Chapter 10
Unsupervised Learning & Clustering

- ✓ Introduction
- ✓ Mixture Densities and Identifiability
- ✓ ML Estimates
- ✓ Application to Normal Mixtures
- ✓ K-means algorithm
- ✓ Unsupervised Bayesian Learning
- ✓ Data description and clustering
- ✓ Criterion function for clustering
- ✓ Hierarchical clustering
- ✓ The number of cluster problem and cluster validation
- ✓ On-line clustering
- ✓ Graph-theoretic methods
- ✓ PCA and ICA
- ✓ Low-dim reps and multidimensional scaling (self-organizing maps)
- ✓ Clustering and dimensionality reduction

Introduction

- Previously, all our training samples were labeled: these procedures were said “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 large amounts 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

- 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 an unsupervised mode are used

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

- 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, …,c)*
  - $P(\mathbf{x} | \omega_j, \theta_j)$ (j = 1, …,c) are known
  - The values of the c parameter vectors $\theta_1, \theta_2, \ldots, \theta_c$ are unknown

– The category labels are unknown

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

$$\text{where } \boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_c)^t$$

- 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 $\boldsymbol{\theta}$.
- Once $\boldsymbol{\theta}$ is known, we can decompose the mixture into its components and use a MAP classifier on the derived densities.

- Definition

A density $P(\mathbf{x} \mid \theta)$ is said to be identifiable if

$\theta \neq \theta'$ implies that there exists an $\mathbf{x}$ such that:

$$P(\mathbf{x} \mid \theta) \neq P(\mathbf{x} \mid \theta')$$

As a simple example, consider the case where $\mathbf{x}$ is binary and $P(\mathbf{x} \mid \theta)$ is the mixture:

$$\boxed{\begin{aligned} P(x \mid \boldsymbol{\theta}) &= \frac{1}{2}\theta_1^x(1-\theta_1)^{1-x} + \frac{1}{2}\theta_2^x(1-\theta_2)^{1-x} \\ &= \begin{cases} \frac{1}{2}(\theta_1+\theta_2) \text{ if } x=1 \\ 1\text{-}\frac{1}{2}(\theta_1+\theta_2) \text{ if } x=0 \end{cases} \end{aligned}}$$

Assume that:

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

by replacing these probabilities values, we obtain:

$$\theta_1 + \theta_2 = 1.2$$

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

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

  cannot be uniquely identified if $P(\omega_1) = P(\omega_2)$
  (we cannot recover a unique $\boldsymbol{\theta}$ even from an infinite amount of data!)
- $\boldsymbol{\theta} = (\theta_1, \theta_2)$ and $\boldsymbol{\theta} = (\theta_2, \theta_1)$ are two possible vectors that can be interchanged without affecting $P(x \mid \boldsymbol{\theta})$.
- 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 $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n unlabeled samples drawn independently from the mixture density

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

($\boldsymbol{\theta}$ is fixed but unknown!)

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

The gradient of the log-likelihood is:

$$\left(l = \sum_{k=1}^{n} \ln p(\mathbf{x}_k \mid \boldsymbol{\theta})\right) \Rightarrow \boxed{\nabla_{\boldsymbol{\theta}_i} l = \sum_{k=1}^{n} \frac{1}{p(\mathbf{x}_k|\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}_i} \left[\sum_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \boldsymbol{\theta}_j) P(\omega_j) \right]}$$

$$\boxed{P(\omega_i \mid \mathbf{x}_k, \boldsymbol{\theta}) = \frac{p(\mathbf{x}_k \mid \omega_i, \boldsymbol{\theta}_i) P(\omega_i)}{p(\mathbf{x}_k \mid \boldsymbol{\theta})}}$$

$$\nabla_{\boldsymbol{\theta}_i} l = \sum_{k=1}^{n} \frac{1}{p(\mathbf{x}_k \mid \boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}_i} \left[p(\mathbf{x}_k \mid \omega_i, \boldsymbol{\theta}_i) P(\omega_i) \right]$$

$$= \sum_{k=1}^{n} \frac{P(\omega_i)}{p(\mathbf{x}_k \mid \boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}_i} \left[p(\mathbf{x}_k \mid \omega_i, \boldsymbol{\theta}_i) \right]$$

$$= \sum_{k=1}^{n} \frac{P(\omega_i \mid \mathbf{x}_k, \boldsymbol{\theta})}{p(\mathbf{x}_k \mid \omega_i, \boldsymbol{\theta}_i)} \nabla_{\boldsymbol{\theta}_i} \left[p(\mathbf{x}_k \mid \omega_i, \boldsymbol{\theta}_i) \right]$$

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

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

Since the gradient must vanish at the value of θ_i

that maximizes l $\left(l = \sum_{k=1}^{n} \ln p(\mathbf{x}_k \mid \boldsymbol{\theta})\right)$, therefore,

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

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

By including the prior probabilities as unknown variables, we finally obtain (problem 6):

Constraints: $P(\omega_i) \geq 0 \quad i = 1, \ldots, c \qquad \sum_{i=1}^{c} P(\omega_i) = 1.$

$$\hat{P}(\omega_i) = \frac{1}{n}\sum_{k=1}^{n} \hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}}) \qquad (11)$$

$$\sum_{k=1}^{n} \hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\theta}) \nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k \mid \omega_i, \hat{\boldsymbol{\theta}}_i) = 0 \qquad (12)$$

$$where\text{:}\ \ \hat{\mathrm{P}}(\omega_{\mathrm{i}} \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}}) = \frac{p(\mathbf{x}_k \mid \omega_i, \hat{\boldsymbol{\theta}}_i)\hat{P}(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}_k \mid \omega_j, \hat{\boldsymbol{\theta}}_j)\hat{P}(\omega_j)} \qquad (13)$$

Equation 11 states that the maximum-likelihood estimate of the probability of a category is the average over the entire data set of the estimate derived from each sample — each sample is weighted equally.

Equation 13 is ultimately related to Bayes Theorem, but notice that in estimating the probability for class ω_i, the numerator on the right-hand side depends on $\hat{\boldsymbol{\theta}}_i$ and not the full $\hat{\boldsymbol{\theta}}$ directly.

Applications to Normal Mixtures

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

Case	$\boldsymbol{\mu}_i$	$\boldsymbol{\Sigma}_i$	$P(\omega_i)$	c
1	?	√	√	√
2	?	?	?	√
3	?	?	?	?

Case 1 = Simplest case

Case 1: Unknown mean vectors

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

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

its derivative is

$$\nabla_{\boldsymbol{\mu}_i} \ln p(\mathbf{x} \mid \omega_i, \boldsymbol{\mu}_i) = \Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)$$

the maximum-likelihood estimate $\hat{\boldsymbol{\mu}}_i$ must satisfy

$$\sum_{k=1}^{n} P(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\mu}}) \Sigma_i^{-1} (\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i) = 0 \qquad \text{where} \qquad \hat{\boldsymbol{\mu}} = (\hat{\boldsymbol{\mu}}_1, \cdots, \hat{\boldsymbol{\mu}}_c)^t.$$

After multiplying by $\boldsymbol{\Sigma}_i$ and rearranging terms, we obtain the solution:

$$\hat{\boldsymbol{\mu}}_i = \frac{\sum_{k=1}^{n} P(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\mu}})\mathbf{x}_k}{\sum_{k=1}^{n} P(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\mu}})} \qquad (17)$$

The ML estimate for $\boldsymbol{\mu}_i$ is merely a weighted average of the samples; the weight for the *k*th sample is an estimate of how likely it is that $\mathbf{x}_k$ belongs to the *i*th class. If $P\left(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\mu}}\right)$ happened to be 1.0 for some of the samples and 0.0 for the rest, then $\hat{\boldsymbol{\mu}}_i$ would be the mean of those samples estimated to belong to the *i*th class. More generally, suppose that $\hat{\boldsymbol{\mu}}_i$ is sufficiently close to the true value of $\boldsymbol{\mu}_\mathbf{i}$ that $P\left(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\mu}}\right)$ is essentially the true posterior probability for ω_i.

$P(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\mu}})$is the fraction of those samples having value $\mathbf{x}_k$ that come from the *i*-th class, and $\hat{\boldsymbol{\mu}}_i$ is the average of the samples coming from the *i*-th class. Unfortunately, equation (1) does not give $\hat{\boldsymbol{\mu}}_i$ explicitly and if we substitute

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

with $p(\mathbf{x}|\omega_i, \hat{\boldsymbol{\mu}}_i) \sim N(\hat{\boldsymbol{\mu}}_i, \boldsymbol{\Sigma}_i)$, we obtain a tangled snarl of coupled simultaneous nonlinear equations.

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

$$\hat{\boldsymbol{\mu}}_i(j+1) = \frac{\sum_{k=1}^{n} P(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\mu}}(j))\mathbf{x}_k}{\sum_{k=1}^{n} P(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\mu}}(j))} \qquad (18)$$

- This is a gradient ascent for maximizing the log-likelihood function
- If the overlap between component densities is small, then the coupling between classes will be small and convergence will be fast.

- Example:

  Consider the simple two-component one-dimensional normal mixture

$$p(x \mid \mu_1, \mu_2) = \underbrace{\frac{1}{3\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\mu_1)^2\right]}_{\omega_1} + \underbrace{\frac{2}{3\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\mu_2)^2\right]}_{\omega_2}$$

  (2 clusters!)

