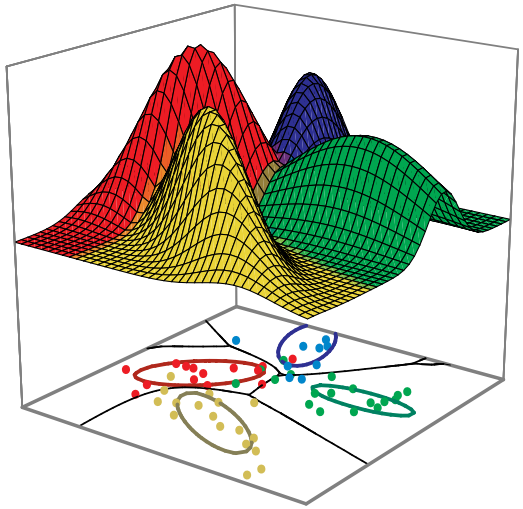
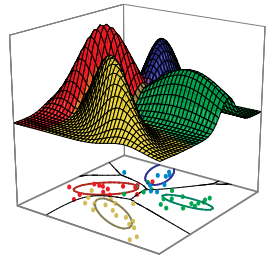
Part 13: Unsupervised Learning



Introduction
Mixture Densities
ML Estimates
Application to Normal Mixtures
K-means algorithm
Data description and clustering
Criterion function for clustering
Hierarchical clustering
The number of cluster problem and cluster validation
Graph-theoretic methods

Some materials in these slides were taken from Pattern Classification (2nd ed) by R. O. Duda, P. E. Hart and D. G. Stork, John Wiley & Sons, 2000, **Chapter 10.1-10.4, 10.6-10.10, 10.12**.

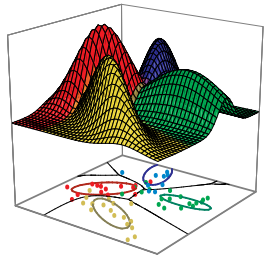
Unsupervised Learning



- Cluster these items:

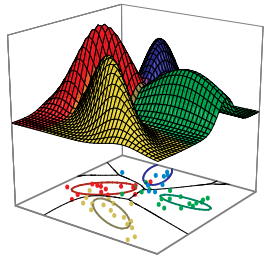
http://www.lifeofcharmings.com/2012/11/obsessed-random-favs-from-our-trip-to.html

Unsupervised Learning (10.1)



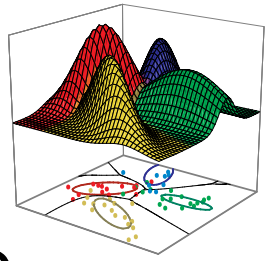
- Until now, have assumed that class of training data samples was known → *Supervised Learning*
- Instead, apply *Unsupervised Learning* to discover patterns, clusters, groupings, or relationships in your data.
  - Largely a task of clustering '*like*' samples together
  - May cluster on an unexpected feature!

Unsupervised Learning (10.1)

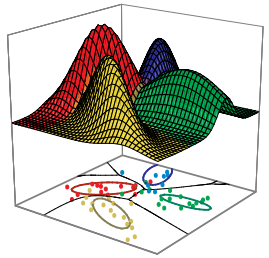


- 5 reasons to use unsupervised learning:
  - 1) Collecting and labeling a large set of sample patterns can be costly
    - Train on small amount of labeled data, then ‘tune up’ on large unlabeled dataset.
  - 2) Train on large amounts of (less expensive) unlabeled data, and only then use supervision to label the groupings
    - Appropriate for large “data mining” applications where the contents of a large database are not known beforehand
  - 3) Allow tuning over time using unsupervised classifier.
    - e.g. fruit classifier changing over the seasons
  - 4) Use UL to find features to later use for SL
  - 5) Use UL for exploratory data analysis early on
    - Gain insight into nature/structure of data

Mixture Densities (10.2)



- We shall begin with the assumption that the functional forms for the underlying probability densities are known and that the only thing that must be learned is the value of an unknown parameter vector
- We make the following assumptions:
  1. The samples come from a known number *c* of classes
  2. The prior probabilities $P(\omega_j)$ for each class are known $(j = 1, \ldots, c)$
  3. $P(x \mid \omega_j, \theta_j)$ $(j = 1, \ldots, c)$ are known
     - i.e. the form of the class-conditional dist known
  4. The values of the c parameter vectors $\theta_1, \theta_2, \ldots, \theta_c$ are unknown



- The category labels are unknown

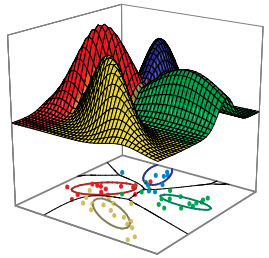
$$p(x \mid \theta) = \sum_{j=1}^{c} \overbrace{P(x \mid \omega_j, \theta_j)}^{component\ densities}. \quad \underbrace{P(\omega_j)}_{mixing\ parameters}$$

$$where\ \theta = (\theta_1, \theta_2, \ldots, \theta_c)^t$$

- This density function is called a mixture density

- Our goal will be to use samples drawn from this mixture density to estimate the unknown parameter vector θ.
- Once θ is known, we can decompose the mixture into its components and use a MAP classifier on the derived densities.

ML Estimates (10.3)



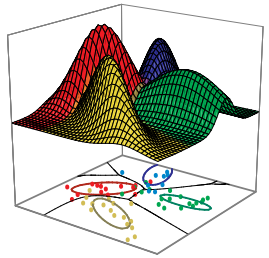
- Suppose that we have a set $D = \{x_1, \ldots, x_n\}$ of n unlabeled samples drawn independently from the mixture density $$p(x \mid \theta) = \sum_{j=1}^{c} p(x \mid \omega_j, \theta_j) P(\omega_j)$$
  - (θ is fixed but unknown!)

$$\hat{\theta} = \arg\max_{\theta} p(D \mid \theta) \text{ with } p(D \mid \theta) = \prod_{k=1}^{n} p(x_k \mid \theta)$$

$$l = \ln(p(D \mid \theta)) = \sum_{k=1}^{n} \ln\left(p\left(x_k \mid \theta\right)\right)$$

- The gradient of the log-likelihood is *(skipping steps)*:

$$\nabla_{\theta_i} l = \sum_{k=1}^{n} P(\omega_i \mid x_k, \theta) \nabla_{\theta_i} \ln p(x_k \mid \omega_i, \theta_i)$$



Since the gradient must vanish at the value of θ_i that maximizes $\ell\ (l = \sum_{k=1}^{n} \ln p(x_k \mid \theta))$ therefore, the ML estimate $\hat{\theta}_i$ must satisfy the conditions

$$\sum_{k=1}^{n} P(\omega_i \mid x_k, \hat{\theta}) \nabla_{\theta_i} \ln p(x_k \mid \omega_i, \hat{\theta}_i) = 0 \quad (\mathrm{i} = 1, \ldots, \mathrm{c}) \qquad (a)$$

- One solution to these constraints will be ML estimate for θ

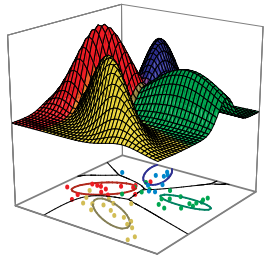
- Can generalize to include the prior probabilities as unknown variables:

$$\hat{P}(\omega_i) = \frac{1}{n} \sum \hat{P}(\omega_i \mid x_k, \hat{\theta})$$

$$\text{and} \sum_{k=1}^{n} \hat{P}(\omega_i \mid x_k, \hat{\theta}) \nabla_{\theta_i} \ln p(x_k \mid \omega_i, \hat{\theta}_i) = 0$$

$$\text{where :}\ \hat{P}(\omega_i \mid x_k, \hat{\theta}) = \frac{p(x_k \mid \omega_i, \hat{\theta}_i) \hat{P}(\omega_i)}{\sum_{j=1}^{c} p(x_k \mid \omega_j, \hat{\theta}_j) \hat{P}(\omega_j)}$$

Applications to Normal Mixtures (10.4)

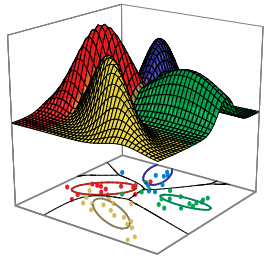


$p(x \mid \omega_i, \theta_i) \sim N(\mu_i, \Sigma_i)$

Case	μ_i	Σ_i	$P(\omega_i)$	c
1	?	x	x	x
2	?	?	?	x
3	?	?	?	?

Case 1 = Simplest case

Case 1: Unknown mean vectors



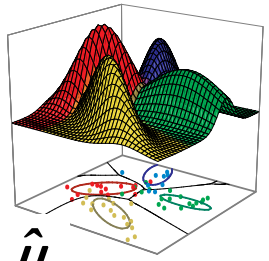
$$\mu_i = \theta_i \quad \forall i = 1, \ldots, c$$

$$\ln p(x \mid \omega_i, \mu_i) = -\ln\left[(2\pi)^{d/2}|\Sigma_i|^{1/2}\right] - \frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i)$$

ML estimate of $\mu = (\mu_i)$ is:

$$\hat{\mu}_i = \frac{\sum_{k=1}^{n} P(\omega_i \mid x_k, \hat{\mu}) x_k}{\sum_{k=1}^{n} P(\omega_i \mid x_k, \hat{\mu})} \qquad (1)$$

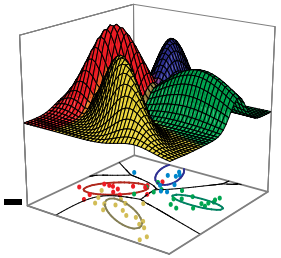
$\boldsymbol{P(\omega_i \mid x_k, \hat{\mu})}$ is the fraction of those samples having value x_k that come from the ith class, and $\boldsymbol{\hat{\mu}_i}$ is the average of the samples coming from the ith class.



