8. Unsupervised Analysis

Unsupervised Analysis

Previously, all our *training* samples were *labeled*: these samples were said to be "*supervised*"

We now investigate a number of "*unsupervised*" procedures which use *unlabeled* samples

Collecting and labeling a large set of sample patterns can be costly

We can train with a large amount of (less expensive) unlabeled data, and only then use supervision to label the groupings found, this is appropriate for large "data mining" applications where the contents of a large database are not known beforehand

Unsupervised Analysis

This is also appropriate in many applications when the characteristics of the patterns can change slowly with time

Improved performance can be achieved if classifiers running in a unsupervised mode are used

We can use unsupervised methods to identify features that will then be useful for classification

We gain some insight into the nature (or structure) of the data

Mixture Densities & Identifiability

- **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:**

  - The samples come from a known number c of classes

  - The prior probabilities $P(\omega_j)$ for each class are known $(j = 1, \ldots, c)$

  - $P(x \mid \omega_j, \theta_j)$ $(j = 1, \ldots, c)$ are known

  - The values of the c parameter vectors $\theta_1, \theta_2, \ldots, \theta_c$ are unknown

Mixture Densities & Identifiability

- **The category labels are unknown**

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

$$\text{where } \theta = (\theta_1, \theta_2, ..., \theta_c)^t$$

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

Mixture Densities & Identifiability

- **Definition**

  - **A density $p(x \mid \theta)$ is said to be *identifiable* if $\theta \neq \theta'$ implies that there exists an x such that:**

$$p(x \mid \theta) \neq p(x \mid \theta')$$

Mixture Densities & Identifiability

As a simple example, consider the case where x is binary and P(*x* | θ) is the mixture:

$$P(x|\theta) = \frac{1}{2}\theta_1^x(1-\theta_1)^{1-x} + \frac{1}{2}\theta_2^x(1-\theta_2)^{1-x}$$

$$= \begin{cases} \frac{1}{2}(\theta_1+\theta_2) \text{ if } x=1 \\ 1-\frac{1}{2}(\theta_1+\theta_2) \text{ if } x=0 \end{cases}$$

Assume that:

$$P(x = 1 \mid \theta) = 0.6 \Rightarrow P(x = 0 \mid \theta) = 0.4$$

by replacing these probabilities values, we obtain:

$$\theta_1 + \theta_2 = 1.2$$

Mixture Densities & Identifiability

- Thus, we have a case in which the mixture distribution is completely unidentifiable, and therefore unsupervised learning is impossible

- In the discrete distributions, if there are too many components in the mixture, there may be more unknowns than independent equations, and identifiability can become a serious problem!

Mixture Densities & Identifiability

- While it can be shown that mixtures of normal densities are usually identifiable, the parameters in the simple mixture density

$$P(x \mid \theta) = \frac{P(\omega_1)}{\sqrt{2\pi}} exp\left[-\frac{1}{2}(x-\theta_1)^2 \right] + \frac{P(\omega_2)}{\sqrt{2\pi}} exp\left[-\frac{1}{2}(x-\theta_2)^2 \right]$$

- Cannot be uniquely identified if $P(\omega_1) = P(\omega_2)$

  (we cannot recover a unique θ even from an infinite amount of data!)

- $\theta = (\theta_1, \theta_2)$ and $\theta = (\theta_2, \theta_1)$ are two possible vectors that can be interchanged without affecting $P(x \mid \theta)$

- Identifiability can be a problem, we always assume that the densities we are dealing with are identifiable!

ML Estimates

Suppose that we have a set $\mathcal{D} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ of n unlabeled samples drawn independently from the mixture density

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j) P(\omega_j)$$

(θ is fixed but unknown!)

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta}) \quad \textbf{with} \quad p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta})$$

The gradient of the log-likelihood is:

$$\nabla_{\boldsymbol{\theta}_i} l = \sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k|\omega_i, \boldsymbol{\theta}_i)$$

ML Estimates

Since the gradient must vanish at the value of θ_i that maximizes $l = \sum_{k=1}^{n} \ln p(\mathbf{x}_k|\theta)$, therefore,

the ML estimate $\hat{\theta}_i$ must satisfy the conditions:

$$\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\theta}) \nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i) = 0 (i = 1, ..., c)$$