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

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

k	x_k	ω_1	ω_2
1	0.608		$\times$
2	-1.590	$\times$	
3	0.235		$\times$
4	3.949		$\times$
5	-2.249	$\times$	
6	2.704		$\times$
7	-2.473	$\times$	
8	0.672		$\times$

k	x_k	ω_1	ω_2
9	0.262		$\times$
10	1.072		$\times$
11	-1.773	$\times$	
12	0.537		$\times$
13	3.240		$\times$
14	2.400		$\times$
15	-2.499	$\times$	
16	2.608		$\times$

k	x_k	ω_1	ω_2
17	-3.458	$\times$	
18	0.257		$\times$
19	2.569		$\times$
20	1.415		$\times$
21	1.410		$\times$
22	-2.653	$\times$	
23	1.396		$\times$
24	3.286		$\times$
25	-0.712	$\times$	

The maximum value of l occurs at:

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

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

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

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

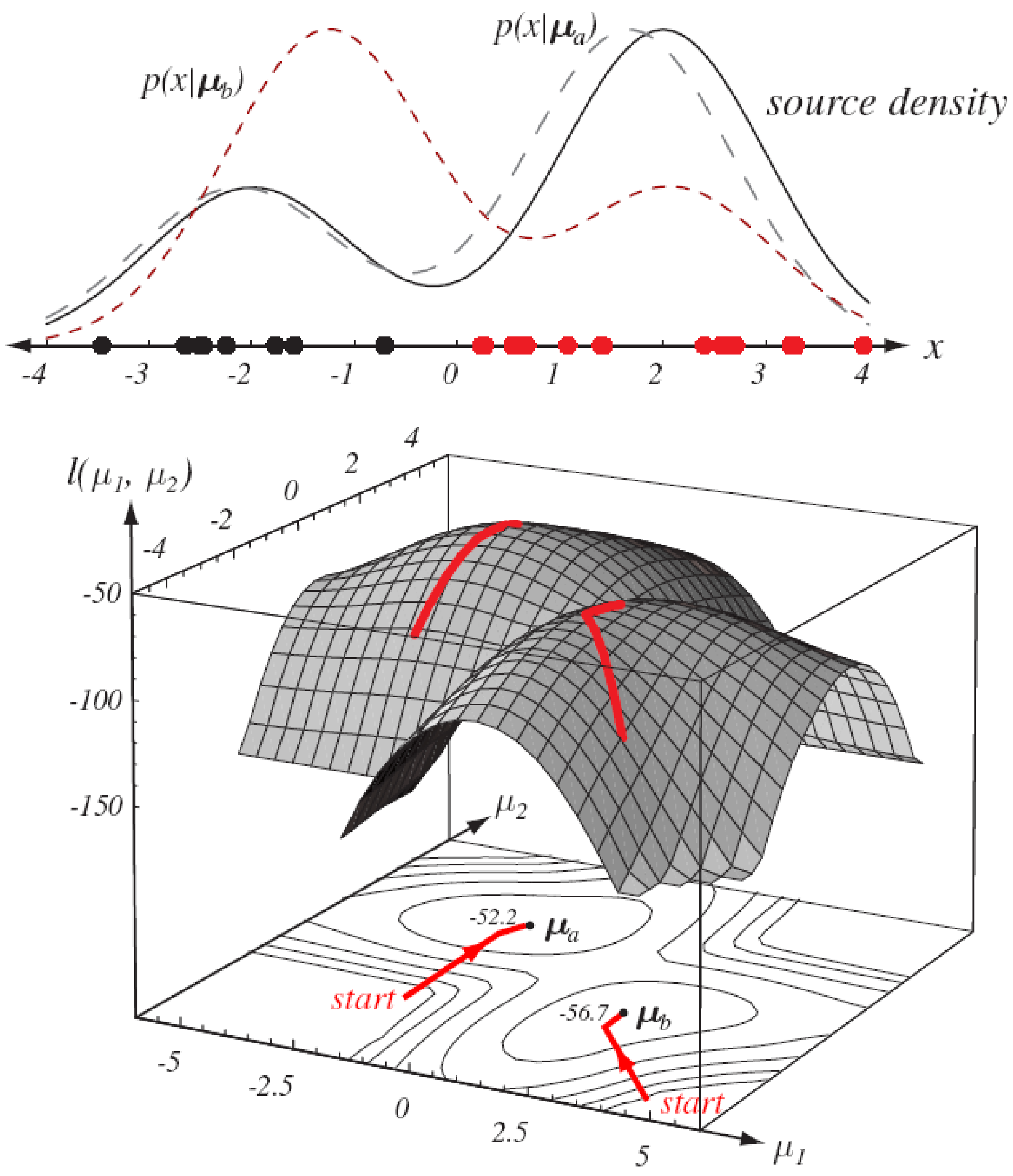


FIGURE 10.1. (Above) The source mixture density used to generate sample data, and two maximum likelihood estimates based on the data in the table. (Bottom) Loglikelihood of a mixture model consisting of two univariate Gaussians as a function of their means, for the data in the table. Trajectories for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

Case 2: All parameters unknown (c is known)

– If no constraints are placed on the covariance matrix, the ML principle yields useless singular solutions.

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

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

Suppose $\mu = x_1$, therefore:

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

For the rest of the samples:

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

Finally,

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

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

We had

$$\ln p(\mathbf{x}_k|\omega_i, \boldsymbol{\theta}_i) = \ln \frac{|\boldsymbol{\Sigma}_i^{-1}|^{1/2}}{(2\pi)^{d/2}} - \frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_i)$$

Note:

1. Only half of the off-diagonal elements of $\boldsymbol{\Sigma}_i$ are independent.
2. It is convenient to let the independent elements of $\boldsymbol{\Sigma}_i^{-1}$ rather than $\boldsymbol{\Sigma}_i$ be the unknown.

The actual differentiation of the above equation with respect to the elements of $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i^{-1}$ is relatively routine.

Let $x_p(k)$ be the pth element of $\mathbf{x}_k$, $\mu_p(i)$ be the pth element of $\boldsymbol{\mu}_i$, $\sigma_{pq}(i)$ be the pqth element of $\boldsymbol{\Sigma}_i$, and $\sigma^{pq}(i)$be the pqth element of $\boldsymbol{\Sigma}_i^{-1}$. Then differentiation gives:

$$\nabla_{\boldsymbol{\mu}_i} \ln p(\mathbf{x}_k|\omega_i, \boldsymbol{\theta}_i) = \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_i)$$

and

$$\frac{\partial \ln p(\mathbf{x}_k|\omega_i, \boldsymbol{\theta}_i)}{\partial \sigma^{pq}(i)} = \left(1 - \frac{\delta_{pq}}{2}\right)\left[\sigma_{pq}(i) - (x_p(k) - \mu_p(i))(x_q(k) - \mu_q(i))\right],$$

where δ_{pq} is the Kronecker delta.

We substitute these results in Eq. 12 and perform a small amount of algebraic manipulation, then we have Equations (24-26).

- Adding an assumption

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

  We obtain the following (after algebraic manipulations):

$$
\text{Iterative scheme}\left\{
\begin{aligned}
&\hat{P}(\omega_i) = \frac{1}{n}\sum_{k=1}^{n}\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}}) && (24)\\
&\hat{\boldsymbol{\mu}}_i = \frac{\sum_{k=1}^{n}\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}})\mathbf{x}_k}{\sum_{k=1}^{n}\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}})} && (25)\\
&\hat{\boldsymbol{\Sigma}}_i = \frac{\sum_{k=1}^{n}\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)^t}{\sum_{k=1}^{n}\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}})} && (26)
\end{aligned}
\right.
$$

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

$$= \frac{\left|\hat{\boldsymbol{\Sigma}}_i\right|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)^t \hat{\boldsymbol{\Sigma}}_i^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)\right]\hat{P}(\omega_i)}{\sum_{j=1}^{c}\left|\hat{\boldsymbol{\Sigma}}_j\right|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_j)^t \hat{\boldsymbol{\Sigma}}_j^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_j)\right]\hat{P}(\omega_j)} \tag{27}$$

In the extreme case where $\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}})$ is 1.0 when $\mathbf{x}_k$ is from Class ω_i and 0.0 otherwise, $\hat{P}(\omega_i)$ is the fraction of samples from ω_i, $\hat{\boldsymbol{\mu}}_i$ is the mean of those samples, and $\hat{\boldsymbol{\Sigma}}_i$ is the corresponding sample covariance matrix. More generally, $\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}})$ is between 0.0 and 1.0, and all of the samples play some role in the estimates. However, the estimates are basically still frequency ratios, sample means, and sample covariance matrices.

Of the various techniques that can be used to obtain a solution, the most obvious approach is to use initial estimates to evaluate Eq. 27 for $\hat{P}(\omega_i \mid \mathbf{x}_k, \hat{\boldsymbol{\theta}})$ and then to use Eqs. 24 – 26 to update these estimates.

If the initial estimates are very good, having perhaps been obtained from a fairly large set of labeled samples, convergence can be quite rapid.

Considerable simplification can be obtained if it is possible to assume that the covariance matrices are diagonal. This has the added virtue of reducing the number of unknown parameters, which is very important when the number of samples is not large.