- Unfortunately, equation (1) does not give $\hat{\mu}_i$ explicitly

- However, if we have some way of obtaining good initial estimates $\hat{\mu}_i(0)$ for the unknown means, therefore equation (1) can be seen as an iterative process for improving the estimates

$$\hat{\mu}_i(j+1) = \frac{\sum_{k=1}^{n} P(\omega_i \mid x_k, \hat{\mu}(j))x_k}{\sum_{k=1}^{n} P(\omega_i \mid x_k, \hat{\mu}(j))}$$



- This is a gradient ascent for maximizing the log-likelihood function
- Example:

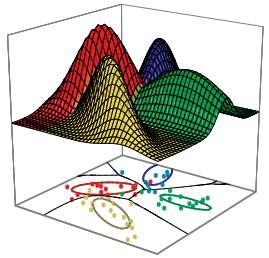
  Consider the simple two-component one-dimensional normal mixture

$$p(x \mid \mu_1, \mu_2) = \frac{1}{3\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\mu_1)^2\right] + \frac{2}{3\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\mu_2)^2\right]$$

  (2 clusters!)

  Let's set $\mu_1 = -2$, $\mu_2 = 2$ and draw 25 samples sequentially from this mixture. The log-likelihood function is:

$$l(\mu_1, \mu_2) = \sum_{k=1}^{n} \ln p(x_k \mid \mu_1, \mu_2)$$



The maximum value of l occurs at:

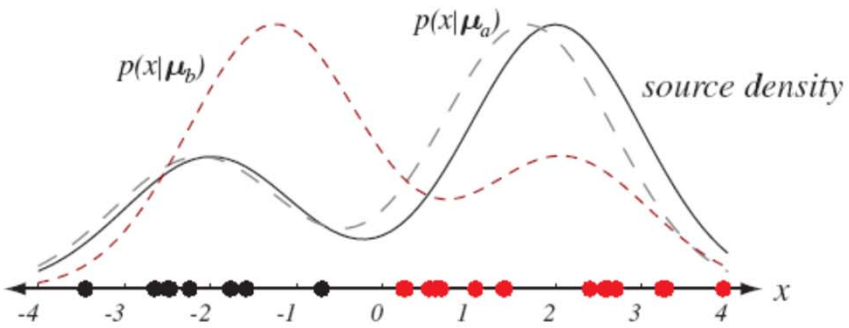
$$\hat{\mu}_1 = -2.130 \text{ and } \hat{\mu}_2 = 1.668$$

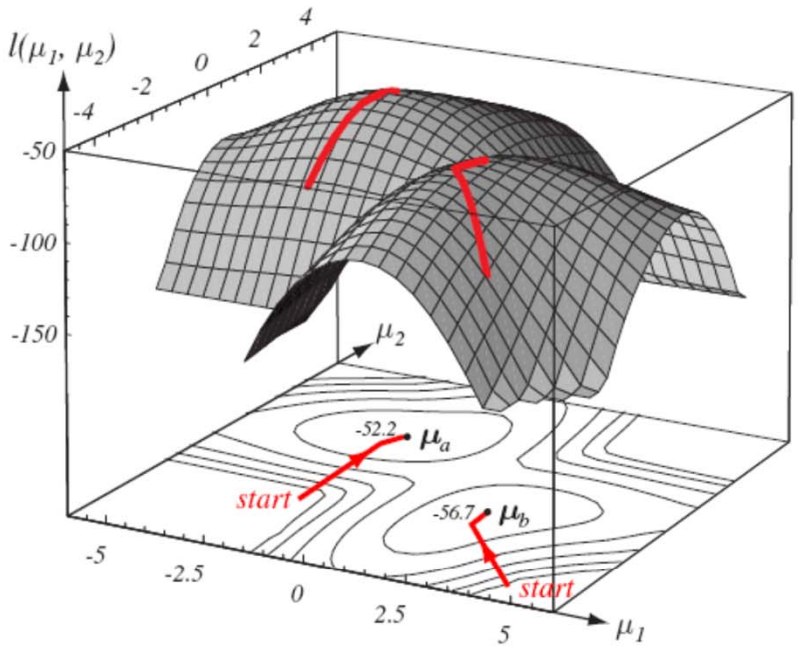
(which are not far from the true values: μ_1 = *-2 and* μ_2 = *+2)*

There is another peak at $\hat{\mu}_1 = 2.085 \text{ and } \hat{\mu}_2 = -1.257$ which has almost the same height as can be seen from the following figure.

This mixture of normal densities is identifiable

When the mixture density is not identifiable, the ML solution is not unique





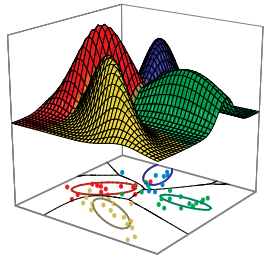
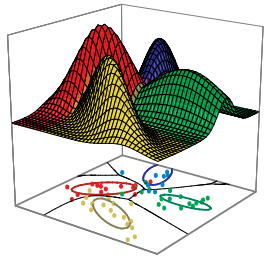


FIGURE 10.1. (Above) The source mixture density used to generate sample data, and two maximum-likelihood estimates based on the data in the table. (Bottom) Log-likelihood of a mixture model consisting of two univariate Gaussians as a function of their means, for the data in the table. Trajectories for the iterative maximum-likelihood estimation of the means of a two-Gaussian mixture model based on the data are shown as red lines. Two local optima (with log-likelihoods −52.2 and −56.7) correspond to the two density estimates shown above. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

K-Means Clustering (10.4.3)



- Goal: find the c mean vectors $\mu_1, \mu_2, \ldots, \mu_c$
- Difference #1: Replace the squared Mahalanobis distance:

$$(x_k - \hat{\mu}_i)^t \hat{\Sigma}_i^{-1} (x_k - \hat{\mu}_i)$$

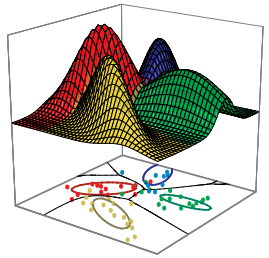
  with squared Euclidean distance: $\left\| \mathrm{x}_k - \hat{\mu}_i \right\|^2$

- Difference #2: points can only belong to a single cluster.
  - Find the mean $\hat{\mu}_m$ nearest to x_k and approximate

$$\hat{P}(\omega_i \mid x_k, \hat{\theta}) \text{ as: } \hat{P}(\omega_i \mid x_k, \theta) \cong \begin{cases} 1 & \text{if } i = m \\ 0 & \text{otherwise} \end{cases}$$

- Use the iterative scheme to find $\hat{\mu}_1, \hat{\mu}_2, \ldots, \hat{\mu}_c$

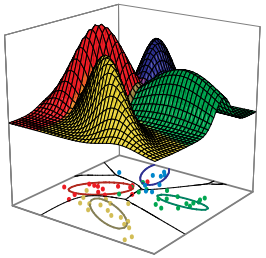
K-Means Clustering (10.4.3)



- If n is the known number of patterns and c the desired number of clusters, the k-means algorithm is:

```
Begin
        initialize n, c, μ1, μ2, …, μc (randomly selected)
              do classify n samples according to
                    nearest μi
                    recompute μi
              until no change in μi
        return μ1, μ2, …, μc
End
```

- Considering the example in the previous figure



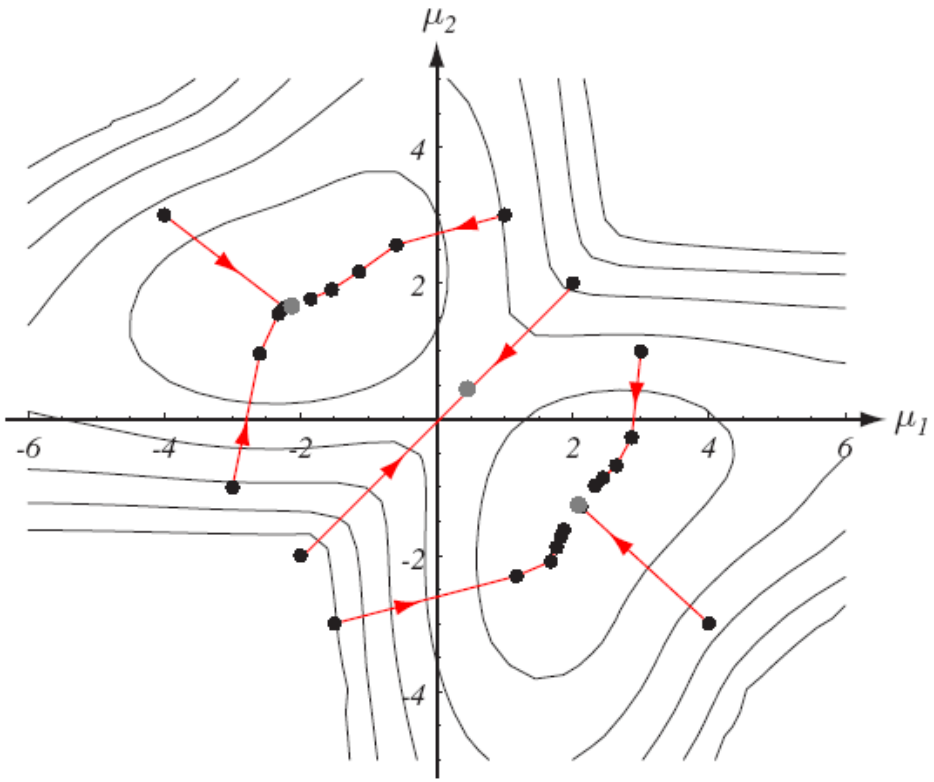
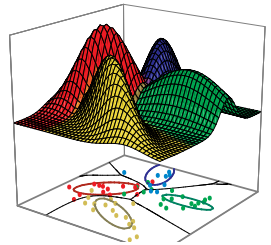


