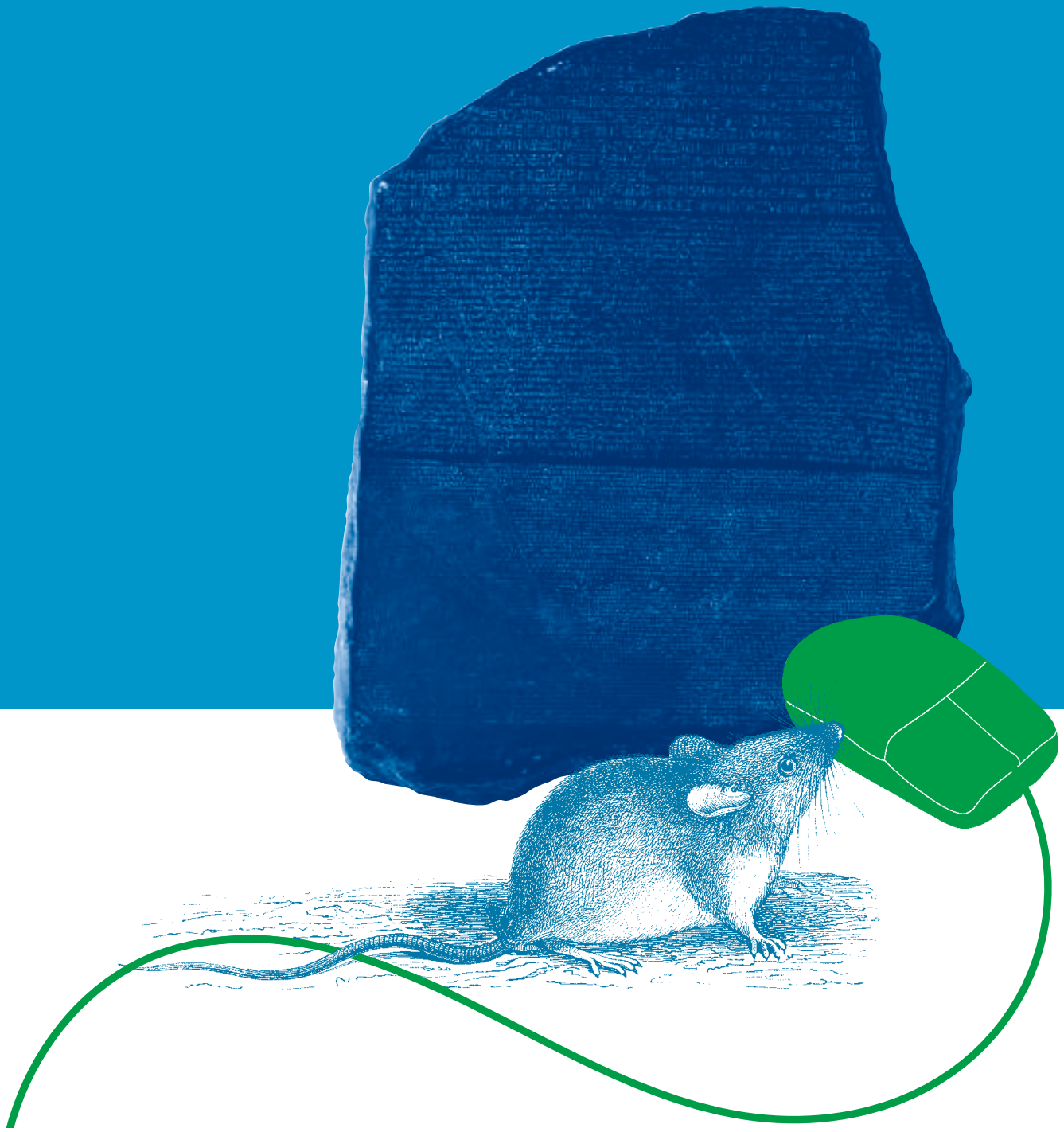
AN INTRODUCTION TO **BIOINFORMATICS ALGORITHMS**

NEIL C. JONES AND PAVEL A. PEVZNER



10 *Clustering and Trees*

A common problem in biology is to partition a set of experimental data into groups (clusters) in such a way that the data points within the same cluster are highly similar while data points in different clusters are very different. This problem is far from simple, and this chapter covers several algorithms that perform different types of clustering. There is no simple recipe for choosing one particular approach over another for a particular clustering problem, just as there is no universal notion of what constitutes a "good cluster." Nonetheless, these algorithms can yield significant insight into data and allow one, for example, to identify clusters of genes with similar functions even when it is not clear what particular role these genes play. We conclude this chapter with studies of evolutionary tree reconstruction, which is closely related to clustering.

