

# Data Complexity in Clustering Analysis of Gene Microarray Expression Profiles

Feng Luo and Latifur Khan

**Summary.** The increasing application of microarray technology is generating large amounts of high dimensional gene expression data. Genes participating in the same biological process tend to have similar expression patterns, and clustering is one of the most useful and efficient methods for identifying these patterns. Due to the complexity of microarray profiles, there are some limitations in directly applying traditional clustering techniques to the microarray data. Recently, researchers have proposed clustering algorithms custom tailored to overcome their limitations for microarray analysis. In this chapter, we first introduce the microarray technique. Next, we review seven representative clustering algorithms: K-means, quality-based clustering, hierarchical agglomerative clustering, self-organizing neural network-based clustering, graph-theory-based clustering, model-based clustering, and subspace clustering. All these algorithms have shown their applicability to the microarray profiles. We also survey several criteria for evaluating clustering results. Biology plays an important role in the evaluation of clustering results. We discuss possible research directions to equip clustering techniques with underlying biological interpretations for better microarray profile analysis.

## 11.1 Introduction

Understanding the system-level characteristics of biological organization is a key issue in the post-genome era. In every living organism, subsets of its gene expressions differ across types, stages, and conditions. Given a specific condition and stage, there are particular genes that are expressed. Measuring these gene expression levels across different stages in different tissues or cells, or under different conditions, is very important and useful for understanding and interpreting biological processes. For a long time, biologists dreamed of getting information about all genes in a genome and the ability to study the complex interplay of all genes simultaneously. The emergence of the microarray technique [26, 35] has brought this to realization. The microarray technology enables the massively parallel measurement of expressions of thousands of genes simultaneously. There are many potential applications for the microarray technique, such as identification of genetic diseases [32], discovery of new drugs [10], toxicology studies [31], etc.

The application of microarray technology now generates very large amounts of gene expression data. One microarray chip has anywhere from thousands to tens of thousands of genes on it. Thus a series of microarray experiments will generate ten thousand to a million

data points. As a result, there is an increasing need for technologies that can extract useful and rational fundamental knowledge of gene expression from the microarray data.

The biological system, like many other engineering synthetic systems, is modular [19]. The expression patterns of genes inside the same module are similar to each other. Thus, the genes can be grouped according to their gene expressions. Clustering analysis is one of the most useful methods for grouping data in large data sets. It has been shown that the clustering techniques can identify biologically meaningful gene groups from microarray profiles.

However, due to the complexity of biological systems, the microarray data are also very involved. Conventional clustering algorithms may have some limitation to deal with such intricacies and need modifications. First, the size of the data is large. A typical microarray data set will have thousands to ten thousands of data series and up to millions of data points. So the computation time for clustering is definitely an important issue. Second, although the experiments in a microarray profile may relate to each other, like a time series experiments of cell cycle or stress response, the degree of expression in different experiments can be dramatically different. Furthermore, in many cases, there is seldom a correlation between the experiments in a microarray data set, like microarray experiments of a set of mutants or microarray experiments of tumor cell and normal cell in cancer research; or there are even several kinds of experiments in one microarray profile. Usually, microarray profiles by nature are multidimensional. Third, for one specific experimental condition, not every gene is expressed. The microarray expression data of these genes are more like random outliers. Either some pre-processing of the microarray data set is needed to remove the outliers as much as possible or the clustering algorithms themselves need to have some mechanism to handle these outliers. All these characteristics of microarray data need to be considered for successful clustering.

In this chapter, we review clustering analysis for microarray profiles. First, we introduce the microarray technology. Second, we present some traditional and newly developed clustering algorithms that have been applied to microarray profile analysis. Third, we describe some methods to evaluate the clustering results. Finally, we provide some discussions.

## 11.2 Microarray technology

The microarray technology refers to the use of the microarray chips to measure gene expressions. Microarray chips are commonly small slides that are made of chemically coated glass, nylon membrane, or silicon, onto which a matrix of spots are printed. Each spot on a microarray chip is less than 250  $\mu\text{m}$  in diameter and contains millions of identical DNA molecules (probes). The number of spots on a single microarray chip ranges from thousands to tens of thousands, which depends on the size of the chip and the spotting technique used. In each spot there are only specific kinds of DNA sequences that correspond to a specific gene. The hybridization between the DNA sequence and the complement DNA (cDNA) that represent the messenger RNA (mRNA) from test samples (targets) is quantified by the signal of labeling molecules bound to cDNA, which is the essential principle employed by the microarray technology to measure the gene expression level of the targets.

There are two major types of microarray technology. One is the cDNA microarray [35]; the other is the oligonucleotide microarray [26]. Although there are some differences in the details of experiment protocols, the gene expressions measured by the cDNA microarray and the oligonucleotide microarray have the same biological meanings. A typical microarray experiment consists of four basic steps [40]:

**Target preparation:** Extract mRNA from both the test and the control samples. Then transcribe the mRNA to cDNA in the presence of nucleotides that are labeled with fluorescent

dyes or radioactive isotopes. For cDNA microarray, the test and reference cDNA are labeled differently. For oligonucleotide microarray, this differential labeling is not necessary.

**Hybridization:** The labeled test and control cDNA samples are mixed in equal amounts and incubated with a microarray slide for hybridization for a given time. Then the microarray slide is washed to get rid of the surplus cDNA.

**Scanning:** Once the hybridized microarray is ready, the expression level of genes is detected by monitoring the intensity of the labeling fluorescent or radioactive molecules. For fluorescence labeling, a laser beam is used to excite the fluorescent dyes and the degree of fluorescence correlates with the abundance of target molecules at a specific spot. The fluorescent emission is monitored by a scanner, which also produces a digital image file to store signals.

**Normalization:** To correct the differences in overall array intensity, such as background noise and different efficiency in detection, the raw signal intensity, either from cDNA or from oligonucleotide microarray, must be normalized to a common standard. The normalization can make the gene expression profile from different experiments comparable. After normalization, the final gene expression levels are presented as an expression ratio of test versus control sample and are ready for further analysis.

A typical microarray gene expression profile is represented by a real matrix  $E$ , the *expression matrix*. Each cell  $e_{ij}$  of the expression matrix  $E$  represents an expression value of a gene (row) in a certain experiment (column). Generally, there is no biological difference between the expression profiles measured by cDNA microarray and those measured by oligonucleotide microarray. In this chapter, the term *microarray* means both cDNA and oligonucleotide microarray.

## 11.3 Introduction to Clustering

Clustering is defined as a process of partitioning a set of data  $S = \{D_1, D_2, \dots, D_n\}$  into a number of subclusters  $C_1, C_2, \dots, C_m$  based on a measure of similarity between the data (distance based) or based on the probability of data following certain distribution (model-based). For a survey of clustering analysis techniques, see Jain et al. [23]. The distance-based clustering techniques are broadly divided into hierarchical clustering and partitioning clustering. As opposed to partitioning clustering algorithms, the hierarchical clustering can discover trends and subtle common characteristics, and thus can provide more useful information for biologists. Hierarchical clustering algorithms are further subdivided into agglomerative algorithms and divisive algorithms. Recently, as more and more high-dimensional microarray expressions data become available, subspace clustering algorithms have been developed to find clusters in the subdimension of the whole space.

### 11.3.1 Similarity Measurements of Microarray Profiles

The similarity measurement is essential for the clustering analysis, which controls the definition of a cluster. Different similarity measurement may generate widely different values for the same set of data. Most commonly used similarity measurements for microarray profiles are Euclidean distance and Pearson correlation coefficient.

The Euclidean distance actually measures the geometric distance between data points. The distance between expression  $E_i$  of gene  $g_i$  and expression ( $E_j$ ) of gene  $g_j$  in  $D$ -dimensions is defined as

$$Euclidean(E_i, E_j) = \sqrt{\sum_{d=1}^D (E_{id} - E_{jd})^2} \quad (11.1)$$

The Euclidean distance is simple and works well when a data set has “compact” or “isolated” clusters. However, the distances can be greatly affected by differences in scale among the dimensions from which the distances are computed. To make the distance insensitive to the absolute level of expression and sensitive to the degree of change [14], the microarray expression profiles are always standardized by z-score normalization (zero mean and unit variance) [39] before the distance is computed.