FIGURE 10.2. The *k*-means clustering procedure is a form of stochastic hill climbing in the log-likelihood function. The contours represent equal log-likelihood values for the one-dimensional data in Fig. 10.1. The dots indicate parameter values after different iterations of the *k*-means algorithm. Six of the starting points shown lead to local maxima, whereas two (i.e., $\mu_1(0) = \mu_2(0)$) lead to a saddle point near $\boldsymbol{\mu} = \mathbf{0}$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

K-Means clustering



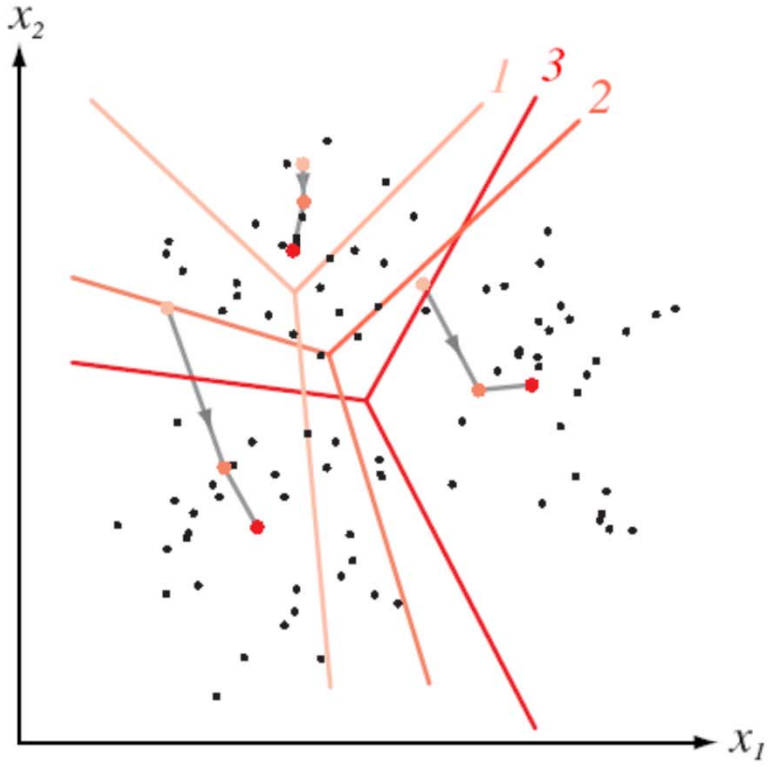
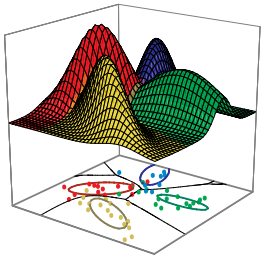


FIGURE 10.3. Trajectories for the means of the *k*-means clustering procedure applied to two-dimensional data. The final Voronoi tesselation (for classification) is also shown—the means correspond to the "centers" of the Voronoi cells. In this case, convergence is obtained in three iterations. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Data Clustering (10.6)



- Structures of multidimensional patterns are important for clustering
- If we know that data come from a specific distribution, such data can be *represented* by a compact set of parameters (*sufficient statistics*)

- If samples are considered coming from a specific distribution, but actually they are not, these statistics are a misleading representation of the data

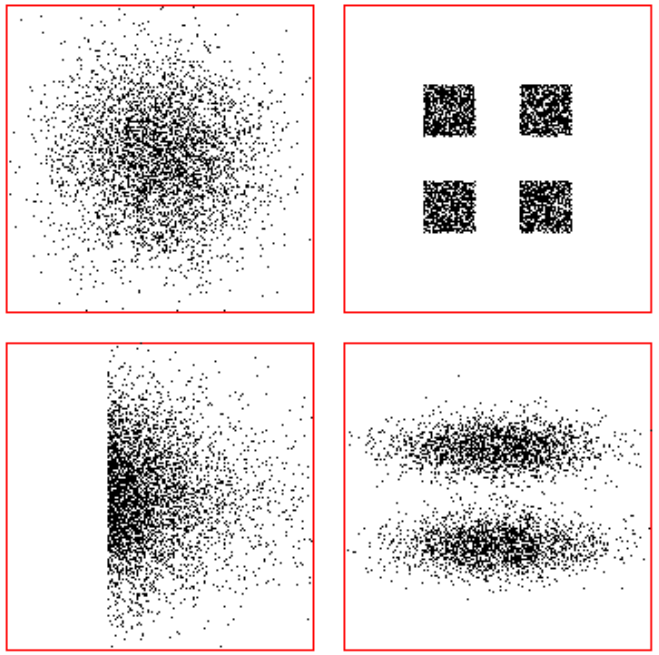
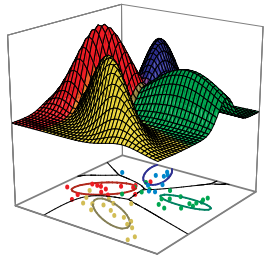


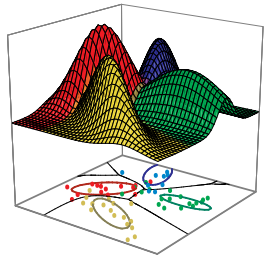
FIGURE 10.6. These four data sets have identical statistics up to second-order—that is, the same mean and covariance. In such cases it is important to include in the model more parameters to represent the structure more completely.

Data Clustering (10.6)



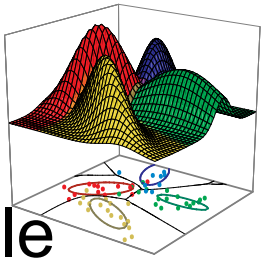
- Mixture of normal distributions can approximate a large variety of situations (i.e., any density function).
- In these cases, one can use *parametric* methods to estimate the parameters of the mixture density.
- If little prior knowledge can be assumed, the assumption of a parametric form is meaningless:
  - we are actually imposing structure on data, not finding structure on it!
- In these cases, one can use *non-parametric* methods to estimate the unknown mixture density.
- If the goal is to find subclasses, one can use a *clustering procedure* to identify groups of data points having strong internal similarities

Similarity measures (10.6.1)



- The question is how to evaluate that the samples in one cluster are more similar among them than samples in other clusters.
- Two isses:
  - How to measure the similarity between samples?
  - How to evaluate a partitioning of a set into clusters?
- The most obvious measure of similarity (or dissimilarity) between 2 samples is the distance between them, i.e., define a *metric*.
  - Metric: non-negative, d(x,y)=0 → x=y, symmetry d(x,y)=d(y,x), subadditivity/triangle inequality (d(x,z) <= d(x,y)+d(y,z)
- Once metric defined, expect the distance between samples of the same cluster to be significantly less than the distance between samples in different classes.

Similarity measures (10.6.1)



- Euclidean distance is a possible metric: a possible criterion is to assume samples belonging to same cluster if their distance is less than a threshold d_0

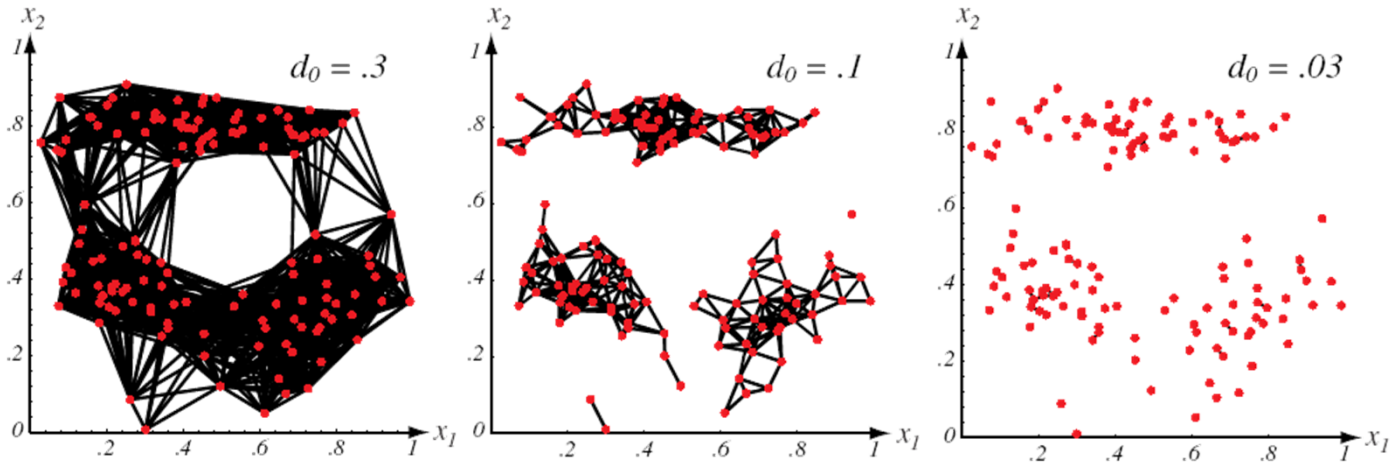


FIGURE 10.7. The distance threshold affects the number and size of clusters in similarity based clustering methods. For three different values of distance d_0, lines are drawn between points closer than d_0—the smaller the value of d_0, the smaller and more numerous the clusters. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