10.1 Gene Expression Analysis

Sequence comparison often helps to discover the function of a newly sequenced gene by finding similarities between the new gene and previously sequenced genes with known functions. However, for many genes, the sequence similarity of genes in a functional family is so weak that one cannot reliably derive the function of the newly sequenced gene based on sequence alone. Moreover, genes with the same function sometimes have no sequence similarity at all. As a result, the functions of more than 40% of the genes in sequenced genomes are still unknown.

In the last decade, a new approach to analyzing gene functions has emerged. DNA arrays allow one to analyze the *expression levels* (amount of mRNA produced in the cell) of many genes under many time points and conditions and

to reveal which genes are switched on and switched off in the cell.[1] The outcome of this type of study is an $n \times m$ *expression matrix* $\mathbf{I}$, with the n rows corresponding to genes, and the m columns corresponding to different time points and different conditions. The expression matrix $\mathbf{I}$ represents *intensities* of hybridization signals as provided by a DNA array. In reality, expression matrices usually represent transformed and normalized intensities rather than the raw intensities obtained as a result of a DNA array experiment, but we will not discuss this transformation.

The element $I_{i,j}$ of the expression matrix represents the expression level of gene i in experiment j; the entire ith row of the expression matrix is called the *expression pattern* of gene i. One can look for pairs of genes in an expression matrix with similar expression patterns, which would be manifested as two similar rows. Therefore, if the expression patterns of two genes are similar, there is a good chance that these genes are somehow related, that is, they either perform similar functions or are involved in the same biological process.[2] Accordingly, if the expression pattern of a newly sequenced gene is similar to the expression pattern of a gene with known function, a biologist may have reason to suspect that these genes perform similar or related functions. Another important application of expression analysis is in the deciphering of regulatory pathways; similar expression patterns usually imply coregulation. However, expression analysis should be done with caution since DNA arrays typically produce noisy data with high error rates.

Clustering algorithms group genes with similar expression patterns into *clusters* with the hope that these clusters correspond to groups of functionally related genes. To cluster the expression data, the $n \times m$ expression matrix is often transformed into an $n \times n$ *distance matrix* $\mathbf{d} = (d_{i,j})$ where $d_{i,j}$ reflects how similar the expression patterns of genes i and j are (see figure 10.1).[3] The goal of clustering is to group genes into clusters satisfying the following two conditions:

- *Homogeneity.* Genes (rather, their expression patterns) within a cluster

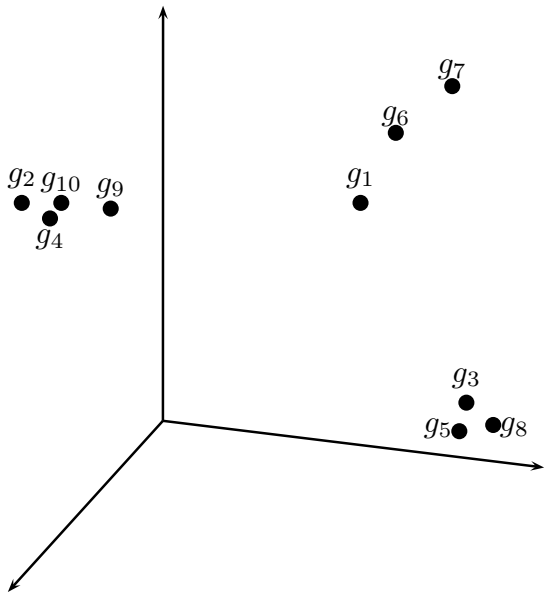
1. Expression analysis studies implicitly assume that the amount of mRNA (as measured by a DNA array) is correlated with the amount of its protein produced by the cell. We emphasize that a number of processes affect the production of proteins in the cell (transcription, splicing, translation, post-translational modifications, protein degradation, etc.) and therefore this correlation may not be straightforward, but it is still significant.
2. Of course, we do not exclude the possibility that expression patterns of two genes may look similar simply by chance, but the probability of such a chance similarity decreases as we increase the number of time points.
3. The distance matrix is typically larger than the expression matrix since $n \gg m$ in most expression studies.