The Pearson correlation coefficient measures the strength and direction of a *linear* relationship between two data. It assumes that both data are approximately normally distributed and their joint distribution is bivariate normal. The Pearson correlation coefficient between expression ( $E_i$ ) of gene  $g_i$  and expression ( $E_j$ ) of gene  $g_j$  in  $D$ -dimensions is defined as

$$Pearson(E_i, E_j) = \frac{1}{D} \sum_{d=1, D} \left( \frac{E_{id} - M_{E_i}}{\sigma_{E_i}} \right) \left( \frac{E_{jd} - M_{E_j}}{\sigma_{E_j}} \right) \quad (11.2)$$

where  $M_{E_i}, M_{E_j}$  are the average gene expression level of gene  $g_i$  and gene  $g_j$ , respectively, and  $\sigma_{E_i}, \sigma_{E_j}$  are the standard deviation of the gene expression level of gene  $g_i$  and gene  $g_j$ , respectively. The Pearson correlation ranges from +1 to -1. A correlation of +1 means a perfect positive linear relationship between two expressions.

Although the Pearson correlation coefficient is an effective similarity measure, it is sensitive to outlier expression data. Especially if the number of data of two data series is limited, the effects of outlier points become more significant. Heyer et al. [22] observed that if the expression of two genes has a high peak or valley in one of the experiments but is unrelated in all others, the Pearson correlation coefficient still will be very high, which results as a false positive. To remove the effect of a single outlier, Heyer et al. proposed a similarity measurement called the Jackknife correlation, which is based on the Jackknife sampling procedure from computational statistics [13]. The Jackknife correlation between expression ( $E_i$ ) of gene  $g_i$  and expression ( $E_j$ ) of gene  $g_j$  in  $D$ -dimensions is defined as

$$Jackknife(E_i, E_j) = \min\{P_{ij}^1, \dots, P_{ij}^d, \dots, P_{ij}^D\} \quad (11.3)$$

where  $P_{ij}^d$  is the Pearson correlation coefficient between expression  $E_i$  and expression  $E_j$  with  $d$ th experiment value deleted. More general Jackknife correlation by deleting every subset of size  $n$  will be robust to  $n$  outliers, but they become computationally unrealistic to implement.

### 11.3.2 Missing Values in Microarray Profiles

There may be some missing expression values in the microarray expression profiles. These missing values may have been caused by various kinds of experimental errors. When calculating the similarity between two gene expressions with missing values, one way is to consider only the expressions present in both genes. The other way is to replace the missing value with estimated values. Troyanskaya et al. [43] evaluated three estimation methods: singular value decomposition based method, weighted K-nearest neighbors, and row average. Based on the robust and sensitive test, they demonstrated that the weighted K-nearest neighbors method is the best one for missing value estimation in microarray expression profiles.

## 11.4 Clustering Algorithms

In this section, we present several representative clustering algorithms that have been applied to gene expression analysis.

### 11.4.1 K-Means Algorithms

The K-means algorithm [30] is one of the most popular partitioning clustering methods. The K-means partition the data set into predefined K clusters by minimizing the average dissimilarity of each cluster, which is defined as

$$E = \sum_{k=1}^K \sum_{C(i)=k} \|x_{ik} - \bar{x}_k\|^2 \quad \text{where } \bar{x}_k = \frac{1}{N_k} \sum_{i=1}^N x_{ik} \quad (11.4)$$

$\bar{x}_k$  is the mean of all data belonging to the k th cluster, which represents the centroid of the cluster. Thus, the K-means algorithm tries to minimize the average distance of data within each cluster from the cluster mean. The general K-means algorithm is as follows [23]:

1. Initialization: Choose  $K$  cluster centroids (e.g., randomly chosen  $K$  data points).
2. Assign each data-point to the closest cluster centroid.
3. Recompute the cluster centroid using the current cluster memberships.
4. If the convergence criterion is not met, go to step 2.

The K-means algorithm converges if every data-point is assigned to the same cluster during iteration, or the decrease in squared error is less than a predefined threshold. In addition, to avoid the local suboptimal minimum, one should run the K-means algorithm with different random initializing values for the centroids, and then choose the one with the best clustering result (smallest average dissimilarity). Furthermore, since the number of clusters inside the data set is unknown, to find the proper number of clusters, one needs to run the K-means algorithm with different values of  $K$  to determine the best  $K$  value.

Tavazoie et al. [39] selected the most highly expressed 3000 yeast genes from the yeast cell cycle microarray profiles and applied the K-means algorithm to cluster them into 30 clusters. They successfully found that some of the clusters were significantly enriched with homo functional genes, but not all clusters showed the presence of functionally meaningful genes. This is because some clusters of genes participate in multiple functional processes, or some clusters of genes are not significantly expressed, and the number of clusters (30) may also not be the optimal one.

The K-means algorithm is most widely used because of its simplicity. However, it has several weak points when it is used to cluster gene expression profiles. First, it is sensitive to outliers. As there are a lot of noise and experimental errors among the gene expression data, the clustering result produced by K-means algorithms may be difficult to explain. Second, since not every gene expresses significantly under certain conditions, the expressions of this kind of genes do not have a clear pattern but are more likely to have random patterns. This kind of gene must be considered as noisy and should not be assigned to any cluster. However, the K-means algorithm forces each gene into a cluster, which decreases the accuracy of the clusters and renders some of the clusters meaningless and not suitable for further analysis.

### 11.4.2 Quality-Based Clustering

Recently, several quality-based clustering algorithms have been proposed to overcome the drawbacks of K-means. These kinds of algorithms attempt to find quality-controlled clusters from the gene expression profiles. The number of clusters does not have to be predefined and is automatically determined during the clustering procedure. Genes whose expressions are not similar to the expressions of other genes will not be assigned to any of the clusters.

Heyer et al. [22] proposed a new quality clustering algorithm called *QT\_Clust*. They used the *QT\_Clust* to cluster 4169 yeast gene cell cycle microarray expression profiles and discovered 24 large quality clusters. The *QT\_Clust* algorithm uses two steps to produce quality clusters: (1) For each gene, a cluster seeded by this gene is formed. This is done by iteratively adding genes to the cluster. During each iteration, the gene that minimizes the increase in cluster diameter is added. The process continues until no gene can be added without surpassing the predefined cluster diameter threshold. After the first step, a set of candidate clusters is created. (2) Quality clusters are selected one by one from the candidates. Each time, the largest cluster is selected and retained. Then the genes that belong to this cluster are removed from the other clusters. This procedure continues until a certain termination criterion is satisfied. One criterion suggested by Heyer et al. is the minimum number of clusters. The total number of clusters is controlled by the termination criterion and is not required to be predefined like K-means algorithm. The quality of the clusters in *QT\_Clust* is controlled by the cluster diameter, which is a user-defined parameter. However, determination of this parameter is not an easy task. It may need either the biological knowledge of the data set to evaluate the clustering result, or use of some cluster validation criterion to evaluate the clustering result. In either case one may need to use the clustering algorithm several times to obtain a proper clustering result. Furthermore, as pointed out by Smet et al. [37], the diameter threshold may also be different for each cluster. Then, one cluster diameter cannot be proper for all clusters.

Smet et al. [37] proposed a new heuristic adaptive quality-based clustering algorithm called *Adap\_Cluster* to improve the method proposed by Heyer et al. The *Adap\_Cluster* algorithm constructs clusters sequentially with a two-step approach. In the first step (quality-based step), the algorithm finds a cluster center  $C_K$  using a preestimated radius ( $R_K$  **PRELIM**) with a method similar to the algorithm proposed by Heyer et al. If the size of the cluster is less than a user-defined parameter **MIN\_NR\_GENES**, the clustering result is discarded. Otherwise, a second step (adaptive step) is invoked. The adaptive step uses an expectation maximization (EM) algorithm to optimize the radius  $R_K$  of the cluster center  $C_K$  for a certain significant level **S**. The significant level is the probability that a gene belongs to a cluster. The default value of the significant level is 95%, which means that a gene has less than a 5% probability of being a false positive. The significant level can determine the number of genes that belong to the cluster. Then, the significant level **S** is the quality-control criterion. All clusters have the same significant level. However, the cluster radius may differ among the clusters. The advantage of using a significant level as a quality-control criterion is that it has easy-to-understand statistical meaning and is independent of the data set. Smet et al. applied the *Adap\_Cluster* to 3000 yeast gene cell cycle microarray expression profiles. Compared to the result of K-means, the result of *Adap\_Cluster* has a higher degree of enrichment based on the P-value significant measure (see section 11.5) and is more biologically consistent.

