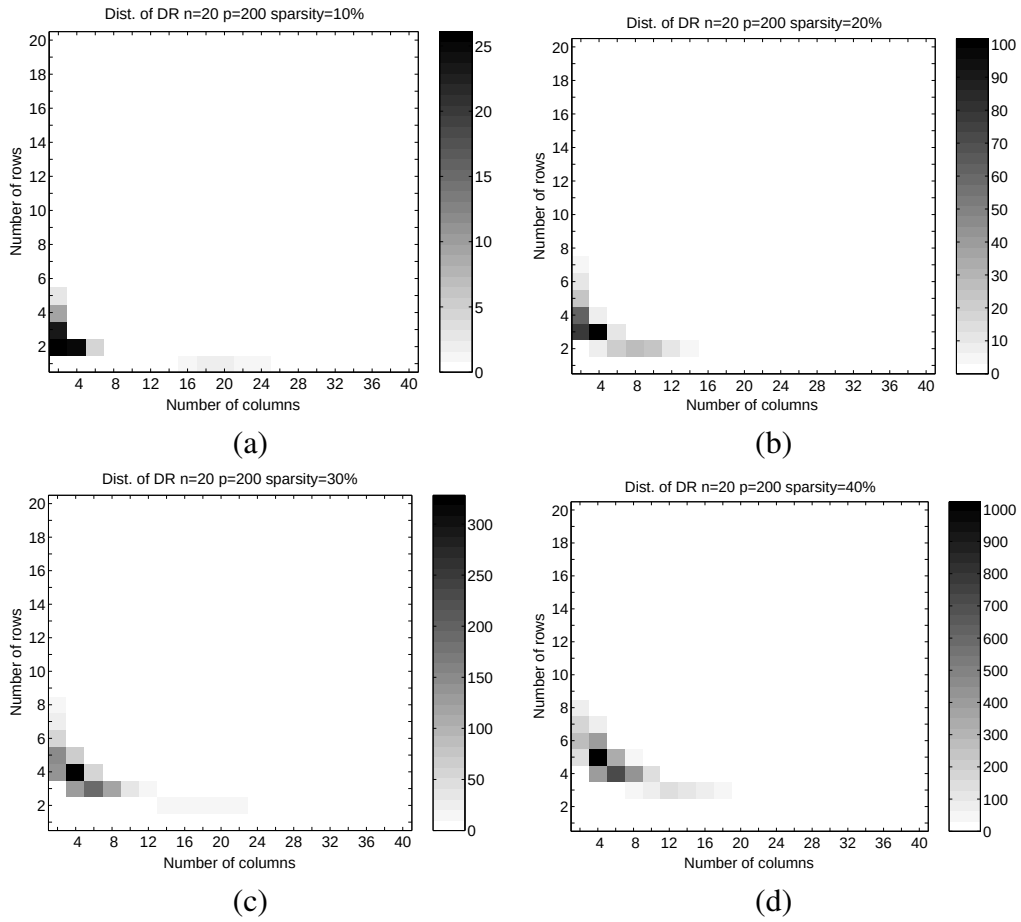

Fig. 2. Frequency distributions of the size of the DRs in random matrices with size 20×200. (a) Sparsity $= 10\%$. (b) Sparsity $= 20\%$. (c) Sparsity $= 30\%$. (d) Sparsity $= 40\%$.

to speed up the algorithms presented in the previous sections.

A. Data for Simulations

We use data matrices of size 20×200. Four kinds of random binary matrices with sparsity (percentage of the number of 1's) 10%, 20%, 30% and 40% respectively are generated. The DRIFT algorithm (with minimum size threshold 1×1) is applied to each of these four kinds of matrices and the size of the identified DRs are recorded. These procedures are repeated 10 times and the counts are averaged. The resulting (average) frequency distributions of the size of the DRs are depicted as heat-maps in Fig. 2.

B. Assessing Statistical Significance of DRs

In dense region discovery, some DRs may correspond merely to random patterns. For example, when the sparsity is s, it is easy to find DRs with size $1 \times sp$ and $sn \times 1$ for the expected number of 1's in a row is sp and the expected number of 1's in a column is sn. Thus, it is essential to identify which DRs are statistically meaningful. To this end, we propose to use the following simple non-parametric method to estimate the p-value [21, p.440] of a given DR $R \times C$ found in X:

$$p\text{-value} = \text{Prob(finding a DR in } X' \text{ with size at least } |R| \times |C| \mid X' \text{ is random).} \tag{3}$$

Here, X' is a random binary matrix whose size and sparsity equal to that of X. If the p-value of a DR is small, say 10^{-3}, then the region is significant. Conversely, if the p-value of a DR is large, say 10^{-1}, then the region is insignificant.

The use of random matrices allows us to assess the size of those DRs which would have been obtained merely by chance. In practice, such a probability may be estimated efficiently from empirical distributions obtained from matrices of moderate size. More precisely, if the given data matrix has a size $n \times p$, then an integer m ($m > 1$) is chosen and a random matrix of moderate size $(n/m) \times (p/m)$, whose sparsity of the target value equals to that of the original matrix, is generated. Next, we compute the distribution of the size of its DRs from which the p-values can be obtained by summing all probabilities of the DRs with a minimum size. Finally, the p-value of a DR with size $|R| \times |C|$ in the original data matrix is approximated by the p-value of the DR with size $(|R|/m) \times (|C|/m)$ in the smaller matrix used to compute the distribution.

Example 7: The significance levels of a data matrix of size 20×200 and sparsity 20% can be obtained from its frequency distribution of the size of the DRs (c.f. Fig. 2):

p-value	$\|C\| = 3$	$\|C\| = 5$	$\|C\| = 7$
$\|R\| = 1$	0.5368	0.2517	0.1760
$\|R\| = 2$	0.5247	0.2396	0.1639
$\|R\| = 3$	0.2983	0.0304	0.0017

These values are obtained by summing the (normalized) frequencies corresponding to all DRs with size larger than or equal to a threshold, c.f. Formula (3). □

C. Pruning Schemes

A computational obstacle of the DRIFT algorithm is that the problems of finding all DRs and the finding the DR(s) with the largest size are NP-hard. Such a problem is also encountered in bi-clustering [15]. A common approach is to fix the number of DRs to be found to a small number. However, the resulting DRs may be of very small size and may not be of interest at all. We propose to use the statistical significance of the DRs to reduce the number of DRs to a manageable size. For instance, we may restrict our attention to the top 5% most significant DRs. By the definition of the p-value in Formula (3), this corresponds to requiring the DRs to have a minimum size. Thus, one may lookup the distribution of the size of DRs and choose a minimum size threshold at the 5% significance level.