Chapter 10-part2

- **K-Means Clustering**

  – Goal: find the *c* mean vectors $\mu_1, \mu_2, \ldots, \mu_c$

  – Replace the squared Mahalanobis distance

$$(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)^t \hat{\boldsymbol{\Sigma}}_i^{-1} (\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i) \text{ by the sq. Euc. distance } \left\| \mathbf{x}_\mathrm{k} - \hat{\boldsymbol{\mu}}_i \right\|^2$$

  – Find the mean $\hat{\boldsymbol{\mu}}_m$ nearest to $\mathbf{x}_k$ and approximate

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

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

– If n is the known number of patterns and c the desired number of clusters, the k-means algorithm is:
(Note: c samples randomly chosen from the dataset as initial cluster centers)

```
Begin
    initialize n, c, $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, …, $\boldsymbol{\mu}_c$ (randomly
    selected)
            do classify n samples according to
                    nearest $\boldsymbol{\mu}_i$
                    recompute $\boldsymbol{\mu}_i$
            until no change in $\boldsymbol{\mu}_i$
    return $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, …, $\boldsymbol{\mu}_c$
End
```

Exercise 2 p.594 (Textbook)

- Considering the example in the previous figure

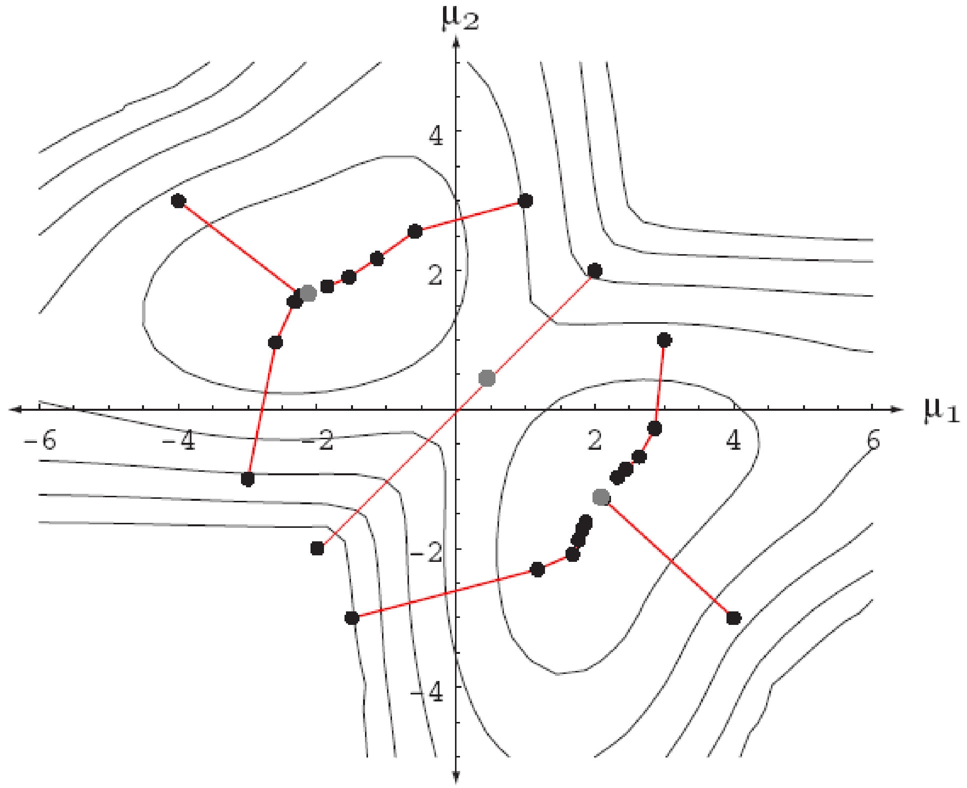
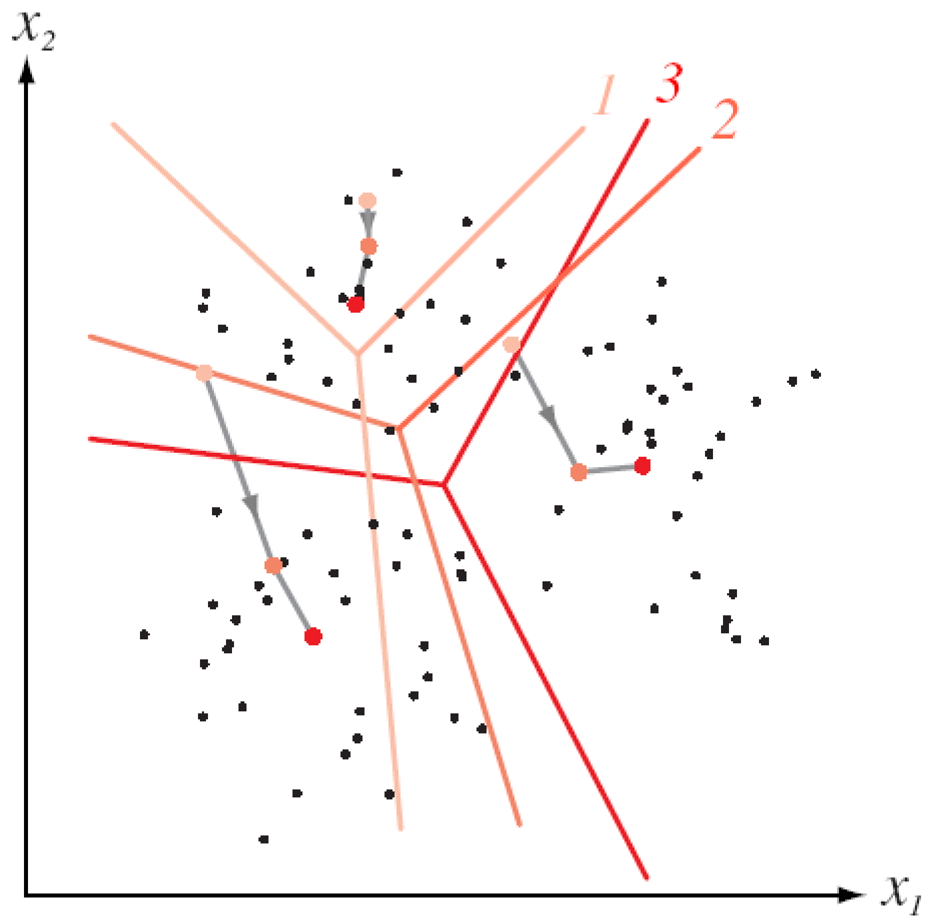


Figure 10.1: The k-means clustering procedure is a form of stochastic hill climbing in the log-likelihood function. The contours represent equal log-likelihood values for the one-dimensional data in Example 1. The dots indicate parameter values after different iterations of the k-means algorithm. Six 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* $\boldsymbol{\mu} = \mathbf{0}$.

- Figure 10.1 shows the sequence of values for $\hat{\boldsymbol{\mu}}_1$ and $\hat{\boldsymbol{\mu}}_2$ obtained for several different starting points. Since interchanging $\hat{\boldsymbol{\mu}}_1$ and $\hat{\boldsymbol{\mu}}_2$ merely interchanges the labels assigned to the data, the trajectories are symmetric about the line $\hat{\boldsymbol{\mu}}_1 = \hat{\boldsymbol{\mu}}_2$. The trajectories lead either to the point $\hat{\boldsymbol{\mu}}_1 = -2.176$, $\hat{\boldsymbol{\mu}}_2 = 1.684$ or to its symmetric image. This is close to the solution found by the maximum-likelihood method (viz., $\hat{\boldsymbol{\mu}}_2 = -2.130$ and $\hat{\boldsymbol{\mu}}_1 = 1.688$), and the trajectories show a general resemblance to those shown in Example 1.



C=3

The three initial cluster centers, chosen randomly from the training points.

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.

Fuzzy k-means clustering

- In every iteration of the classical k-means procedure, each data point is assumed to be in exactly one cluster
- We can relax this condition and assume that each sample $\mathbf{x}_j$ has some graded or “fuzzy” cluster membership $\mu_i(\mathbf{x}_j)$ in cluster ω_i, where $0 \leq \mu_i(\mathbf{x}_j) \leq 1$.
- At root, these “memberships” are equivalent to the probabilities $\hat{P}(\omega_i \mid \mathbf{x}_j, \hat{\boldsymbol{\theta}})$
- In the resulting fuzzy k-means clustering algorithm we seek a minimum of a global cost function

$$L = \sum_{i=1}^{c} \sum_{j=1}^{n} [\hat{P}(\omega_i|\mathbf{x}_j, \hat{\theta})]^b ||\mathbf{x}_j - \boldsymbol{\mu}_i||^2,$$

where *b* is a free parameter chosen to adjust the "blending" of different clusters. If *b* is set to 0, this criterion function is merely a sum-of-squared errors criterion we shall see again in Eq. 49. If *b*>1, criterion allows each pattern to belong to multiple clusters.
The probabilities of cluster membership for each point are normalized as

$$\sum_{i=1}^{c} \hat{P}(\omega_i|\mathbf{x}_j) = 1, \qquad j = 1, \ldots, n. \tag{25}$$

At the solution, i.e., the minimum of L, we have

$$\partial L / \partial \boldsymbol{\mu}_i = 0 \quad \text{and} \quad \partial L / \partial \hat{P}_j = 0,$$

Then we have

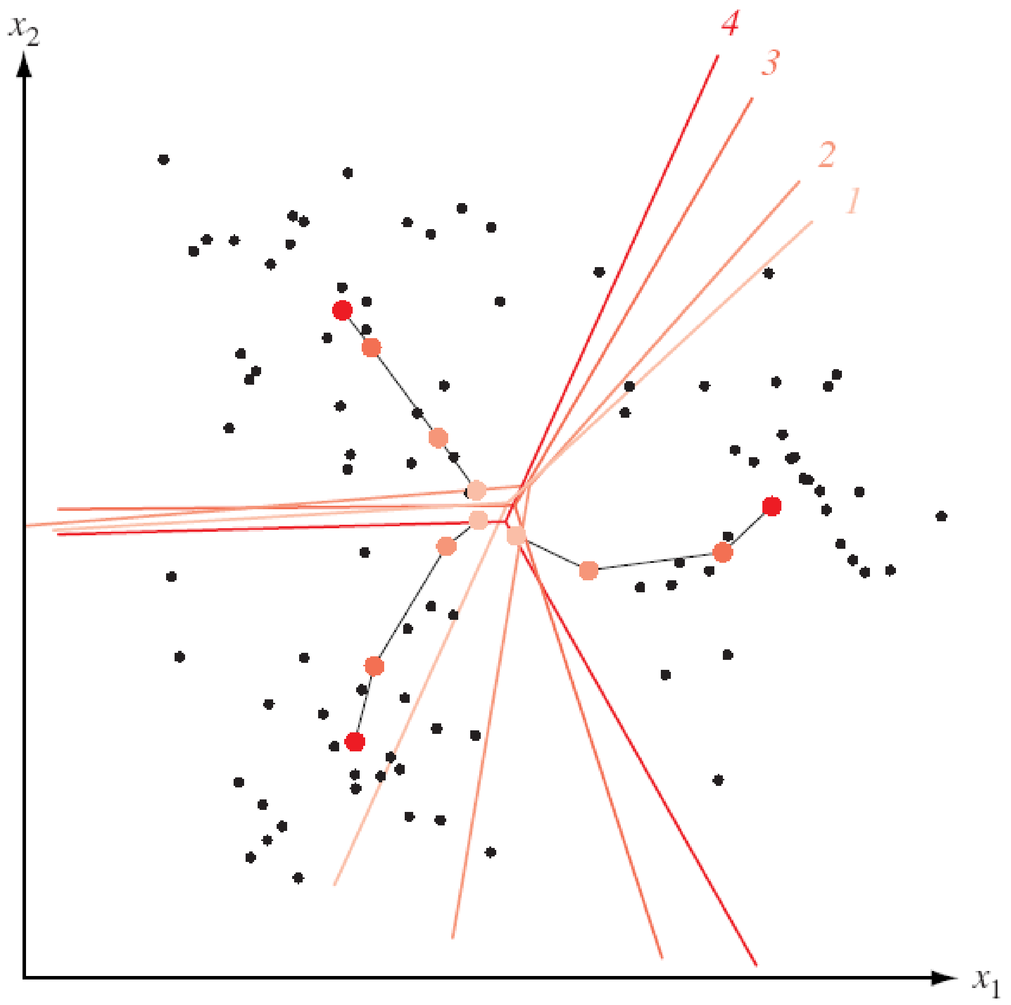
$$\boldsymbol{\mu}_j = \frac{\sum_{j=1}^{n} [P(\omega_i|\mathbf{x}_j)]^b \mathbf{x}_j}{\sum_{j=1}^{n} [P(\omega_i|\mathbf{x}_j)]^b} \qquad (27)$$