Time	1 hr	2 hr	3 hr
g_1	10.0	8.0	10.0
g_2	10.0	0.0	9.0
g_3	4.0	8.5	3.0
g_4	9.5	0.5	8.5
g_5	4.5	8.5	2.5
g_6	10.5	9.0	12.0
g_7	5.0	8.5	11.0
g_8	2.7	8.7	2.0
g_9	9.7	2.0	9.0
g_{10}	10.2	1.0	9.2

(a) Intensity matrix, $\mathbf{I}$

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0

(b) Distance matrix, $\mathbf{d}$



(c) Expression patterns as points in three-dimentsional space.

Figure 10.1 An "expression" matrix of ten genes measured at three time points, and the corresponding distance matrix. Distances are calculated as the Euclidean distance in three-dimensional space.

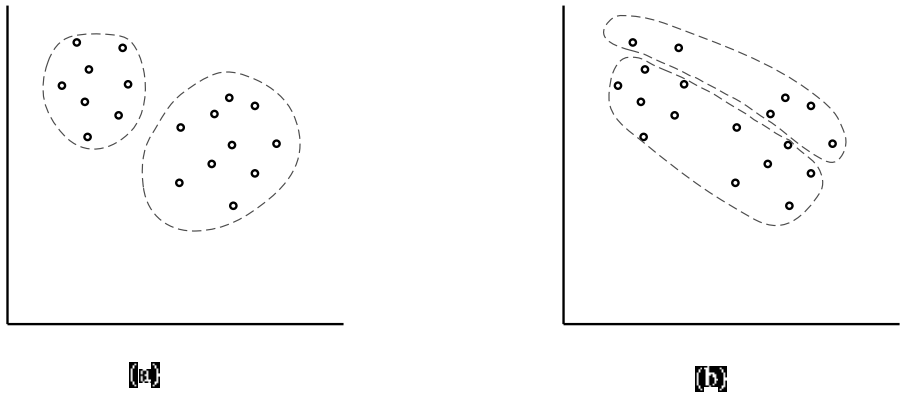


Figure 10.2 Data can be grouped into clusters. Some clusters are better than others: the two clusters in a) exhibit good homogeneity and separation, while the clusters in b) do not.

should be highly similar to each other. That is, $d_{i,j}$ should be small if i and j belong to the same cluster.

- *Separation.* Genes from different clusters should be very different. That is, $d_{i,j}$ should be large if i and j belong to different clusters.

An example of clustering is shown in figure 10.2. Figure 10.2 (a) shows a good partition according to the above two properties, while (b) shows a bad one. Clustering algorithms try to find a good partition.

A "good" clustering of data is one that adheres to these goals. While we hope that a better clustering of genes gives rise to a better grouping of genes on a functional level, the final analysis of resulting clusters is left to biologists.

Different tissues express different genes, and there are typically over 10,000 genes expressed in any one tissue. Since there are about 100 different tissue types, and since expression levels are often measured over many time points, gene expression experiments can generate vast amounts of data which can be hard to interpret. Compounding these difficulties, expression levels of related genes may vary by several orders of magnitude, thus creating the problem of achieving accurate measurements over a large range of expression levels; genes with low expression levels may be related to genes with high expression levels.