Example 8: To obtain the threshold for a data matrix of size 40×400 and sparsity 20% at the significance level of 0.05, we lookup the table in Example 7 and find the largest p-value that is smaller than 0.05 (0.0304 in this case). The value 0.0304 in the table corresponds to the threshold 3×5. Since the matrix used in Example 7 has a size of 20×200, we need to compensate the threshold by a factor of 2. This gives the threshold $|R| \times |C| = (3 \cdot 2) \times (5 \cdot 2)$. It means that the probability of finding of a DR with size at least 6×10 in a random matrix is smaller than 0.05. Thus the DRs found by the DRIFT with the threshold 6×10 are far from random occurrences, assuming that the data matrix has a size of 40×400 and a sparsity of 20%. □

VII. Experimental Results

A. Primate Dataset

We employ the dataset consisting of gene expression measurements of 23 primate brain samples (7 human, 8 chimpanzees, 8 Rhesus macaques) studied by Cáceres *et al*. in [22]. Oligonucleotide microarrays were used to measure expression levels of $\sim$10000 genes simultaneously. The purpose of the study was to explain the phenotypic differences between human and chimpanzees at the level of gene regulation using macaques as an outgroup, despite the fact that the two species have $\sim 99\%$ of their DNA sequences in common.

For illustration purposes, a subset of 376 genes is selected based on the coefficient of variation and the percentage of present calls generated by the dChip 1.3 software [23]. Next, the model-based expression indices are calculated and all replicates are pooled resulting in a dataset with 13 samples and 376 genes.

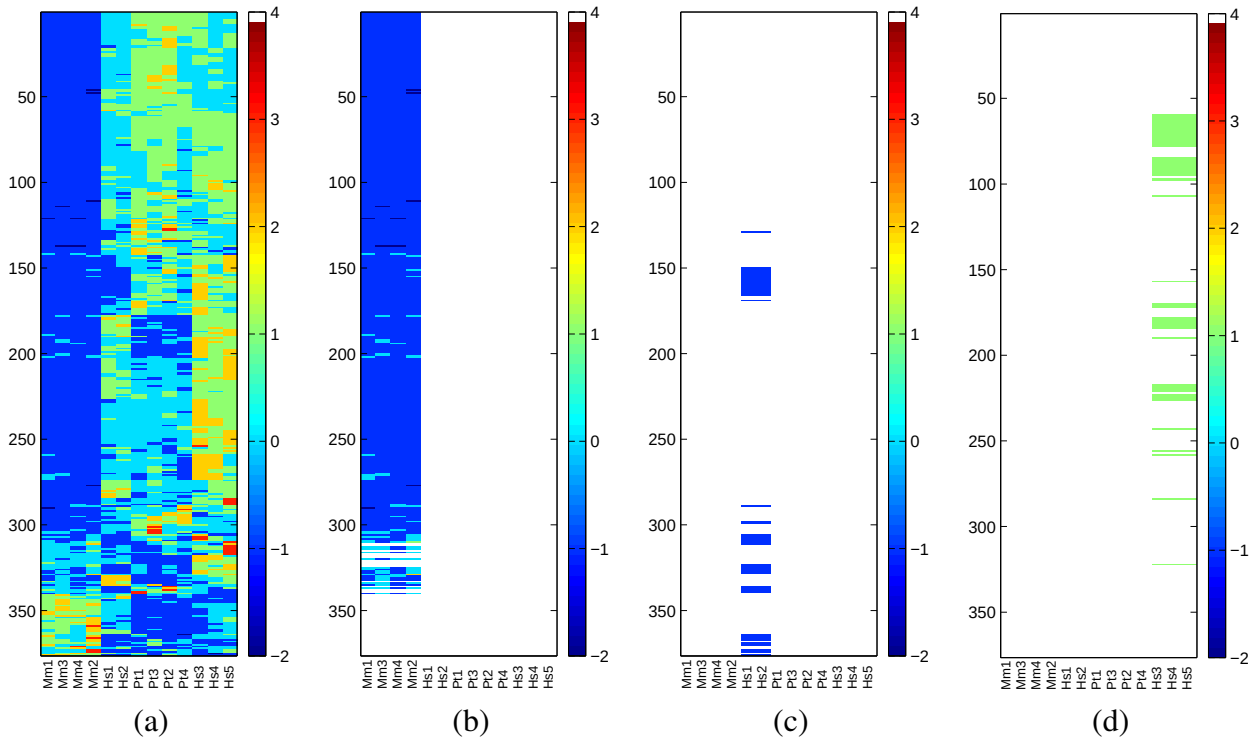

Fig. 3. (a) Heat map of the expression level of 376 genes and 13 samples. The genes and samples are reordered according the results of the average linkage hierarchical clustering. (b) A 90%-DR with value -1 (326 genes, 4 samples). (c) A 90%-DR with value -1 (47 genes, 2 samples). (d) A 90%-DR with value 1 (59 genes, 3 samples).

Each gene is then normalized to have mean 0 and standard deviation 1 across the samples. Finally, the values are rounded off to integers.

We apply our algorithms to find DRs in the dataset. The results are shown in Fig. 3(a–d). The heat map of the dataset is shown in (a) where the genes and the samples are ordered according to the results of the average linkage hierarchical clustering [24]. Three sample DRs identified by μDRIFT are illustrated in (b–d). Moreover, we apply the annotation tool DAVID [25] to find the functional categories of the genes in each DR.

The region in (b) suggests that most of the genes under the study are down-regulated in the macaque brain samples (Mm1–Mm4). Among the 326 genes in this region, 130 (39.9%) of them are involved metabolism and 96 (29.4%) in cellular physiological process.

The region in (c) mostly consists of genes that are down-regulated in the two human samples (Hs1–Hs2) but not in other human samples (Hs3–Hs5). The samples Hs1 and Hs2 differ from the other three human samples by (i) they had longer postmortem intervals ($\sim$13 hrs) so that the degradation of the RNA samples may have been more pronounced; (ii) they were collected from a different region (the frontal pole) which may show a different pattern of gene expression than samples collected from other regions. If case (i) is the major reason that causes the deviation of Hs1 and Hs2 from Hs3–Hs5, then one may want to discard Hs1 and Hs2 before any further analysis. Thus, it will be helpful to look at the functional categories of the genes in this region. Indeed, 15 genes (31.9%) are involved metabolism while 14 genes (29.8%) in cellular physiological process.