ML Estimates

By including the prior probabilities as unknown variables, we finally obtain:

$$\hat{P}(\omega_i) = \frac{1}{n} \sum_{k=1}^{n} \hat{P}(\omega_i | \mathbf{x}_k, \hat{\boldsymbol{\theta}})$$

and

$$\sum_{k=1}^{n} \hat{P}(\omega_i | \mathbf{x}_k, \hat{\boldsymbol{\theta}}) \nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k | \omega_i, \hat{\boldsymbol{\theta}}_i) = 0$$

where

$$\hat{P}(\omega_i | \mathbf{x}_k, \hat{\boldsymbol{\theta}}) = \frac{p(\mathbf{x}_k | \omega_i, \hat{\boldsymbol{\theta}}_i) \hat{P}(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}_k | \omega_j, \hat{\boldsymbol{\theta}}_j) \hat{P}(\omega_j)}$$

Applications to Normal Mixtures

$$p(\mathbf{x}|\omega_i, \boldsymbol{\theta}_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

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

Case 1 = Simplest case

Applications to Normal Mixtures

- **Case 1: Unknown mean vectors**

$$\boldsymbol{\mu}_i = \boldsymbol{\theta}_i, \forall i = 1, ..., c$$

$$\ln p(\mathbf{x}|\omega_i, \boldsymbol{\mu}_i) = -\ln\left[(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}\right] - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)$$

Applications to Normal Mixtures

- **Case 1: Unknown mean vectors (continued)**

ML estimate:

$$\hat{\boldsymbol{\mu}}_i = \frac{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}})\mathbf{x}_k}{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}})} \qquad (1)$$

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

Applications to Normal Mixtures

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

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

$$\hat{\boldsymbol{\mu}}_i(j+1) = \frac{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}}(j))\mathbf{x}_k}{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}}(j))}$$

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

Applications to Normal Mixtures

- **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.

Applications to Normal Mixtures

The log-likelihood function is:

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

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

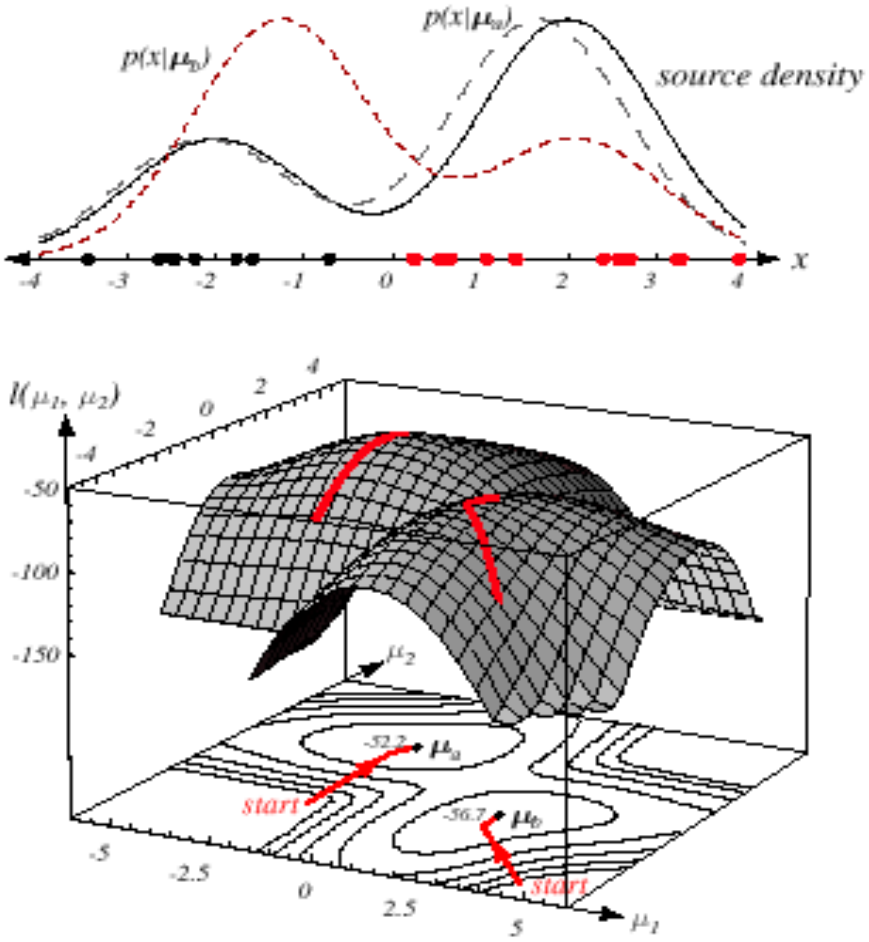


FIGURE 10.1. (Above) The source mixture density used to generate sample data, and two maximum-likelihood estimates based on the data in the table. (Bottom) Log-likelihood of a mixture model consisting of two univariate Gaussians as a function of their means, for the data in the table. Trajectories for 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.

