CHAPTER

Clustering Algorithms I: Sequential Algorithms 12

12.1 INTRODUCTION

In the previous chapter, our major focus was on introducing a number of proximity measures. Each of these measures gives a different interpretation of the terms *similar* and *dissimilar*, associated with the types of clusters that our clustering procedure has to reveal. In the current and the following three chapters, the emphasis is on the various clustering algorithmic schemes and criteria that are available to the analyst. As has already been stated, different combinations of a proximity measure and a clustering scheme will lead to different results, which the expert has to interpret.

This chapter begins with a general overview of the various clustering algorithmic schemes and then focuses on one category, known as sequential algorithms.

12.1.1 Number of Possible Clusterings

Given the time and resources, the best way to assign the feature vectors $\boldsymbol{x}_i$, $i = 1, \ldots, N$, of a set X to clusters would be to identify all possible partitions and to select the most sensible one according to a preselected criterion. However, this is not possible even for moderate values of N. Indeed, let $S(N, m)$ denote the number of all possible clusterings of N vectors into m groups. Remember that, by definition, no cluster is empty. It is clear that the following conditions hold [Spat 80, Jain 88]:

- $S(N, 1) = 1$
- $S(N, N) = 1$
- $S(N, m) = 0, \quad \text{for } m > N$

Let L_{N-1}^k be the list containing all possible clusterings of the $N - 1$ vectors into k clusters, for $k = m, m - 1$. The Nth vector

- Either will be added to one of the clusters of any member of L_{N-1}^m
- Or will form a new cluster to each member of L_{N-1}^{m-1}

Thus, we may write

$$S(N, m) = mS(N-1, m) + S(N-1, m-1) \tag{12.1}$$

The solutions of (12.1) are the so-called Stirling numbers of the second kind (e.g., see [Liu 68]):[1]

$$S(N, m) = \frac{1}{m!}\sum_{i=0}^{m}(-1)^{m-i}\binom{m}{i}i^N \tag{12.2}$$

Example 12.1

Assume that $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$. We seek to find all possible clusterings of the elements of X in two clusters. It is easy to deduce that

$$L_2^1 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2\}\}$$

and

$$L_2^2 = \{\{\boldsymbol{x}_1\}, \{\boldsymbol{x}_2\}\}$$

Taking into account (12.1), we easily find that $S(3, 2) = 2 \times 1 + = 3$. Indeed, the L_3^2 list is

$$L_3^2 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_3\}, \{\boldsymbol{x}_2\}\}, \{\{\boldsymbol{x}_1\}, \{\boldsymbol{x}_2, \boldsymbol{x}_3\}\}, \{\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3\}\}$$

Especially for $m = 2$, (12.2) becomes

$$S(N, 2) = 2^{N-1} - 1 \tag{12.3}$$

(see Problem 12.1). Some numerical values of (12.2) are [Spat 80]

- $S(15, 3) = 2375101$
- $S(20, 4) = 45232115901$
- $S(25, 8) = 690223721118368580$
- $S(100, 5) \simeq 10^{68}$

It is clear that these calculations are valid for the case in which the number of clusters is fixed. If this is not the case, one has to enumerate all possible clusterings for all possible values of m. From the preceding analysis, it is obvious that evaluating all of them to identify the most sensible one is impractical even for moderate values of N. Indeed, if, for example, one has to evaluate all possible clusterings of 100 objects into five clusters with a computer that evaluates each single clustering in 10^{-12} seconds, the most "sensible" clustering would be available after approximately 10^{48} years!

[1] Compare it with the number of dichotomies in Cover's theorem.

12.2 CATEGORIES OF CLUSTERING ALGORITHMS

Clustering algorithms may be viewed as schemes that provide us with sensible clusterings by considering *only a small fraction of the set containing all possible partitions of X. The result depends on the specific algorithm and the criteria used.* Thus a clustering algorithm is a learning procedure that tries to identify the specific characteristics of the clusters underlying the data set. Clustering algorithms may be divided into the following major categories.

- *Sequential algorithms.* These algorithms produce a single clustering. They are quite straightforward and fast methods. In most of them, all the feature vectors are presented to the algorithm once or a few times (typically no more than five or six times). The final result is, usually, dependent on the order in which the vectors are presented to the algorithm. These schemes tend to produce compact and hyperspherically or hyperellipsoidally shaped clusters, depending on the distance metric used. This category will be studied at the end of this chapter.
- *Hierarchical clustering algorithms.* These schemes are further divided into
  - *Agglomerative algorithms.* These algorithms produce a sequence of clusterings of decreasing number of clusters, m, at each step. The clustering produced at each step results from the previous one by merging two clusters into one. The main representatives of the agglomerative algorithms are the *single and complete link* algorithms. The agglomerative algorithms may be further divided into the following subcategories:
    - Algorithms that stem from the matrix theory
    - Algorithms that stem from graph theory

    These algorithms are appropriate for the recovery of elongated clusters (as is the case with the single link algorithm) and compact clusters (as is the case with the complete link algorithm).
  - *Divisive algorithms.* These algorithms act in the opposite direction; that is, they produce a sequence of clusterings of increasing m at each step. The clustering produced at each step results from the previous one by splitting a single cluster into two.
- *Clustering algorithms based on cost function optimization.* This category contains algorithms in which "sensible" is quantified by a cost function, J, in terms of which a clustering is evaluated. Usually, the number of clusters m is kept fixed. Most of these algorithms use differential calculus concepts and produce successive clusterings while trying to optimize J. They terminate when a local optimum of J is determined. Algorithms of this category are also called *iterative function optimization schemes.* This category includes the following subcategories:

  - *Hard or crisp clustering algorithms*, where a vector belongs exclusively to a specific cluster. The assignment of the vectors to individual clusters is carried out optimally, according to the adopted optimality criterion. The most famous algorithm of this category is the Isodata or Lloyd algorithm [Lloy 82, Duda 01].
  - *Probabilistic clustering algorithms*, are a special type of hard clustering algorithms that follow Bayesian classification arguments and each vector $\boldsymbol{x}$ is assigned to the cluster C_i for which $P(C_i|\boldsymbol{x})$ (i.e., the *a posteriori* probability) is maximum. These probabilities are estimated via an appropriately defined optimization task.
  - *Fuzzy clustering algorithms*, where a vector belongs to a specific cluster up to a certain degree.
  - *Possibilistic clustering algorithms*. In this case we measure the possibility for a feature vector $\boldsymbol{x}$ to belong to a cluster C_i.
  - *Boundary detection algorithms.* Instead of determining the clusters by the feature vectors themselves, these algorithms adjust iteratively the boundaries of the regions where clusters lie. These algorithms, although they evolve from a cost function optimization philosophy, are different from the above algorithms. All the aforementioned schemes use cluster representatives, and the goal is to locate them in space in an optimal way. In contrast, boundary detection algorithms seek ways of placing optimally boundaries between clusters. This has led us to the decision to treat these algorithms in a separate chapter, together with algorithms to be discussed next.

- *Other*: This last category contains some special clustering techniques that do not fit nicely in any of the previous categories. These include:
  - *Branch and bound clustering algorithms.* These algorithms provide us with *the globally optimal clustering without having to consider all possible clusterings*, for fixed number m of clusters, and for a prespecified criterion that satisfies certain conditions. However, their computational burden is excessive.
  - *Genetic clustering algorithms.* These algorithms use an initial population of possible clusterings and iteratively generate new populations, which, in general, contain better clusterings than those of the previous generations, according to a prespecified criterion.
  - *Stochastic relaxation methods.* These are methods that guarantee, under certain conditions, convergence in probability to the globally optimum clustering, with respect to a prespecified criterion, at the expense of intensive computations.

It must be pointed out that stochastic relaxation methods (as well as genetic algorithms and branch and bound techniques) are cost function optimization methods. However, each follows a conceptually different approach to the problem compared to the methods of the previous category. This is why we chose to treat them separately.

- *Valley-seeking clustering algorithms.* These algorithms treat the feature vectors as instances of a (multidimensional) random variable $\boldsymbol{x}$. They are based on the commonly accepted assumption that regions of $\boldsymbol{x}$ where many vectors reside correspond to regions of increased values of the respective probability density function (pdf) of $\boldsymbol{x}$. Therefore, the estimation of the pdf may highlight the regions where clusters are formed.