The region in (d) consists of genes that are consistently up-regulated in the human samples (Hs3–Hs5) but not in other samples. This gives a list of candidate genes to analyze the difference between human and chimpanzees while reducing the effects (i) and (ii) in Hs1 and Hs2 mentioned above. In this region, out of the 59 genes, 30 (50.8%) of them are involved in metabolism and 17 (28.8%) in cellular physiological process.

B. Ageing Human Brain Dataset

This dataset has been reported in [26]. It contains the gene expression levels of 30 human frontal cortex samples. The measurements were done using microarrays with 12,625 probe sets. In [26], a subset of 463 genes have been reported to be age-regulated (i.e. the expression levels correlate significantly with age). These 463 genes were identified by performing a 2-sample comparison where the "young group" contains individuals with age ≤ 42 and the "old group" contains individuals with age ≥ 73. In our tests, we try to refine the findings by applying the DRIFT algorithm on these 463 genes across the 30 samples. We also use the annotation tool DAVID [25] to find the functional categories of the genes in each DR.

The dataset was first normalized so that for each gene the mean and the standard derivation of the levels across the samples are 0 and 1 respectively. Then we discretize the levels by rounding them off to integers. The discretized data are displayed as a heat map in Fig. 4(a). The rows (genes) and the columns (samples) are independently reordered according to the results of the average linkage hierarchical clustering [24]. The samples are labeled according the age and the sex of the individuals. In the figure, we see that there are roughly two large clusters of genes: one cluster contains genes that are mostly up-regulated among the older individuals but down-regulated among the younger individuals; another cluster is the other way around. This is simply because the genes used here are differentially expressed among the two age groups.

Two identified dense regions (80%-DRs) corresponding to the expression levels -2 and 2 are shown in Fig. 4(b). For the Region A in Fig. 4(b), all the 3 samples (M73, M95 and F106) belong to the "old group" and the genes are consistently down-regulated in these samples. Among the 23 genes in the DR, 15 of them have been annotated, 7 of them are involved in "ion transport" and 5 of them in "phosphorylation". For the Region B in Fig. 3(b), all the 3 samples (M26, M26B and F27) belong to the "young group". Indeed, these three individuals are the youngest among all 30 individuals. There are 8 genes in this DR and they are consistently down-regulated. 5 of them are annotated, 2 of them are involved in "cell cycle".

C. Yeast Cell Cycle Dataset

Our next dataset is the microarray recordings of gene expression levels during yeast's cell cycles [27]. The original data consist of 6178 genes over 77 conditions. We first remove those conditions with more than 5% missing values, resulting in 44 conditions. The 1000 most varying genes (measured by variance) are chosen for our analysis. Each gene is then normalized to have mean 0 and standard deviation 1 across the conditions. Finally, the values are rounded off to integers.

In this experiment, we first transform the expression data into a binary matrix by thresholding the absolute expression levels at the value of 3. Thus the DRs with value 1 are the subsets of genes and conditions where the expression levels are either larger than or equal to 3 or smaller than or equal to -3. This corresponds to finding genes that are simultaneously over-expressed or under-expressed across some conditions.

By requiring the DRs to have a size at least 6×6, we obtain 5 (perfect) DRs. The expression profiles (before rounding off to integers) of DRs are shown as parallel plots in Fig. 5. In each DR, the profiles of the genes are very coherent.

D. Comparisons with Traditional and Projected Clustering Methods

To confirm that DRs cannot be effectively discovered by clustering algorithms, we compare the performance of DRIFT with 6 clustering algorithms, k-means [24], k-modes [16], average linkage hierarchical clustering [24], CAST [28], HARP [14] (projected clustering), and Cheng and Church's bi-clustering [15] on two synthetic datasets with known DRs and one real dataset.

Experiment 1. In this experiment, we consider the scenario that the data matrix contains several DRs and that the set of rows (genes) of each DR is the pattern that we are interested. Our goal is to discover all of these subsets of genes. To simulate this, we first construct a zero matrix of size 200-by-50. Then 10 random DRs with value 1 are embedded into the matrix. These 10 DRs have size ranging from 10-by-5 to