Applications to Normal Mixtures

- This mixture of normal densities is identifiable
- When the mixture density is not identifiable, the ML solution is not unique

- **Case 2: All parameters unknown**

  - No constraints are placed on the covariance matrix

  Let $p(x \mid \mu, \sigma^2)$ be the two-component normal mixture:

$$p(x|\mu,\sigma^2) = \frac{1}{2\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] + \frac{1}{2\sqrt{2\pi}}\exp\left[-\frac{1}{2}x^2\right]$$

Applications to Normal Mixtures

Suppose $\mu = x_1$, therefore:

$$p(x_1|\mu,\sigma^2) = \frac{1}{2\sqrt{2\pi}\sigma} + \frac{1}{2\sqrt{2\pi}}\exp\left[-\frac{1}{2}x_1^2\right]$$

For the rest of the samples:

$$p(x_k|\mu,\sigma^2) \geq \frac{1}{2\sqrt{2\pi}}\exp\left[-\frac{1}{2}x_k^2\right]$$

Finally,

$$p(x_1,\ldots,x_n|\mu,\sigma^2) \geq \underbrace{\left\{\frac{1}{\sigma}+\exp\left[-\frac{1}{2}x_1^2\right]\right\}}_{\left(\begin{array}{c}\text{this term}\to\infty\\ \sigma\to 0\end{array}\right)} \frac{1}{(2\sqrt{2\pi})^n}\exp\left[-\frac{1}{2}\sum_{k=2}^{n}x_k^2\right]$$

The likelihood is therefore large and the maximum-likelihood solution becomes singular.

Applications to Normal Mixtures

- **Adding an assumption**

  Consider the largest of the finite local maxima of the likelihood function and use the ML estimation

  We obtain the following:

Iterative scheme

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

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

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

Applications to Normal Mixtures

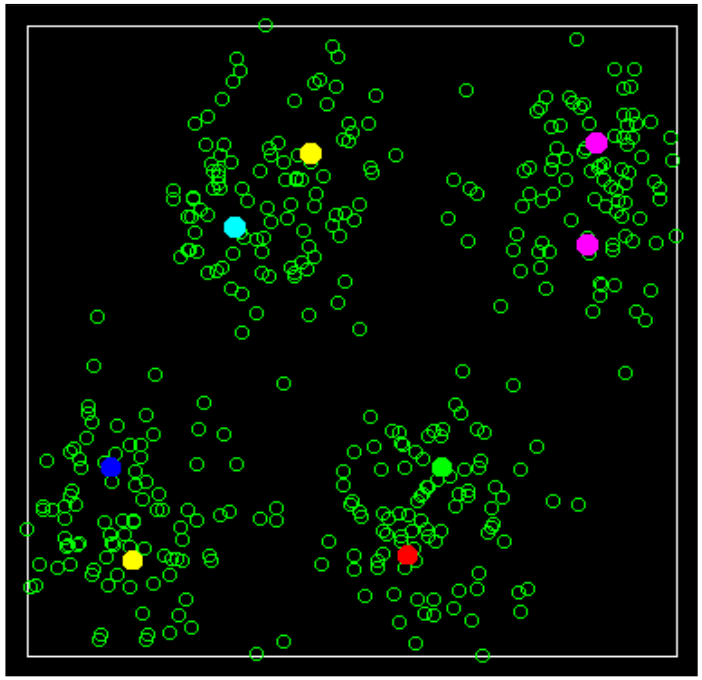
Where:

$$\hat{P}(\omega_i \mid x_k, \hat{\theta}) = \frac{\left|\Sigma_i\right|^{-1/2} exp\left[-\frac{1}{2}(x_k - \hat{\mu}_i)^t \hat{\Sigma}_i^{-1}(x_k - \hat{\mu}_i)\right]\hat{P}(\omega_i)}{\sum_{j=1}^{c}\left|\hat{\Sigma}_j\right|^{-1/2} exp\left[-\frac{1}{2}(x_k - \hat{\mu}_j)^t \hat{\Sigma}_j^{-1}(x_k - \hat{\mu}_j)\right]\hat{P}(\omega_j)}$$

Clustering

What is a Cluster?

A cluster is a collection of data points with similar properties.