- Clusters defined by Euclidean distance are invariant to translations and rotation of the feature space, but not invariant to general transformations that distort the distance relationship

Effect of scaling on clusters

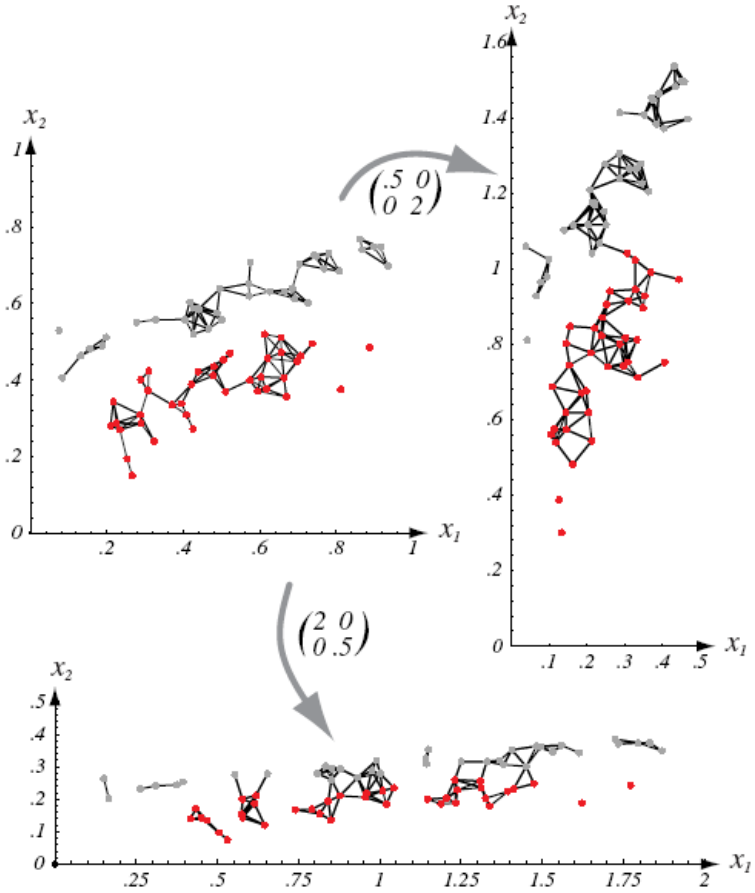
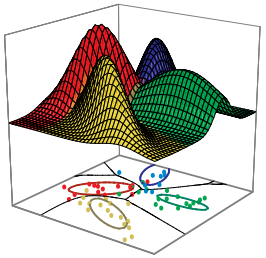


FIGURE 10.8. Scaling axes affects the clusters in a minimum distance cluster method. The original data and minimum-distance clusters are shown in the upper left; points in one cluster are shown in red, while the others are shown in gray. When the vertical axis is expanded by a factor of 2.0 and the horizontal axis shrunk by a factor of 0.5, the clustering is altered (as shown at the right). Alternatively, if the vertical axis is shrunk by a factor of 0.5 and the horizontal axis is expanded by a factor of 2.0, smaller more numerous clusters result (shown at the bottom). In both these scaled cases, the assignment of points to clusters differ from that in the original space. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Effect of scaling on clusters



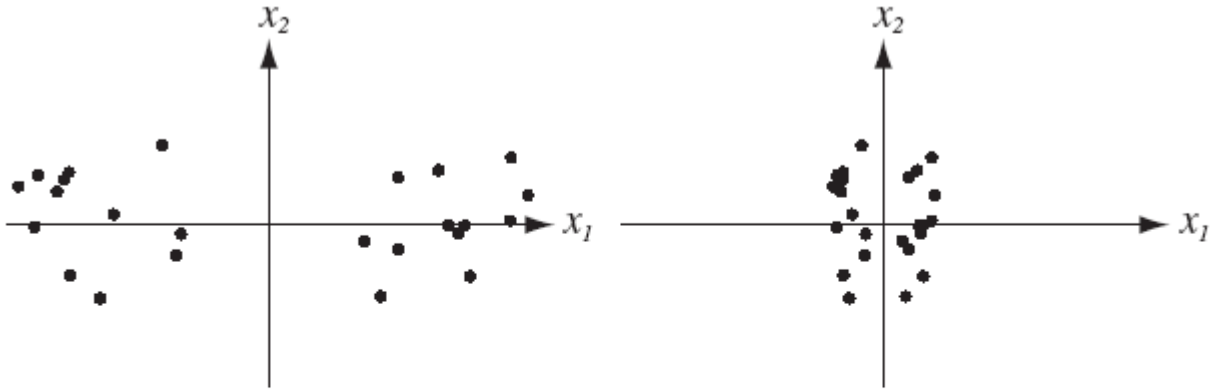
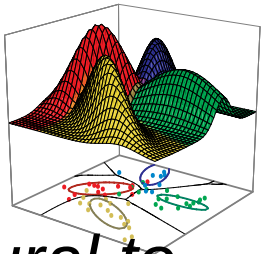


FIGURE 10.9. If the data fall into well-separated clusters (left), normalization by scaling for unit variance for the full data may reduce the separation, and hence be undesirable (right). Such a normalization may in fact be appropriate if the full data set arises from a single fundamental process (with noise), but inappropriate if there are several different processes, as shown here. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Similarity measures (10.6.1)



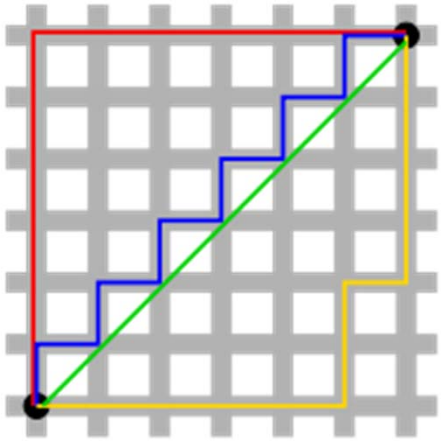
- Metric should be invariant to transformations *natural to the problem*
  - Scale/displacement invariance: normalize the data (zero means and unit variance)
  - Rotation: Use principal components
- A broad class of metrics is the Minkowsky metric

$$d(\mathbf{x},\mathbf{x}') = \left(\sum_{k=1}^{d} \left| x_k - x_k' \right|^q \right)^{1/q}$$

where $q \geq 1$ is a selectable parameter:

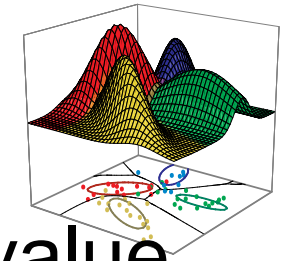
$q = 1 \Rightarrow$ Manhattan or city block metric

$q = 2 \Rightarrow$ Euclidean metric



- One can also used a *nonmetric* similarity function s(**x**,**x'**) to compare 2 vectors.

Similarity measures (10.6.1)



- It is typically a symmetric function whose value is large when **x** and **x'** are similar.
- For example, the normalized inner product (cosine)

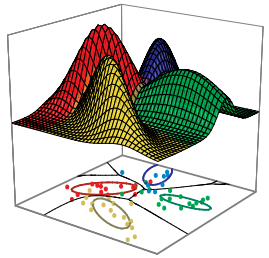
$$s(\mathbf{x},\mathbf{x}') = \frac{\mathbf{x}^t\mathbf{x}'}{\|\mathbf{x}\|\|\mathbf{x}'\|}$$

- In case of binary-valued features, becomes a count of shared attributes.
- Variations include:

$$s(\mathbf{x},\mathbf{x}') = \frac{\mathbf{x}^t\mathbf{x}'}{d}$$

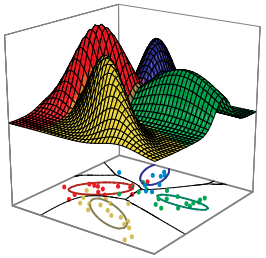
Tanimoto distance $$s(\mathbf{x},\mathbf{x}') = \frac{\mathbf{x}^t\mathbf{x}'}{\mathbf{x}^t\mathbf{x} + \mathbf{x}'^t\mathbf{x}' + \mathbf{x}^t\mathbf{x}'}$$

Clustering as optimization (10.7)



- The second issue: how to evaluate a partitioning of a set into clusters?
- Clustering can be framed as an optimization of a criterion function
  - The sum-of-squared-error criterion and its variants
  - Scatter criteria
- The sum-of-squared-error criterion
  - Let n_i the number of samples in D_i, and $\mathbf{m}_i$ the mean of those samples