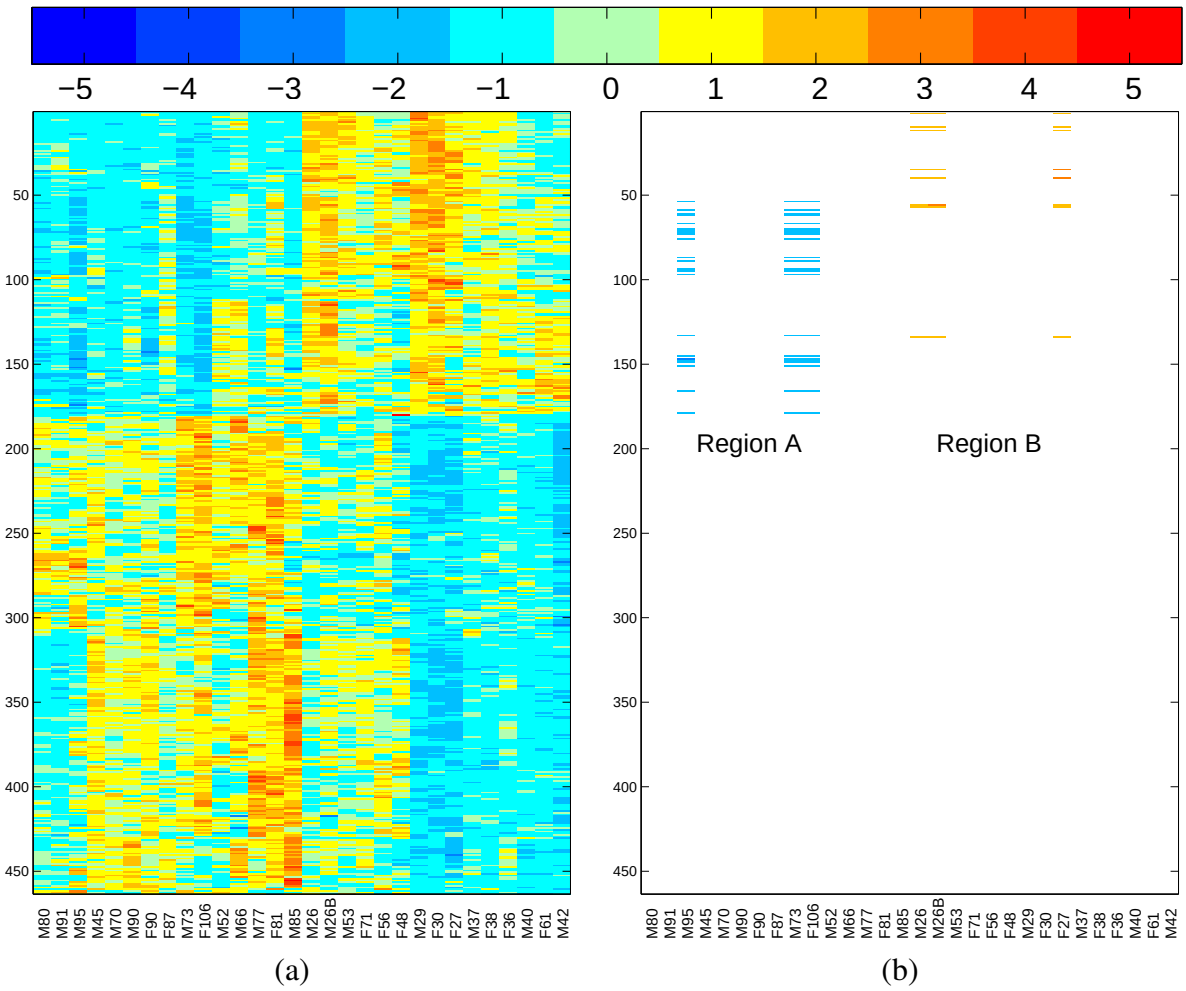

Fig. 4. (a) Heat map of the expression level of 463 genes and 30 samples. The genes and samples are reordered using the average linkage hierarchical clustering. (b) Region A: An 80%-DR with value -2 (23 genes, 3 samples); Region B: An 80%-DR with value 2 (8 genes, 3 samples).

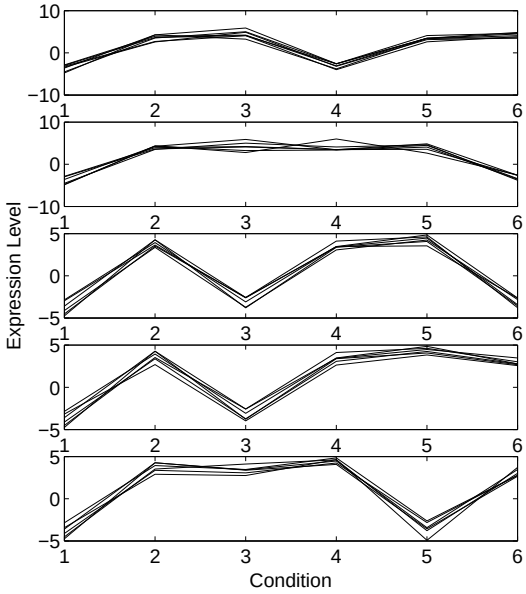

Fig. 5. Expression profiles of 5 identified DRs. The DRs are defined by the subsets of genes and subsets of conditions whose corresponding absolute expression levels are at least 3.

30-by-10. They may overlap with each other. However, due to the relatively sparse structure of the data matrices, the DRs only overlap to a small extent. The matrix is then input to the DRIFT to identify the DRs and to the 6 clustering algorithms to cluster the rows.

Finally, we calculate the *discovery rate* for each algorithm as follows. Let $T_1, \ldots, T_{10}$ be the true clusters of genes that we embedded into the data matrix. Let $C_1, \ldots, C_k$ be the set of clusters of genes discovered by one of the algorithms. The relative error between T_i and C_j is defined to be

$$\mathrm{err}(T_i, C_j) := \frac{|(T_i \setminus C_j) \cup (C_j \setminus T_i)|}{|T_i \cup C_j|}$$

where $|\cdot|$ denotes the cardinality of its argument. The discovery rate of T_i is defined to be

$$1 - \min_{j=1,\ldots,k} \{\mathrm{err}(T_i, C_j)\},$$

i.e., using the C_j that best matches T_i. Finally, the overall discovery rate of an algorithm is defined to be the average rate of $T_1, \ldots, T_{10}$:

$$\frac{1}{10} \sum_{i=1}^{10} \left[1 - \min_{j=1,\ldots,k} \{\mathrm{err}(T_i, C_j)\} \right].$$

If all the 10 true clusters are found, then the discovery rate attains it maximum value 1. If the clusters found by an algorithm contains no genes in the true clusters (i.e. $T_i \cap C_j = \emptyset$ for all i, j), then the discovery rate is 0.

Such an experiment is repeated for 10 times where each time different randomly generated DRs are used.

For each algorithm, the parameters are tuned to obtain optimal results. In particular, for the k-means, k-modes, hierarchical, CAST, and HARP algorithms, we first construct k clusters with $k = 6, 7, \ldots, 15$. Then the k which gives the best discovery rate is chosen. For the bi-clustering algorithm, the MSR in (2) is used since the true clusters are defined by constant regions. Moreover, we try $\delta = 0.001, 0.01, 0.1$ and the δ which gives the best discovery rate is chosen.

The boxplots for the 10 runs are shown in Fig. 6(a). As expected, the DRIFT achieves a perfect score since it searches for all DRs. For all the clustering algorithms, we found that some of the overlapped true clusters are split into pieces. Interestingly, the bi-clustering algorithm shows the worst performance. We found that the main reason was that more than half of the bi-clusters contain a majority of 0 entries. Indeed, even if we use the algorithm to find 100 bi-clusters, less than 10 out of the 100 bi-clusters have a majority of 1 entries. This shows that the bi-clustering algorithm is not effective for searching bi-clusters with a specific target value.

Experiment 2. In the previous experiment, the constructed true clusters may be overlapped. Since 5 of the clustering algorithms (with the exception of the bi-clustering algorithm) cannot discover overlapping clusters, we perform a second experiment where the subsets of rows defined by the embedded DRs do not overlap. The random matrices are similarly generated as in Experiment 1, except for (a) the sets of rows of the DRs do not overlap (b) after the 10 DRs are embedded, 20% of the entries with value 0 (i.e. entries that do not belong to any DR) are randomly selected and changed to the value 1. The step in (b) is to avoid all the rows corresponding to a true cluster being identical, which is an uninteresting case.