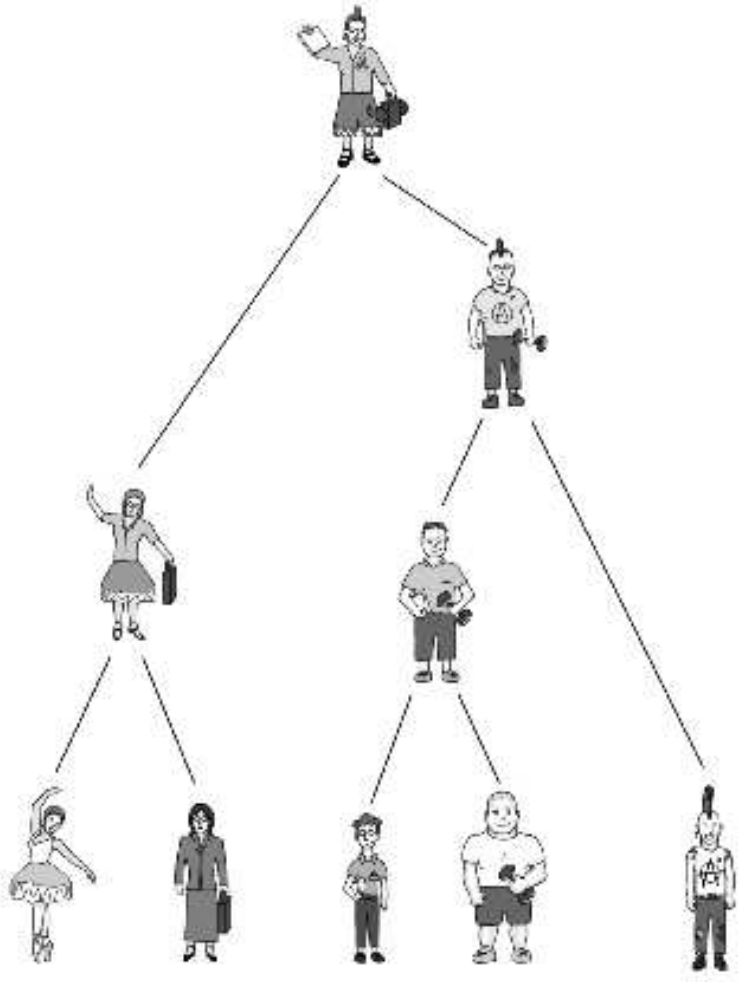


Figure 10.3 A schematic of hierarchical clustering.

10.2 Hierarchical Clustering

In many cases clusters have subclusters, these have subsubclusters, and so on. For example, mammals can be broken down into primates, carnivora, bats, marsupials, and many other orders. The order carnivora can be further broken down into cats, hyenas, bears, seals, and many others. Finally, cats can be broken into thirty seven species.[4]

4. Lions, tigers, leopards, jaguars, lynx, cheetahs, pumas, golden cats, domestic cats, small wildcats, ocelots, and many others.