### 11.4.3 Hierarchical Agglomerative Clustering (HAC) Algorithm

The HAC algorithm is a classical and most commonly used hierarchical clustering algorithm. The HAC algorithm iteratively joins the closest subelements into a hierarchical tree. The general HAC algorithm is as follows:

1. Put each data element into a singleton cluster; compute a list of intercluster distances for all singleton cluster; then sort the list in ascending order.
2. Find a pair of clusters having the most similarity; merge them into one cluster and calculate the similarity between the new cluster and the remaining clusters.
3. When there is more than one cluster remaining, go to step 2; otherwise stop.

Based on the calculation of similarity between the nonsingleton clusters, a variety of hierarchical agglomerative techniques have been proposed. Single-link, complete-link, and group-average-link clustering are commonly used. In the single-link clustering the similarity between two clusters is the maximum similarity of all pairs of data that are in different clusters. In the complete-link clustering the similarity between two clusters is the minimum similarity of all pairs of data that are in different clusters. In the group-average-link clustering the similarity between two clusters is the mean similarity of all pairs of data that are in different clusters [45]. Lance and Williams [25] show that many HAC algorithms can be derived from the following general combinatorial formula:

$$d_{k,i \cup j} = \alpha_i \cdot d_{k,i} + \alpha_j \cdot d_{k,j} + \beta \cdot d_{i,j} + \gamma \cdot |d_{k,i} - d_{k,j}| \quad (11.5)$$

where  $i \cup j$  is a cluster constructed by merging cluster  $i$  and cluster  $j$  and  $d_{k,i \cup j}$  is the distance between cluster  $i \cup j$  and an existing cluster  $k$ . The  $\alpha$ ,  $\beta$ ,  $\gamma$  parameters characterize the different HAC algorithms [25].

The single-link and complete-link clustering simply use the similarity information of minimum or maximum of a cluster; therefore, these methods perform less well than the group-average-link clustering. But the simple-link algorithm is easier to implement, has some theoretical characteristic, and has been widely used. The single-link clustering tends to build a long chaining cluster, which makes it suitable for delineating ellipsoidal clusters but not suitable for poorly separated clusters.

The clustering result of HAC can be represented by a dendrogram, which provides a natural way to graphically represent the data set. Eisen et al. [14] used the average-link HAC algorithm to analyze the human gene growth response and the yeast cell cycle microarray expression profiles. They also proposed using a colored matrix to visualize the clustering result. Each row of the matrix represents expressions of a gene. Each column of the matrix represents the expressions of an experiment. Each cell of the matrix is colored according to the expression ratio. Expressions of log ratio equal to 0 are colored black, increasingly positive ratios are colored with increasing intensity of reds, and increasingly negative ratios are colored with increasing intensity of greens. The rows in the matrix are ordered based on the dendrogram of the clustering result, so that genes with similar expressions patterns are adjacent in the matrix. This graphical view presents an intuitive understanding of the clustering result of the data set, which is most favored by biologists. A program called TreeView is available on Eisen's web site [15].

Although the HAC algorithm has been widely used for clustering gene microarray expression profiles, it has several drawbacks. First, as Tamayo et al. [38] have noted, HAC suffers from a lack of robustness when dealing with data containing noise, so that preprocessing data to filter out noise is needed. Second, unlike the division hierarchical clustering algorithm (such

as SOTA and DGSOT in section 11.4) that can stop the hierarchical tree construction in any level, HAC needs to construct the hierarchical tree that includes the whole data set before extracting the patterns. This becomes very computationally expensive for a large data set when only the brief upper level patterns of the data set are needed. Third, since HAC is unable to reevaluate the results, some clusters of patterns are based on local decisions that will produce difficult-to-interpret patterns when HAC is applied to a large array of data. Fourth, the number of clusters is decided by cutting the tree structure at a certain level. Biological knowledge may be needed to determine the cut.

#### 11.4.4 Self-Organizing Neural Network-Based Clustering Algorithm

##### Self-Organizing Map (SOM)

The SOM is a self-organizing neural network introduced by Kohonen [24]. It maps the high-dimensional input data into the low-dimensional output topology space, which usually is a two-dimensional grid. Furthermore, the SOM can be thought of as a “nonlinear projection” of probability density function  $p(x)$  of the high-dimensional input data vector  $x$  onto the two dimensional display. This makes SOM optimally suitable for applying to the problem of the visualization and clustering of complex data.

Each SOM node in the output map has a reference vector  $w$ . The reference vector has the same dimension as the feature vector of input data. Initially the reference vector is assigned random values. During the learning process an input data vector is randomly chosen from the input data set and compared with all  $w$ . The *best matching node*  $c$  is the node that has the minimum distance with the input data:

$$c : \|x - w_c\| = \min_i \{\|x - w_i\|\} \quad (11.6)$$

Then, equation (11.7) is used to update the reference vectors of the best matching node and its neighboring nodes, which are topologically close in the map. In this way, eventually neighboring nodes will become more similar to the best match nodes. Therefore, the topologically close regions of the output map gain an *affinity* for clusters of similar data vectors [24]:

$$\Delta w_i = \eta(t) \times \Lambda(i, c) \times (x - w_i) \quad (11.7)$$

where  $i$ ,  $t$ , and  $\eta(t)$  denote the neighboring node, discrete time coordinate, and learning rate function, respectively. The convergence of the algorithm depends on the proper choice of  $\eta$ . At the beginning of the learning process,  $\eta$  should be chosen close to 1. Thereafter, it should decrease monotonically. One choice can be  $\eta(t) = 1/t$ . Note that in equation (11.7), the  $\Lambda(i, c)$  is the neighborhood function. A Gaussian function can be used to define  $\Lambda(i, c)$ :

$$\Lambda(i, c) = \exp\left(-\frac{\|r_i - r_c\|^2}{2\sigma(t)^2}\right) \quad (11.8)$$

where  $\|r_i - r_c\|$  denotes the distance between the best match node and the neighboring node  $i$  and  $\sigma(t)$  denotes the width of the neighbor. At the beginning of the learning process the width of the neighborhood is fairly large, but it decreases during the learning process. Therefore,  $\sigma(t)$  decreases monotonically with  $t$ . Thus the size of the neighborhood also monotonically decreases. At the end of learning, only the best match node is updated. The learning steps will stop when the weight update is insignificant.



Finally, each data-point is assigned to its best match node to form final clusters. Furthermore, there is a geometrical relationship between the clusters. Clusters that are close to each other in the output grid are more similar to each other than those that are further apart. This makes it easy to find some cluster relationship during the visualization of the SOM. Tamayo et al. [38] applied the SOM to analyze yeast gene cell cycle and different human cell culture microarray expression profiles. The SOM successfully identified the predominant gene expression pattern in these microarray expression profiles.

### Self-Organizing Tree Algorithm (SOTA)

Dopazo et al. [11] introduced a new unsupervised growing and tree-structured self-organizing neural network called self-organizing tree algorithm (SOTA) for hierarchical clustering. SOTA is based on Kohonen's [24] self-organizing map (SOM) and Fritzke's [16] growing cell structures. The topology of SOTA is a binary tree.