Clustering

Partitional clustering

- C-means algorithm (Hard)
- Isodata algorithm
- Fuzzy C-means Clustering

Hierarchical clustering

- Single-linkage algorithm (minimum method)
- Complete-linkage algorithm (maximum method)
- Average linkage algorithm
- Minimum-variance method (Ward's method)

C-Means Clustering

- Goal: find the c mean vectors $\mu_1, \mu_2, \ldots, \mu_c$
- Replace the squared Mahalanobis distance in the previous discussion $(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)^T \hat{\boldsymbol{\Sigma}}_i^{-1} (\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)$ by the squared Euclidean distance $\|\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i\|^2$
- Find the mean $\hat{\mu}_m$ nearest to x_k and approximate

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

C-Means Clustering

- **Use the iterative scheme to find** $\hat{\mu}_1, \hat{\mu}_2, ..., \hat{\mu}_c$
- **if n is the known number of patterns and c the desired number of clusters, the C-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
```

C-Means Clustering

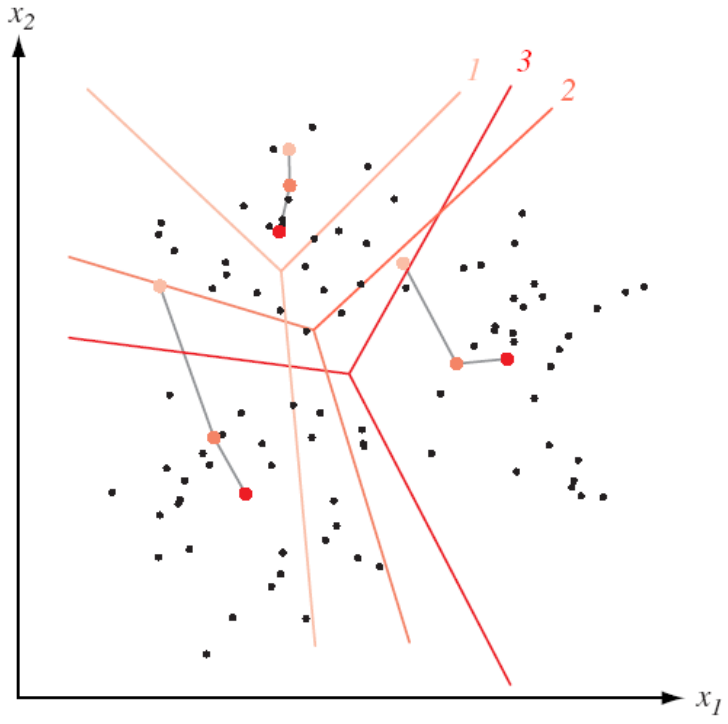


Figure 10.2: Trajectories for the means of the k-means clustering procedure applied to two-dimensional data. The final Voronoi tesselation (for classification) is also shown — the means correspond to the "centers" of the Voronoi cells.

C-Means Clustering

C-means algorithm:

1. Begin with C cluster centers
2. For each sample, find the cluster center nearest to it. Put the sample in the cluster represented by the just-found cluster center.
3. If no samples changed clusters, stop.
4. Recompute cluster centers of altered clusters and go back to step 2.

Properties:

- The number of cluster C must be given in advance.
- The goal is to minimize the square error, but it could end up in a local minimum.

Demo:
http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

Clustering: ISODATA