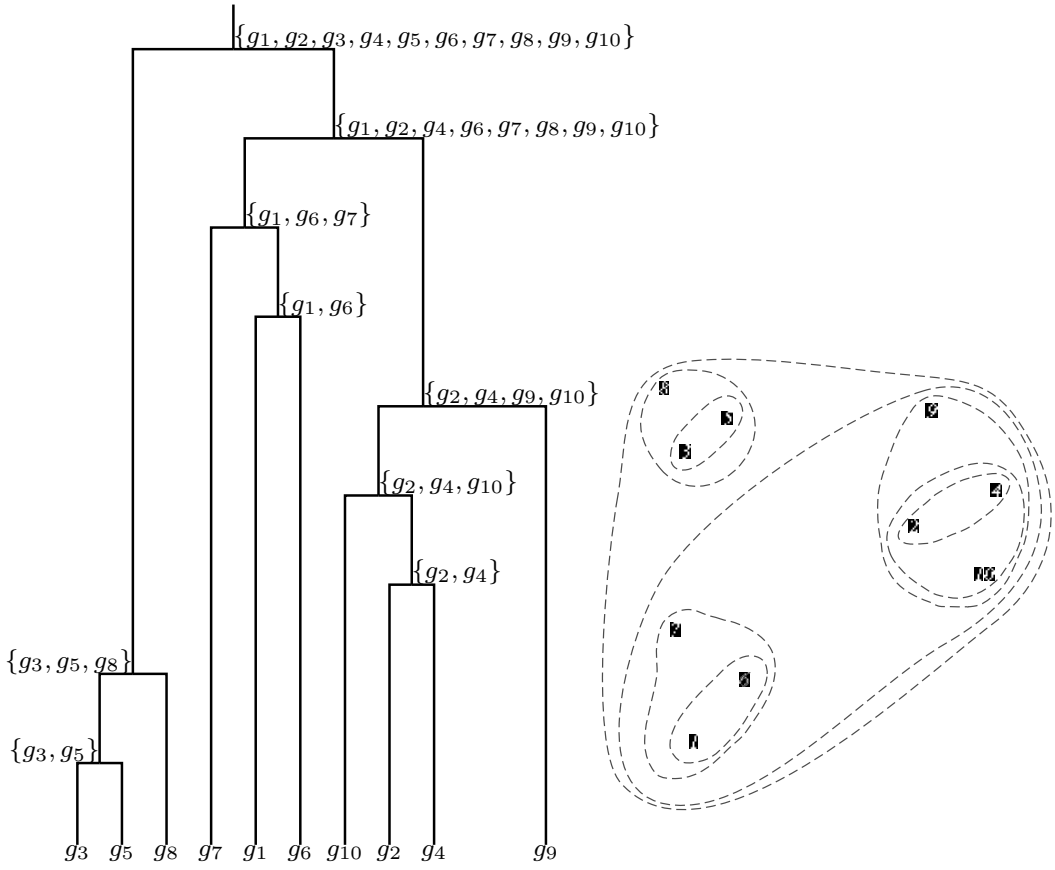


Figure 10.4 A hierarchical clustering of the data in figure 10.1

Hierarchical clustering (fig. 10.3) is a technique that organizes elements into a *tree*, rather than forming an explicit partitioning of the elements into clusters. In this case, the genes are represented as the leaves of a tree. The edges of the trees are assigned *lengths* and the distances between leaves—that is, the length of the path in the tree that connects two leaves—correlate with entries in the distance matrix. Such trees are used in both the analysis of expression data and in studies of molecular evolution which we will discuss below.

Figure 10.4 shows a tree that represents clustering of the data in figure 10.1. This tree actually describes a family of different partitions into clusters, each with a different number of clusters (one for each value from 1 to n). You can see what these partitions by drawing a horizontal line through the tree. Each line crosses the tree at i points ($1 \leq i \leq k$) and correspond to i clusters.

The HIERARCHICALCLUSTERING algorithm below takes an $n \times n$ distance matrix $\mathbf{d}$ as an input, and progressively generates n different partitions of the data as the tree it outputs. The largest partition has n single-element clusters, with every element forming its own cluster. The second-largest partition combines the two closest clusters from the largest partition, and thus has $n - 1$ clusters. In general, the ith partition combines the two closest clusters from the $(i - 1)$th partition and has $n - i + 1$ clusters.

```
HIERARCHICALCLUSTERING(d, n)
 1  Form n clusters, each with 1 element
 2  Construct a graph T by assigning an isolated vertex to each cluster
 3  while there is more than 1 cluster
 4      Find the two closest clusters C1 and C2
 5      Merge C1 and C2 into new cluster C with |C1| + |C2| elements
 6      Compute distance from C to all other clusters
 7      Add a new vertex C to T and connect to vertices C1 and C2
 8      Remove rows and columns of d corresponding to C1 and C2
 9      Add a row and column to d for the new cluster C
10  return T
```

Line 6 in the algorithm is (intentionally) left ambiguous; clustering algorithms vary in how they compute the distance between the newly formed cluster and any other cluster. Different formulas for recomputing distances yield different answers from the same hierarchical clustering algorithm. For example, one can define the distance between two clusters as the smallest distance between any pair of their elements

$$d_{min}(C^*, C) = \min_{x \in C^*, y \in C} d(x, y)$$

or the average distance between their elements

$$d_{avg}(C^*, C) = \frac{1}{|C^*||C|} \sum_{x \in C^*, y \in C} d(x, y).$$

Another distance function estimates distance based on the separation of C_1 and C_2 in HIERARCHICALCLUSTERING:

$$d(C^*, C) = \frac{d(C^*, C_1) + d(C^*, C_2) - d(C_1, C_2)}{2}$$

In one of the first expression analysis studies, Michael Eisen and colleagues used hierarchical clustering to analyze the expression profiles of 8600 genes over thirteen time points to find the genes responsible for the growth response of starved human cells. The HIERARCHICALCLUSTERING resulted in a tree consisting of five main subtrees and many smaller subtrees. The genes within these five clusters had similar functions, thus confirming that the resulting clusters are biologically sensible.

10.3 *k*-Means Clustering

One can view the n rows of the $n \times m$ expression matrix as a set of n points in m-dimensional space and partition them into k subsets, pretending that k—the number of clusters—is known in advance.

One of the most popular clustering methods for points in multidimensional spaces is called *k-Means clustering*. Given a set of n data points in m-dimensional space and an integer k, the problem is to determine a set of k points, or *centers*, in m-dimensional space that minimize the *squared error distortion* defined below. Given a data point v and a set of k centers $\mathcal{X} = \{x_1, \ldots x_k\}$, define the distance from v to the centers $\mathcal{X}$ as the distance from v to the closest point in $\mathcal{X}$, that is, $d(v, \mathcal{X}) = \min_{1 \leq i \leq k} d(v, x_i)$. We will assume for now that $d(v, x_i)$ is just the Euclidean[5] distance in m dimensions. The squared error distortion for a set of n points $\mathcal{V} = \{v_1, \ldots v_n\}$, and a set of k centers $\mathcal{X} = \{x_1, \ldots x_k\}$, is defined as the mean squared distance from each data point to its nearest center:

$$d(\mathcal{V}, \mathcal{X}) = \frac{\sum_{i=1}^{n} d(v_i, \mathcal{X})^2}{n}$$

---

k-Means Clustering Problem:
Given n data points, find k center points minimizing the squared error distortion.