The resulting discovery rates are shown in Fig. 6(b). On average, the rates of the 4 traditional clustering methods are higher than that in Experiment 1. This is because the true clusters become non-overlapping. For CAST and average linkage, the rate is close to 1. Although HARP uses only a relevant subspace for each cluster, the performance is not as good as the traditional clustering algorithms. For bi-clustering, it often finds very small-sized regions which correspond to noise. We also found that most of the true clusters that cannot be found by the 6 clustering algorithms have a relatively small number of columns

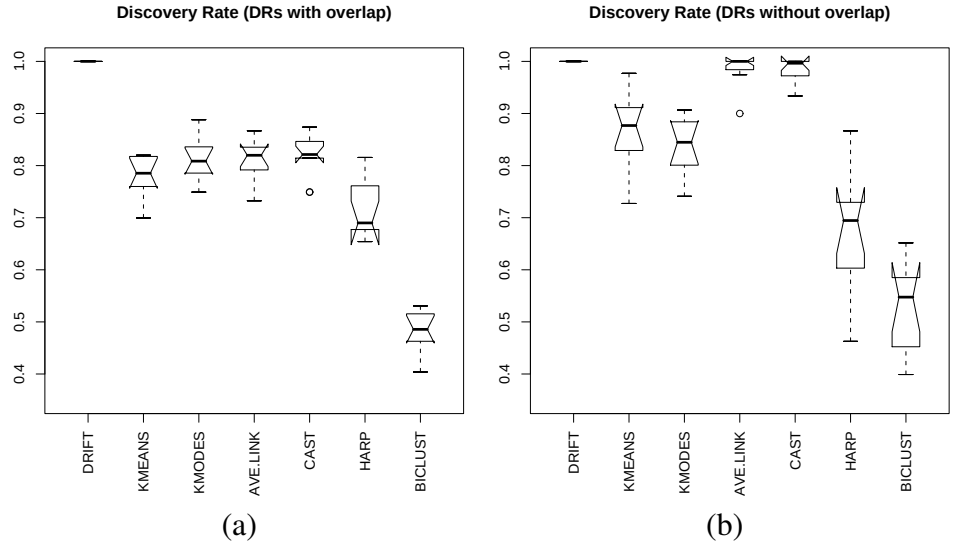

Fig. 6. Boxplots of the discovery rates of the DRIFT, k-means, k-modes, average linkage, CAST, HARP, and bi-clustering algorithms. (a) Discovery rates of 10 different datasets where each dataset contains 10 possibly overlapping DRs. (b) Discovery rates of 10 different datasets where each dataset contains 10 non-overlapping DRs.

(of the corresponding DRs). In this case, clustering using all variables [1] cannot capture at all the genes that are similar only in a subspace.

We note that, however, a subset of genes having very coherent expression profiles may form a cluster, but may not form any DR when the expression levels vary a lot across the samples. In this case, such clusters are better discovered by traditional or projected clustering algorithms. In practice, it will be a judgement call to the user which algorithm(s) to apply and which pattern(s) are more useful for the problem in hand. But we have illustrated that if DRs are desirable, then clustering algorithms do not work well.

Experiment 3. In this experiment, we compare the algorithms on clustering a real dataset, the primate dataset. We would like to test if the algorithms can discover the 3 gene clusters corresponding to the 3 large DRs shown in 3(b–d). For each algorithm and for each of the 3 gene clusters T_i, the following discovery rate is computed:

$$1 - \min_{j=1,\ldots,k} \{\mathrm{err}(T_i, C_j)\}.$$

For k-means, k-modes, hierarchical, HARP and CAST algorithms, the parameter k (number of clusters) is varied from 2 to 20. Then, the best results are reported. For the bi-clustering algorithm, the number of bi-clusters is fixed at 100 so that we allow many more candidates to match with the 3 desirable clusters. The parameter δ is varied between $0.001, 0.01, 0.1$. Again, the δ giving the best results is reported.

The discovery rates are depicted in the Table I. The results show that the largest gene cluster (Fig. 3(b)) is faithfully discovered by all algorithms. For example, if k-means with $k = 2$ is used, then one of the two clusters has 97.6% overlap with the gene cluster in Fig. 3(b). This cluster can be discovered by all algorithms for it is clearly defined even if all samples are considered. However, the other two gene clusters are essentially missed by all algorithms, even if the best parameters are used. Traditional clustering methods missed the two clusters because they are incapable of selecting subsets of samples. The projected clustering algorithm HARP fails for it returns the cluster in Fig. 3(b) as one of its clusters which hinders the search of the two other overlapped clusters. The bi-clustering algorithm uses all samples and genes to make selection/deletion decisions and thus moderate-sized clusters are missed. This confirms that gene clusters defined by DRs are better discovered by the proposed algorithm. Of course, if gene

[1]For the bi-clustering algorithm, although a bi-cluster is defined only in a subset of samples, the searching algorithm makes gene selection/deletion decisions using all samples.

clusters defined based on all samples are desired, then traditional clustering algorithms are usually more effective.

TABLE I

DISCOVERY RATES OF VARIOUS ALGORITHMS IN DETECTING THE 3 LARGE GENE CLUSTERS IN FIG. 3(b–d).

Gene Cluster	DRIFT	KMEANS	KMODES	AVE.LINK	CAST	HARP	BICLUST
Fig. 3(b)	1.00	0.976	0.948	0.956	0.681	0.914	0.883
	($\mu = 90\%, v = -1$)	($k = 2$)	($k = 2$)	($k = 3$)	($k = 2$)	($k = 4$)	($\delta = 0.1$)
Fig. 3(c)	1.00	0.360	0.490	0.161	0.189	0.357	0.196
	($\mu = 90\%, v = -1$)	($k = 12$)	($k = 5$)	($k = 5$)	($k = 4$)	($k = 2$)	($\delta = 0.1$)
Fig. 3(d)	1.00	0.402	0.539	0.201	0.203	0.243	0.492
	($\mu = 90\%, v = 1$)	($k = 10$)	($k = 19$)	($k = 12$)	($k = 5$)	($k = 20$)	($\delta = 0.01$)

E. The Effect of the Model Parameters

In this subsection, we study the effect of the density threshold μ used in the μ-DRIFT algorithm and the size threshold $MinRow \times MinCol$ used for pruning.