Similar to C-means with some enhancements:

- Clusters with too few elements are discarded.
- Clusters are merged if the number of clusters grows too large or if clusters are too close together.
- A cluster is split if the number of clusters is too few or if the cluster contains very dissimilar samples

Properties:

- The number of clusters C is not given exactly in advance.
- The algorithm may requires extensive computations.

Fuzzy C-Means Clustering

$X = \{ x_1, x_k, \ldots, x_n \}$ = Object Data in $\mathfrak{R}^p$

$\{ u_1, u_2, \ldots, u_c \}$ = Fuzzy Subsets $\in \mathfrak{I}(X)$

$u_i(x_k) = u_{ik}$ = membership of x_k in class (i)

$$U = \begin{array}{c|c|c|c|} & x_1 & x_k & x_n \\ & 1 & \downarrow & \\ & & \downarrow & \\ & i & \rightarrow\rightarrow\rightarrow\ u_{ik} & \rightarrow\rightarrow\rightarrow \\ & c & \downarrow & \end{array} \quad \textit{objects}$$

classes

Fuzzy C-Means Clustering

⇨ Row (i) ➠ Membership of all x_k's in class i

⇨ Col (k) ➠ Membership of x_k in each class i

U is a [Constrained] Probabilistic or
Fuzzy *c-Partition of X* iff:

1)	$0 < u_{ik} < 1$	any i,k (at least)
2)	$0 < \sum u_{ik}$	all (rowsums) k
3)	$1 = \sum u_{ik}$	all (colsums) i

Solution Spaces for Clustering

Constrained c-Partition Matrices

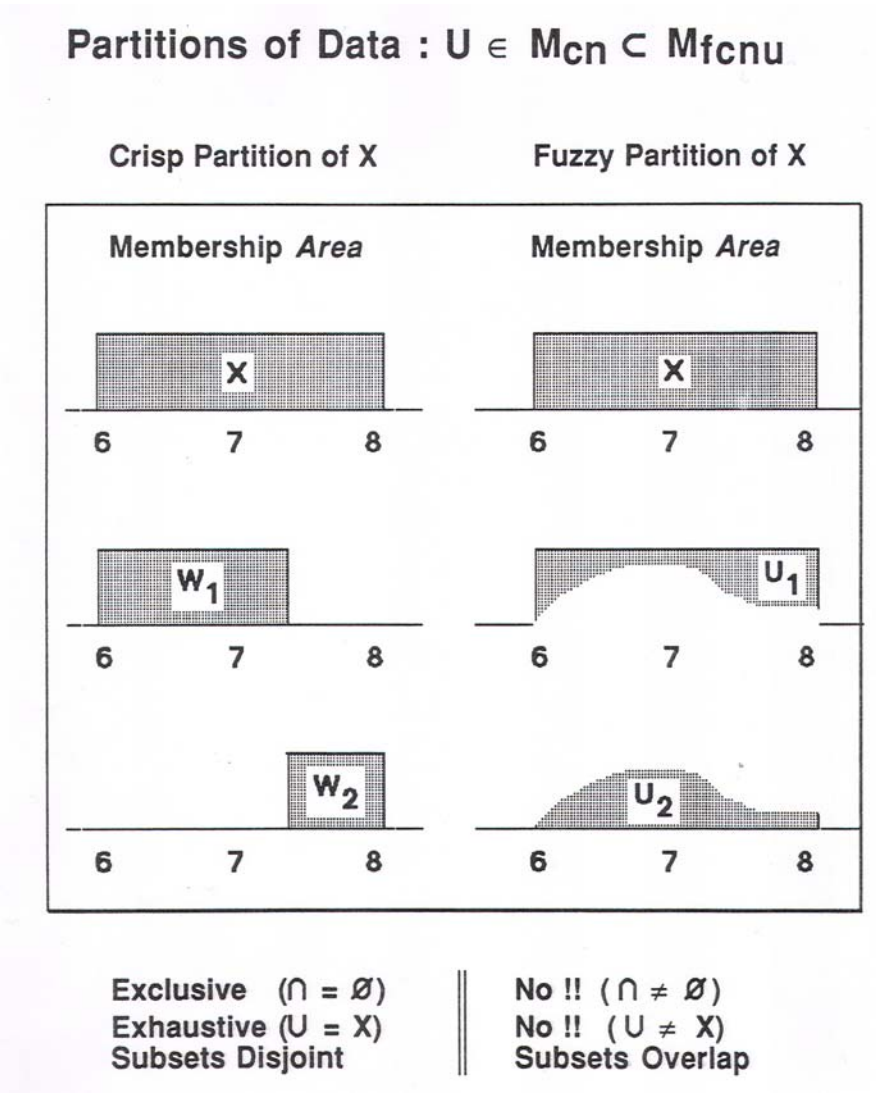
$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{vmatrix} \qquad \begin{vmatrix} 1 & .07 & 0 & .44 \\ 0 & .91 & 0 & .06 \\ 0 & .02 & 1 & .50 \end{vmatrix}$$

"Take a Second Look"

$U \in M_{cn}$ ↕ Crisp Partition Matrices

$U \in M_{fcn}$ ↕ Fuzzy Partition Matrices

Partition of Data



Defuzzification of Fuzzy Partitions

Conversion via max membership and α-cuts

$$\begin{vmatrix} 1 & .07 & 0 & .34 \\ 0 & .81 & 0 & .06 \\ 0 & .12 & 1 & .60 \end{vmatrix} \quad \text{MM} \Rightarrow \quad \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{vmatrix}$$