- *Competitive learning algorithms.* These are iterative schemes that do not employ cost functions. They produce several clusterings and they converge to the most "sensible" one, according to a distance metric. Typical representatives of this category are the *basic competitive learning scheme* and the *leaky learning algorithm*.

- *Algorithms based on morphological transformation techniques*. These algorithms use morphological transformations in order to achieve better separation of the involved clusters.

- *Density-based algorithms.* These algorithms view the clusters as regions in the l-dimensional space that are "dense" in data. From this point of view there is an affinity with the valley-seeking algorithms. However, now the approach to the problem is achieved via an alternative route. Algorithmic variants within this family spring from the different way each of them quantifies the term *density*. Because most of them require only a few passes on the data set X (some of them consider the data points only once), they are serious candidates for processing large data sets.

- *Subspace clustering algorithms.* These algorithms are well suited for processing high-dimensional data sets. In some applications the dimension of the feature space can even be of the order of a few thousands. A major problem one has to face is the "curse of dimensionality" and one is forced to equip his/her arsenal with tools tailored for such demanding tasks.

- *Kernel-based methods.* The essence behind these methods is to adopt the "kernel trick," discussed in Chapter 4 in the context of nonlinear support vector machines, to perform a mapping of the original space, X, into a high-dimensional space and to exploit the nonlinear power of this tool.

Advances in database and Internet technologies over the past years have made data collection easier and faster, resulting in large and complex data sets with many patterns and/or dimensions ([Pars 04]). Such very large data sets are met, for example, in *Web mining*, where the goal is to extract knowledge from the Web ([Pier 03]). Two significant branches of this area are *Web content mining* (which aims at the extraction of useful knowledge from the content of Web pages) and *Web usage mining* (which aims at the discovery of interesting patterns of use by analyzing Web usage data). The sizes of web data are, in general, orders of magnitude larger than those encountered in more common clustering applications. Thus, the task of clustering Web pages in order to categorize them according to their content (Web content mining) or to categorize users according to the pages they visit most often (Web usage mining) becomes a very challenging problem. In addition, if in Web content mining each page is represented by a significant number of the words it contains, the dimension of the data space can become very high.

Another typical example of a computational resource-demanding clustering application comes from the area of *bioinformatics*, especially from *DNA microarray analysis*. This is a scientific field of enormous interest and significance that has already attracted a lot of research effort and investment. In such applications, data sets of dimensionality as high as 4000 can be encountered ([Pars 04]).

The need for efficient processing of data sets large in size and/or dimensionality has led to the development of clustering algorithms tailored for such complex tasks. Although many of these algorithms fall under the umbrella of one of the previously mentioned categories, we have chosen to discuss them separately at each related chapter to emphasize their specific focus and characteristics.

Several books—including [Ande 73, Dura 74, Ever 01, Gord 99, Hart 75, Jain 88, Kauf 90, and Spat 80]—are dedicated to the clustering problem. In addition, several survey papers on clustering algorithms have also been written. Specifically, a presentation of the clustering algorithms from a statistical point of view is given in [Jain 99]. In [Hans 97], the clustering problem is presented in a mathematical programming framework. In [Kola 01], applications of clustering algorithms for spatial database systems are discussed. Other survey papers are [Berk 02, Murt 83, Bara 99], and [Xu 05].

In addition, papers dealing with comparative studies among different clustering methods have also appeared in the literature. For example, in [Raub 00] the comparison of five typical clustering algorithms and their relative merits are discussed. Computationally efficient algorithms for large databases are compared in [Wei 00].

Finally, evaluations of different clustering techniques in the context of specific applications have also been conducted. For example, clustering applications for gene-expression data from DNA microarray experiments are discussed in [Jian 04, Made 04], and an experimental evaluation of document clustering techniques is given in [Stei 00].

12.3 SEQUENTIAL CLUSTERING ALGORITHMS

In this section we describe a basic sequential algorithmic scheme, (BSAS), (which is a generalization of that discussed in [Hall 67]), and we also give some variants of it. First, we consider the case where all the vectors are presented to the algorithm *only once*. *The number of clusters is not known a priori in this case*. In fact, new clusters are created as the algorithm evolves.

Let $d(\boldsymbol{x}, C)$ denote the distance (or dissimilarity) between a feature vector $\boldsymbol{x}$ and a cluster C. This may be defined by taking into account either all vectors of C or a representative vector of it (see Chapter 11). *The user-defined parameters required by the algorithmic scheme are the threshold of dissimilarity* Θ *and the maximum allowable number of clusters, q.* The basic idea of the algorithm is the following: As each new vector is considered, it is assigned either to an existing cluster or to a newly created cluster, depending on its distance from the already formed ones. Let m be the number of clusters that the algorithm has created up to now. Then the algorithmic scheme may be stated as:

Basic Sequential Algorithmic Scheme (BSAS)

```
■ $m = 1$
■ $C_m = \{\boldsymbol{x}_1\}$
■ For $i = 2$ to $N$
    • Find $C_k$: $d(\boldsymbol{x}_i, C_k) = \min_{1 \le j \le m} d(\boldsymbol{x}_i, C_j)$.
    • If $(d(\boldsymbol{x}_i, C_k) > \Theta)$ AND $(m < q)$ then
        ○ $m = m + 1$
        ○ $C_m = \{\boldsymbol{x}_i\}$
    • Else
        ○ $C_k = C_k \cup \{\boldsymbol{x}_i\}$
        ○ Where necessary, update representatives[2]
    • End {if}
■ End {For}
```

Different choices of $d(\boldsymbol{x}, C)$ lead to different algorithms, and any of the measures introduced in Chapter 11 can be employed. When C is represented by a single vector, $d(\boldsymbol{x}, C)$ becomes

$$d(\boldsymbol{x}, C) = d(\boldsymbol{x}, \boldsymbol{m}_C) \tag{12.4}$$

[2] This statement is activated in the cases where each cluster is represented by a single vector. For example, if each cluster is represented by its mean vector, this must be updated each time a new vector becomes a member of the cluster.

where $\boldsymbol{m}_C$ is the representative of C. In the case in which the mean vector is used as a representative, the updating may take place in an iterative fashion, that is,

$$\boldsymbol{m}_{C_k}^{new} = \frac{(n_{C_k^{new}} - 1)\boldsymbol{m}_{C_k}^{old} + \boldsymbol{x}}{n_{C_k^{new}}} \tag{12.5}$$

where $n_{C_k^{new}}$ is the cardinality of C_k after the assignment of $\boldsymbol{x}$ to it and $\boldsymbol{m}_{C_k}^{new}$ ($\boldsymbol{m}_{C_k}^{old}$) is the representative of C_k after (before) the assignment of $\boldsymbol{x}$ to it (Problem 12.2).

It is not difficult to realize that the order in which the vectors are presented to the BSAS plays an important role in the clustering results. Different presentation ordering may lead to totally different clustering results, in terms of the number of clusters as well as the clusters themselves (see Problem 12.3).

Another important factor affecting the result of the clustering algorithm is the choice of the threshold Θ. This value directly affects the number of clusters formed by BSAS. If Θ is too small, unnecessary clusters will be created. On the other hand, if Θ is too large a smaller than appropriate number of clusters will be created. In both cases, the number of clusters that best fits the data set is missed.

If the number q of the maximum allowable number of clusters is not constrained, we leave it to the algorithm to "decide" about the appropriate number of clusters. Consider, for example, Figure 12.1, where three compact and well-separated clusters are formed by the points of X. If the maximum allowable number of clusters is set equal to two, the BSAS algorithm will be unable to discover three clusters. Probably, in this case the two rightmost groups of points will form a single cluster. On the other hand, if q is unconstrained, the BSAS algorithm will probably form three clusters (with an appropriate choice of Θ), at least for the case in which the mean vector is used as a representative. However, constraining q becomes necessary when dealing with implementations where the available computational resources are limited. In the next subsection, a simple technique is given for determining the number of clusters.[3]

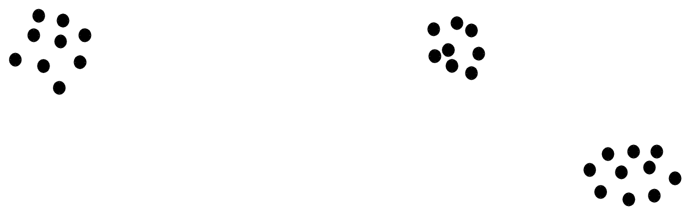

FIGURE 12.1

Three clusters are formed by the feature vectors. When q is constrained to a value less than 3, the BSAS algorithm will not be able to reveal them.