Experiment 4. In this experiment, we investigate the distribution of the size of the DRs identified by the μ-DRIFT algorithm as μ is varied. We use random matrices of size 20-by-200 and sparsity 40%. The value of μ is varied between 30% and 90%. The size of the DRs identified by the μ-DRIFT algorithm is recorded. We repeat the experiment 10 times and the frequency counts are averaged. The resulting frequency plots for $\mu = 0.4, 0.6, 0.8$ are shown in Fig.7(a)–(c). See also Fig.2(d) for the result of $\mu = 100\%$. The average number of DRs for $\mu = 30\%, \ldots, 90\%$ is shown in Fig.7(d).

The result shows that the overall number of DRs increases with μ, except for a drop at $\mu = 80\%$. The reason is that as μ increases, the DRs are split into smaller DRs. Indeed, when $\mu = 30\%$, there is only 1 DR, namely the whole matrix. For the same reason, the size of the DRs decreases as μ increases, as we can see from Fig.7(a)–(c). We also observe that when $\mu = 60\%, 80\%$, most DRs have small size. Thus moderate- and large-sized DRs are more statistically significant. However, if μ is set to be very small, e.g. 40%, then small-sized DRs are statistically significant too. But for practical use, we recommend setting μ to be 70%–90%.

Experiment 5. In this experiment, we test the effectiveness of the pruning scheme given in §VI. We use random matrices of size 1000-by-20 and sparsity $10\%, 20\%, 30\%, 40\%$ for our tests. These matrices are input to the DRIFT algorithm to identify (perfect) DRs. The minimum-size threshold $MinRow \times MinCol$ is varied between $2, 4, 6, 8$ with $MinRow = MinCol$. The CPU time for the DRIFT algorithm is recorded. For each chosen value of the sparsity and minimum-size threshold, the experiment is repeated 10 times and the CPU times are averaged. The algorithms are implemented in C but called via Matlab 6.5. The experiments are done using a Pentium Dual Core 3.2 GHz machine.

In Fig.8, the graph of the average CPU time versus the minimum-size threshold for each sparsity is shown. We observe that the CPU time drops significantly as the threshold increases, especially when the sparsity is high (e.g. 30% and 40%). The main reason is that the search space in the Extended Basic DRIFT is greatly reduced. Another reason is that in our implementation, the DRs are stored and sorted in order to remove duplicates. Increasing the threshold causes a decrease in the number of DRs and thus shortens the time for sorting the DRs as well. When the sparsity is low (e.g. 10% and 20%), much of the CPU time is spent on the (repeated) Basic DRIFT algorithm which is independent of the minimum-size threshold. So, the CPU time is not reduced at all. This is expected since Our pruning scheme is designed to reduce the time spent on the Extended Basic DRIFT. In any case, it takes less than 3 seconds in such a low sparsity. In general, we recommend setting the threshold to be as large as possible so as to focus on the most statistically significant DRs and reduce the computation cost.

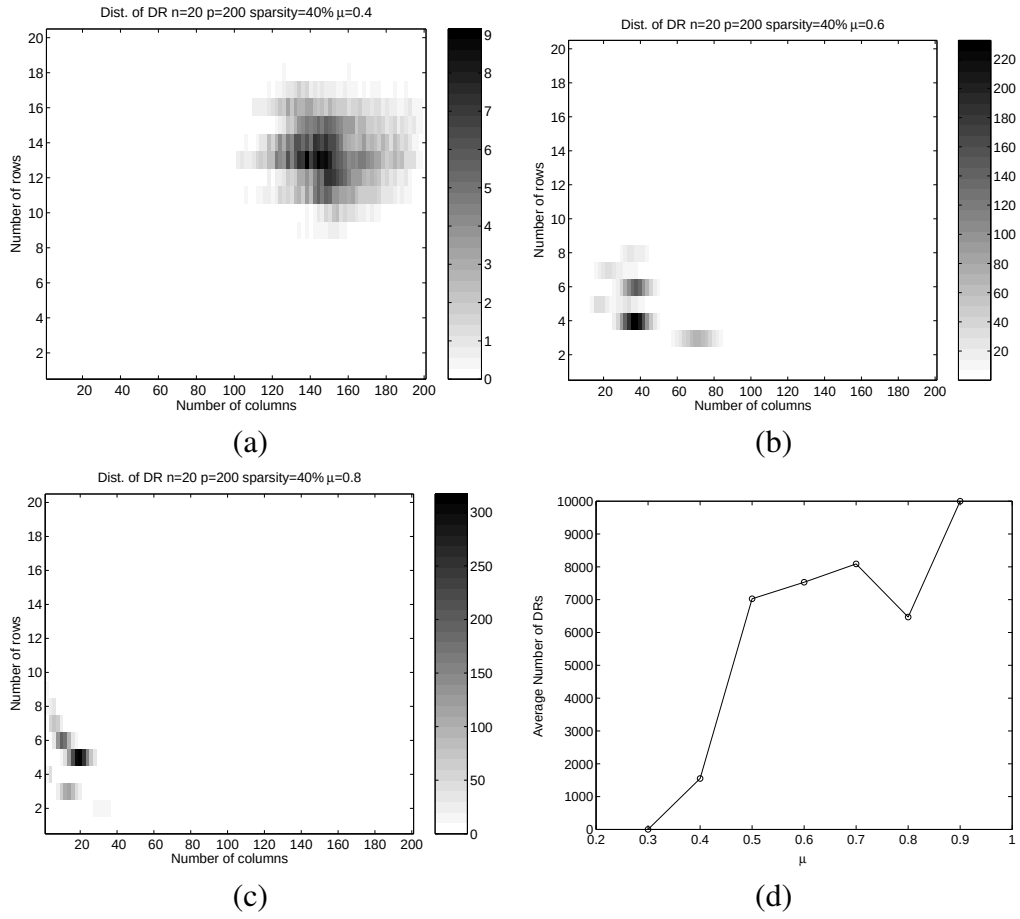

Fig. 7. The effect of μ on the size of the DRs. Random matrices of size 20-by-200 and sparsity 40% are used. (a)–(c) Frequency distribution for various μ. (d) Average number of DRs versus μ.