$$\begin{vmatrix} 1 & .07 & 0 & .34 \\ 0 & .81 & 0 & .06 \\ 0 & .12 & 1 & .60 \end{vmatrix} \quad \alpha \geq .9 \Rightarrow \quad \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

$$\begin{vmatrix} 1 & .07 & 0 & .34 \\ 0 & .81 & 0 & .06 \\ 0 & .12 & 1 & .60 \end{vmatrix} \quad \alpha \geq .8 \Rightarrow \quad \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

Fuzzy c-Means Clustering

The Fuzzy c-Means (FCM) Model

$X \subset \Re^p$ = Unlabeled Object Data

$U \in M_{fcn}$ = Unknown *Fuzzy* c-partition of X

$\underline{v} = \{v_i\}$ = Unknown "cluster centers": $v_i \in \Re^p$

$$\text{Min } J_m(U,\underline{v}\,;X) = \Sigma\Sigma\, u_{ik}{}^m\, (\|x_k - v_i\|_A)^2$$

Dunn	*('73)*	m = 2
Bezdek	*('73)*	m > 1

Gustafson/Kessel	('79)	Variable Norms $\{A_i\}$
Bezdek, et al	('80)	Linear Varieties for $\{v_i\}$
Pedrycz	('85)	Labeled/Unlabeled X
Yang and Yu	('89)	Integrals
Dave	('89)	Hyperspheres for $\{v_i\}$
Bezdek/Bobrowski	('90)	q =1 and q = ∞ norms
Krishnapuram	('91)	Hyperquadrics for $\{v_i\}$

Fuzzy c-Means (FCM) Clustering

Let's now assume that a sample can belong to *all* the clusters. Define u_{ij} where

$$\sum_{i=1}^{c} u_{ij} = 1, \ \forall j = 1, \ldots, n.$$

The cost function (or objective function) for FCM can then be written as

$$J\left(U, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_c\right) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j}^{n} u_{ij}^{m} \, d_{ij}^{2},$$

where u_{ij} is between 0 and 1; c_i is the cluster center of fuzzy group i; $d_{ij} = \|\mathbf{c}_i - \mathbf{x}_j\|$ is the Euclidean distance between *i*-th cluster center and *j*th data point; and $m \in [1, \infty)$ is a weighting exponent.

Fuzzy c-Means (FCM) Clustering

The updated mean vector is

$$\boldsymbol{c}_i = \frac{\sum_{j=1}^{n} u_{ij}^m \boldsymbol{x}_j}{\sum_{j=1}^{n} u_{ij}^m},$$

where

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}} \right)^{2/(m-1)}}.$$