[3] This problem is also treated in Chapter 16.

Remarks

- The BSAS scheme may be used with similarity instead of dissimilarity measures with appropriate modification; that is, the min operator is replaced by max.
- It turns out that BSAS, with point cluster representatives, favors compact clusters. Thus, it is not recommended if there is strong evidence that other types of clusters are present.
- The BSAS algorithm performs a single pass on the entire data set, X. For each iteration, the distance of the vector currently considered from each of the clusters defined so far is computed. Because the final number of clusters m is expected to be much smaller than N, the time complexity of BSAS is $O(N)$.
- The preceding algorithm is closely related to the algorithm implemented by the ART2 (adaptive resonance theory) neural architecture [Carp 87, Burk 91].

12.3.1 Estimation of the Number of Clusters

In this subsection, a simple method is described for determining the number of clusters (other such methods are discussed in Chapter 16). The method is suitable for BSAS as well as other algorithms, for which the number of clusters is not required as an input parameter. In what follows, BSAS(Θ) denotes the BSAS algorithm with a specific threshold of dissimilarity Θ.

```
For $\Theta = a$ to $b$ step $c$
    Run $s$ times the algorithm BSAS($\Theta$), each time presenting the data in a different order.
    Estimate the number of clusters, $m_\Theta$, as the most frequent number resulting from the $s$ runs of BSAS($\Theta$).
Next $\Theta$
```

The values a and b are the minimum and maximum dissimilarity levels among all pairs of vectors in X, that is, $a = \min_{i,j=1,\ldots,N} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $b = \max_{i,j=1,\ldots,N} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The choice of c is directly influenced by the choice of $d(\boldsymbol{x}, C)$. As far as the value of s is concerned, the greater the s, the larger the statistical sample and, thus, the higher the accuracy of the results. In the sequel, we plot the number of clusters m_Θ versus Θ. This plot has a number of flat regions. We estimate the number of clusters as the number that corresponds to the widest flat region. It is expected that at least for the case in which the vectors form well-separated compact clusters, this is the desired number. Let us explain this argument intuitively. Suppose that the data form two compact and well-separated clusters C_1 and C_2. Let the maximum distance between two vectors in C_1 (C_2) be r_1 (r_2) and suppose that $r_1 < r_2$. Also let r ($>r_2$) be the minimum among all distances $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, with $\boldsymbol{x}_i \in C_1$ and $\boldsymbol{x}_j \in C_2$. It is clear that for $\Theta \in [r_2, r - r_2]$, the number of clusters created by BSAS is 2. In addition, if $r \gg r_2$, the interval has a wide range, and thus it corresponds to a wide flat region in the plot of m_Θ versus Θ. Example 12.2 illustrates the idea.

Example 12.2

Consider two 2-dimensional Gaussian distributions with means $[0, 0]^T$ and $[20, 20]^T$, respectively. The covariance matrices are $\Sigma = 0.5I$ for both distributions, where I is the 2×2 identity matrix. Generate 50 points from each distribution (Figure 12.2a). The number of underlying clusters is 2. The plot resulting from the application of the previously described procedure is shown in Figure 12.2b, with $a = \min_{\boldsymbol{x}_i, \boldsymbol{x}_j \in X} d_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $b = \max_{\boldsymbol{x}_i, \boldsymbol{x}_j \in X} d_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and $c \simeq 0.3$. It can be seen that the widest flat region corresponds to the number 2, which is the number of underlying clusters.

In the foregoing procedure, we have implicitly assumed that the feature vectors do form clusters. If this is not the case, the method is useless. Methods that deal with the problem of discovering whether any clusters exist are discussed in Chapter 16. Moreover, if the vectors form compact clusters, which are not well separated, the procedure may give unreliable results, since it is unlikely for the plot of m_Θ versus Θ to contain wide flat regions.

In some cases, it may be advisable to consider all the numbers of clusters, m_Θ, that correspond to all flat regions of considerable size in the plot of m_Θ versus Θ. If, for example, we have three clusters and the first two of them lie close to each other and away from the third, the flattest region may occur for $m_\Theta = 2$ and the second flattest for $m_\Theta = 3$. If we discard the second flattest region, we will miss the three-cluster solution (Problem 12.6).

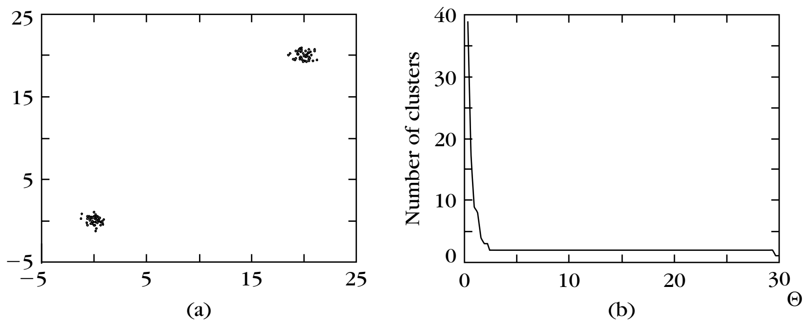

FIGURE 12.2

(a) The data set. (b) The plot of the number of clusters versus Θ. It can be seen that for a wide range of values of Θ, the number of clusters, m, is 2.

12.4 A MODIFICATION OF BSAS

As has already been stated, the basic idea behind BSAS is that each input vector $\boldsymbol{x}$ is assigned to an already created cluster or a new one is formed. Therefore, a decision for the vector $\boldsymbol{x}$ is reached *prior to the final cluster formation*, which is determined after all vectors have been presented. The following refinement of BSAS, which will be called modified BSAS (MBSAS), overcomes this drawback. The cost we pay for it is that the vectors of X have to be presented twice to the algorithm. The algorithmic scheme consists of two phases. The first phase involves the determination of the clusters, via the assignment of *some* of the vectors of X to them. During the second phase, the unassigned vectors are presented for a second time to the algorithm and are assigned to the appropriate cluster. The MBSAS may be written as follows:

Modified Basic Sequential Algorithmic Scheme (MBSAS)

```
■ Cluster Determination
■ $m = 1$
■ $C_m = \{\boldsymbol{x}_1\}$
    • For $i = 2$ to $N$
    • Find $C_k$: $d(\boldsymbol{x}_i, C_k) = \min_{1 \leq j \leq m} d(\boldsymbol{x}_i, C_j)$.
    • If $(d(\boldsymbol{x}_i, C_k) > \Theta)$ AND $(m < q)$ then
        ○ $m = m + 1$
        ○ $C_m = \{\boldsymbol{x}_i\}$
    • End {if}
■ End {For}
```

Pattern Classification

```
■ For $i = 1$ to $N$
    • If $\boldsymbol{x}_i$ has not been assigned to a cluster, then
        ○ Find $C_k$: $d(\boldsymbol{x}_i, C_k) = \min_{1 \leq j \leq m} d(\boldsymbol{x}_i, C_j)$
        ○ $C_k = C_k \cup \{\boldsymbol{x}_i\}$
        ○ Where necessary, update representatives
    • End {if}
■ End {For}
```

The number of clusters is determined in the first phase, and then it is frozen. Thus, the decision taken during the second phase for each vector takes into account all clusters.

When the mean vector of a cluster is used as its representative, the appropriate cluster representative has to be adjusted using Eq. (12.5), after the assignment of each vector in a cluster.

Also, as it was the case with BSAS, MBSAS is sensitive to the order in which the vectors are presented. In addition, because MBSAS performs two passes (one in each phase) on the data set X, it is expected to be slower than BSAS. However, its time complexity is of the same order; that is, $O(N)$.