and

$$P(\omega_i|\mathbf{x}_j) = \frac{(1/d_{ij})^{1/(b-1)}}{\sum_{r=1}^{c} (1/d_{rj})^{1/(b-1)}}, \qquad d_{ij} = ||\mathbf{x}_j - \boldsymbol{\mu}_i||^2. \qquad (28)$$

Algorithm 2 (Fuzzy k-means clustering)

```
1 begin initialize $n, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c, P(\omega_i \mid \mathbf{x}_j), i = 1 \ldots, c;\ j = 1, \ldots, n$
2         normalize proabilities of cluster memberships by Eq. 25
3         do classify $n$ samples according to nearest $\boldsymbol{\mu}_i$
4             recompute $\boldsymbol{\mu}_i$ by Eq. 27
5             recompute $P(\omega_i \mid \mathbf{x}_j)$ by Eq. 28
6         until no change in $\boldsymbol{\mu}_i$ and $P(\omega_i \mid \mathbf{x}_j)$
7     return $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_c$
8 end
```



At early iterations the means lie near the center of the full data set because each point has a non-negligible “membership” (i.e., probability) in each cluster. At later iterations the means separate and each membership tends toward the value 1.0 or 0.0.

the classical k-means algorithm is just of special case where the memberships for all points obey

$$P(\omega_i|\mathbf{x}_j) = \begin{cases} 1 & \text{if } \|\mathbf{x}_j - \boldsymbol{\mu}_i\| < \|\mathbf{x}_j - \boldsymbol{\mu}_{i'}\| \text{ for all } i' \neq i \\ 0 & \text{otherwise,} \end{cases}$$

*Unsupervised Bayesian Learning

- Other than the ML estimate, the Bayesian estimation technique can also be used in the unsupervised case (see Ch: 3 - ML & Bayesian methods)
  - The number of classes is known
  - So as their prior distributions and the forms of the class-conditional probability densities $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$
  - The parameters' vector $\boldsymbol{\theta}$ is unknown and is assumed as a random variable
  - Part of the knowledge about $\boldsymbol{\theta}$ is contained in the priors $p(\boldsymbol{\theta})$, and the rest is the training samples
  - We compute the posterior distribution using the training samples

the familiar mixture density is

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

$$P(\omega_i \mid \mathbf{x}, D) = \frac{p(\mathbf{x} \mid \omega_i, D)P(\omega_i \mid D)}{\sum_{j=1}^{c} p(\mathbf{x} \mid \omega_j, D)P(\omega_j \mid D)} = \frac{p(\mathbf{x} \mid \omega_i, D)P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x} \mid \omega_j, D)P(\omega_j)} \quad (37)$$

$$p(\mathbf{x} \mid \omega_i, D) = \int p(\mathbf{x}, \boldsymbol{\theta} \mid \omega_i, D)\, d\boldsymbol{\theta} = \int p(\mathbf{x} \mid \boldsymbol{\theta}, \omega_i, D)\, p(\boldsymbol{\theta} \mid \omega_i, D)\, d\boldsymbol{\theta}$$

$$= \int p(\mathbf{x} \mid \omega_i, \boldsymbol{\theta}_i)\, p(\boldsymbol{\theta} \mid D)\, d\boldsymbol{\theta} \qquad (38\text{-}39)$$

Since the selection of ***x*** is independent of the samples, we have $p(\mathbf{x}|\boldsymbol{\theta},\square_i,D) = p(\mathbf{x}|\square_i, \boldsymbol{\theta}_i)$. Similarly, since knowledge of the state of nature when **x** is selected tells us nothing about the distribution of $\boldsymbol{\theta}$, we have $p(\boldsymbol{\theta}|\square_i,D) = p(\boldsymbol{\theta}|D)$

- **Learning the Parameter Vector :**
- Using Bayes

$$p(\boldsymbol{\theta} \mid D) = \frac{p(D \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{\int p(D \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})\, d\boldsymbol{\theta}}$$

- Assuming the independence of the samples

$$p(D \mid \boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k \mid \boldsymbol{\theta})$$

or alternately (denoting D^n the set of n samples)

$$p(\boldsymbol{\theta} \mid D^n) = \frac{p(\mathbf{x}_n \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid D^{n-1})}{\int p(\mathbf{x}_n \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid D^{n-1})\, d\boldsymbol{\theta}}$$

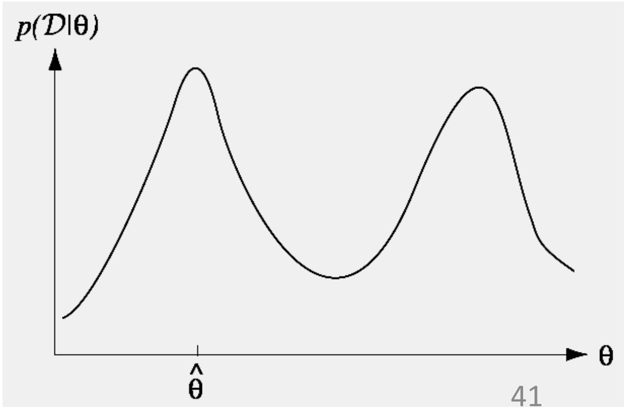
- If $p(\theta)$ is almost uniform in the region where $p(D|\theta)$ peaks, then $p(\theta|D)$ peaks in the same place.

- If the only significant peak occurs at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ and the peak is very sharp, then Eqs 37 & 39 yield

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

- Therefore, the ML estimate is justified.
- Both approaches coincide if large amounts of data are available.
- In small sample size problems they can agree or not, depending on the form of the distributions
- The ML method is typically easier

  to implement than the Bayesian one



- The formal Bayesian solution to the unsupervised learning of the parameters of a mixture density is similar to the supervised learning of the parameters of a component density.
- But, there are significant differences: the issue of identifiability, and the computational complexity
- The issue of identifiability
  - With SL, the lack of identifiability means that we do not obtain a unique vector, but an equivalence class, which does not present theoretical difficulty as all yield the same component density.
  - With UL, the lack of identifiability means that the mixture cannot be decomposed into its true components

    $\Rightarrow p(\mathbf{x} / D^n)$ may still converge to $p(\mathbf{x})$, but $p(\mathbf{x} / \omega_{\mathrm{i}}, D^n)$ will not in general converge to $p(\mathbf{x} / \omega_{\mathrm{i}})$, hence there is a theoretical barrier.
  - It is here that a few labeled training samples would be valuable: for “decomposing” the mixture into its components.

- The computational complexity
  - With SL, the sufficient statistics allows the solutions to be computationally feasible. With UL, samples comes from a mixture density and there is little hope of finding simple exact sol. for $p(D/\boldsymbol{\theta})$.
- Another way of comparing the UL and SL is to consider the usual equation in which the mixture density is explicit

$$p(\boldsymbol{\theta} \mid D^n) = \frac{p(\mathbf{x}_n \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid D^{n-1})}{\int p(\mathbf{x}_n \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid D^{n-1}) d\boldsymbol{\theta}} =$$

$$= \frac{\sum_{j=1}^{c} p(\mathbf{x}_n \mid \omega_j, \boldsymbol{\theta}_j) P(\omega_j)}{\sum_{j=1}^{c} \int p(\mathbf{x}_n \mid \omega_j, \boldsymbol{\theta}_j) P(\omega_j)\, p(\boldsymbol{\theta} \mid D^{n-1}) d\boldsymbol{\theta}}\, p(\boldsymbol{\theta} \mid D^{n-1})$$

- If we consider the case in which $P(\omega_1)=1$ and all other prior probabilities as zero, corresponding to the supervised case in which all samples comes from the class ω_1, then we get

$$p(\boldsymbol{\theta} \mid D^n) = \frac{p(\mathbf{x}_n \mid \omega_1, \boldsymbol{\theta}_1)}{\int p(\mathbf{x}_n \mid \omega_1, \boldsymbol{\theta}_1) p(\boldsymbol{\theta} \mid D^{n-1}) d\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid D^{n-1})$$

- Comparing the two equations, we see that observing an additional sample changes the estimate of $\boldsymbol{\theta}$.
- Ignoring the denominator which is independent of $\boldsymbol{\theta}$, the only significant difference is that
  - in the SL, we multiply the "prior" density for $\boldsymbol{\theta}$ by the component density $p(\mathbf{x}_n \mid \omega_1, \boldsymbol{\theta}_1)$
  - In the UL, we multiply the "prior" density by the whole mixture

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

- Assuming that the sample did come from class ω_1, the effect of not knowing this category is to diminish the influence of $\mathbf{x}_n$ in changing $\boldsymbol{\theta}$.