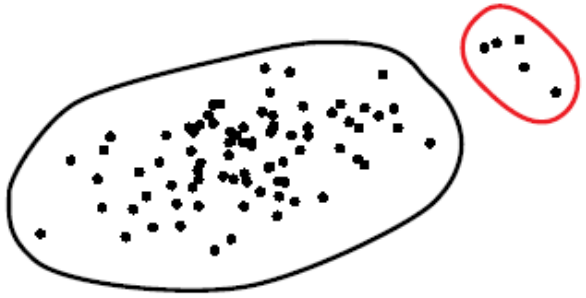
$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{x}$$



- The sum of squared error is defined as

$$J_e = \sum_{i=1}^{c} \sum_{\mathbf{x} \in D_i} \left\| \mathbf{x} - \mathbf{m}_i \right\|$$

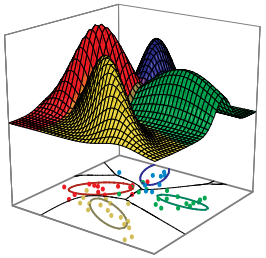
- This criterion defines clusters as their mean vectors $\mathbf{m}_i$ in the sense that it minimizes the sum of the squared lengths of the error $\mathbf{x} - \mathbf{m}_i$.
- The optimal partition is defined as one that minimizes J_e
  - Equivalent to *minimum variance* partition, which minimizes the sum of distances between pairs of points within a cluster.
- Works well when clusters form well separated compact clouds, less when there are great differences in the number of samples in different clusters.



$J_e = large$

$J_e = small$

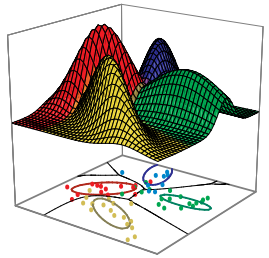
Scatter criteria (10.7.3)



- Scatter matrices used in multiple discriminant analysis, i.e., the within-scatter matrix $\mathbf{S}_W$ and the between-scatter matrix $\mathbf{S}_B$

$$\mathbf{S}_T = \mathbf{S}_B + \mathbf{S}_W \qquad S_W \sum_{i=1}^{c} S_i \qquad S_i \sum_{x \in D_i} (x - m_i)(x - m_i)^t$$

- S_T depends only on the set of samples (not on the partitioning)
- The criterion can be to minimize the within-cluster or maximize the between-cluster scatter
- The trace (sum of diagonal elements) is the simplest scalar measure of the scatter matrix, as it is proportional to the sum of the variances in the coordinate directions



$$tr[S_W] = \sum_{i=1}^{c} tr[S_i] = \sum_{i=1}^{c} \sum_{\mathbf{x} \in D_i} \|\mathbf{x} - \mathbf{m}_i\|^2 = J_e$$

that is in practice the sum-of-squared-error criterion.

- As $tr[\mathbf{S}_T] = tr[\mathbf{S}_W] + tr[\mathbf{S}_B]$ and $tr[\mathbf{S}_T]$ is independent from the partitioning, no new results can be derived by minimizing $tr[\mathbf{S}_B]$

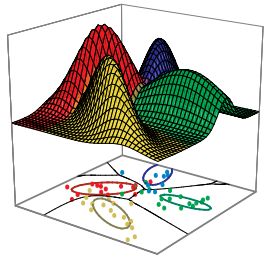
- However, seeking to minimize the within-cluster criterion $J_e = tr[\mathbf{S}_W]$, is equivalent to maximise the between-cluster criterion

$$tr[S_B] = \sum_{i=1}^{c} n_i \|\mathbf{m}_i - \mathbf{m}\|^2$$

where $\mathbf{m}$ is the total mean vector:

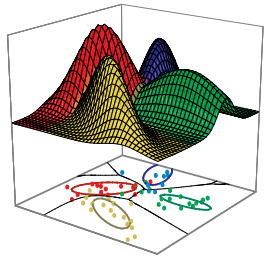
$$\mathbf{m} = \frac{1}{n} \sum_{D} \mathbf{x} = \frac{1}{n} \sum_{i=1}^{c} n_i \mathbf{m}_i$$

Iterative Optimization (10.8)



- Once a criterion function has been selected, clustering becomes a problem of discrete optimization.
- As the sample set is finite there is a finite number of possible partitions, and the optimal one can be always found by exhaustive search.
- Most frequently, it is adopted an iterative optimization procedure to select the optimal partitions
- The basic idea lies in starting from a reasonable initial partition and “move” samples from one cluster to another trying to minimize the criterion function.
- In general, these kinds of approaches guarantee local, not global, optimization.

Iterative Optimization (10.8)



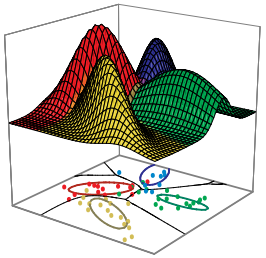
- Let us consider an iterative procedure to minimize the sum-of-squared-error criterion J_e

$$J_e = \sum_{i=1}^{c} J_i \quad \text{where} \quad J_i = \sum_{\mathbf{x} \in D_i} \left\| \mathbf{x} - \mathbf{m}_i \right\|^2$$

where J_i is the effective error per cluster.

- It can be proven that if a sample $\hat{\mathbf{x}}$, currently in cluster D_i, is tentatively moved in D_j, the change of the errors in the 2 clusters is

$$J_j^* = J_j + \frac{n_j}{n_j + 1} \left\| \hat{\mathbf{x}} - \mathbf{m}_j \right\|^2 \qquad J_i^* = J_i - \frac{n_i}{n_i - 1} \left\| \hat{\mathbf{x}} - \mathbf{m}_i \right\|^2$$



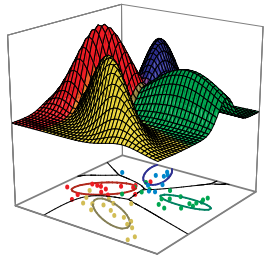
- Hence, the transfer is advantageous if the decrease in J_i is larger than the increase in J_j

$$\frac{n_i}{n_i - 1}\left\|\hat{\mathbf{x}} - \mathbf{m}_i\right\|^2 > \frac{n_j}{n_j + 1}\left\|\hat{\mathbf{x}} - \mathbf{m}_j\right\|^2$$

■ **Algorithm 3. (Basic Iterative Minimum-Squared-Error Clustering)**

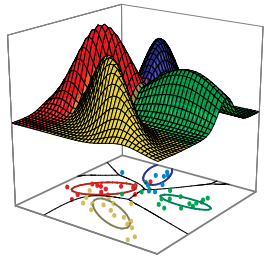
1 **begin initialize** $n, c, \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$
2 **do** randomly select a sample $\hat{\mathbf{x}}$
3 $i \leftarrow \arg\min_{i'} \|\mathbf{m}_{i'} - \hat{\mathbf{x}}\|$ (classify $\hat{\mathbf{x}}$)
4 **if** $n_i \neq 1$ **then** compute
5 $$\rho_j = \begin{cases} \frac{n_j}{n_j+1}\|\hat{\mathbf{x}} - \mathbf{m}_j\|^2 & j \neq i \\ \frac{n_j}{n_j-1}\|\hat{\mathbf{x}} - \mathbf{m}_i\|^2 & j = i \end{cases}$$
6 **if** $\rho_k \leq \rho_j$ for all j **then** transfer $\hat{\mathbf{x}}$ to $\mathcal{D}_k$
7 recompute $J_e, \mathbf{m}_i, \mathbf{m}_k$
8 **until** no change in J_e in n attempts
9 **return** $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$
10 **end**

Iterative Optimization (10.8)



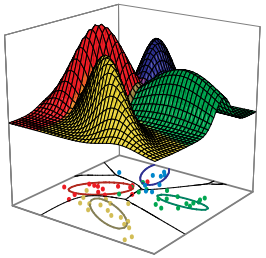
- This procedure is a sequential version of the *k-means* algorithm, with the difference that k-means waits until all n samples have been reclassified before updating, whereas the latter updates each time a sample is reclassified.
- This procedure is more prone to be trapped in local minima, and depends on the order of presentation of the samples, but it is *online!*
- Starting point is always a problem:
  - Random centers of clusters
  - Repetition with different random initialization
  - *c*-cluster starting point as the solution of the (*c-1*)-cluster problem plus the sample farthest from all cluster centers

Hierarchical Clustering (10.9)