Finally, it must be stated that, after minor modifications, MBSAS may be used when a similarity measure is employed (see Problem 12.7).

Another algorithm that falls under the MBSAS rationale is the so-called *maxmin* algorithm [Kats 94, Juan 00]. In the MBSAS scheme, a cluster is formed during the first pass, every time the distance of a vector from the already formed clusters is larger than a threshold. In contrast, the max-min algorithm follows a different strategy during the first phase. Let W be the set of all points that have been selected to form clusters, up to the current iteration step. To form a new cluster, we compute the distance of every point in $X - W$ from every point in W. If $\boldsymbol{x} \in X - W$, let $d_{\boldsymbol{x}}$ be the minimum distance of $\boldsymbol{x}$ from all the points in W. This is performed for all points in $X - W$. Then we select the point (say, $\boldsymbol{y}$) whose minimum distance (from the vectors in W) is maximum; that is,

$$d_{\boldsymbol{y}} = \max_{\boldsymbol{x}} d_{\boldsymbol{x}}, \ \ \boldsymbol{x} \in X - W$$

If this is greater than a threshold, this vector forms a new cluster. Otherwise, the first phase of the algorithm terminates. It must be emphasized that in contrast to BSAS and MBSAS, the max-min algorithm employs a threshold that is data dependent. During the second pass, points that have not yet been assigned to clusters are assigned to the created clusters as in the MBSAS method. The max-min algorithm, although computationally more demanding than MBSAS, is expected to produce clusterings of better quality.

12.5 A TWO-THRESHOLD SEQUENTIAL SCHEME

As already has been pointed out, the results of BSAS and MBSAS are strongly dependent on the order in which the vectors are presented to the algorithm, as well as on the value of Θ. Improper choice of Θ may lead to meaningless clustering results. One way to overcome these difficulties is to define a "gray" region (see [Trah 89]). This is achieved by employing two thresholds, Θ_1 and $\Theta_2 (> \Theta_1)$. If the dissimilarity level $d(\boldsymbol{x}, C)$ of a vector $\boldsymbol{x}$ from its closest cluster C is less than Θ_1, $\boldsymbol{x}$ is assigned to C. If $d(\boldsymbol{x}, C) > \Theta_2$, a new cluster is formed and $\boldsymbol{x}$ is placed in it. Otherwise, if $\Theta_1 \leq d(\boldsymbol{x}, C) \leq \Theta_2$, there exists uncertainty, and the assignment of $\boldsymbol{x}$ to a cluster will take place at a later stage. Let *clas*$(\boldsymbol{x})$ be a flag that indicates whether $\boldsymbol{x}$ has been classified (1) or not (0). Again, we denote by m the number of clusters that have been formed up to now. In the following, we assume no bounds to the number of clusters (i.e., $q = N$). The algorithmic scheme is:

The Two-Threshold Sequential Algorithmic Scheme (TTSAS)

```
  $m = 0$
  $clas(\boldsymbol{x}) = 0, \quad \forall \boldsymbol{x} \in X$
  $prev\_change = 0$
  $cur\_change = 0$
  $exists\_change = 0$

While (there exists at least one feature vector $\boldsymbol{x}$ with $clas(\boldsymbol{x}) = 0$) do

    ■ For $i = 1$ to $N$
      • if $clas(\boldsymbol{x}_i) = 0$ AND it is the first in the new while loop AND
        $exists\_change = 0$ then
        ○ $m = m + 1$
        ○ $C_m = \{\boldsymbol{x}_i\}$
        ○ $clas(\boldsymbol{x}_i) = 1$
        ○ $cur\_change = cur\_change + 1$
      • Else if $clas(\boldsymbol{x}_i) = 0$ then
        ○ Find $d(\boldsymbol{x}_i, C_k) = \min_{1 \le j \le m} d(\boldsymbol{x}_i, C_j)$
        ○ if $d(\boldsymbol{x}_i, C_k) < \Theta_1$ then
          — $C_k = C_k \cup \{\boldsymbol{x}_i\}$
          — $clas(\boldsymbol{x}_i) = 1$
          — $cur\_change = cur\_change + 1$
        ○ else if $d(\boldsymbol{x}_i, C_k) > \Theta_2$ then
          — $m = m + 1$
          — $C_m = \{\boldsymbol{x}_i\}$
          — $clas(\boldsymbol{x}_i) = 1$
          — $cur\_change = cur\_change + 1$
        ○ End {If}
      • Else if $clas(\boldsymbol{x}_i) = 1$ then
        ○ $cur\_change = cur\_change + 1$
      • End {If}
    ■ End {For}
    ■ $exists\_change = |cur\_change - prev\_change|$
    ■ $prev\_change = cur\_change$
    ■ $cur\_change = 0$
End {While}
```

The *exists_change* checks whether there exists at least one vector that has been classified at the current pass on X (i.e., the current iteration of the while loop). This is achieved by comparing the number of vectors that have been classified up to the current pass on X, *cur_change*, with the number of vectors that have been classified up to the previous pass on X, *prev_change*. If *exists_change* $= 0$, that is, no vector has been assigned to a cluster during the last pass on X, the first unclassified vector is used for the formation of a new cluster.

The first *if* condition in the *For* loop ensures that the algorithm terminates after N passes on X (N executions of the *while* loop) at the most. Indeed, this condition forces the first unassigned vector to a new cluster when no vector has been assigned during the last pass on X. This gives a way out to the case in which no vector has been assigned at a given circle.

However, in practice, the number of required passes is much less than N. It should be pointed out that this scheme is almost always at least as expensive as the previous two schemes, because in general it requires at least two passes on X. Moreover, since the assignment of a vector is postponed until enough information becomes available, it turns out that this algorithm is less sensitive to the order of data presentation.

As in the previous case, different choices of the dissimilarity between a vector and a cluster lead to different results. This algorithm also favors compact clusters, when used with point cluster representatives.

Remark

- Note that for all these algorithms no deadlock state occurs. That is, none of the algorithms enters into a state where there exist unassigned vectors that cannot be assigned either to existing clusters or to new ones, regardless of the number of passes of the data to the algorithm. The BSAS and MBSAS algorithms are guaranteed to terminate after a single and after two passes on X, respectively. In TTSAS the deadlock situation is avoided, as we arbitrarily assign the first unassigned vector at the current pass to a new cluster if no assignment of vectors occurred in the previous pass.

Example 12.3

Consider the vectors $\boldsymbol{x}_1 = [2,5]^T$, $\boldsymbol{x}_2 = [6,4]^T$, $\boldsymbol{x}_3 = [5,3]^T$, $\boldsymbol{x}_4 = [2,2]^T$, $\boldsymbol{x}_5 = [1,4]^T$, $\boldsymbol{x}_6 = [5,2]^T$, $\boldsymbol{x}_7 = [3,3]^T$, and $\boldsymbol{x}_8 = [2,3]^T$. The distance from a vector $\boldsymbol{x}$ to a cluster C

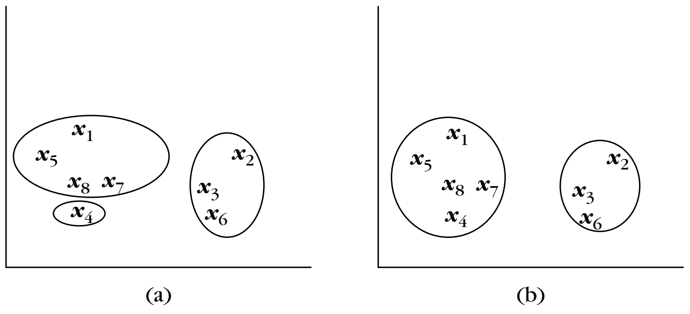

FIGURE 12.3

(a) The clustering produced by the MBSAS. (b) The clustering produced by the TTSAS.

is taken to be the Euclidean distance between $\boldsymbol{x}$ and the mean vector of C. If we present the vectors in the above order to the MBSAS algorithm and we set $\Theta = 2.5$, we obtain three clusters, $C_1 = \{\boldsymbol{x}_1, \boldsymbol{x}_5, \boldsymbol{x}_7, \boldsymbol{x}_8\}$, $C_2 = \{\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_6\}$, and $C_3 = \{\boldsymbol{x}_4\}$ (see Figure 12.3a).