Summary of discussions

- If $p(\boldsymbol{\theta})$ has been obtained by supervised learning from a large set of labeled samples, it will be far from uniform and it will have a dominant influence on $p(\boldsymbol{\theta}|D^n)$ when n is small.
- Each sample sharpens $p(\boldsymbol{\theta}|D^n)$. In the limit it will converge to a Dirac delta function centered at the true value of $\boldsymbol{\theta}$.
- Thus, even though we don't know the categories of the samples, identifiabiliy assures us that we can learn the unknown parameter $\boldsymbol{\theta}$.
- Unsupervised learning of parameters is very similar to supervised learning.
- One significant difference: with supervised learning the lack of identifiability means that instead of obtaining a unique parameter vector, we obtain an equivalent class of parameter vectors.
- For unsupervised training, a lack of identifiability means that even though $p(\boldsymbol{\theta}|D^n)$ might converge to $p(\mathbf{x})$, $p(\mathbf{x}|\omega_i,D^n)$ will not in general converge to $p(\mathbf{x}|\omega_i)$. In such cases a few labeled training samples can have a big impact on your ability to decompose the mixture distribution into its components.

Example 2: Unsupervised learning of Gaussian data

- Consider the one-dimensional, two-component mixture with $p(x|\omega_1) \sim N(\mu, 1)$, $p(x|\omega_2,\theta) \sim N(\theta,1)$, where μ, $P(\omega_1)$ and $P(\omega_2)$ are known.

$$p(x|\theta) = \underbrace{\frac{P(\omega_1)}{\sqrt{2\pi}}\exp\left[-\frac{1}{2}(x-\mu)^2\right]}_{\omega_1} + \underbrace{\frac{P(\omega_2)}{\sqrt{2\pi}}\exp\left[-\frac{1}{2}(x-\theta)^2\right]}_{\omega_2},$$

we seek the mean of the second component.

- Suppose that the prior density $p(\theta)$ is uniform from a to b. Then after one observation ($x = x_1$) we have

$$\begin{aligned} p(\theta|x_1) &= \alpha p(x_1|\theta)p(\theta) \\ &= \left\{ \begin{array}{ll} \alpha'\{P(\omega_1)\exp[-\frac{1}{2}(x_1-\mu)^2]+ & \\ \qquad P(\omega_2)\exp[-\frac{1}{2}(x_1-\theta)^2]\} & a \le \theta \le b \\ 0 \qquad \text{otherwise} & \end{array} \right\} \end{aligned}$$

- If the sample x_1 is in the range $a \leq x \leq b$, then $p(\theta|x_1)$ peaks at $\theta = x_1$, of course. Otherwise it peaks either at $\theta = a$ if $x_1 < a$ or at $\theta = b$ if $x_1 > b$.
- With the addition of a second sample x_2, $p(\theta|x_1)$ changes to

$$\begin{aligned} p(\theta|x_1, x_2) &= \beta p(x_2|\theta) p(\theta|x_1) \\ &= \begin{cases} \beta' \{P(\omega_1)P(\omega_1)\exp\left[-\frac{1}{2}(x_1-\mu)^2 - \frac{1}{2}(x_2-\mu)^2\right] \\ \quad +[P(\omega_1)P(\omega_2)\exp\left[-\frac{1}{2}(x_1-\mu)^2 - \frac{1}{2}(x_2-\theta)^2\right] \\ \quad +[P(\omega_2)P(\omega_1)\exp\left[-\frac{1}{2}(x_1-\theta)^2 - \frac{1}{2}(x_2-\mu)^2\right] \\ \quad +[P(\omega_2)P(\omega_2)\exp\left[-\frac{1}{2}(x_1-\theta)^2 - \frac{1}{2}(x_2-\theta)^2\right]\} & a \leq \theta \leq b \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

- With n samples there will be 2^n terms, and no simple sufficient statistics can be found to facilitate understanding or to simplify computations.

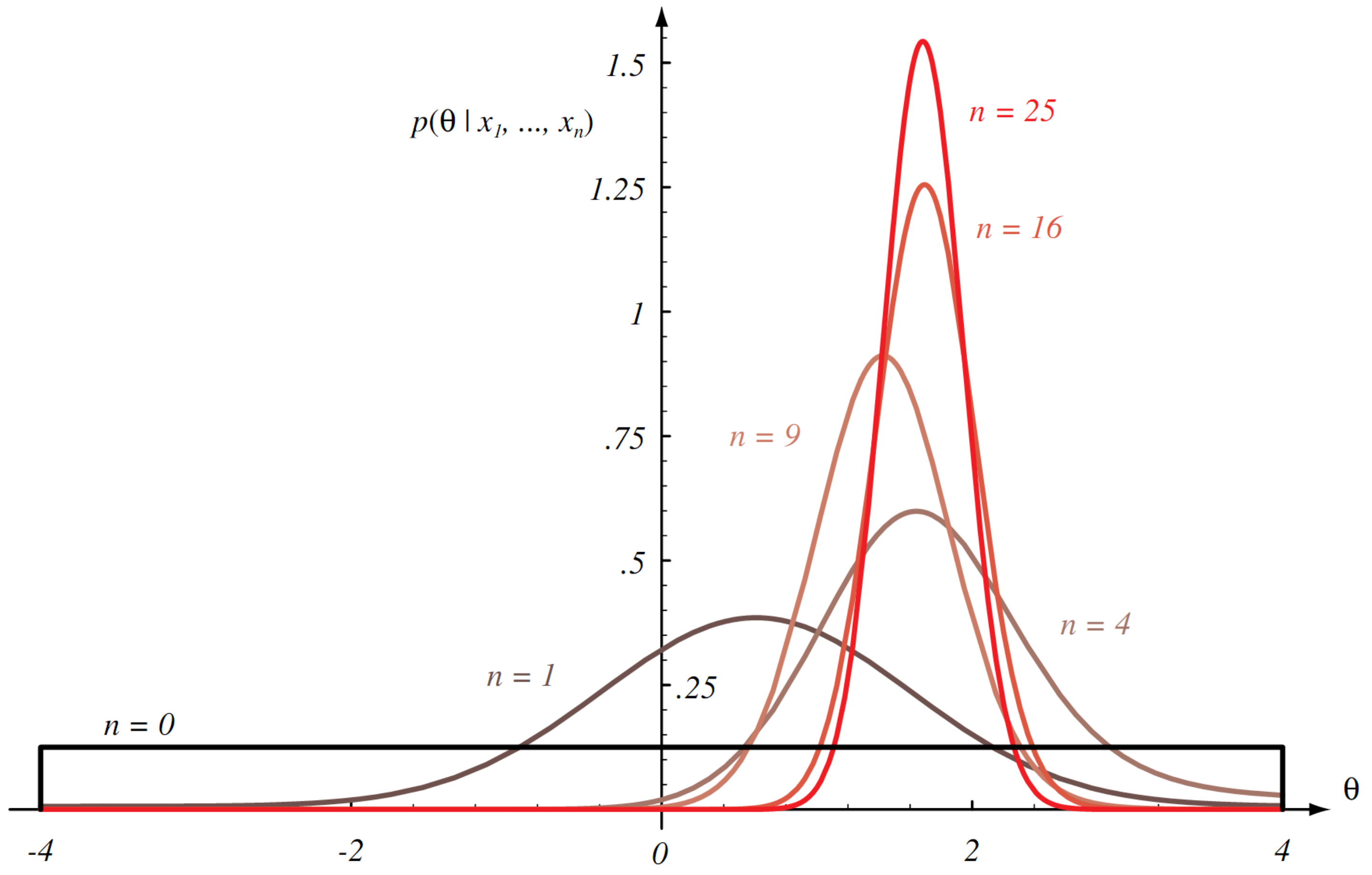
It is possible to use the relation

$$p(\theta|\mathcal{D}^n) = \frac{p(x_n|\theta)p(\theta|\mathcal{D}^{n-1})}{\int p(x_n|\theta)p(\theta|\mathcal{D}^{n-1})\ d\theta}$$