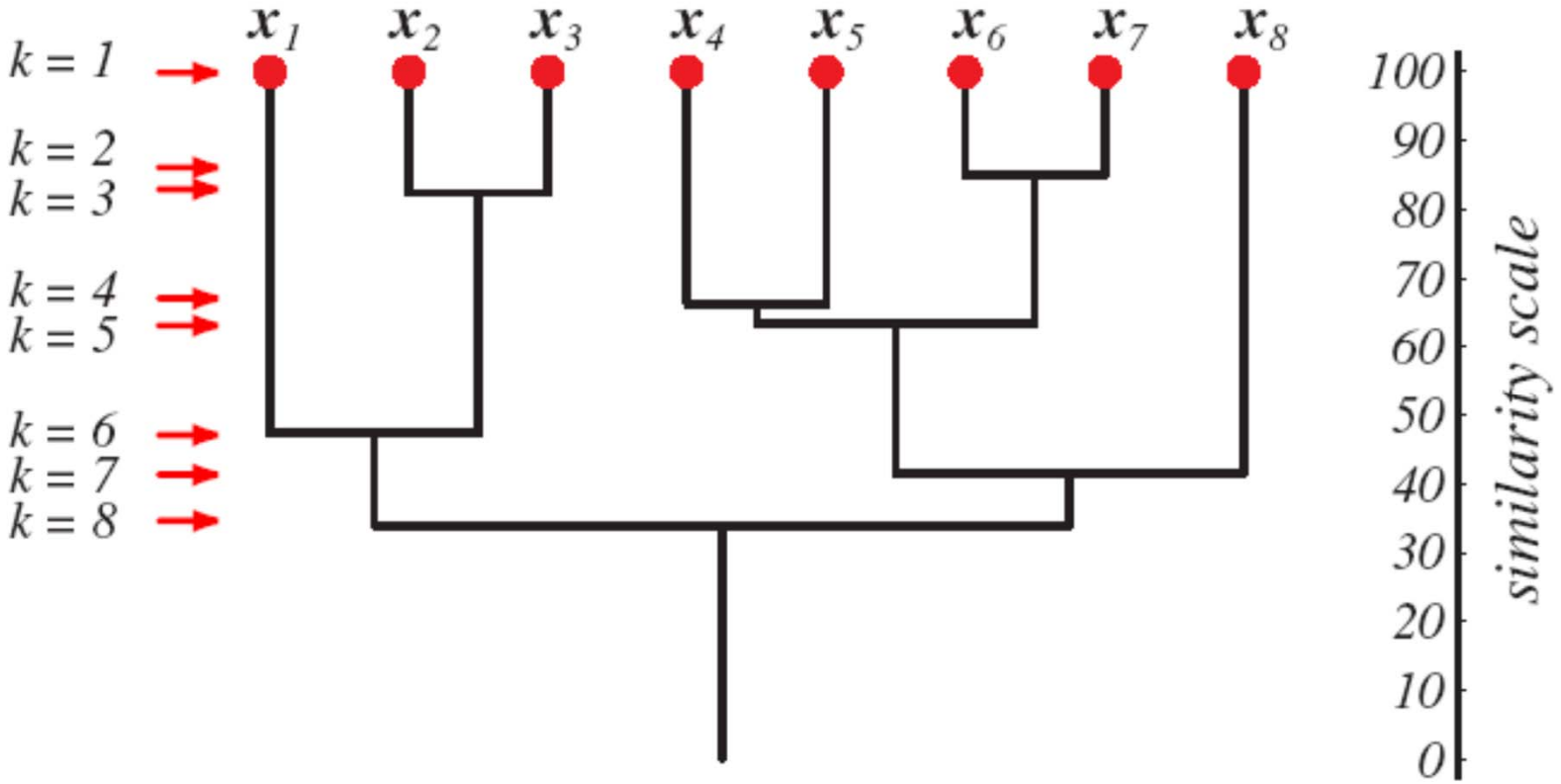


- Many times, clusters are not disjoint, but a cluster may have subclusters, in turn having sub-subclusters, etc.
- Consider a sequence of partitions of the *n* samples into *c* clusters
  - The first is a partition into *n* cluster, each one containing exactly one sample
  - The second is a partition into *n-1* clusters, the third into *n-2*, and so on, until the *n-th* in which there is only one cluster containing all of the samples
  - At the level *k* in the sequence, $c = n-k+1$.

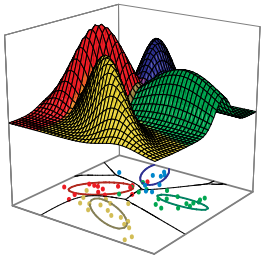
Hierarchical Clustering (10.9)



- Given any two samples **x** and **x'**, they will be grouped together *at some level*, and if they are grouped a level *k*, they remain grouped for all higher levels
- Hierarchical clustering ⇒ tree representation called *dendrogram*



Hierarchical Clustering (10.9)



- The similarity values may help to determine if the grouping are natural or forced, but if they are evenly distributed no information can be gained
- Another representation is based on set, e.g., on the Venn diagrams

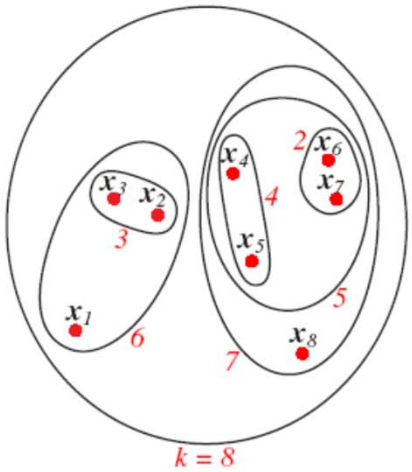
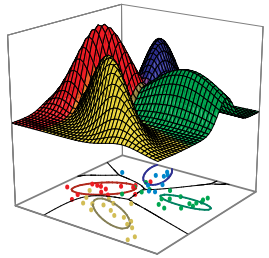


FIGURE 10.12. A set or Venn diagram representation of two-dimensional data (which was used in the dendrogram of Fig. 10.11) reveals the hierarchical structure but not the quantitative distances between clusters. The levels are numbered by k, in red. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Hierarchical Clustering (10.9)

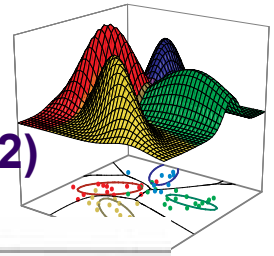


- Hierarchical clustering can be divided in *agglomerative* and *divisive*.

- Agglomerative (bottom up, clumping): start with *n* singleton cluster and form the sequence by merging clusters

- Divisive (top down, splitting): start with all of the samples in one cluster and form the sequence by successively splitting clusters

Agglomerative Hierarchical Clustering (10.9.2)

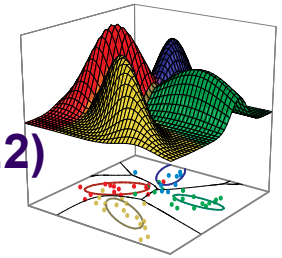


■ **Algorithm 4. (Agglomerative Hierarchical Clustering)**

```
1  begin initialize c, ĉ ← n, D_i ← {x_i}, i = 1, ..., n
2        do ĉ ← ĉ − 1
3           find nearest clusters, say, D_i and D_j
4           merge D_i and D_j
5        until c = ĉ
6     return c clusters
7  end
```

- The procedure terminates when the specified number of clusters has been obtained, and returns the cluster as sets of points, rather than the mean or a representative vector for each cluster

Agglomerative Hierarchical Clustering (10.9.2)



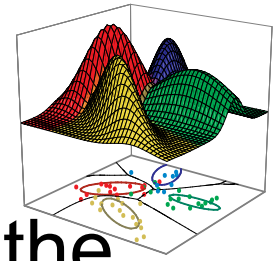
- At any level, the distance between nearest clusters can provide the dissimilarity value for that level
- To find the nearest clusters, one can use

$$d_{\min}(D_i, D_j) = \min_{\mathbf{x} \in D_i, \mathbf{x}' \in D_j} \|\mathbf{x} - \mathbf{x}'\| \qquad d_{\max}(D_i, D_j) = \max_{\mathbf{x} \in D_i, \mathbf{x}' \in D_j} \|\mathbf{x} - \mathbf{x}'\|$$

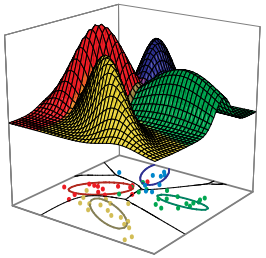
$$d_{avg}(D_i, D_j) = \frac{1}{n_i n_j} \sum_{\mathbf{x} \in D_i} \sum_{\mathbf{x}' \in D_j} \|\mathbf{x} - \mathbf{x}'\| \qquad d_{mean}(D_i, D_j) = \|\mathbf{m}_i - \mathbf{m}_j\|$$

  which behave quite similar if the clusters are hyperspherical and well separated.
- The computational complexity is $O(cn^2d^2)$, $n>>c$

Nearest-neighbour algorithm (10.9.2)



- When d_{min} is used, the algorithm is called the *nearest neighbour* algorithm
- If it is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called *single-linkage* algorithm
- If data points are thought as nodes of a graph with edges forming a path between the nodes in the same subset D_i, the merging of D_i and D_j corresponds to adding an edge between the nearest pair of node in D_i and D_j
- The resulting graph has any closed loop and it is a *tree*, if all subsets are linked we have a *spanning tree*



Nearest-neighbour algorithm (10.9.2)

- The use of d_{min} as a distance measure and the agglomerative clustering generate a *minimal spanning tree*
  - *Chaining* effect: defect of this distance measure (right)