On the other hand, if we present the vectors in the above order to the TTSAS algorithm, with $\Theta_1 = 2.2$ and $\Theta_2 = 4$, we obtain $C_1 = \{\boldsymbol{x}_1, \boldsymbol{x}_5, \boldsymbol{x}_7, \boldsymbol{x}_8, \boldsymbol{x}_4\}$ and $C_2 = \{\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_6\}$ (see Figure 12.3b). In this case, all vectors were assigned to clusters during the first pass on X, except $\boldsymbol{x}_4$. This was assigned to cluster C_1 during the second pass on X. At each pass on X, we had at least one vector assignment to a cluster. Thus, no vector is forced to a new cluster arbitrarily.

It is clear that the last algorithm leads to more reasonable results than MBSAS. However, it should be noted that MBSAS also leads to the same clustering if, for example, the vectors are presented with the following order: $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_5$, $\boldsymbol{x}_3$, $\boldsymbol{x}_8$, $\boldsymbol{x}_6$, $\boldsymbol{x}_7$, $\boldsymbol{x}_4$.

12.6 REFINEMENT STAGES

In all the preceding algorithms, it may happen that two of the formed clusters are very closely located, and it may be desirable to merge them into a single one. Such cases cannot be handled by these algorithms. One way out of this problem is to run the following simple merging procedure, after the termination of the preceding schemes (see [Fu 93]).

Merging procedure

```
■ (A) Find $C_i, C_j$ ($i < j$) such that $d(C_i, C_j) = \min_{k,r=1,\ldots,m,\, k \neq r} d(C_k, C_r)$
■ If $d(C_i, C_j) \leq M_1$ then
    • Merge $C_i, C_j$ to $C_i$ and eliminate $C_j$.
    • Update the cluster representative of $C_i$ (if cluster representatives are used).
    • Rename the clusters $C_{j+1}, \ldots, C_m$ to $C_j, \ldots, C_{m-1}$, respectively
```

```
        • $m = m - 1$
        • Go to (A)
    ■ Else
        • Stop
    ■ End {If}
```

M_1 is a user-defined parameter that quantifies the closeness of two clusters, C_i and C_j. The dissimilarity $d(C_i, C_j)$ between the clusters can be defined using the definitions given in Chapter 11.

The other drawback of the sequential algorithms is their sensitivity to the order of presentation of vectors. Suppose, for example, that in using BSAS, $\boldsymbol{x}_2$ is assigned to the first cluster, C_1, and after the termination of the algorithm four clusters are formed. Then it is possible for $\boldsymbol{x}_2$ to be closer to a cluster different from C_1. However, there is no way for $\boldsymbol{x}_2$ to move to its closest cluster once assigned to another one. A simple way to face this problem is to use the following reassignment procedure:

Reassignment procedure

```
    ■ For $i = 1$ to $N$
        • Find $C_j$ such that $d(\boldsymbol{x}_i, C_j) = \min_{k=1,\ldots,m} d(\boldsymbol{x}_i, C_k)$.
        • Set $b(i) = j$.
    ■ End {For}
    ■ For $j = 1$ to $m$
        • Set $C_j = \{\boldsymbol{x}_i \in X\colon b(i) = j\}$.
        • Update the representatives (if used).
    ■ End {For}
```

In this procedure, $b(i)$ denotes the closest to $\boldsymbol{x}_i$ cluster. This procedure may be used after the termination of the algorithms or, if the merging procedure is also used, after the termination of the merging procedure.

A variant of the BSAS algorithm combining the two refinement procedures has been proposed in [MacQ 67]. Only the case in which point representatives are used is considered. According to this algorithm, instead of starting with a single cluster, we start with $m > 1$ clusters, each containing one of the first m of the vectors in X. We apply the merging procedure and then we present each of the remaining vectors to the algorithm. After assigning the current vector to a cluster and updating its representative, we run the merging procedure again. If the distance between a vector $\boldsymbol{x}_i$ and its closest cluster is greater than a prespecified threshold, we form a new cluster which contains only $\boldsymbol{x}_i$. Finally, after all vectors have been presented to the algorithm, we run the reassignment procedure once. The merging procedure is applied $N - m + 1$ times. A variant of the algorithm is given in [Ande 73].

A different sequential clustering algorithm that requires a single pass on X is discussed in [Mant 85]. More specifically, it is assumed that the vectors are produced by a mixture of k Gaussian probability densities, $p(\boldsymbol{x}|C_i)$, that is,

$$p(\boldsymbol{x}) = \sum_{j=1}^{k} P(C_j)p(\boldsymbol{x}|C_j;\, \mu_j, \Sigma_j) \tag{12.6}$$

where μ_j and Σ_j are the mean and the covariance matrix of the jth Gaussian distribution, respectively. Also, $P(C_j)$ is the *a priori* probability for C_j. For convenience, let us assume that all $P(C_j)$'s are equal to each other. The clusters formed by the algorithm are assumed to follow the Gaussian distribution. At the beginning, a single cluster is formed using the first vector. Then, for each newly arrived vector, $\boldsymbol{x}_i$, the mean vector and covariance matrix of each of the m clusters, formed up to now, are appropriately updated and the conditional probabilities $P(C_j|\boldsymbol{x}_i)$ are estimated. If $P(C_q|\boldsymbol{x}_i) = \max_{j=1,\ldots,m} P(C_j|\boldsymbol{x}_i)$ is greater than a prespecifed threshold a, then $\boldsymbol{x}_i$ is assigned to C_q. Otherwise, a new cluster is formed where $\boldsymbol{x}_i$ is assigned. An alternative sequential clustering method that uses statistical tools is presented in [Amad 05].

12.7 NEURAL NETWORK IMPLEMENTATION

In this section, a neural network architecture is introduced and is then used to implement BSAS.

12.7.1 Description of the Architecture

The architecture is shown in Figure 12.4a. It consists of two modules, the matching score generator (MSG) and the MaxNet network (MN).[4]

The first module stores q parameter vectors[5] $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_q$ of dimension $l \times 1$ and implements a function $f(\boldsymbol{x}, \boldsymbol{w})$, which indicates the similarity between $\boldsymbol{x}$ and $\boldsymbol{w}$. The higher the value of $f(\boldsymbol{x}, \boldsymbol{w})$, the more similar $\boldsymbol{x}$ and $\boldsymbol{w}$ are.

When a vector $\boldsymbol{x}$ is presented to the network, the MSG module outputs a $q \times 1$ vector $\boldsymbol{v}$, with its ith coordinate being equal to $f(\boldsymbol{x}, \boldsymbol{w}_i)$, $i = 1, \ldots, q$.

The second module takes as input the vector $\boldsymbol{v}$ and identifies its maximum coordinate. Its output is a $q \times 1$ vector $\boldsymbol{s}$ with all its components equal to 0 except one that corresponds to the maximum coordinate of $\boldsymbol{v}$. This is set equal to 1. Most of the modules of this type require at least one coordinate of $\boldsymbol{v}$ to be positive.

Different implementations of the MSG can be used, depending on the proximity measure adopted. For example, if the function f is the inner product, the MSG

[4] This is a generalization of the Hamming network proposed in [Lipp 87].
[5] These are also called *exemplar patterns*.

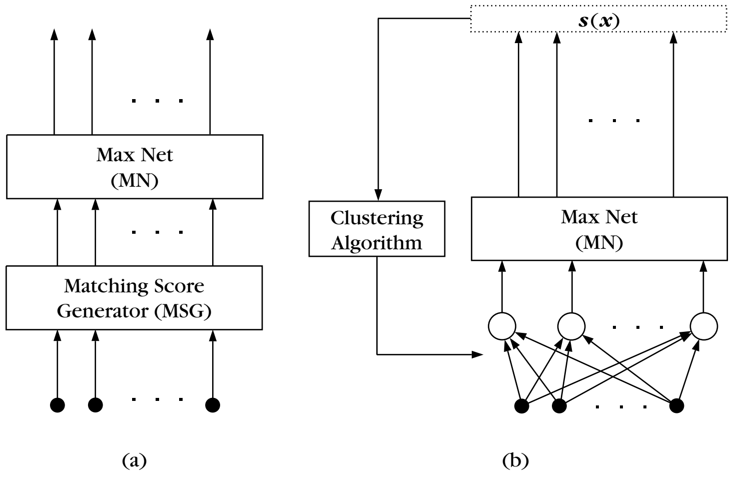

FIGURE 12.4

(a) The neural architecture. (b) Implementation of the BSAS algorithm when each cluster is represented by its mean vector and the Euclidean distance between two vectors is used.

module consists of q *linear* nodes with their threshold being equal to 0. Each of these nodes is associated with a parameter vector $\boldsymbol{w}_i$, and its output is the inner product of the input vector $\boldsymbol{x}$ with $\boldsymbol{w}_i$.