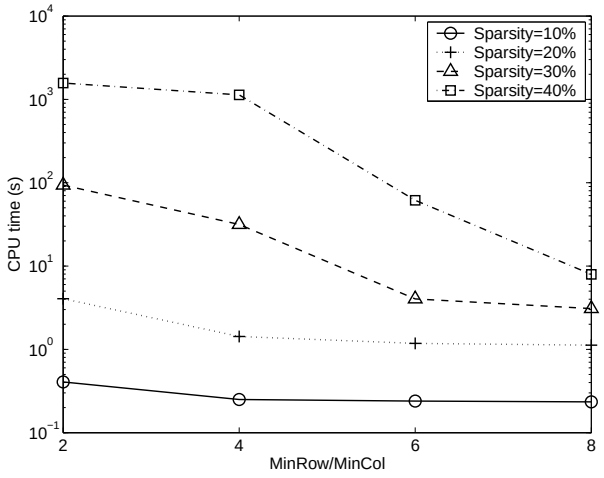

Fig. 8. CPU time of the DRIFT algorithm versus the minimum-size threshold. Random matrices of size 1000-by-20 and sparsity $10\%, 20\%, 30\%, 40\%$ are used.

VIII. Conclusion

In this paper, we study the notion of dense regions for categorized microarray data and present effective and efficient algorithms for discovering them. Such microarray patterns are very useful that practitioners may want to look into at the beginning of high-level analysis. Moreover, traditional statistical and machine learning methods fail to identify these patterns.

We demonstrate the usefulness of our algorithms on real datasets where several biologically interesting dense regions are discovered and on synthetic datasets where comparisons with clustering methods are carried out. We also study the empirical distribution of the size of dense regions in order to assess their statistical significance.

Although it is not our purpose to come up with any biological findings from the DRs, we aim to show that the subsets of samples and genes simultaneously identified by dense regions could be biologically meaningful and could not be identified by traditional methods such as clustering.

The input data to the algorithms can be a matrix recording the discretized gene expression levels. But it is equally-well to apply transformed data. In our yeast cell cycle example, the input is a binary matrix encoding whether the expression level is high or not. Thus, our method can be applied to answer a wide range of queries.

In terms of computational time, the algorithms are generally quite fast for typical microarray data. In our experiments on real data, it usually takes less than 1 minute even without any pruning. However, in some simulated data where the sparsity of the data matrix is very high, say 90%, then the algorithms may take a few hours if no pruning is applied. It is because there are a large number of (small-sized) dense regions exist. One can, of course, apply our pruning strategies to reduce the computational load. Alternatively, one can also use the idea from Cheng and Church's bi-clustering: fix the number of DRs to be discovered. But we feel that some further research is needed which focuses on the searching for a small number of most significant DRs.

We remark that Cheng and Church's bi-clustering algorithm allows the identification of genes that are anti-correlated over a subset of samples, i.e. co-regulated genes. In their mathematical definition of a bi-cluster, anti-correlated genes are not allowed. Such genes are found by an algorithmic trick which is used at the final stage of the algorithm. Roughly speaking, after a bi-cluster is found, each row which is not in the bi-cluster is "flipped" around its row mean. Then, the transformed rows that are similar to the given bi-clusters are added. While our current algorithm does not perform such a post-processing step, such an interesting idea of flipping the data can be easily adopted into our algorithm. Perhaps the most straightforward way is that after a DR is found, the categories of the remaining rows are rotated so that each category has a chance to become the target category. In this way, we can identify regions where each row is roughly constant but different rows may have a different constant. However, further research is needed to confirm the effectiveness and usefulness of such a procedure.

Acknowledgements

The authors would like to thank the anonymous referees for the helpful suggestions to improve the quality of the manuscript. We would also like to thank Kevin Yip for providing an implementation of the HARP and CAST algorithms and thank Ivy Law for her suggestions on the implementation of our algorithms.

M. Ng's research is supported in part by the Hong Kong Research Grant Council under contracts 7035/04P and 7035/05P and by HKBU FRGs. T. Chan's research is supported in part by grants from ONR under contract N00014-06-1-0345, NIH under contract U54 RR021813, and NSF under contract CCF-0528586.

References

[1] M. Eisen, P. Spellman, P. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proc. of National Academy of Sciences*, vol. 85, pp. 14 863–14 868, 1998.

[2] T. F. Cox and M. A. A. Cox, *Multidimensional Scaling*, 2nd ed. Chapman & Hall/CRC, 2000.
[3] S. Dudoit, J. Fridlyand, and T. Speed, "Comparison of discrimination methods for the classification of tumors using gene expression data," *Journal of the American Statistical Association*, vol. 97, no. 457, pp. 77–87, 2002.
[4] K. Fellenberg, N. Hauser, B. Brors, A. Neutzner, J. Hoheisel, and M. Vingron, "Correspondence analysis applied to microarray data," *Proc. of National Academy of Sciences*, vol. 98, no. 19, pp. 10 781–10 786, 2001.
[5] M. T. Lee, *Analysis of Microarray Gene Expression Data*. Kluwer Academic Publishers, 2004.
[6] R. Somogyi and C. Sniegoski, "Modeling the complexity of genetic networks: Understanding multigenetic and pleiotropic regulation," *Complexity*, vol. 1, pp. 45–63, 1996.
[7] S. Yeung, J. Tegner, and J. Collins, "Reverse engineering gene networks using simgular value decomposition and robust regression," *Proc. of National Academy of Sciences*, vol. 99, no. 9, pp. 6163–6168, 2002.
[8] C. Becquet, S. Blachon, B. Jeudy, J. Boulicaut, and O. Gandrillon, "Strong-association-rule mining for large-scale gene-expression data analysis: A case study on human SAGE data," *Genome Biology*, vol. 3, no. 12, pp. 67.1–67.16, 2002.
[9] B. Zhang and S. Horvath, "A general framework for automatic construction of gene co-expression networks," *Statistical Applications in Genetics and Molecular Biology*, vol. 4, no. 1, 2005.
[10] T. Mitchell, *Machine Learning*. McGraw-Hill, 1997.
[11] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of Bioinformatics and Computational Biology*, vol. 3, no. 2, pp. 185–205, 2005.
[12] I. Shmulevich and W. Zhang, "Binary analysis and optimization-based normalization of gene expression data," *Bioinformatics*, vol. 18, no. 4, pp. 555–565, 2002.
[13] H. Liu, F. Hussain, C. L. Tan, and M. Dash, "A systematic study of discretization methods," *Journal of Data Mining and Knowledge Discovery*, vol. 6, pp. 393–423, 2002.
[14] K. Y. Yip, D. W. Cheung, and M. K. Ng, "HARP: A practical projected clustering algorithm," *IEEE Transaction Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1387–1397, 2004.
[15] Y. Cheng and G. Church, "Biclustering of expression data," in *Proc. of the 8th Int. Conf. on Intel. Sys. for Mol. Biol.*, 2000.
[16] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data Mining and Knowledge Discovery*, vol. 2, pp. 283–304, 1998.
[17] S. Guha, R. Rastogi, and K. Shim, "ROCK: A robust clustering algorithm for categorical attributes," in *Proceedings of the 15th International Conference on Data Engineering (ICDE)*. IEEE Press, 1999, pp. 512–521.
[18] D. Gibson, J. Kleinberg, and P. Raghavan, "Clustering categorical data: An approach based on dynamical systems," in *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB)*. Morgan Kaufmann, 1998, pp. 311–322.
[19] D. Barbará, J. Couto, and Y. Li, "COOLCAT: An entropy-based algorithm for categorical clustering," in *Proceedings of 11th International Conference on Information and Knowledge Management (CIKM)*. ACM Press, 2002, pp. 582–589.
[20] R. G. Pensa, C. Robardet, and J.-F. Boulicaut, "A bi-clustering framework for categorical data," in *Knowledge Discovery and Databases: PKDD 2005*, ser. Lecture Notes in Artificial Intelligence, A. M. Jorge, L. Torgo, P. B. Brazdil, R. Camacho, and J. Gama, Eds., vol. 3721. Berlin: Springer, 2005, pp. 643–650.
[21] D. S. Moore and G. P. McCabe, *Introduction to the Practice of Statistics*, 4th ed. W. H. Freeman, 2002.
[22] M. Cáceres, "Elevated gene expression levels distinguish human from non-human primate brains," *PNAS*, vol. 100, pp. 13 030–13 035, 2003.
[23] C. Li and W. H. Wong, "Model-based analysis of oligonucleotide arrays: Expression index computation and outlier detection," *Proc. National Academy of Science*, vol. 98, pp. 31–36, 2001.
[24] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: John Wiley & Sons, Inc., 1990.
[25] G. Dennis, B. Sherman, D. Hosack, J. Yang, W. Gao, H. Lane, and R. Lempicki, "DAVID: Database for annotation, visualization, and integrated discovery," *Genome Biology*, vol. 4, no. 9, 2003.
[26] T. Lu *et al.*, "Gene regulation and DNA damage in the ageing human brain," *Nature*, vol. 429, no. 24, pp. 883–891, 2004.
[27] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, and K. Anders, "Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization," *Mol. Biol. Cell*, vol. 9, no. 12, pp. 3273–3297, 1998.
[28] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering gene expression patterns," *J. Comp. Bio.*, vol. 6, no. 3, pp. 281–297, 1999.