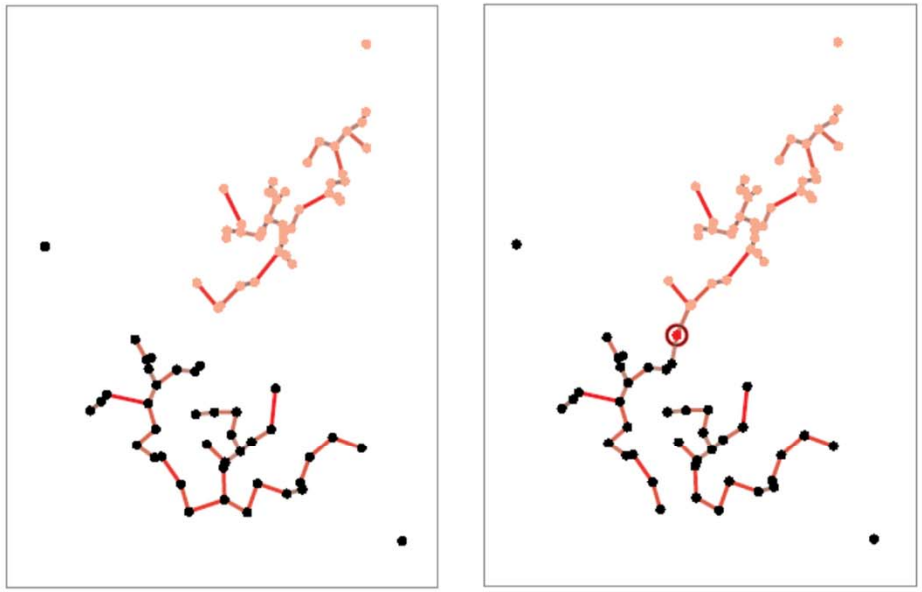
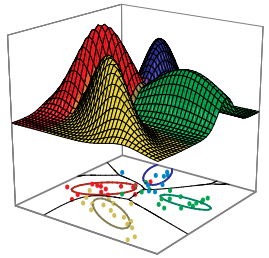
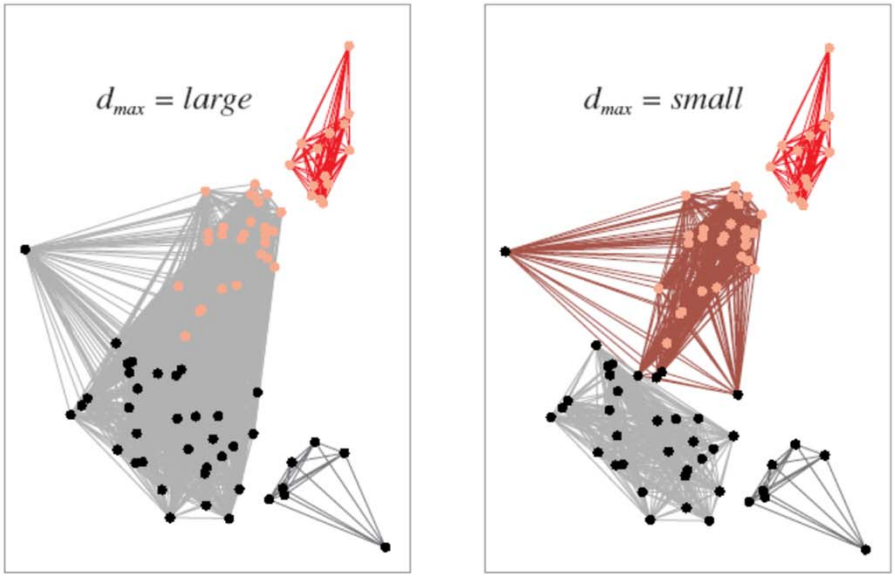


FIGURE 10.13. Two Gaussians were used to generate two-dimensional samples, shown in pink and black. The nearest-neighbor clustering algorithm gives two clusters that well approximate the generating Gaussians (left). If, however, another particular sample is generated (circled red point at the right) and the procedure is restarted, the clusters do not well approximate the Gaussians. This illustrates how the algorithm is sensitive to the details of the samples. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Farthest-neighbour algorithm (10.9.2)



- When d_{max} is used, the algorithm is called the *farthest neighbour* algorithm
- If it is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called *complete-linkage* algorithm
- This method discourages the growth of elongated clusters
- In the terminology of the graph theory, every cluster constitutes a complete subgraph, and the distance between two clusters is determined by the most distant nodes in the 2 clusters



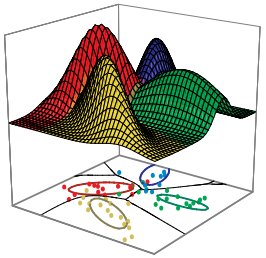
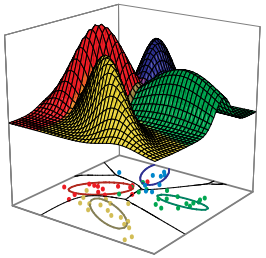
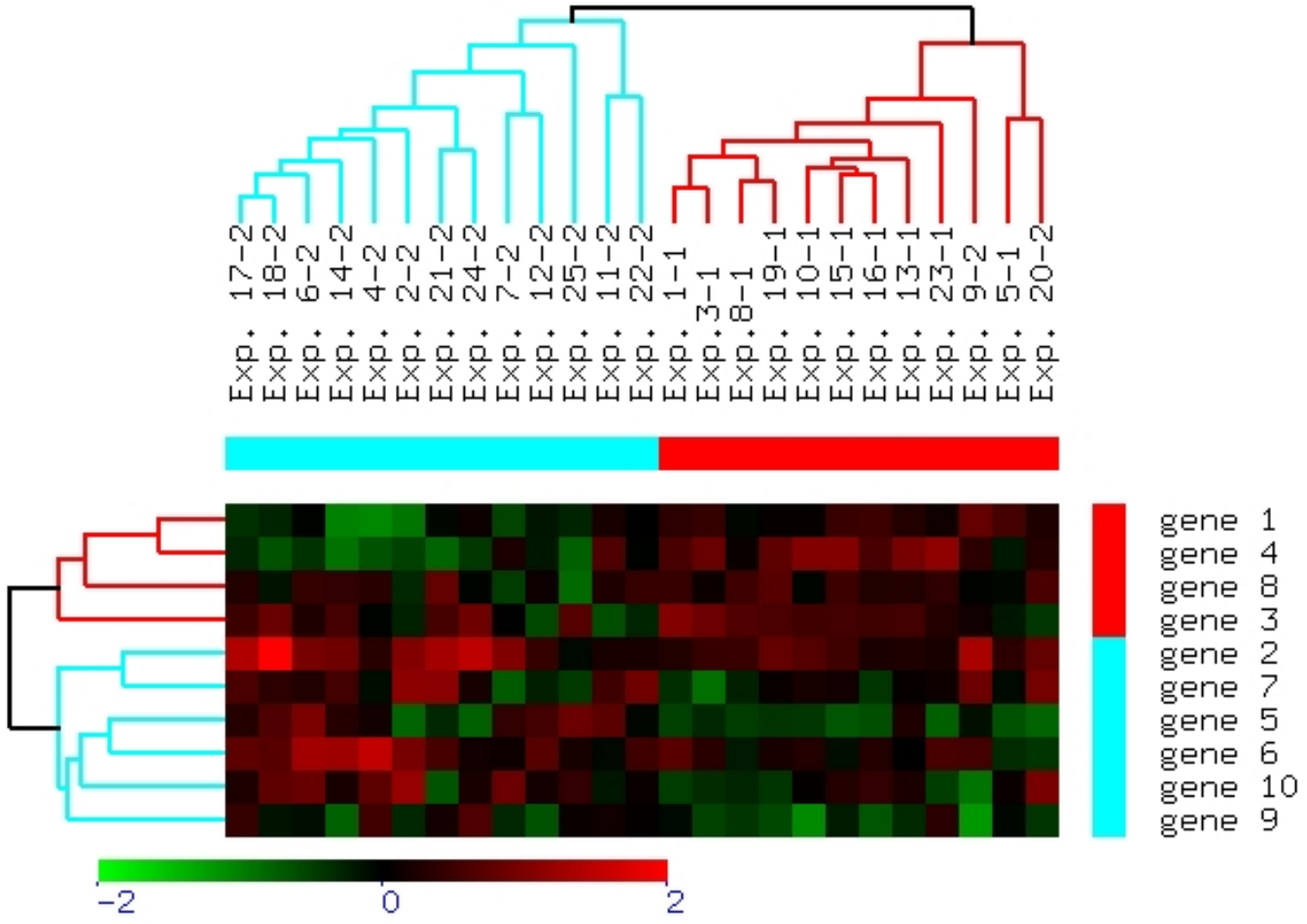


FIGURE 10.14. The farthest-neighbor clustering algorithm uses the separation between the most distant points as a criterion for cluster membership. If this distance is set very large, then all points lie in the same cluster. In the case shown at the left, a fairly large d_{max} leads to three clusters; a smaller d_{max} gives four clusters, as shown at the right. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

- When two clusters are merged, the graph is changed by adding edges between every pair of nodes in the 2 clusters
- All the procedures involving minima or maxima are sensitive to outliers.
  - The use of d_{mean} or d_{avg} are natural compromises

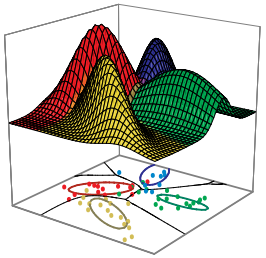
Hierarchical clustering of gene expression profiles