C-means Algorithm (FCM/HCM)

c-Means Algorithms (FCM/HCM)

Given unlabeled data set $X = \{x_1, x_2, ..., x_n\}$ in $\Re^p$

Fix : $1 < c < n$ and $1 \le m < \infty$

A-norm : $\|x_k - v_j\|_A = \sqrt{(x_k - v_j)^T A (x_k - v_j)}$

ε, a small positive constant

FCM/HCM 1

Guess $\underline{v}_0 = (v_{1,0}, v_{2,0}, ..., v_{c,0}) \in \Re^{cp}$

FCM/HCM 2

For t = 1 to tmax:

Calculate U_t with $\{v_{i,t-1}\}$

Update $\{v_{i,t-1}\}$ to $\{v_{i,t}\}$ with U_t

If $\Sigma \|v_{i,t-1} - v_{i,t}\|^2 \} \le \varepsilon$,

Then stop and put $(U^*, v^*) = (U_t, v_t)$; Else

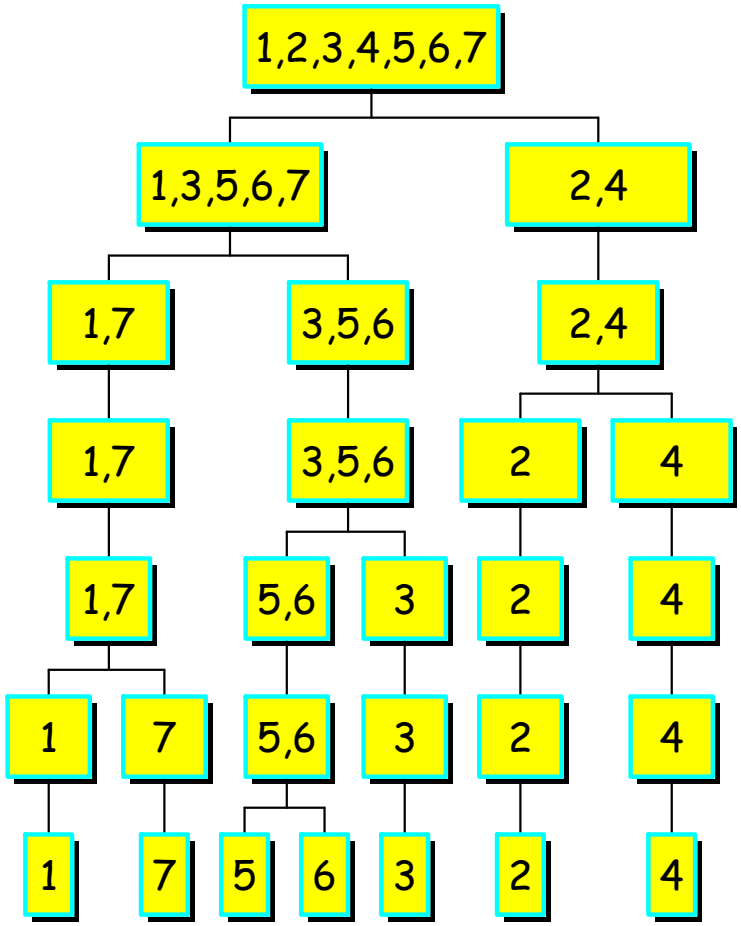
Next t

Hierarchical Clustering

Agglomerative clustering (bottom up)

1. Begin with n clusters; each containing one sample
2. Merge the most similar two clusters into one.
3. Repeat the previous step until done

Divisive clustering (top down)



Hierarchical Clustering

The most natural representation of hierarchical clustering is a corresponding tree, called a *dendrogram*, which shows how the samples are grouped

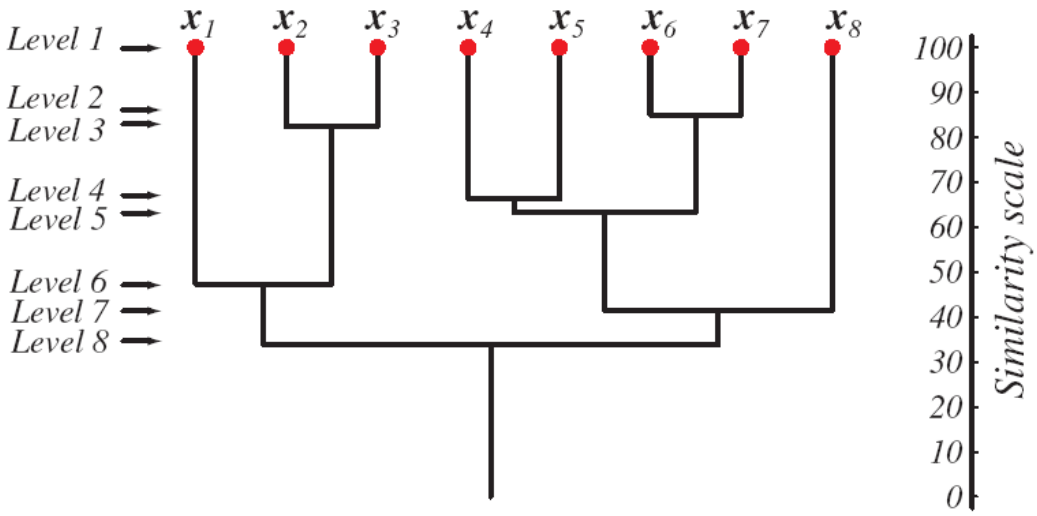


Figure 10.10: A dendrogram can represent the results of hierarchical clustering algorithms. The vertical axis shows a generalized measure of similarity among clusters. Here, at level 1 all eight points lie in singleton clusters; each point in a cluster is highly similar to itself, of course. Points $\mathbf{x}_6$ and $\mathbf{x}_7$ happen to be the most similar, and are merged at level 2, and so forth.

Hierarchical Clustering

Demo:
http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletH.html

Hierarchical Clustering

Single-linkage algorithm (minimum method)

$$D_{\min}(C_i, C_j) = \min_{a \in C_i, b \in C_j} d(a,b)$$

Complete-linkage algorithm (maximum method)

$$D_{\max}(C_i, C_j) = \max_{a \in C_i, b \in C_j} d(a,b)$$

Average-linkage algorithm (average method)

$$D_{ave}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{a \in C_i, b \in C_j} d(a,b)$$