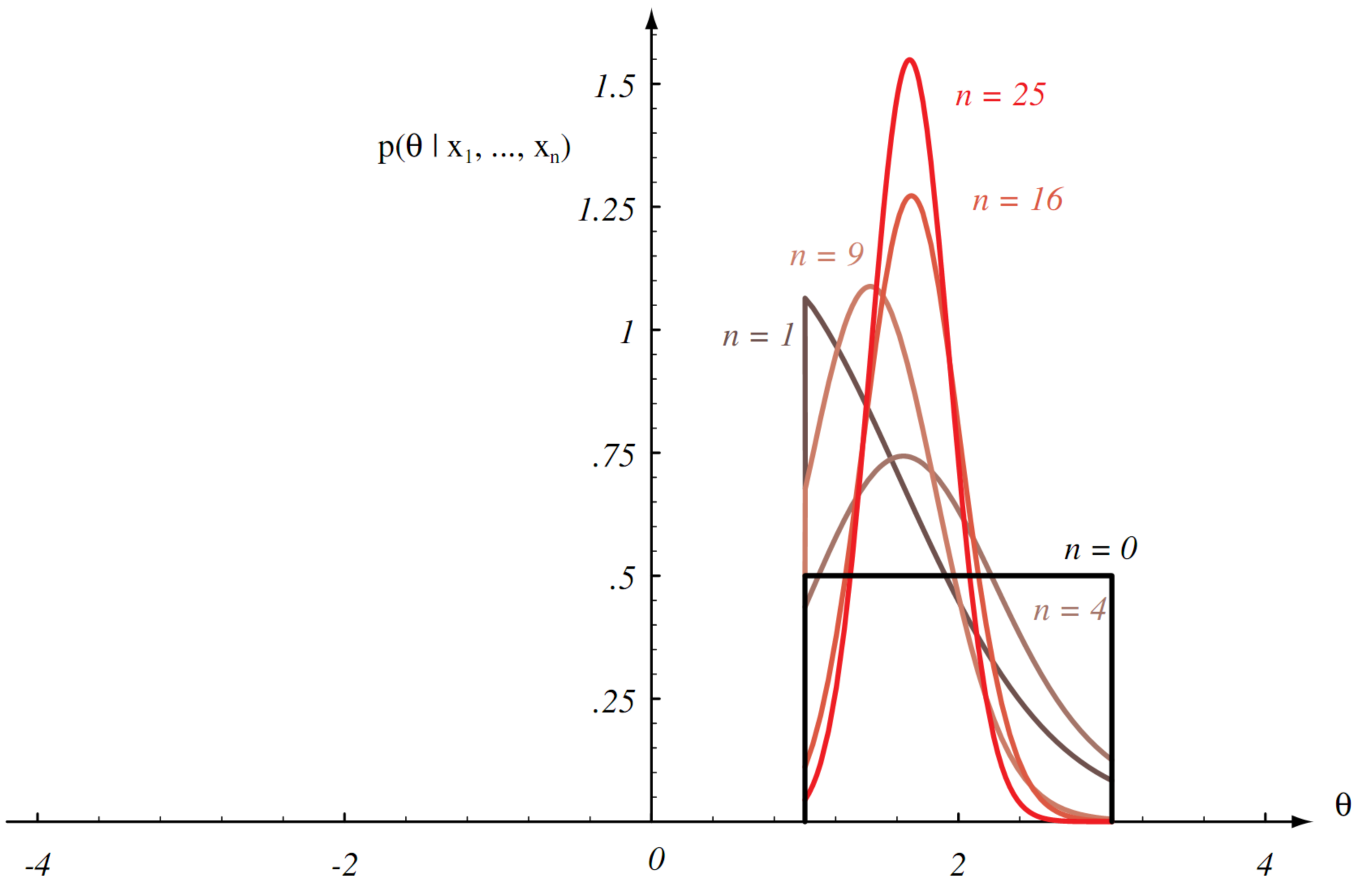
and numerical integration to obtain an approximate numerical solution for $p(\theta|D^n)$.

- Data in previous Example used with:
- $\mu = 2$, $P(\omega_1)=1/3$, $P(\omega_2)=2/3$
- prior density $p(\theta)$ uniform from −4 to +4

- As n goes to infinity we can confidently expect $p(\theta|D^n)$ to approach an impulse centered at $\theta = 2$. This graph gives some idea of the rate of convergence.



In unsupervised Bayesian learning of the parameter θ, the density becomes more peaked as the number of samples increases. A wide uniform prior $p(\theta)=1/8, -4 \le \theta \le 4$ has been used.



A narrower one, $p(\theta)=1/2$, $1 \leq \theta \leq 3$ has been used.
After all 25 samples have been used, the posterior densities are virtually identical in the two cases — the information in the samples overwhelms the prior information.

* Decision-Directed Approximation

- Because the difference between supervised and unsupervised learning is the presence of labels, it is natural to propose the following:
  - Use prior information to train a classifier.
  - Label new data with this classifier.
  - Use the new labeled samples to train a new (supervised) classifier.
- This approach is known as the decision-directed approach to unsupervised learning.
- Obvious dangers include:
  - If the initial classifier is not reasonably good, the process can diverge.
  - The tails of the distribution tend not to be modeled well this way, which results in significant overlap between the component densities.
- In practice, this approach works well because it is easy to leverage previous work for the initial classifier.
- Also, it is less computationally expensive than the pure Bayesian unsupervised learning approach.

Data Clustering

- 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 is a misleading representation of the data

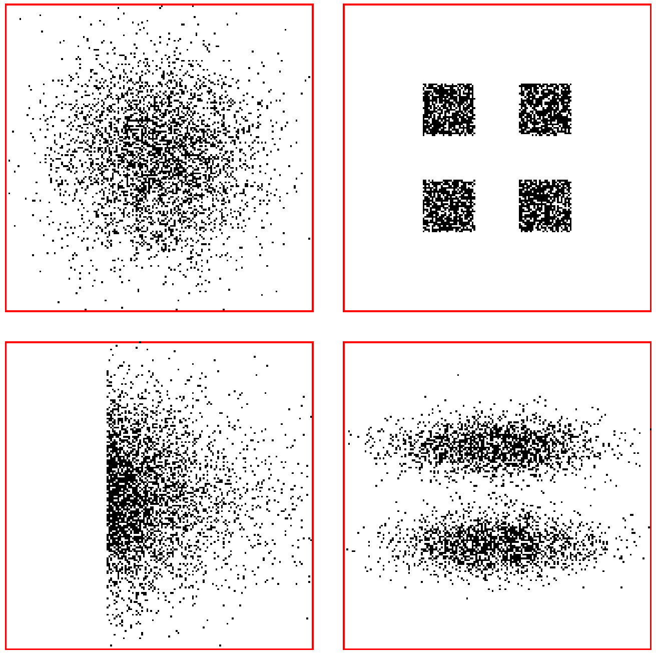


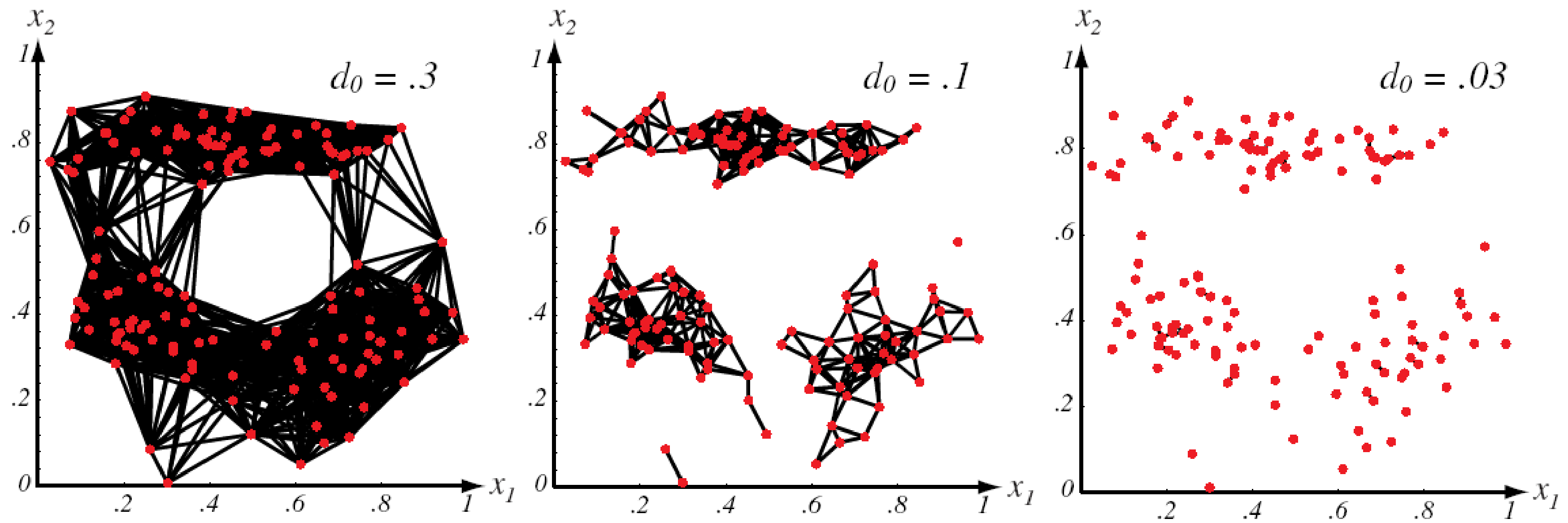
Figure 10.5: These four data sets have identical statistics up to second-order, i.e., the same mean **μ** and covariance **Σ**. In such cases it is important to include in the model more parameters to represent the structure more completely.

- Mixture of normal distributions can approximate a large variety of situations (i.e., any density functions).
- 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 of 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

- 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*.
- Once defined this measure, one would expect the distance between samples of the same cluster to be significantly less than the distance between samples in different classes.