Initially the system is a binary tree with three nodes (Fig. 11.1a). The leaf of the tree is called a *cell* and internal node of the tree is called a *node*. Each cell and node has a reference vector  $w$ . The values of the reference vector are randomly initialized. In SOTA only cells are used for comparison with the input data. The procedure that distribute all data into cells is called a *cycle*. Each adaptation *cycle* contains a series of *epochs*. Each *epoch* consists of presentation of all the input data, and each presentation has two steps. First, the best match cell, which is known as the *winning cell*, is found. This is similar to the SOM. The cell that has the minimum distance from the input data is the best match cell/winning cell. The distance between the cell and data is the distance between the data vector and the reference vector of the cell. Once the *winning cell* of a data is found, the data is assigned to the cell. Second, update the reference vector  $w_i$  of the winning cell and its neighborhood using the following function:

$$\Delta w_i = \varphi(t) \times (x - w_i) \quad (11.9)$$

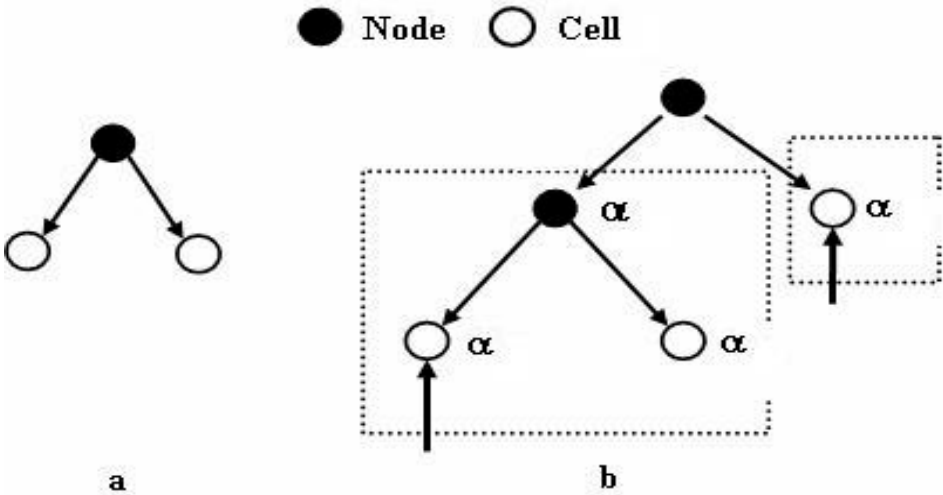
where  $\varphi(t)$  is the learning function:

$$\varphi(t) = \alpha \times \eta(t) \quad (11.10)$$

where  $\eta(t)$  is the function similar in SOM and  $\alpha$  is a learning constant. For different neighborhoods  $\alpha$  has different values. Two different neighborhoods are here. If the sibling of the winning cell is a cell, then the neighborhood includes the winning cell, the parent node, and the sibling cell. Otherwise, it includes only the winning cell itself [11] (Fig. 11.1b). Furthermore, parameters  $\alpha_w$ ,  $\alpha_m$ , and  $\alpha_s$  are used for the winning cell, the ancestor node, and the sibling cell, respectively. For example, values of  $\alpha_w$ ,  $\alpha_m$ , and  $\alpha_s$  can be set as 0.1, 0.05, and 0.01, respectively. Note that the parameter values are not equal. These nonequal values are critical to partition the input data set into various cells. A *cycle* converges when the relative increase in total error falls below a certain threshold.

After distributing all the input data into two cells, the cell that is most heterogeneous will be changed to a node and two descendent cells will be created. To determine heterogeneity, the *resource* of a cell is introduced. The *resource* of a cell  $i$  is the average of the distances between the input data assigned to the cell and the cell:

$$Resource_i = \sum_{i=1}^D \frac{d(x_i, w_i)}{D} \quad (11.11)$$



**Fig. 11.1.** (a) Initial architecture of SOTA. (b) Two different reference vector updating schemas.

where  $D$  is the total number of input data associated with the cell. A cell that has the maximum *resource* will expand. Therefore, the algorithm proceeds through the *cycle* until each input data-point is associated with a single cell or it reaches the desired level of heterogeneity.

SOTA uses a neural network mechanism and it is robust to noisy data. The time complexity of SOTA is  $O(N \log N)$ , where  $N$  is the number of genes. If only upper-level patterns are needed, the heterogeneity threshold can be set to a larger value and the SOTA can stop in the early stages. Then, the time complexity of SOTA can be reduced to  $O(N)$ , namely approximately linear. Herrero et al. [20] applied SOTA to cluster 800 yeast gene microarray expression profiles. An online web site tool is also available [21].

### DGSOT Algorithm

Nearly all hierarchical clustering techniques that include the tree structure have two shortcomings: (1) they do not properly represent hierarchical relationships, and (2) once the data are assigned improperly to a given cluster, they cannot later be reevaluated and placed in another cluster. Recently, Luo et al. [27, 28] proposed a new tree-structured self-organizing neural network, called the dynamically growing self-organizing tree (DGSOT) algorithm, to overcome these two drawbacks of hierarchical clustering.

DGSOT is a tree-structured self-organizing neural network designed to discover the proper hierarchical structure of the underlying data. DGSOT grows vertically and horizontally. In each vertical growth, DGSOT adds two children to the leaf whose heterogeneity is greater than a threshold and turns it into a node. In each horizontal growth, DGSOT dynamically finds the proper number of children (subclusters) of the lowest-level nodes. Each vertical growth step is followed by a horizontal growth step. This process continues until the heterogeneity of all leaves is less than a threshold  $T_R$ . During vertical and horizontal growth, a learning process similar to SOTA is adopted. Figure 11.2 shows an example of DGSOT algorithm in action. Initially there is only one root node (Fig. 11.2a). All the input data are associated with the

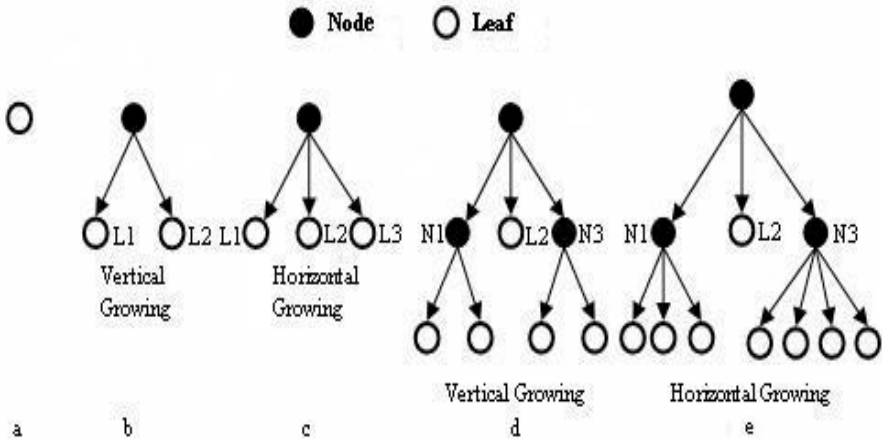


Fig. 11.2. Illustration of the DGSOT algorithm.

root, and the reference vector of the root node is initialized with the centroid of the data. When vertical growth is invoked, two children are added to the root node. All input data associated with the root node are distributed between these children by employing a learning process (Fig. 11.2b). Following vertical growth, horizontal growth is invoked to determine the proper number of children for the root. In this example, three leaves are used (Fig. 11.2c). After this, the heterogeneities of the leaves are checked to determine whether or not to expand to another level. The answer is yes in this example, and a new vertical growth step is invoked. Two children are added to the leaves (L1, L3) whose heterogeneity are greater than the threshold (Fig. 11.2d) and are turned to nodes (N1, N3). All the input data are distributed again with the learning process, and a new horizontal growth begins (Fig. 11.2e). This process continues until the heterogeneity of all the leaves (indicated by the empty cycle in Fig. 11.2) are less than the threshold.

The DGSOT algorithm combines the horizontal growth and vertical growth to construct a multifurcating hierarchical tree from top to bottom to cluster the data. If the number or the size of the clusters and subclusters of the data set are not even, for example, there is a very large cluster in the data set, the combination of horizontal growth and vertical growth lets the DGSOT algorithm find the proper hierarchical structure of the underlying data set, and then find a more reasonable final clustering result. The harmonization of the vertical growth and the horizontal growth is important in the DGSOT algorithm to find the proper structure of the underlying data set. The balance of vertical and horizontal growth is controlled by the clustering validation criterion, which determines the number of horizontal growth. Therefore, the cluster validation criterion is critical in the DGSOT algorithm. In DGSOT, the cluster validation criterion is used to determine the proper number of clusters in each hierarchical level rather than in the whole data set. For a data set containing an even number of clusters along with similar size, a proper cluster validation criterion must not allow the horizontal growth to continue forever without the vertical growth. On the other hand, for a data set containing an uneven number of clusters or an uneven size of clusters, a proper cluster validation criterion must be able to detect that uneven behavior and find the best representation in each hierarchical level.

To improve the clustering accuracy, Luo et al. [27, 28] proposed a K-level up distribution (KLD) mechanism. For KLD, data associated with a parent node will be distributed among its children leaves and also among its neighboring leaves. The following is the KLD strategy:

- For a selected node, its K level up ancestor node is determined.
- The subtree rooted by the ancestor node is determined.
- Data associated with the selected node is distributed among all leaves of the subtree.