If the Euclidean distance is used, the MSG module also consists of q linear nodes. However, a different setup is required. The weight vector associated with the ith node is $\boldsymbol{w}_i$ and its threshold is set equal to $T_i = \frac{1}{2}(Q - \|\boldsymbol{w}_i\|^2)$, where Q is a positive constant that ensures that at least one of the first layer nodes will output a positive matching score, and $\|\boldsymbol{w}_i\|$ is the Euclidean norm of $\boldsymbol{w}_i$. Thus, the output of the node is

$$f(\boldsymbol{x}, \boldsymbol{w}_i) = \boldsymbol{x}^T \boldsymbol{w}_i + \frac{1}{2}(Q - \|\boldsymbol{w}_i\|^2) \tag{12.7}$$

It is easy to show that $d_2(\boldsymbol{x}, \boldsymbol{w}_i) < d_2(\boldsymbol{x}, \boldsymbol{w}_j)$ is equivalent to $f(\boldsymbol{x}, \boldsymbol{w}_i) > f(\boldsymbol{x}, \boldsymbol{w}_j)$ and thus the output of MSG corresponds to the $\boldsymbol{w}_i$ with the minimum Euclidean distance from $\boldsymbol{x}$ (see Problem 12.8).

The MN module can be implemented via a number of alternatives. One can use either neural network comparators such as the Hamming MaxNet, its generalizations and other feed-forward architectures [Lipp 87, Kout 95, Kout 05, Kout 98] or conventional comparators [Mano 79].

12.7.2 Implementation of the BSAS Algorithm

In this section, we demonstrate how the BSAS algorithm can be mapped to the neural network architecture when (a) each cluster is represented by its mean vector and (b) the Euclidean distance between two vectors is used (see Figure 12.4b). The structure of the Hamming network must also be slightly modified, so that each node in the first layer to has as an extra input the term $-\frac{1}{2}\|\boldsymbol{x}\|^2$. Let $\boldsymbol{w}_i$ and T_i be the weight vector and the threshold of the ith node in the MSG module, respectively. Also let $\boldsymbol{a}$ be a $q \times 1$ vector whose ith component indicates the number of vectors contained in the ith cluster. Also, let $\boldsymbol{s}(\boldsymbol{x})$ be the output of the MN module when the input to the network is $\boldsymbol{x}$. In addition, let t_i be the connection between the ith node of the MSG and its corresponding node in the MN module. Finally, let $sgn(z)$ be the step function that returns 1 if $z > 0$ and 0 otherwise.

The first m of the q $\boldsymbol{w}_i$'s correspond to the representatives of the clusters defined so far by the algorithm. At each iteration step either one of the first m $\boldsymbol{w}_i$'s is updated or a new parameter vector $\boldsymbol{w}_{m+1}$ is employed, whenever a new cluster is created (if $m < q$). The algorithm may be stated as follows.

```
■ Initialization
  • $\boldsymbol{a} = \boldsymbol{0}$
  • $\boldsymbol{w}_i = \boldsymbol{0}, i = 1, \ldots, q$
  • $t_i = 0, i = 1, \ldots, q$
  • $m = 1$
  • For the first vector $\boldsymbol{x}_1$ set
    ○ $\boldsymbol{w}_1 = \boldsymbol{x}_1$
    ○ $a_1 = 1$
    ○ $t_1 = 1$
■ Main Phase
  • Repeat
    ○ Present the next vector $\boldsymbol{x}$ to the network
    ○ Compute the output vector $\boldsymbol{s}(\boldsymbol{x})$
    ○ $GATE(\boldsymbol{x}) = AND((1 - \sum_{j=1}^{q}(s_j(\boldsymbol{x}))), sgn(q - m))$
    ○ $m = m + GATE(\boldsymbol{x})$
    ○ $a_m = a_m + GATE(\boldsymbol{x})$
    ○ $\boldsymbol{w}_m = \boldsymbol{w}_m + GATE(\boldsymbol{x})\boldsymbol{x}$
    ○ $T_m = \Theta - \frac{1}{2}\|\boldsymbol{w}_m\|^2$
    ○ $t_m = 1$
    ○ For $j = 1$ to $m$
      — $a_j = a_j + (1 - GATE(\boldsymbol{x}))s_j(\boldsymbol{x})$
      — $\boldsymbol{w}_j = \boldsymbol{w}_j - (1 - GATE(\boldsymbol{x}))s_j(\boldsymbol{x})(\frac{1}{a_j}(\boldsymbol{w}_j - \boldsymbol{x}))$
      — $T_j = \Theta - \frac{1}{2}\|\boldsymbol{w}_j\|^2$
```

```
        ○ Next $j$
    • Until all vectors have been presented once to the network
```

Note that only the outputs of the m first nodes of the MSG module are taken into account, because only these correspond to clusters. The outputs of the remaining nodes are not taken into account, since $t_k = 0, k = m+1, \ldots, q$. Assume that a new vector is presented to the network such that $\min_{1 \le j \le m} d(\boldsymbol{x}, \boldsymbol{w}_j) > \Theta$ and $m < q$. Then $GATE(\boldsymbol{x}) = 1$. Therefore, a new cluster is created and the next node is activated in order to represent it. Since $1 - GATE(\boldsymbol{x}) = 0$, the execution of the instructions in the *For* loop does not affect any of the parameters of the network.

Suppose next that $GATE(\boldsymbol{x}) = 0$. This is equivalent to the fact that either $\min_{1 \le j \le m} d(\boldsymbol{x}, \boldsymbol{w}_j) \le \Theta$ or there are no more nodes available to represent additional clusters. Then the execution of the instructions in the For loop results in updating the weight vector and the threshold of the node, k, for which $d(\boldsymbol{x}, \boldsymbol{w}_k) = \min_{1 \le j \le m} d(\boldsymbol{x}, \boldsymbol{w}_j)$. This happens because $s_k(\boldsymbol{x}) = 1$ and $s_j(\boldsymbol{x}) = 0, j = 1, \ldots, q$, $j \neq k$.

12.8 PROBLEMS

12.1 Prove Eq. (12.3) using induction.

12.2 Prove Eq. (12.5).

12.3 This problem aims at the investigation of the effects of the ordering of presentation of the vectors in the BSAS and MBSAS algorithms. Consider the following two-dimensional vectors: $\boldsymbol{x}_1 = [1,1]^T$, $\boldsymbol{x}_2 = [1,2]^T$, $\boldsymbol{x}_3 = [2,2]^T$, $\boldsymbol{x}_4 = [2,3]^T$, $\boldsymbol{x}_5 = [3,3]^T$, $\boldsymbol{x}_6 = [3,4]^T$, $\boldsymbol{x}_7 = [4,4]^T$, $\boldsymbol{x}_8 = [4,5]^T$, $\boldsymbol{x}_9 = [5,5]^T$, $\boldsymbol{x}_{10} = [5,6]^T$, $\boldsymbol{x}_{11} = [-4,5]^T$, $\boldsymbol{x}_{12} = [-3,5]^T$, $\boldsymbol{x}_{13} = [-4,4]^T$, $\boldsymbol{x}_{14} = [-3,4]^T$. Also consider the case that each cluster is represented by its mean vector.

a. Run the BSAS and the MBSAS algorithms when the vectors are presented in the given order. Use the Euclidean distance between two vectors and take $\Theta = \sqrt{2}$.

b. Change the order of presentation to $\boldsymbol{x}_1, \boldsymbol{x}_{10}, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_{11}, \boldsymbol{x}_{12}, \boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_{13}, \boldsymbol{x}_8, \boldsymbol{x}_{14}, \boldsymbol{x}_9$ and rerun the algorithms.

c. Run the algorithms for the following order of presentation: $\boldsymbol{x}_1, \boldsymbol{x}_{10}, \boldsymbol{x}_5, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_{11}, \boldsymbol{x}_{12}, \boldsymbol{x}_4, \boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_{13}, \boldsymbol{x}_{14}, \boldsymbol{x}_8, \boldsymbol{x}_9$.

d. Plot the given vectors and discuss the results of these runs.

e. Perform a visual clustering of the data. How many clusters do you claim are formed by the given vectors?