Andy M. Yip received his Bachelor's degree in Mathematics in 1998 from the Chinese University of Hong Kong. Two years later, he got his Master's degree in Mathematics from the University of Hong Kong. He then went on to pursue his Ph.D. in Mathematics at University of California, Los Angeles and graduated in 2005. He joined the Department of Mathematics at National University of Singapore in July 2005 as an assistant professor. His research interests include variational and PDE methods in image processing and data mining algorithms.

Michael Ng is a Professor in the Department of Mathematics at the Hong Kong Baptist University. He obtained his B.Sc. degree in 1990 and M.Phil. degree in 1992 at the University of Hong Kong, and Ph.D. degree in 1995 at Chinese University of Hong Kong. He was a Research Fellow of Computer Sciences Laboratory at Australian National University (1995-1997), and an Associate Professor (1997-2005) of the University of Hong Kong, before joining Hong Kong Baptist University.

Michael won the Honourable Mention of Householder Award IX, in 1996 at Switzerland, an excellent young researcher's presentation at Nanjing International Conference on Optimization and Numerical Algebra, 1999, and the Outstanding Young Researcher Award of the University of Hong Kong. He supervised more than 20 graduate students. His master's student, Eric Fung, won the Outstanding Research Postgraduate Student Award in 2004 at the University of Hong Kong.

Michael has published and edited 5 books, published more than 140 journal papers. He has reviewed papers for more than 40 international journals. He currently serves on the editorial boards of SIAM Journal on Scientific Computing, Numerical Linear Algebra with Applications, International Journal of Data Mining and Bioinformatics, Multidimensional Systems and Signal Processing, International Journal of Computational Science and Engineering, Numerical Mathematics, A journal of Chinese Universities (English Series) Applied Mathematical Sciences, and was guest editors of several special issues of the international journals (Journal of Computational Mathematics, International Journal of Applied Mathematics, Applied Mathematics and Computation, EURASIP Journal on Applied Signal Processing, International Journal of Imaging Systems and Technology).

Edmond Haocun Wu received the B.Sc. degree in applied mathematics & computer science from South China University of Technology, China in 2002 and the M.Phil. degree in Mathematics from The University of Hong Kong, Hong Kong in 2004. Currently, he is pursuing the Ph.D. degree in Department of Statistics & Actuarial Science, The University of Hong Kong, Hong Kong. His research interests include data mining, time series models, independent component analysis, financial data analysis, and risk management.

Tony Chan 's scientific background is in mathematics, computer science and engineering. He received his BS and MS degrees (in engineering) from CalTech and his PhD (in computer science) from Stanford University, worked at CalTech (Applied Math) as a Research Fellow, and taught at Yale before joining the UCLA faculty in 1986. He became Chair of the Department of Mathematics in 1997. He was one of the principal investigators who made the successful proposal to NSF to form the Institute for Pure and Applied Mathematics at UCLA, with a vision to promote collaborations between the mathematical sciences with the general scientific and engineering disciplines. He served as IPAM's Director from 2000-2001. From July 2001 to June 2006, he served as Dean of Physical Sciences at UCLA. He is also co-Director of the Center for Computational Biology at UCLA, an NIH-funded interdisciplinary center under the NIH Roadmap initiative. His current research interests include mathematical image processing and computer vision, VLSI physical design and computational brain mapping. He is an active member of many scientific societies, including SIAM (where he is currently a member of the Board of Trustees), AMS and IEEE. He serves on the editorial boards of several journals in mathematics and computing, and is one of the Editors-in-Chief of Numerisch Mathematik. He formerly served on the NSF Mathematical and Physical Sciences Advisory Committee and is a current member of the US National Committee on Mathematics. Since October 2006, he has been on temporary leave from UCLA to serve as Assistant Director of the Mathematical and Physical Sciences Directorate at the National Science Foundation. He has published over 200 refereed papers and is one of the most cited mathematicians according to http://isihighlycited.com/. He has mentored over 25 PhD students and 15 postdocs.