The KLD scheme increases the scope in the hierarchy for data distribution, which will give the data mis-clustered at an early stage a chance to be reevaluated. The DGSOT algorithm combined with the KLD mechanism overcomes the drawbacks of the traditional neural tree-based hierarchical clustering algorithms.

In DGSOT, each leaf represents a cluster that includes all data associated with it. The reference vector of a leaf is the centroid of all data associated with it. Therefore, all reference vectors of the leaves form a *Voronoi* set of the original data set, and each internal node represents a cluster that includes all data associated with its leaf descendants. The reference vector of an internal node is the centroid of all data associated with its leaf descendants. The internal nodes of each hierarchical level also form a *Voronoi* set of the data set with a different resolution.

The DGSOT algorithm is applied to cluster the same 3000 yeast gene cell cycle microarray expression data that Tavazoie et al. [39] used. The clustering results of DGSOT are compared with SOTA when both have 25 clusters. For these low-level resolution clustering results, DGSOT, with the multipartition strategy, can successfully establish the hierarchical structure of these data, and then get more reasonable results than those obtained by SOTA, which is a pure bipartitions method. This is more drastic if the structure and substructure of the data contains an uneven number of clusters and subclusters or contains dramatically different sizes of clusters and subclusters [28]. Furthermore, the biological functionality enrichment in the clustering result of DGSOT is considerably higher in the degree of enrichment based on the P-value significant measure (see section 11.5) than the clustering result of SOTA and the K-means [28].

### 11.4.5 Graph-Based Clustering Algorithm

Given a set of gene expression data profiles  $D = \{d_i\}$ , we can define a weighted undirected complete graph  $G = (V, E)$  to represent it. Each vertex in graph represents a gene. And each edge  $(u, v) \in E$  has a weight that represents the similarity (or distance) between  $u$  and  $v$ . The graph-based clustering algorithms solve the problem of clustering a data set based on some graph theoretical problems, such as finding the minimum cut and finding the maximal cliques, or according to the properties of the graph, such as the minimum spanning tree of graph  $G$ .

#### **CLICK (CLuster Identification via Connectivity Kernels)**

The CLICK algorithm [36] recursively partitions the graph  $G$  into highly connected subgraphs, which represent clusters. The partition that removes a subset of edges in the graph to disconnect the graph is called a cut. C. Shamir and Sharan [36] defined the weight of the edge  $(v, u)$  as the probability that vertices  $u$  and  $v$  are in the same cluster. Then, the weight of  $C$  is the sum of the weights of its edges. And a minimum weight cut in  $G$  is a cut with the minimum weight. The CLICK algorithm recursively finds the minimum cut in the graph and partitions the current graph into two subgraphs. After each cut, a stopping criterion is used

to check the subgraph. If the subgraph satisfies the stopping criterion, then it is a kernel, and no more split is needed. Otherwise, the partitioning procedure is continued. The full CLICK algorithm also has two postprocessing steps to improve the clustering results. The adoption step adds the singletons into the cluster if the similarity between the singleton and the cluster is beyond a predefined threshold. The merging step merges two clusters that are similar. Shamir and Sharan [36] applied the CLICK to two gene expression profiles. The clustering results of CLICK are shown to perform better in homogeneity and separation than the result of SOM and HAC on two expression profiles, respectively.

### CAST (Cluster Affinity Search Technique)

Ben-Dor et al. [4] proposed a corrupted clique graph model. In this model, a cluster can be represented by a clique graph, which is a disjoint union of complete graphs. The true clusters of genes are denoted by the clique graph  $H$ . Then, the input graph  $G$  of gene expression profiles can be obtained from  $H$  by flipping each edge/non-edge with probability  $\alpha$ . Therefore, the clustering procedure can be translated into obtaining a clique graph as good as true clique graph  $H$  with high probability from the input graph  $G$ .

Based on the corrupted graph model, Ben-Dor et al. [4] proposed a theoretical algorithm that can discover the clusters with high probability and practical heuristic algorithm called CAST that can run in polynomial time. The input of CAST is a complete graph  $G$ . The weight of the edge is the similarity between two gene expressions. Ben-Dor et al. defined the *affinity value*  $a(v)$  of a vertex  $v$  with respect to a cluster  $C$  as  $a(v) = \sum_{u \in C} S(u, v)$ . The CAST constructs one cluster at a time. When data have not been clustered, the CAST algorithm randomly picks up a data-point and starts a new cluster. Then, a free data-point whose affinity is greater than a predefined affinity threshold  $t$  is added to the cluster, and a data-point in the cluster whose affinity is less than  $t$  is removed from the cluster. This process continues until it is stabilized, and one cluster is created. Ben-Dor et al. applied CAST to several gene expression profiles and showed very promising results.

### Minimum Spanning Tree (MST)-Based Clustering Algorithm

A spanning tree,  $T$ , of a weighted undirected graph  $G$  is a minimal subgraph of  $G$ , which connects all vertices. A MST is the spanning tree with the smallest total weight. The MST can represent the binary relationship of vertices in the graph. Xu et al. [46] observed that the tree edges connecting data of the same cluster are short whereas the tree edges linking different clusters are long. Based on this observation, Xu et al. defined the separation condition of a cluster as follows: “Let  $D$  be a data set and  $s$  represent the distance between two data in  $D$ . Then,  $C \subseteq D$  forms a cluster in  $D$  only if for any partition  $C = C_1 \cup C_2$ , the closest data point  $d$  to  $C_1$ ,  $d \in D - C_1$ , is from  $C_2$ .” Then any cluster  $C$  corresponds to a subtree of its MST. That is, “if  $d_1$  and  $d_2$  are two points of a cluster  $C$ , then all data points in the tree path  $P$ , connecting  $d_1$  and  $d_2$  in the MST, must be from  $C$ .”

Based on above cluster definition of the MST framework, Xu et al. [46] proposed three clustering algorithms according to different objective functions. The first objective function is to minimize the total edge weight of all  $K$  subtrees when partitioning MST into  $K$  subtrees. This can be easily realized by finding the  $K - 1$  longest MST-edges and by cutting them. If there is an outlier data-point that is far away from any cluster, this simple algorithm will identify the outlier data-point as a cluster and fail to identify real clusters when the number  $K$

is chosen improperly. The second objective function minimizes the total distance between the data in each cluster and its center:

$$\sum_{i=1}^K \sum_{d \in C_i} \text{dist}(d, \text{center}(C_i)) \quad (11.12)$$

Xu et al. proposed an iterative algorithm for this objective function. Initially, the MST is randomly partitioned into  $K$  subtrees. Then two adjacent clusters are selected and merged into one cluster. Through all the edges in the merged cluster, the edge that is cut will optimize the objective function and the merged cluster is partitioned into two clusters again. This process continues until it converges.

The third objective function  $i$  minimizes the distance between the data in a cluster and its representative points.

$$\sum_{i=1}^K \sum_{d \in C_i} \text{dist}(d, \text{represent}(C_i)) \quad (11.13)$$

A dynamic programming approach is used by Xu et al. [46] to find the optimal  $K$  clusters and their  $K$  representative data. All these algorithms are implemented in a software package called EXCAVATOR and tested for three microarray expression profiles.

#### 11.4.6 Model-Based Clustering Algorithm

Unlike the distance (similarity)-based algorithm, the model-based algorithms assume a strong structure of the underlying data set, provided that each cluster of the underlying data set is generated from a probability distribution. In the mixture model, the likelihood that data in the set  $x$  belong to these  $K$  distributions is defined as

$$L_{MIX}(\theta_1, \dots, \theta_K | x) = \prod_{i=1}^n \sum_{k=1}^K \tau_k f_k(x_i, \theta_k) \quad (11.14)$$

where  $\tau_k$  is the probability that data  $x_i$  belong to the  $k$ th cluster;  $f_k(x_i, \theta_k)$  is the probability density function (PDF) of the  $k$ th cluster in the data.  $\theta_k$  is the parameter of the probability density function. The model-based algorithms try to maximize the likelihood function over the model parameter  $\theta_k$  and the hidden parameter  $\tau_k$ . Generally, the parameters  $\theta_k$  and  $\tau_k$  are estimated by the EM algorithm.