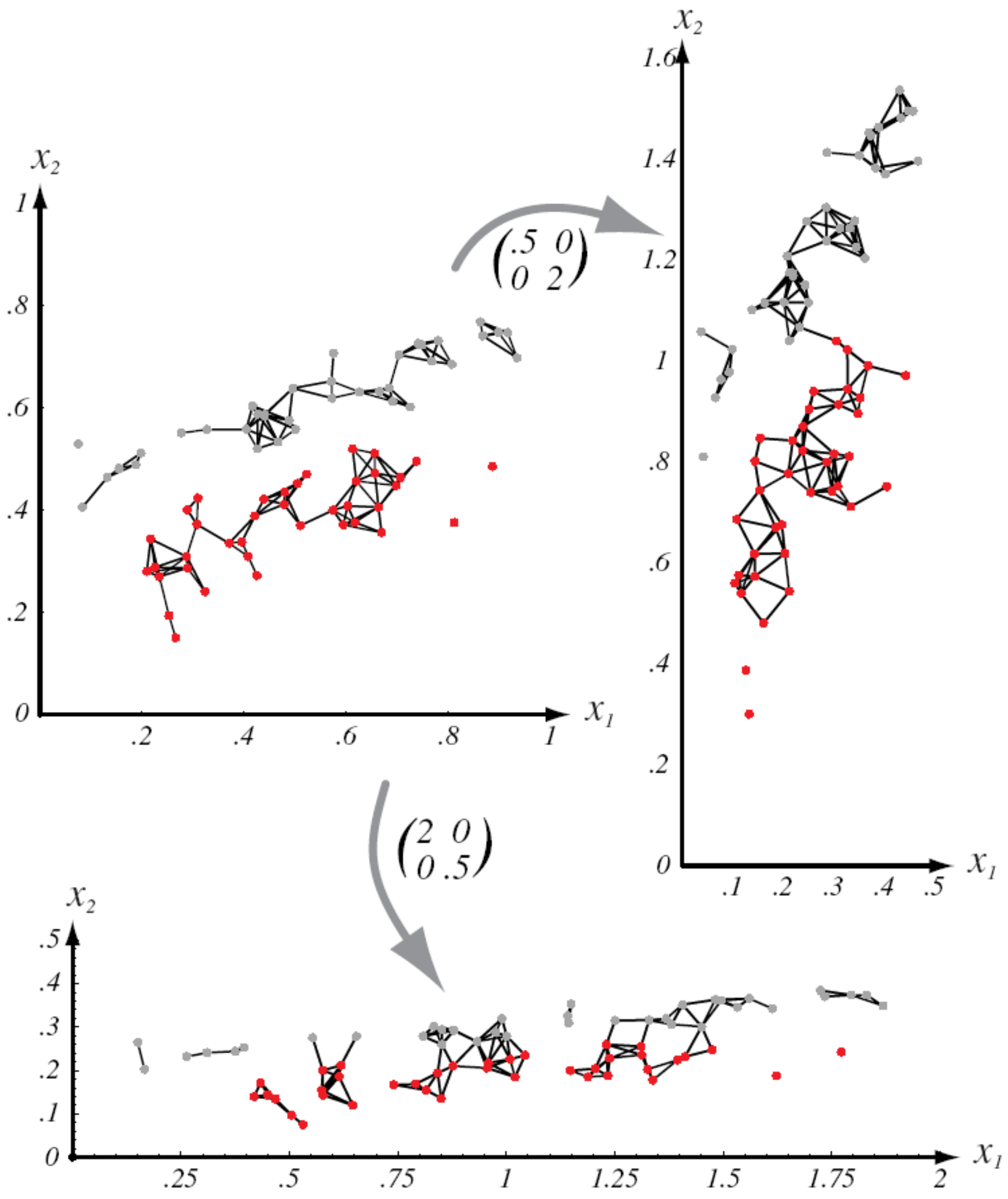
- Euclidean distance is a possible metric: a possible criterion is to assume samples belonging to same cluster if their distance is less than a threshold d_0



- 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

FIGURE 10.8.

Scaling axes affects the clusters in a minimum distance cluster method. In both these scaled cases, the assignment of points to clusters differ from that in the original space.



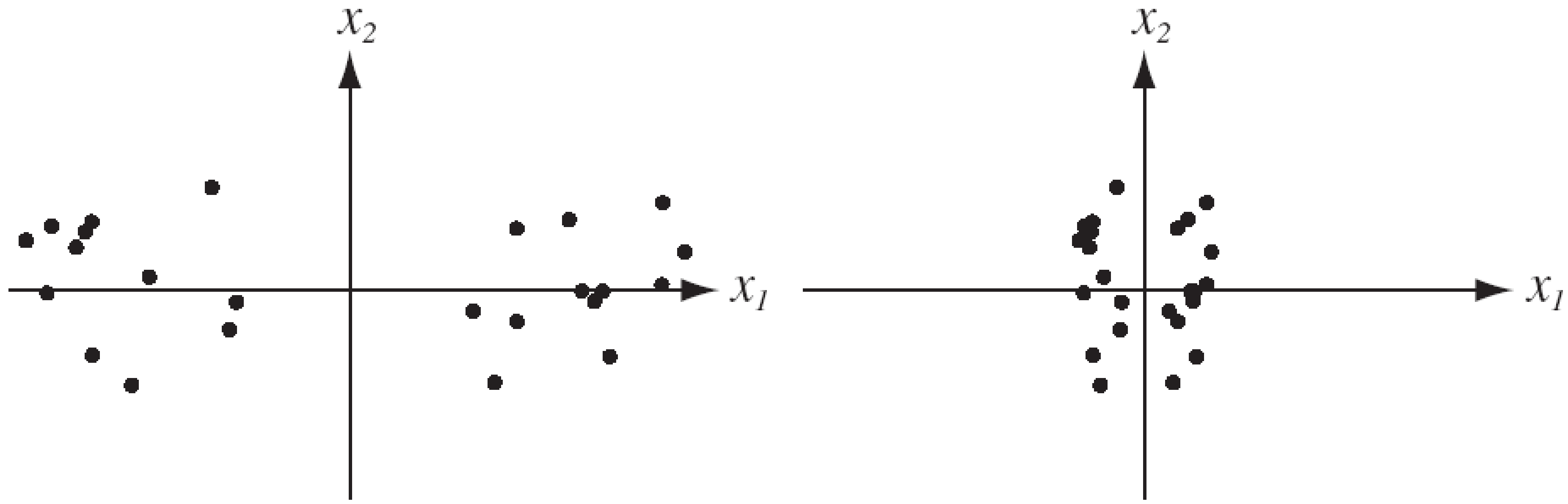


FIGURE 10.9. If the data fall into well-separated clusters (left), normalization by scaling for unit variance for the full data may reduce the separation, and hence be undesirable (right). Such a normalization may in fact be appropriate if the full data set arises from a single fundamental process (with noise), but inappropriate if there are several different processes, as shown here.

- To achieve invariance, one can normalize the data, e.g., such that they all have zero means and unit variance, or use principal components for invariance to rotation
- 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 $s(\mathbf{x},\mathbf{x}')$ to compare 2 vectors.

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

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

If the angle is meaningful

- In case of binary-valued features, we have, e.g.:

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

the fraction of attributes shared

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

the ratio of the number of shared attributes to the number possessed by **x** *or* **x**´.

Tanimoto coefficient or distance used in Information Retrieval and Taxonomy

Ch10-part3: Clustering as optimization

- The first issue: how to measure "similarity"?
- The second issue: how to evaluate a partitioning of a set into clusters?
- Clustering can be posted 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 $$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{x}$$

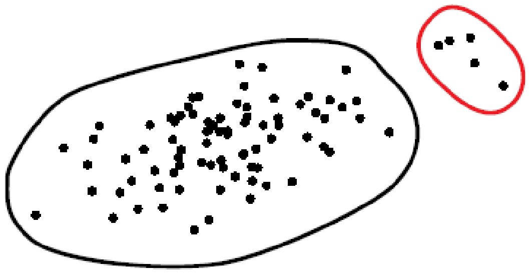
– The sum of squared error is defined as

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

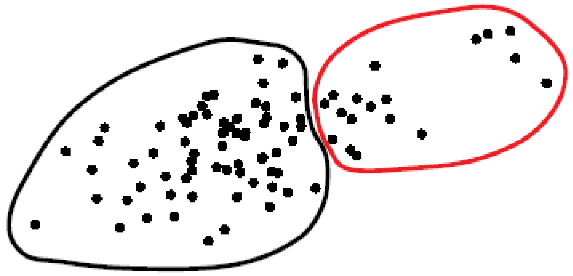
– This criterion defines clusters as their mean vectors $\mathbf{m}_i$ in the sense that it minimizes the sum of the squared lengths of the error $\mathbf{x}$ - $\mathbf{m}_i$.

– The optimal partition is defined as one that minimizes J_e, also called *minimum variance* partition.

– Work fine when clusters form well separated compact clouds, less when there are great differences in the number of samples in different clusters.



$J_e = large$



$J_e = small$

Related Minimum Variance Criteria

By some simple algebraic manipulation we can eliminate the mean vectors from the expression for J_e and obtain the equivalent expression

$$J_e = \frac{1}{2}\sum_{i=1}^{c} n_i \bar{s}_i$$

where

$$\bar{s}_i = \frac{1}{n_i^2}\sum_{\mathbf{x}\in D_i}\sum_{\mathbf{x}'\in D_i}\|\mathbf{x}-\mathbf{x}'\|^2$$

$\bar{s}_i$ is the average squared distance between points in the ith cluster, and emphasizes the fact that the sum-of-squared-error criterion uses Euclidean distance as the measure of similarity. More generally

$$\bar{s}_i = \frac{1}{n_i^2}\sum_{\mathbf{x}\in D_i}\sum_{\mathbf{x}'\in D_i} s(\mathbf{x},\mathbf{x}')$$

or

$$\bar{s}_i = \min_{\mathbf{x},\mathbf{x}'\in \mathcal{D}_i} s(\mathbf{x},\mathbf{x}').$$

Scatter Criteria

Table 10.1: Mean vectors and scatter matrices used in clustering criteria.

	Depend on cluster center?		
	Yes	No	
Mean vector for the ith cluster		$\times$	$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}$ (54)
Total mean vector		$\times$	$\mathbf{m} = \frac{1}{n} \sum_{\mathcal{D}} \mathbf{x} = \frac{1}{n} \sum_{i=1}^{c} n_i \mathbf{m}_i$ (55)
Scatter matrix for the ith cluster	$\times$		$\mathbf{S}_i = \sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t$ (56)
Within-cluster scatter matrix	$\times$		$\mathbf{S}_W = \sum_{i=1}^{c} \mathbf{S}_i$ (57)
Between-cluster scatter matrix	$\times$		$\mathbf{S}_B = \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t$ (58)
Total scatter matrix		$\times$	$\mathbf{S}_T = \sum_{\mathbf{x} \in \mathcal{D}} (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^t$ (59)

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

$$\mathbf{S}_T = \mathbf{S}_B + \mathbf{S}_W$$

  that depends only from the set of samples (not on the partitioning)
- The criteria can be defined 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 it is in practice the sum-of-squared-error criterion.

– As $tr[\mathbf{S}_\mathrm{T}] = tr[\mathbf{S}_\mathrm{W}] + tr[\mathbf{S}_\mathrm{B}]$ and $tr[\mathbf{S}_\mathrm{T}]$ is independent from the partitioning, no new results can be derived by maximizing $tr[\mathbf{S}_\mathrm{B}]$.