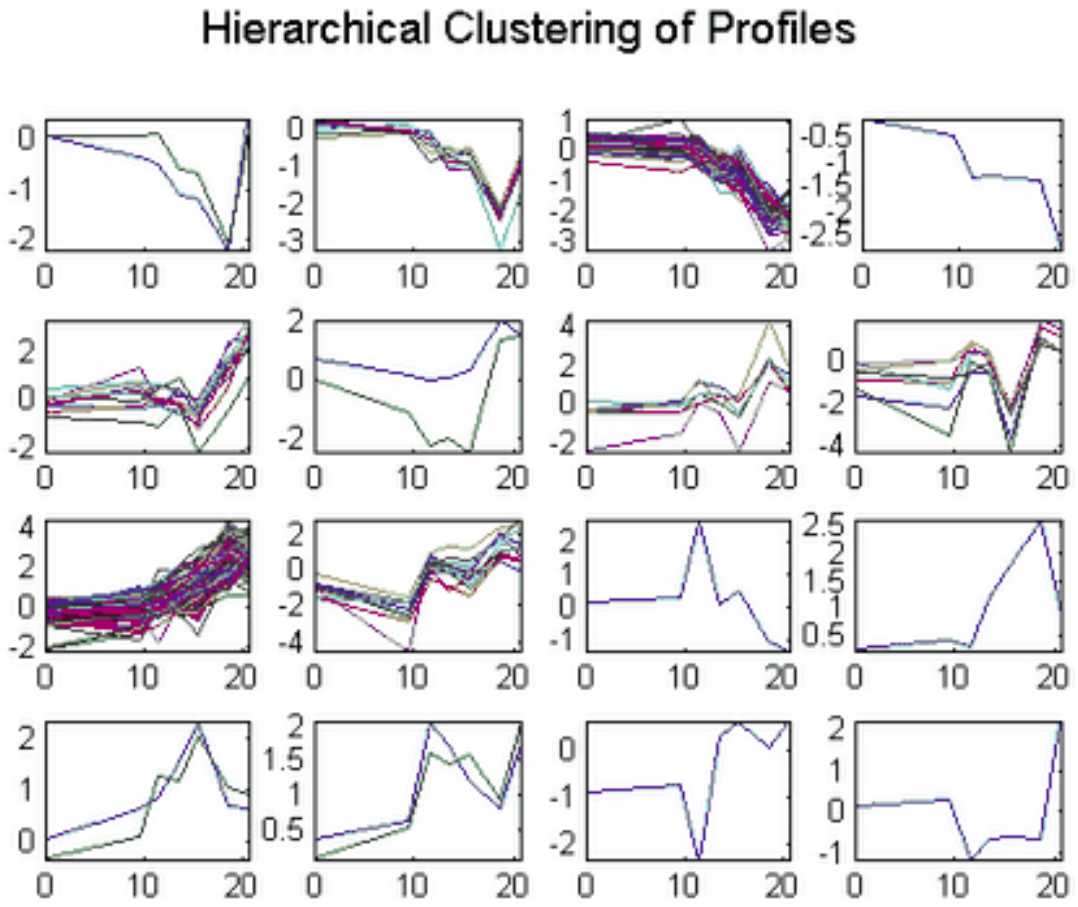




http://dcb.cit.nih.gov/hpcio/projects/images/mds3Dscaled.jpeg

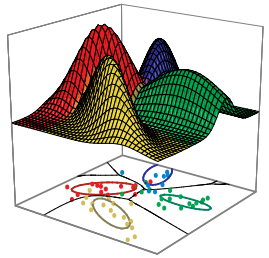
Hierarchical clustering of time-varying gene expression





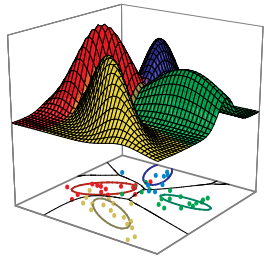
From Mathworks

The problem of the number of clusters (10.10)



- Typically, the number of clusters is known.
- When it's not, there are several ways of proceed.
- When clustering is done by extremizing a criterion function, a common approach is to repeat the clustering with c=1, c=2, c=3, etc.
  - *Plot trend in criterion… rapid decrease in J_e, then slower…*
- Another approach is to state a threshold for the creation of a new cluster;
  - adaptable to online cases but depends on the order of presentation of data.
- These approaches are similar to *model selection* procedures, typically used to determine the topology and number of states (e.g., clusters, parameters) of a model, given a specific application.

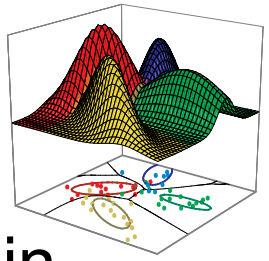
Graph-theoretic methods (10.12)



- Graph theory permits to consider particular structure of data.
- The procedure of setting a distance as a threshold to place 2 points in the same cluster can be generalized to arbitrary similarity measures.
- If s_0 is a threshold value, we can say that $\mathbf{x}_i$ is similar to $\mathbf{x}_j$ if $s(\mathbf{x}_i, \mathbf{x}_j) > s_0$.
- Hence, we define a *similarity <u>matrix</u>* $\mathbf{S} = [s_{ij}]$

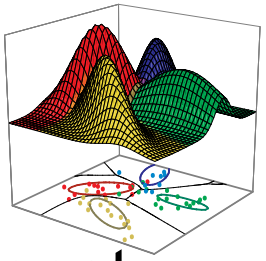
$$s_{ij} = \begin{cases} 1 & \text{if } s(\mathbf{x}_i, \mathbf{x}_j) > s_0 \\ 0 & \text{otherwise} \end{cases}$$

Graph-theoretic methods (10.12)



- This matrix induces a *similarity graph*, dual to **S**, in which nodes corresponds to points and edge joins node *i* and *j* iff s_{ij}=1.
- *Single-linkage* alg.: two samples **x** and **x'** are in the same cluster if there exists a chain **x**, $\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_k$, **x'**, such that **x** is similar to $\mathbf{x}_1$, $\mathbf{x}_1$ to $\mathbf{x}_2$, and so on $\Rightarrow$ *connected components* of the graph
- *Complete-link* alg.: all samples in a given cluster must be similar to one another and no sample can be in more than one cluster.
- *Nearest-neighbour* algorithm is a method to find the minimum spanning tree and vice versa
  - Removal of the longest edge produce a 2-cluster grouping, removal of the next longest edge produces a 3-cluster grouping, and so on.

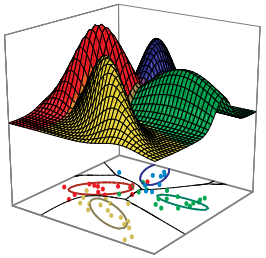
Graph-theoretic methods (10.12)



- This is a *divisive hierarchical* procedure, and suggest ways to dividing the graph in subgraphs
  - E.g., in selecting an edge to remove, comparing its length with the lengths of the other edges incident the nodes

FIGURE 10.20. The removal of inconsistent edges—ones with length significantly larger than the average incident upon a node—may yield natural clusters. The original data are shown at the left, and its minimal spanning tree is shown in the middle. At virtually every node, incident edges are of nearly the same length. Each of the two nodes shown in red are exceptions: their incident edges are of very different lengths. When the two such inconsistent edges are removed, three clusters are produced, as shown at the right. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Graph-theoretic methods (10.12)



- One useful statistics to be estimated from the minimal spanning tree is the edge length distribution
- For instance, in the case of 2 dense cluster immersed in a sparse set of points:

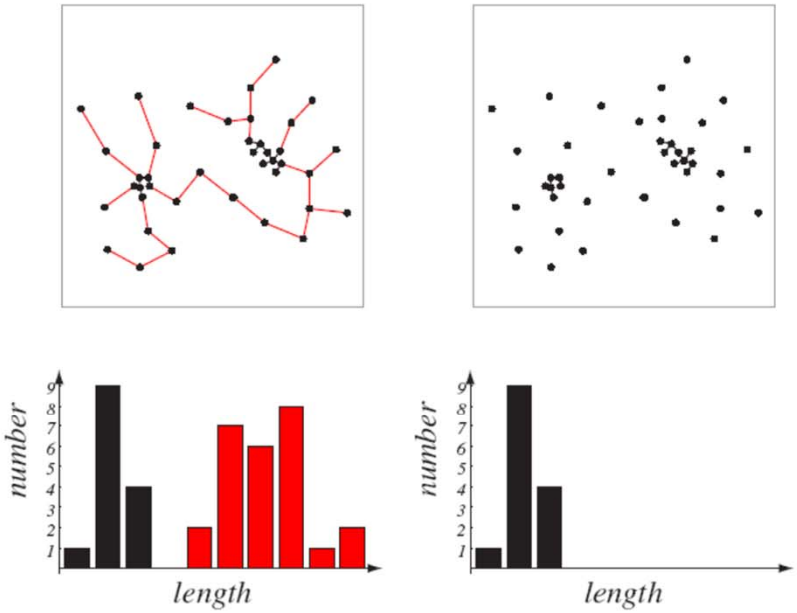
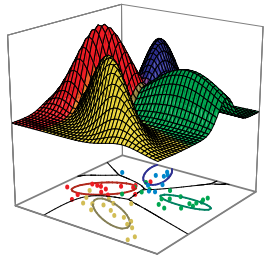


FIGURE 10.21. A minimal spanning tree is shown at the left; its bimodal edge length distribution is evident in the histogram below. If all links of intermediate or high length are removed (red), the two natural clusters are revealed (right). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Other Topics



- Online clustering
  - Learning with a critic
- Component analysis
  - Principal component analysis (PCA)
  - Nonlinear component analysis (NLCA)
  - Independent component analysis (ICA)
- Low-dimensional representations and multidimensional scaling (MDS)
  - Self-organizing maps / Kohonen maps
- Clustering and dimensionality reduction