Usually, multivariate Gaussian distribution with mean vector  $u_k$  and covariance matrix  $\Sigma_k$ :

$$f_k(x_i | u_k, \Sigma_k) = \frac{\exp\{-\frac{1}{2}(x_i - u_k)^T \Sigma_k^{-1} (x_i - u_k)\}}{\sqrt{\det(2\pi \Sigma_k)}} \quad (11.15)$$

The covariance matrix  $\Sigma_k$  controls the geometric feature of the underlying cluster. Banfield and Raftery [2] proposed a general representation of the covariance matrix through the eigenvalue decomposition:

$$\Sigma_k = \lambda_k D_k A_k D_k^T \quad (11.16)$$

where  $D_k$  is the orthogonal matrix of eigenvectors, and  $A_k$  is a diagonal matrix whose elements are proportional to the eigenvalue of  $\lambda_k$ . The matrix  $D_k$  determines the orientation of the cluster,  $A_k$  determines its shape, and  $\lambda_k$  determines its volume. The variation of parameters in equation 11.16 will lead to various models with different characteristics. For the equal volume spherical model, in which each cluster in the data set is spherically symmetric,

the covariance matrix can be simplified to  $\Sigma_k = \lambda I$ , where  $I$  is an identity matrix. If each spherical cluster has different volume, the covariance matrix is  $\Sigma_k = \lambda_k I$ . A different  $\lambda_k$  is used to control each cluster. The variation of parameters of general covariance matrix provides the flexibility to generate different models for different kinds of data set.

Yeung et al. [47] applied different models to cluster ovarian cancer microarray profiles and yeast cell cycle microarray profiles. The clustering results were shown to be comparable to the result of heuristic graph-based algorithm CAST. The advantage of model-based algorithms is that they provide a statistical framework to understand the underlying structure of the gene expression profiles. However, the assumption of model-based clustering that data fit in a specific distribution may not be true in some cases. For example, Yeung et al. [47] found that gene expression data fit the Gaussian model poorly. Finding a more proper model for gene expression data is still an ongoing effort.

### 11.4.7 Subspace Clustering

As the increased application of microarray technologies is generating a huge amount of data, it is easy to collect hundreds or even thousands microarray gene expression experiment data for one genome. For these kinds of high-dimensional gene expression profiles, all the above clustering algorithms that group data based on all dimensions become extremely difficult to implement. First, as the dimensions increase, the irrelevance between dimensions also increases, which will mask the true cluster in noise and eliminate the clustering tendency. Second, there is the *curse of dimensionality*. In high dimensions, the similarity measures become increasingly meaningless. Beyer et al. [6] showed that the distance to the nearest neighbor becomes indistinguishable from the distance to the majority of the points. Then, to find useful clusters, the clustering algorithm must work only on the relevant dimensions. At the same time, it is well known in biology that only a small subset of the genes participates in a certain cellular process. Then, even for a subset of the experiments, only parts of expressions are meaningful. Furthermore, one single gene may participate in multiple processes and can be related to different genes at different process. Thus, it can join different clusters at different processes. So, clustering on all dimensions of high-dimension microarray expression profiles is not meaningful. Recently, several subspace clustering algorithms [33] have been proposed to fit the requirement of mining high-dimensional gene expression profiles.

The subspace clustering concept was first introduced by Agrawal et al. [1] in a general data mining area. In their CLIQUE algorithm, each dimension is divided into a number of equal-length intervals  $\xi$ , and a unit in subspaces is the intersection of intervals in each dimension of sub spaces. A unit is dense if the number of points in it is above a certain threshold  $\tau$ . The CLIQUE discovers the dense units in  $k$ -dimension subspaces from the dense units in  $k - 1$  dimensional subspaces. Next, a cluster is defined as a maximal set of connected dense units. The results of CLIQUE are a series of clusters in different subspaces. The CLIQUE is the first algorithm that combined the clustering with the attribute selection and provides a different view of the data.

A number of subspace clustering algorithms have been proposed to discover subclusters (high coherence submatrix) within high-dimensional gene expression profiles. Here, we only introduce two pioneering works. For more subspace clustering algorithms on gene expression profiles, a detail review is available in [29].

### Coupled Two-Way Clustering (CTWC)

For a microarray expression profile with features  $F$  and objects  $O$ , where  $F$  and  $O$  can be both genes and experiments, Getz et al. [17] proposed a coupled two-way clustering (CTWC) algorithm to find a stable subset ( $F_i$  and  $O_j$ ) of these profiles. A stable subset means that the  $O_j$  will be significant according to some predefined threshold when using only the features  $F_i$  to clustering objects  $O_j$ . The CTWC algorithm iteratively processes two-way clustering to find the stable subset. A clustering algorithm based on the Pott model of statistical physics called the superparamagnetic clustering algorithm (SPC) [7] is used for two-way clustering. Initially, two-way clustering is applied to the full gene expression matrix and generates a set of gene clusters ( $g_i^1$ ) and a set of sample clusters ( $e_j^1$ ). Then, the two-way clustering is applied to submatrices defined by the combination of one of the previously generated gene clusters with one of the previously generated sample clusters. This iteration continues until no new clusters satisfy some predefined criterion.

Getz et al. [17] applied the CTWC algorithm to an acute leukemia microarray profile and a colon caner microarray profile. For the first data set, the CTWC discovered 49 stable gene clusters and 35 stable sample clusters in two iterations. For the second data set, 76 stable sample clusters and 97 stable gene clusters were generated. Several conditionally related gene clusters have been identified, which cannot be identified when all of the samples are used to cluster genes. However, the CTWC also generated some meaningless clusters.

### Biclustering

Given a gene expression profile  $A$  with a set of  $X$  genes and a set of  $Y$  samples, Cheng and Church [8] defined a submatrix  $A_{I,J}(I \subset X, J \subset Y)$  in  $A$  with a high similarity score as a *bicluster*. Cheng and Church introduce the *residue* to measure the coherence of the gene expression. For each element  $a_{ij}$  in the submatrix  $A_{I,J}$ , its mean-squared residue (MSR) is defined as:

$$r_{-}a_{ij} = a_{ij} - a_{iJ} - a_{IJ} + a_{IJ} \tag{11.17}$$

where  $a_{iJ}$  is the mean of the  $i$ th row in the  $A_{I,J}$ ,  $a_{IJ}$  is the mean of the  $j$ th column of  $A_{I,J}$ , and  $a_{IJ}$  is the mean of all elements in the  $A_{I,J}$ . The value of the MSR indicates the coherence of an expression relative to the remaining expressions in the bicluster  $A_{I,J}$  given the biases of the relevant rows and the relevant columns. The lower the residue is, the stronger the coherence will be. Next, the MSR of the submatrix  $A_{I,J}$  can be defined as:

$$MSR_{-}A_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (MSR_{-}a_{ij})^2 \tag{11.18}$$

A submatrix  $A_{I,J}$  is called a  $\delta$ -bicluster if  $MSR_{-}A_{IJ} \leq \delta$  for some  $\delta \geq 0$ .

Cheng and Church [8] showed that finding the largest square  $\delta$ -bicluster is NP-hard and proposed several heuristic greedy row/column removal/addition algorithms to reduce the complexity to polynomial time. The single node deletion algorithm iteratively removes the row or column that gives the most decrease in mean-squared residue. The multiple node deletion algorithm iteratively removes the rows and columns whose mean-squared residues are greater than  $\alpha \times msr_{-}A_{IJ}$ , where  $\alpha > 1$  is a predefined threshold. After the deletion algorithm terminates, a node addition algorithm adds rows and columns that do not increase the mean-squared residue of the bicluster. The algorithm finds one bicluster at a time. After a bicluster is found, the elements of the bicluster are replaced by random values in the original expression matrix. Each iteration bicluster is processed on the full expression matrix. The process will continue



until  $K$  prerequired biclusters are found. Masking elements of already-identified biclusters by random noise will let the biclusters that are already discovered not be reported again. However, highly overlapping biclusters may also not be discovered.

Cheng and Church [8] demonstrated the biclustering algorithm on yeast microarray profiles and a human gene microarray profiles. For each data set, 100 biclusters were discovered.

## 11.5 Cluster Evaluation

The cluster validation evaluates the quality of the clustering result, and then finds the best partition of the underlying data set. A detail review of cluster validation algorithms appears in [18]. An optimal clustering result can be evaluated by two criteria [5]. One is compactness; the data inside the same cluster should be close to each other. The other is separation; different clusters should be separated as wide as possible. Generally, three kinds of approaches have been used to validate the clustering result [41]: the approach based on external criteria, the approach based on internal criteria, and the approach based on relative criteria.