Input: A set, $\mathcal{V}$, consisting of n points and a parameter k.

Output: A set $\mathcal{X}$ consisting of k points (called centers) that minimizes $d(\mathcal{V}, \mathcal{X})$ over all possible choices of $\mathcal{X}$.

---

5. Chapter 2 contains a sample algorithm to calculate this when m is 2.

While the above formulation does not explicitly address *clustering* n points into k clusters, a clustering can be obtained by simply assigning each point to its closest center. Although the k-Means Clustering problem looks relatively simple, there are no efficient (polynomial) algorithms known for it. The *Lloyd* k-Means clustering algorithm is one of the most popular clustering heuristics that often generates good solutions in gene expression analysis. The Lloyd algorithm randomly selects an arbitrary partition of points into k clusters and tries to improve this partition by moving some points between clusters. In the beginning one can choose arbitrary k points as "cluster representatives." The algorithm iteratively performs the following two steps until either it converges or until the fluctuations become very small:

- Assign each data point to the cluster C_i corresponding to the closest cluster representative x_i $(1 \leq i \leq k)$
- After the assignments of all n data points, compute new cluster representatives according to the center of gravity of each cluster, that is, the new cluster representative is $\frac{\sum_{v \in C} v}{|C|}$ for every cluster C.

The Lloyd algorithm often converges to a local minimum of the squared error distortion function rather than the global minimum. Unfortunately, interesting objective functions other than the squared error distortion lead to similarly difficult problems. For example, finding a good clustering can be quite difficult if, instead of the squared error distortion ($\sum_{i=1}^{n} d(v_i, \mathcal{X})^2$), one tries to minimize $\sum_{i=1}^{n} d(v_i, \mathcal{X})$ (*k-Median problem*) or $\max_{1 \leq i \leq n} d(v_i, \mathcal{X})$ (*k-Center problem*). We remark that all of these definitions of clustering cost emphasize the homogeneity condition and more or less ignore the other important goal of clustering, the separation condition. Moreover, in some unlucky instances of the k-Means Clustering problem, the algorithm may converge to a local minimum that is *arbitrarily* bad compared to an optimal solution (a problem at the end of this chapter).

While the Lloyd algorithm is very fast, it can significantly rearrange every cluster in every iteration. A more conservative approach is to move only one element between clusters in each iteration. We assume that every partition P of the n-element set into k clusters has an associated *clustering cost*, denoted $cost(P)$, that measures the quality of the partition P: the smaller the clustering cost of a partition, the better that clustering is.[6] The squared error distortion is one particular choice of $cost(P)$ and assumes that each center

6. "Better" here is better clustering, not a better biological grouping of genes.

point is the center of gravity of its cluster. The pseudocode below implicitly assumes that $cost(P)$ can be efficiently computed based either on the distance matrix or on the expression matrix. Given a partition P, a cluster C within this partition, and an element i outside C, $P_{i \to C}$ denotes the partition obtained from P by moving the element i from its cluster to C. This move improves the clustering cost only if $\Delta(i \to C) = cost(P) - cost(P_{i \to C}) > 0$, and the PROGRESSIVEGREEDYK-MEANS algorithm searches for the "best" move in each step (i.e., a move that maximizes $\Delta(i \to C)$ for all C and for all $i \notin C$).

```
PROGRESSIVEGREEDYK-MEANS(k)
 1  Select an arbitrary partition P into k clusters.
 2  while forever
 3      bestChange ← 0
 4      for every cluster C
 5          for every element i ∉ C
 6              if moving i to cluster C reduces the clustering cost
 7                  if Δ(i → C) > bestChange
 8                      bestChange ← Δ(i → C)
 9                      i* ← i
10                      C* ← C
11      if bestChange > 0
12          change partition P by moving i* to C*
13      else
14          return P
```

Even though line 2 makes an impression that this algorithm may loop endlessly, the return statement on line 14 saves us from an infinitely long wait. We stop iterating when no move allows for an improvement in the score; this eventually has to happen.[7]

10.4 Clustering and Corrupted Cliques

A complete graph, written K_n, is an (undirected) graph on n vertices with every two vertices connected by an edge. A *clique graph* is a graph in which every connected component is a complete graph. Figure 10.5 (a) shows a

7. What would be the natural implication if there could always be an improvement in the score?