12.4 Consider the setup of Example 12.2. Run BSAS and MBSAS algorithms, with $\Theta = 5$, using the mean vector as representative for each cluster. Discuss the results.

12.5 Consider Figure 12.5. The inner square has side $S_1 = 0.3$, and the sides of the inner and outer square of the outer frame are $S_2 = 1$ and $S_3 = 1.3$, respectively. The inner square contains 50 points that stem from a uniform distribution in the square. Similarly, the outer frame contains 50 points that stem from a uniform distribution in the frame.

a. Perform a visual clustering of the data. How many clusters do you claim are formed by the given points?

b. Consider the case in which each cluster is represented by its mean vector and the Euclidean distance between two vectors is employed. Run BSAS and MBSAS algorithms, with

$$\Theta = \min_{i,j=1,\ldots,100} d(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \text{to} \quad \max_{i,j=1,\ldots,100} d(\boldsymbol{x}_i, \boldsymbol{x}_j) \text{ with step } 0.2$$

and with random ordering of the data. Give a quantitative explanation for the results. Compare them with the results obtained from the previous problem.

c. Repeat (b) for the case in which d_{min}^{ps} is chosen as the dissimilarity between a vector and a cluster (see Chapter 11).

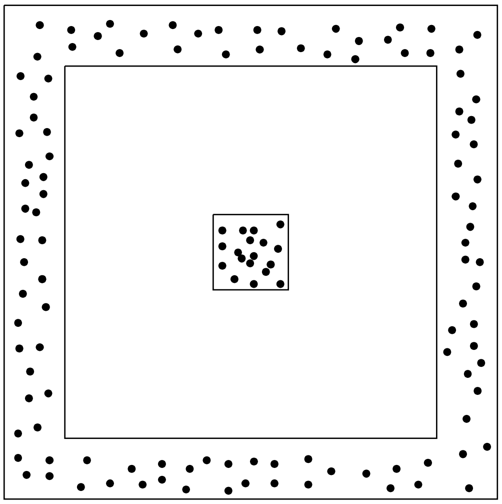

FIGURE 12.5

The setup of Problem 12.5.

12.6 Consider three two-dimensional Gaussian distributions with means $[0,0]^T$, $[6,0]^T$ and $[12,6]^T$, respectively. The covariance matrices for all distributions are equal to the identity matrix I. Generate 30 points from each distribution and let X be the resulting data set. Employ the Euclidean distance and apply the procedure discussed in Section 12.3.1 for the estimation of the number of clusters underlying in X, with $a = \min_{i,j=1,\ldots,100} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $b = \max_{i,j=1,\ldots,100} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $c = 0.3$. Plot m versus Θ and draw your conclusions.

12.7 Let s be a similarity measure between a vector and a cluster. Express the BSAS, MBSAS, and TTSAS algorithms in terms of s.

12.8 Show that when the Euclidean distance between two vectors is in use and the output function of the MSG module is given by Eq. (12.7), the relations $d_2(\boldsymbol{x}, \boldsymbol{w}_1) < d_2(\boldsymbol{x}, \boldsymbol{w}_2)$ and $f(\boldsymbol{x}, \boldsymbol{w}_1) > f(\boldsymbol{x}, \boldsymbol{w}_2)$ are equivalent.

12.9 Describe a neural network implementation similar to the one given in Section 12.7 for the BSAS algorithm when each cluster is represented by the first vector assigned to it.

12.10 The neural network architecture that implements the MBSAS algorithm, if the mean vector is in use, is similar to the one given in Figure 12.4b for the Euclidean distance case. Write the algorithm in a form similar to the one given in Section 12.7 for the MBSAS when the mean vector is in use, and highlight the differences between the two implementations.

MATLAB PROGRAMS AND EXERCISES

Computer Programs

12.1 *MBSAS algorithm.* Write a MATLAB function, named *MBSAS*, that implements the MBSAS algorithm. The function will take as input: (a) an $l \times N$ dimensional matrix, whose ith column is the i-th data vector, (b) the parameter *theta* (it corresponds to Θ in the text), (c) the maximum number of allowable clusters q, (d) an N-dimensional row array, called *order*, that defines the order of presentation of the vectors of X to the algorithm. For example, if *order* = [3 4 1 2], the third vector will be presented first, the fourth vector will be presented second, etc. If *order* = [ ], no reordering takes place. The outputs of the function will be: (a) an N-dimensional row vector *bel*, whose ith component contains the identity of the cluster where the data vector with order of presentation "i" has been assigned (the identity of a cluster is an integer in $\{1, 2, \ldots, n\_clust\}$, where *n_clust* is the number of clusters) and (b) an $l \times n\_clust$ matrix m whose i-th row is the cluster representative of the i-th cluster. Use the Euclidean distance to measure the distance between two vectors.

Solution

In the following code, **do not** type the asterisks. They will be used later on for reference purposes.

```
function [bel, m]=MBSAS(X,theta,q,order)
  % Ordering the data
  [l,N]=size(X);
  if(length(order)==N)
    X1=[];
    for i=1:N
      X1=[X1 X(:,order(i))];
    end
    X=X1;
    clear X1
  end
  % Cluster determination phase
  n_clust=1; % no. of clusters
  [l,N]=size(X);
  bel=zeros(1,N);
  bel(1)=n_clust;
  m=X(:,1);
  for i=2:N
    [m1,m2]=size(m);
    % Determining the closest cluster representative
    [s1,s2]=min(sqrt(sum((m-X(:,i)*ones(1,m2)).^ 2)));
    if(s1>theta) && (n_clust<q)
      n_clust=n_clust+1;
      bel(i)=n_clust;
      m=[m X(:,i)];
    end(*1)
  end(*2)
  [m1,m2]=size(m);(*3)
  % Pattern classification phase(*4)
  for i=1:N(*5)
    if(bel(i)==0)(*6)
      % Determining the closest cluster representative(*7)
      [s1,s2]=min(sqrt(sum((m-X(:,i)*ones(1,m2)).^ 2)));(*8)
      bel(i)=s2;
      m(:,s2)=((sum(bel==s2)-1)*m(:,s2) +
X(:,i))/sum(bel==s2);
    end
  end
```

12.2 *BSAS algorithm.* Write a MATLAB function, named *BSAS*, that implements the BSAS algorithm. Its inputs and outputs are defined exactly as in the *MBSAS* function.

Solution

In the code given for MBSAS replace the line with (*1) with the command

```
else
```

and remove all the other lines with asterisk.

Computer Experiments

12.1 Consider the data set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_8\}$, where $\boldsymbol{x}_1 = [2,\ 5]^T$, $\boldsymbol{x}_2 = [8,\ 4]^T$, $\boldsymbol{x}_3 = [7,\ 3]^T$ $\boldsymbol{x}_4 = [2,\ 2]^T$, $\boldsymbol{x}_5 = [1,\ 4]^T$, $\boldsymbol{x}_6 = [7,\ 2]^T$, $\boldsymbol{x}_7 = [3,\ 3]^T$, $\boldsymbol{x}_8 = [2,\ 3]^T$. Plot the data vectors.

12.2 Run the MBSAS function for $q = 5$ on the above data set for

a. *order* $= [1,\ 5,\ 8,\ 4,\ 7,\ 3,\ 6,\ 2]$, *theta* $= \sqrt{2} + 0.001$

b. *order* $= [5,\ 8,\ 1,\ 4,\ 7,\ 2,\ 3\,6]$, *theta* $= \sqrt{2} + 0.001$

c. *order* $= [1,\ 4,\ 5,\ 7,\ 8,\ 2,\ 3,\ 6]$, *theta* $= 2.5$

d. *order* $= [1,\ 8,\ 4,\ 7,\ 5,\ 2,\ 3,\ 6]$, *theta* $= 2.5$

e. the same order as in (c) and *theta* $= 3$

f. the same order as in (d) and *theta* $= 3$.