The clustering validation using external criteria is based on the null hypothesis, which represents a random structure of a data set. It evaluates the resulting clustering structure by comparing it to an independent partition of the data built according to the null hypothesis of the data set. This kind of test leads to high computation costs. Generally, the Monte Carlo techniques are suitable for the high computation problem and generate the needed probability density function. The clustering validation using internal criteria is to evaluate the clustering result of an algorithm using only quantities and features inherent to the data set [18]. Internal criteria can be applied to two cases when the cluster validity depends on the clustering structure: one is the hierarchy of clustering schemes, and the other is the single clustering scheme [18].

Because of their low computational cost, the clustering validation using relative criteria is more commonly used. Usually the procedure of identifying the best clustering scheme is based on a validity index. In the plotting of validity index versus the number of clusters  $N_c$ , if the validity index does not exhibit an increasing or decreasing trend as  $N_c$  increases, the maximum (minimum) of the plot indicates the best clustering. On the other hand, for indices that increase or decrease as  $N_c$  increases, the value of  $N_c$  at which a significant local change takes place indicates the best clustering. This change appears as a “knee” in the plot, and it is an indication of the number of clusters underlying the data set. Moreover, the absence of a knee may be an indication that the data set has no cluster type structure.

For gene expression profile clustering, there is a fourth clustering validation approach that is based on the biological significance of the clustering results. Several clustering validity indices are presented here.

### 11.5.1 Dunn Indices

Dunn [12] proposed a cluster validity index to identify clusters. The index for a specific number of clusters is defined as

$$D_{N_c} = \min_{i=1, \dots, N_c} \left\{ \min_{j=1, \dots, N_c} \left\{ \frac{d(c_i, c_j)}{\max_{k=1, \dots, N_c} \text{diam}(c_k)} \right\} \right\} \quad (11.19)$$

where  $d(c_i, c_j)$  is the dissimilarity function between two clusters  $c_i$  and  $c_j$  defined as

$$d(c_i, c_j) = \min_{x \in c_i, y \in c_j} d(x, y) \tag{11.20}$$

and  $diam(c)$  is the diameter of a cluster  $c$ , which measures the dispersion of clusters. The diameter of cluster  $c$  can be defined as

$$diam(c) = \max_{x, y \in c} d(x, y) \tag{11.21}$$

For compact and well-separated clusters, the distance between the clusters will be large and the diameter of the clusters will be small. Then, the value of the Dunn index will be large. However, the Dunn index does not exhibit any trend with respect to the number of clusters. Thus, the maximum value in the plot of the  $DN_c$  versus the number of clusters indicate the number of compact and well-separated clusters in the data set.

### 11.5.2 The Davies-Bouldin (DB) Index

Davies and Bouldin proposed a similarity measure  $R_{ij}$  between clusters  $C_i$  and  $C_j$  based on a dispersion measure  $s_i$  of a cluster and a dissimilarity measure,  $d_{ij}$ , between two clusters. The  $R_{ij}$  index is defined to satisfy the following conditions [9]:

- $R_{ij} \geq 0$
- $R_{ij} = R_{ji}$
- If  $s_i = 0$  and  $s_j = 0$  then  $R_{ij} = 0$
- If  $s_j > s_k$  and  $d_{ij} = d_{ik}$  then  $R_{ij} > R_{ik}$
- If  $s_j = s_k$  and  $d_{ij} < d_{ik}$  then  $R_{ij} > R_{ik}$

A  $R_{ij}$  that satisfies the above condition is nonnegative and symmetric. A simple choice for  $R_{ij}$  can be [9]

$$DB_{N_c} = \frac{1}{N_c} \sum_{i=1}^{N_c} R_i \quad R_i = \max_{i, j=1, \dots, N_c, i \neq j} R_{ij} \tag{11.22}$$

The above definition of  $DB_{N_c}$  is the average similarity between each cluster  $C_i, i = 1, \dots, N_c$  and its most similar one. Similar to Dunn index, the  $DB_{N_c}$  index exhibits no trends with respect to the number of clusters. The minimum value of  $DB_{N_c}$  in its plot versus the number of clusters indicates the best clustering.

### 11.5.3 Gap Statistics

Tibshirani et al. [42] proposed estimating the number of clusters in a data set via the gap statistic, which compares the change in within-cluster dispersion with that expected under an appropriate null distribution. The complete description of the gap statistic is as follows:

- Cluster the observed data, calculate the within-dispersion measure  $W_k$  for varying number of clusters  $k = 1, 2, \dots, K$ .

The  $W_k$  is defined as

$$W_k = \sum_{k=1}^K \frac{1}{2|C_k|} \sum_{i, j \in C_k} d_{ij} \tag{11.23}$$

- Generate  $B$  reference data sets, using the uniform prescription, and cluster each one giving within-dispersion measures  $W_{kb}^*$ ,  $b = 1, 2, \dots, B$ ,  $k = 1, 2, \dots, K$ . Compute the (estimated) gap statistic:

$$Gap(k) = (1/B) \sum_b \log(W_{kb}^*) - \log(W_k) \tag{11.24}$$

- Let  $\bar{l} = (1/B) \sum_b \log(W_{kb}^*)$ , compute the standard deviation

$$sd_k = \left[ (1/B) \sum_b (\log(W_{kb}^*) - \bar{l})^2 \right]^{\frac{1}{2}},$$

and define  $s_k = sd_k \sqrt{1 + 1/B}$ . Finally, choose the number of clusters via:

$$\hat{k} = \text{smallest } k \text{ such that } Gap(k) \geq Gap(k + 1) - s_{k+1} \tag{11.25}$$

### Average Silhouette

It has been proposed (see [34, 44]), using average silhouette as a composite index to reflect the compactness and separation of the clusters. A silhouette value  $s(i)$  of data  $i$  that belong to cluster  $A$  is defined as follows

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{11.26}$$

The  $a(i)$  is the average distance between data  $i$  and other data in the same cluster  $A$ :

$$a(i) = \frac{1}{|A| - 1} \sum_{j \in A, j \neq i} d(i, j) \tag{11.27}$$

where  $d(i, j)$  is the distance between data  $i$  and  $j$  and  $|A|$  is the number of data in cluster  $A$ .

The  $b(i)$  is the minimum value of average distance of data  $i$  to data in any cluster  $C$  other than cluster  $A$ .

$$b(i) = \min_{C \neq A} \{d(i, C)\} \tag{11.28}$$

and

$$d(i, C) = \frac{1}{|C|} \sum_{j \in C} d(i, j) \tag{11.29}$$

The average silhouette is the average of the silhouette value of all data in the cluster. The value of the average silhouette lies between -1 and 1. If the average silhouette is great than 0, the cluster is valid. If it is less than 0, the data in this cluster on average are closer to members of some other clusters, making this cluster invalid.

### 11.5.4 Biological Significance-Based on $P$ -Value

Tavazoie et al. [39] proposed evaluating the clustering result based on the biological significance of each category. First, genes are assigned categories based on gene ontology or based on the categories of some public database. Then, for each cluster, a  $P$ -value can be calculated, which is the probability of observing the frequency of genes in a particular functional category in a certain cluster, using the cumulative hypergeometric probability distribution [39]. The  $P$ -value of observing  $k$  genes from a function category within a cluster of size  $n$  is

$$P = 1 - \sum_{i=0}^{k-1} \frac{\binom{f}{i} \binom{g-f}{n-i}}{\binom{g}{n}} \quad (11.30)$$

where  $f$  is the total number of genes within that functional category and  $g$  is the total number of genes within the genome (for yeast,  $n$  equals to 6220). The lower the  $P$ -value is, the higher the biological significance of a cluster.

## 11.6 Discussion

In this chapter, we reviewed clustering analysis for the gene expression profiles. The list of clustering algorithms presented is very limited but representative. Because of the fact that the research in this area is very active, a list of all algorithms developed for clustering gene expression profiles will be prohibitively long.