Hierarchical Clustering

Ward's method (minimum-variance method)

$$D_{Ward}(C_i, C_j) = \sum_{j=1}^{d} \sigma_j^2 = \sum_{j=1}^{d} \sum_{i=1}^{m} \left(x_{ij} - \mu_j \right)^2$$

The distance measure characterizing Ward's method is based on the variance criterion (goal: small variance within each cluster and large variance between the clusters). In each step the two clusters are merged whose merging contributes least to the variance criterion which is increasing in each step. This distance measure is called the Ward distance and is defined as:

$$d_{rs} := \frac{n_r \cdot n_s}{n_r + n_s} \cdot \left\| \bar{x}_r - \bar{x}_s \right\|^2$$

where r and s denote two specific clusters, n_r and n_s denote the number of data points in the two clusters, and $\bar{x}_r$ and $\bar{x}_s$ denote the centers of gravity of the clusters; $\|.\|$ is the Euclidean norm.

Hierarchical Clustering

Single-linkage algorithm

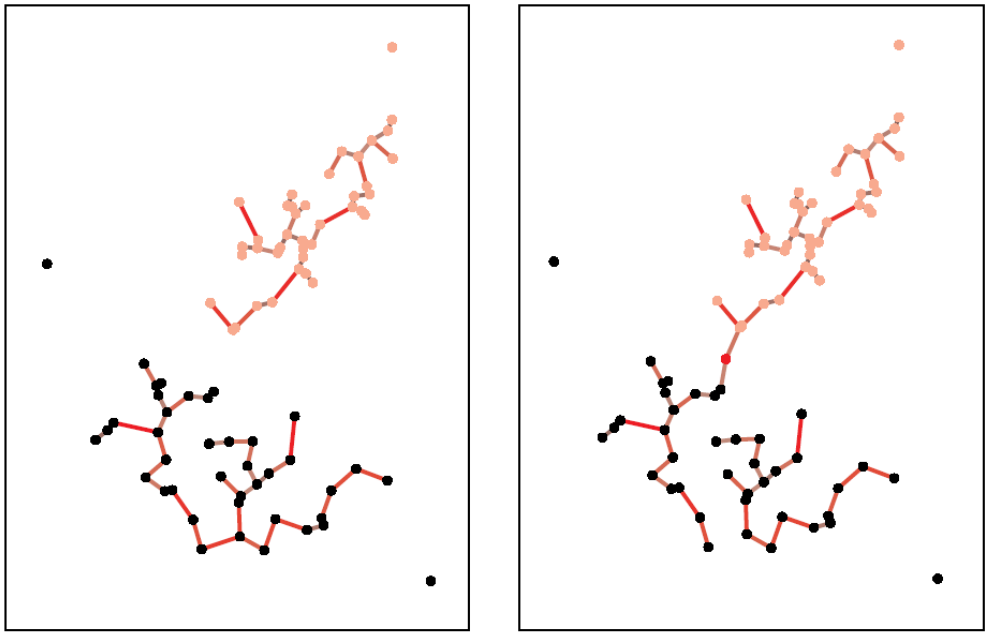


Figure 10.12: Two Gaussians were used to generate two-dimensional samples, shown in pink and black. The nearest-neighbor clustering algorithm gives two clusters that well approximate the generating Gaussians (left). If, however, another particular sample is generated (red point at the right) and the procedure re-started, the clusters do not well approximate the Gaussians. This illustrates how the algorithm is sensitive to the details of the samples.

Hierarchical Clustering

Complete-linkage algorithm

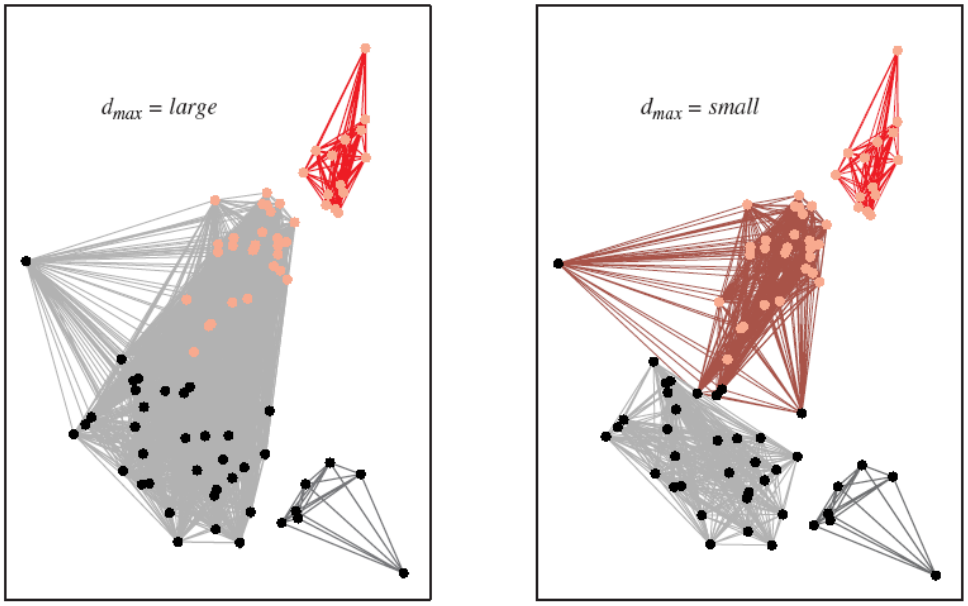


Figure 10.13: The farthest-neighbor clustering algorithm uses the separation between the most distant points as a criterion for cluster membership. If this distance is set very large, then all points lie in the same cluster. In the case shown at the left, a fairly large d_{max} leads to three clusters; a smaller d_{max} gives four clusters, as shown at the right.