Study carefully the results and draw your conclusions.

12.3 Repeat 12.2 for BSAS.

REFERENCES

[Amad 05] Amador J.J. “Sequential clustering by statistical methodology,” *Pattern Recognition Letters*, Vol. 26, pp. 2152–2163, 2005.

[Ande 73] Anderberg M.R. *Cluster Analysis for Applications*, Academic Press, 1973.

[Ball 65] Ball G.H. “Data analysis in social sciences,” *Proceedings FJCC*, Las Vegas, 1965.

[Bara 99] Baraldi A., Blonda P. “A survey of fuzzy clustering algorithms for pattern recognition, Parts I and II,” *IEEE Transactions on Systems, Man and Cybernetics, B. Cybernetics*, Vol. 29(6), pp. 778–801, 1999.

[Bara 99a] Baraldi A., Schenato L. “Soft-to-hard model transition in clustering: a review,” Technical Report TR-99-010, 1999.

[Berk 02] Berkhin P. “Survey of clustering data mining techniques,” *Technical Report*, Accrue Software Inc., 2002.

[Burk 91] Burke L.I. "Clustering characterization of adaptive reasonance," *Neural Networks*, Vol. 4, pp. 485–491, 1991.

[Carp 87] Carpenter G.A., Grossberg S. "ART2: Self-organization of stable category recognition codes for analog input patterns," *Applied Optics*, Vol. 26, pp. 4919–4930, 1987.

[Duda 01] Duda R.O., Hart P., Stork D. *Pattern Classification*, 2nd ed., John Wiley & Sons, 2001.

[Dura 74] Duran B., Odell P. *Cluster Analysis: A Survey*, Springer-Verlag, Berlin, 1974.

[Ever 01] Everitt B., Landau S., Leesse M. *Cluster Analysis*, Arnold, London, 2001.

[Flor 91] Floreen P. "The convergence of the Hamming memory networks," *IEEE Transactions on Neural Networks*, Vol. 2(4), pp. 449–459, July 1991.

[Fu 93] Fu L., Yang M., Braylan R., Benson N. "Real-time adaptive clustering of flow cytometric data," *Pattern Recognition*, Vol. 26(2), pp. 365–373, 1993.

[Gord 99] Gordon A. *Classification*, 2nd ed., Chapman & Hall, London, 1999.

[Hall 67] Hall A.V. "Methods for demonstrating resemblance in taxonomy and ecology," *Nature*, Vol. 214, pp. 830–831, 1967.

[Hans 97] Hansen P., Jaumard B. "Cluster analysis and mathematical programming," *Mathematical Programming*, Vol. 79, pp. 191–215, 1997.

[Hart 75] Hartigan J. *Clustering Algorithms*, John Wiley & Sons, 1975.

[Jain 88] Jain A.K., Dubes R.C. *Algorithms for Clustering Data*, Prentice Hall, 1988.

[Jain 99] Jain A., Muthy M., Flynn P. "Data clustering: A review," *ACM Computational Surveys*, Vol. 31(3), pp. 264–323, 1999.

[Jian 04] Jiang D., Tang C., Zhang A. "Cluster analysis for gene expression data: A survey," *IEEE Transactions on Knowledge Data Engineering*, Vol. 16(11), pp. 1370–1386, 2004.

[Juan 00] Juan A., Vidal E. "Comparison of four initialization techniques for the *k*-medians clustering algorithm," *Proceedings of Joint IAPR International Workshops SSPR2000 and SPR2000, Lecture Notes in Computer Science*, Vol. 1876, pp. 842–852, Springer-Verlag, Alacant (Spain), September 2000.

[Kats 94] Katsavounidis I., Jay Kuo C.-C., Zhang Z., "A new initialization technique for generalized Lloyd iteration," *IEEE Signal Processing Letters*, Vol. 1(10), pp. 144–146, 1994.

[Kauf 90] Kaufman L., Roussseeuw P. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[Kola 01] Kolatch E. "Clustering algorithms for spatial databases: A survey," available at *http://citeseer.nj.nec.com/436843.html*.

[Kout 95] Koutroumbas K. "Hamming neural networks, architecture design and applications," Ph.D. thesis, Department of Informatics, University of Athens, 1995.

[Kout 94] Koutroumbas K., Kalouptsidis N. "Qualitative analysis of the parallel and asynchronous modes of the Hamming network," *IEEE Transactions on Neural Networks*, Vol. 5(3), pp. 380–391, May 1994.

[Kout 98] Koutroumbas K., Kalouptsidis N. "Neural network architectures for selecting the maximum input," *International Journal of Computer Mathematics*, Vol. 68(1–2), 1998.

[Kout 05] Koutroumbas K., Kalouptsidis N., "Generalized Hamming Networks and Applications," *Neural Networks*, Vol. 18, pp. 896–913, 2005.

[Lipp 87] Lippmann R.P. "An introduction to computing with neural nets," *IEEE ASSP Magazine*, Vol. 4(2), April 1987.

[Liu 68] Liu C.L. *Introduction to Combinatorial Mathematics*, McGraw-Hill, 1968.

[Lloy 82] Lloyd S.P. "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, Vol. 28(2), pp. 129–137, March 1982.

[MacQ 67] MacQuenn J.B. "Some methods for classification and analysis of multivariate observations," *Proceedings of the Symposium on Mathematical Statistics and Probability*, 5th ed., Vol. 1, pp. 281–297, AD 669871, University of California Press, Berkeley, 1967.

[Made 04] Madeira S.C., Oliveira A.L. "Biclustering algorithms for biological data analysis: A survey," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 1(1), pp. 24–45, 2004.

[Mano 79] Mano M. *Digital Logic and Computer Design*, Prentice Hall, 1979.

[Mant 85] Mantaras R.L., Aguilar-Martin J. "Self-learning pattern classification using a sequential clustering technique," *Pattern Recognition*, Vol. 18(3/4), pp. 271–277, 1985.

[Murt 83] Murtagh F. "A survey of recent advanced in hierarchical clustering algorithms," *Journal of Computation*, Vol. 26(4), pp. 354–359, 1983.

[Pars 04] Parsons L., Haque E., Liu H. "Subspace clustering for high dimensional data: A review," *ACM SIGKDD Explorations Newsletter*, Vol. 6(1), pp. 90–105, 2004.

[Pier 03] Pierrakos D., Paliouras G., Papatheodorou C., Spyropoulos C.D. "Web usage mining as a tool for personalization: A survey," *User Modelling and User-Adapted Interaction*, Vol. 13(4), pp. 311–372, 2003.

[Raub 00] Rauber A., Paralic J., Pampalk E. "Empirical evaluation of clustering algorithms," *Journal of Inf. Org. Sci.*, Vol. 24(2), pp. 195–209, 2000.

[Sebe 62] Sebestyen G.S. "Pattern recognition by an adaptive process of sample set construction," *IRE Transactions on Information Theory*, Vol. 8(5), pp. S82–S91, 1962.

[Snea 73] Sneath P.H.A., Sokal R.R. *Numerical Taxonomy*, W.H. Freeman, 1973.

[Spat 80] Spath H. *Cluster Analysis Algorithms*, Ellis Horwood, 1980.

[Stei 00] Steinbach M., Karypis G., Kumar V. "A comparison of document clustering techniques," *Technical Report*, 00-034, University of Minnesota, Minneapolis, 2000.

[Trah 89] Trahanias P., Scordalakis E. "An efficient sequential clustering method," *Pattern Recognition*, Vol. 22(4), pp. 449–453, 1989.

[Wei 00] Wei C., Lee Y., Hsu C. "Empirical comparison of fast clustering algorithms for large data sets," *Proceedings of the 33rd Hawaii International Conference on System Sciences*, pp. 1–10, Maui, HI, 2000.

[Xu 05] Xu R., Wunsch D. "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, Vol. 16(3), pp. 645–678, 2005.