For biologists, clustering analysis is only the first step. Biologists want the results from the clustering to provide clues for the further biological research. Because false information will cost a lot of time and money for wet experiments, more accurate results are an absolute necessity. Since only a portion of the genes are expressed under most experimental conditions, to obtain a biologically meaningful result the clustering algorithms need to artificially assign a threshold for controlling the quality of the clusters. To determine the threshold, biological knowledge is needed. For some biological organisms, like yeast, *E. coli*, and *Caenorhabditis elegans*, knowledge of the genes may be enough to determine the threshold. But for most biological organisms knowledge of the genes is limited, so the threshold is determined artificially. The threshold may be either too spurious, which may produce false information, or too strict, which may cause the loss of true information. Moreover, the changing of the threshold will result in different clusters of modules [3]. Finding an objective criterion for deciding what is a biologically meaningful cluster is certainly one of the most important open problems.

With the wide application of the microarray technology, more genome-size gene expression experimental data for an organism are becoming available. Because of the *curse of dimensionality*, traditional clustering algorithms are not suitable for analyzing these high-dimensional data sets. Subspace clustering, which tries to find density sub-“blocks” in the high-dimensional data, is not only appropriate for the purpose of mining high-dimensional large gene expression profiles but also consistent with the knowledge from biology that only part of the genes is expressed at certain experimental conditions. Designing an efficient subspace clustering algorithm that can discover biologically meaningful clusters still is an important research direction.

One other important issue in clustering analysis is how to visualize the clustering result. A good clustering algorithm without a good method to visualize the result will limit its usage. The reason that hierarchical agglomerative clustering is most commonly used in biological data analysis is that it can be visualized by dendrography, which is easily understood by biologists. Developing new methods and tools for visualization of clustering result of biological data can lead to widespread application of the clustering algorithms.

## References

- [1] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.
- [2] J. Banfield, A. Raftery. Model based Gaussian and non-Gaussian clustering. *Biometrics*, 49, 803–821, 1993.
- [3] A.-L. Barabasi, Z.N. Oltvai. Network biology: understanding the cell’s functional organization. *Nature Review*, 5, 101–114, 2004.
- [4] A. Ben-Dor, R. Shamir, Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6, 281–297, 1999.
- [5] M.J.A. Berry, G. Linoff. *Data Mining Techniques For Marketing, Sales and Customer Support*. New York: John Wiley & Sons, USA, 1996.
- [6] K.S. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft. When is “nearest neighbor” meaningful? In *Proceedings of the 7th ICDT*, Jerusalem, Israel, pages 217–235, 1999.
- [7] M. Blat, S. Wiseman, E. Domany. Superparamagnetic clustering of Data, *Physical Review Letters*, 76(18), 3252–3254, 1996.
- [8] Y. Cheng, G.M. Church. Biclustering of expression data. In *Proceedings of ISMB 2000*, pages 93–103, 2000.
- [9] D.L. Davies, D.W. Bouldin. A Cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2), 224–227, 1979.
- [10] C. Debouck, P.N. Goodfellow. DNA microarrays in drug discovery and development. *Nature Genetics supplement*, 21, 48–50, 1999.
- [11] J. Dopazo, J.M. Carazo. Phylogenetic reconstruction using an unsupervised growing neural network that adopts the topology of a phylogenetic Tree. *Journal of Molecular Evolution*, 44, 226–233, 1997.
- [12] J.C. Dunn. Well separated clusters and optimal fuzzy partitions. *J. Cybern.*, 4, 95–104, 1974.
- [13] B. Efron, T. Jackknife. The Bootstrap, and Other Resampling Plans. *CBMS-NSF Regional Conference Series in Applied Mathematics*, 38, 1982.
- [14] M.B. Eisen, P.T. Spellman, P.O. Brown, D. Botstein. Cluster analysis and display of genomewide expression patterns. *Proc. Natl. Acad. Sci.*, 95, 14863–14868, 1998.
- [15] <http://rana.lbl.gov/EisenSoftware.htm>.
- [16] B. Fritzke. Growing cell structures— a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7, 1141–1160, 1994.
- [17] G. Getz, E. Levine E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proc. Natl. Acad. Sci.*, 97, 22, 12079–12084, 2000.
- [18] M. Halkidi, Y. Batistakis, M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17, 107–145, 2001.
- [19] L.H. Hartwell, J.J. Hopfield, S. Leibler, A.W. Murray. From molecular to modular cell biology. *Nature*, 402, C47–C52, 1999.
- [20] J. Herrero, A. Valencia, J. Dopazo. A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17, 126–136, 2001.
- [21] J. Herrero, F.A. Shahrou, R.D. Uriarte et al. GEPAS: a web-based resource for microarray gene expression data analysis. *Nucleic Acids Research*, 31(13), 3461–3467, 2003.
- [22] L.J. Heyer, S. Kruglyak, S. Yooseph. Exploring expression data: identification and analysis of coexpressed Genes. *Genome Research*, 9, 1106–1115, 1999.
- [23] A.K. Jain, M.N. Murty, P.J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3), 264–323, 1999.

- [24] T. Kohonen. *Self-Organizing Maps*. 2nd. New York: Springer 1997.
- [25] G.N. Lance, W.T. Williams. A general theory of classificatory sorting strategies: 1. Hierarchical systems. *Computer Journal*, 9, 373–380, 1966.
- [26] D.J. Lockhart, H. Dong, M.C. Byrne, et al. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology*, 14, 1675–1680, 1996.
- [27] F. Luo, L. Khan, F. Bastani, I.L. Yen. *A dynamical growing self-organizing tree (DG-SOT)*. Technical Report, University of Texas at Dallas, 2003.
- [28] F. Luo, L. Khan, I.L. Yen, F. Bastani, J. Zhou. A dynamical growing self-organizing tree (DGSOT) for hierarchical clustering gene expression profiles. *Bioinformatics*, 20(16), 2605–2617, 2004.
- [29] S.C. Madeira, A.L. Oliveira. Biclustering algorithm for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. 1(1), 1–30, 2004.
- [30] J.B. McQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, University of California, Berkeley, 1967.
- [31] J.M. Naciff, G.J. Overmann, S.M. Torontali, et al. Gene expression Profile induced by 17 $\alpha$ -ethynyl estradiol in the prepubertal female reproductive system of the rat. *Toxicological Science*, 72, 314–330, 2003.
- [32] S.T. Nadler, J.P. Stoehr, K.L. Schueler, et al. The expression of adipogenic genes is decreased in obesity and Diabetes mellitus. *Proc. Natl. Acad. Sci.*, 97, 1371–1376, 2002.
- [33] L. Parsons, E. Haque, H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining*, 2004.
- [34] P.J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65, 1987.
- [35] M. Schena, D. Shalon, R. Davis, P. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270, 467–470, 1995.
- [36] R. Shamir, R. Sharan. Click: a clustering algorithm for gene expression analysis. In *Proceedings of ISMB 2000*, pages 307–316, 2000.
- [37] F. Smet, J. Mathys, K. Marchal, G. Thijs, Y. Moreau. Adaptive quality-based clustering of gene expression profiles. *Bioinformatics*, 18, 735–746, 2002.
- [38] P. Tamayo, D. Slonim, J. Mesirov, et al. Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci.*, 96, 2907–2912, 1999.
- [39] S. Tavazoie, J.D. Hughes, M.J. Campbell, et al. Systematic determination of genetic network architecture. *Nature Genetics*, 22, 281–285, 1999.
- [40] A. Tefferi, M. E. Bolander, S. M. Ansell, et al. Primer on medical genomics part III: microarray experiments and data Analysis. *Mayo Clinic Proc.*, 77, 927–940, 2002.
- [41] S. Theodoridis, K. Koutroubas. *Pattern Recognition*. New York: Academic Press, 1999.
- [42] R. Tibshirani, G. Walther, T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society, Series B*, 63, 411–423, 2001.
- [43] O. Troyanskaya, M. Cantor, G. Sherlock, et al. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6), 520–525, 2001.
- [44] J. Vilo, A. Brazma, I. Jonssen, A. Robinson, E. Ukkonen. Mining for putative regulatory elements in the yeast genome using gene expression data. *Proceedings of ISMB 2000*, 384–394, 2000.

- [45] E.M. Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing & Management*, 22(6), 465–476, 1986.
- [46] Y. Xu, V. Olmam, D. Xu. Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees. *Bioinformatics*, 18, 536–545, 2002.
- [47] K.Y. Yeung, C. Fraley, A. Murua, A.E. Raftery, W.L. Ruzzo. Model-based clustering and data transformations for gene expression data. *Bioinformatics*, 17, 977–987, 2001.