– However, seeking to minimize the within-cluster criterion $J_\mathrm{e}=tr[\mathbf{S}_\mathrm{W}]$, is equivalent to maximize 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:

The Determinant Criterion

- Since S_B will be singular if the number of clusters is less than or equal to the dimensionality, $|S_B|$ is obviously a poor choice for a criterion function. Furthermore, S_W may become singular, and will certainly be so if $n - c$ is less than the dimensionality d. However, if we assume that S_W is nonsingular, we are led to consider the determinant criterion function

$$J_d = |\mathbf{S}_W| = \left| \sum_{i=1}^{c} \mathbf{S}_i \right|.$$

- We observed before that the minimum-squared-error partition might change if the axes are scaled, though this does not happen with J_d. Thus J_d is to be favored under conditions where there may be unknown or irrelevant linear transformations of the data.

Invariant Criteria

- The eigenvalues $\lambda_1, \ldots, \lambda_d$ of $\mathbf{S}_W^{-1}\mathbf{S}_B$ are invariant under nonsingular linear transformations of the data
- Since the trace of a matrix is the sum of its eigenvalues, one might select to maximize the criterion function

$$tr\mathbf{S}_W^{-1}\mathbf{S}_B = \sum_{i=1}^{d} \lambda_i.$$

- By using the relation $S_T = S_W + S_B$

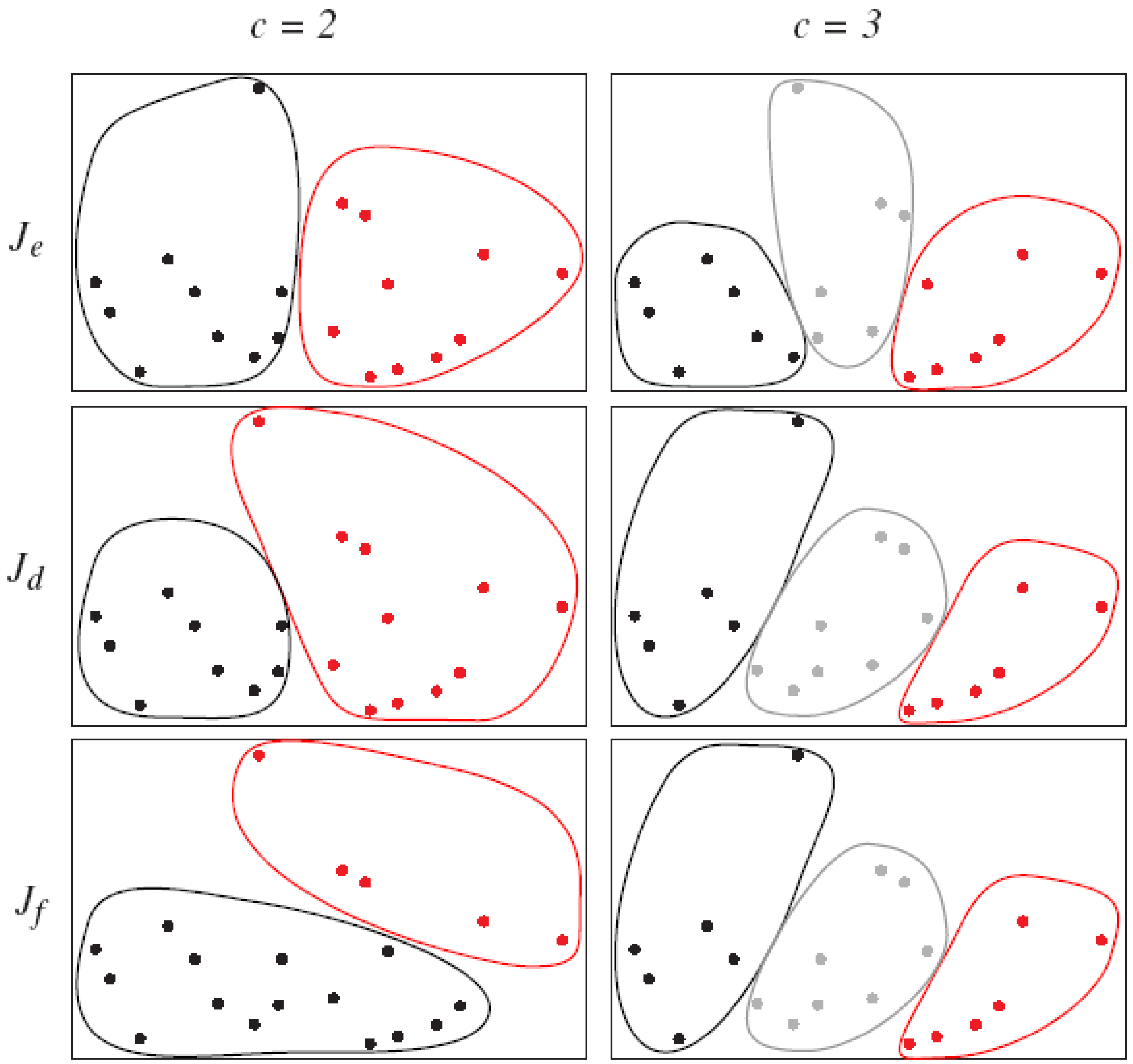
$$J_f = tr\mathbf{S}_T^{-1}\mathbf{S}_W = \sum_{i=1}^{d} \frac{1}{1+\lambda_i}$$

and

$$\frac{|\mathbf{S}_W|}{|\mathbf{S}_T|} = \prod_{i=1}^{d} \frac{1}{1+\lambda_i}.$$

20 points

The clusters found by minimizing a criterion depends upon the criterion function as well as the assumed number of clusters.



Iterative optimization

- 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 ($c^n/n!$ Ways of partitioning).
- 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, this kinds of approaches guarantee local, not global, optimization.

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

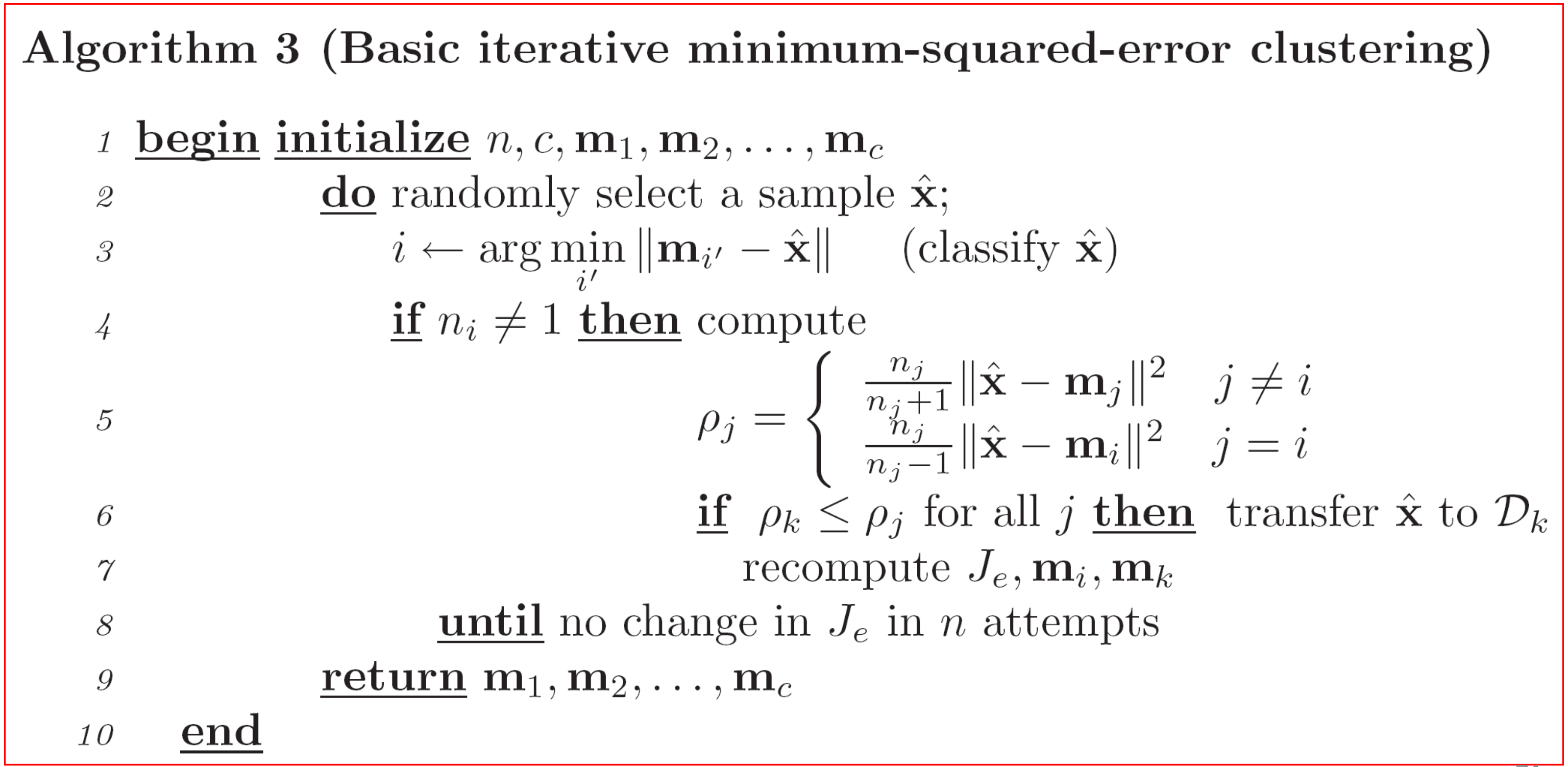
$$J_e = \sum_{i=1}^{c} J_i \quad \text{where} \quad J_i = \sum_{\mathbf{x} \in D_i} \|\mathbf{x} - \mathbf{m}_i\|^2$$

  where J_i is the effective error per cluster.
- It can be proved 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

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

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

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



Algorithm 3 (Basic iterative minimum-squared-error clustering)

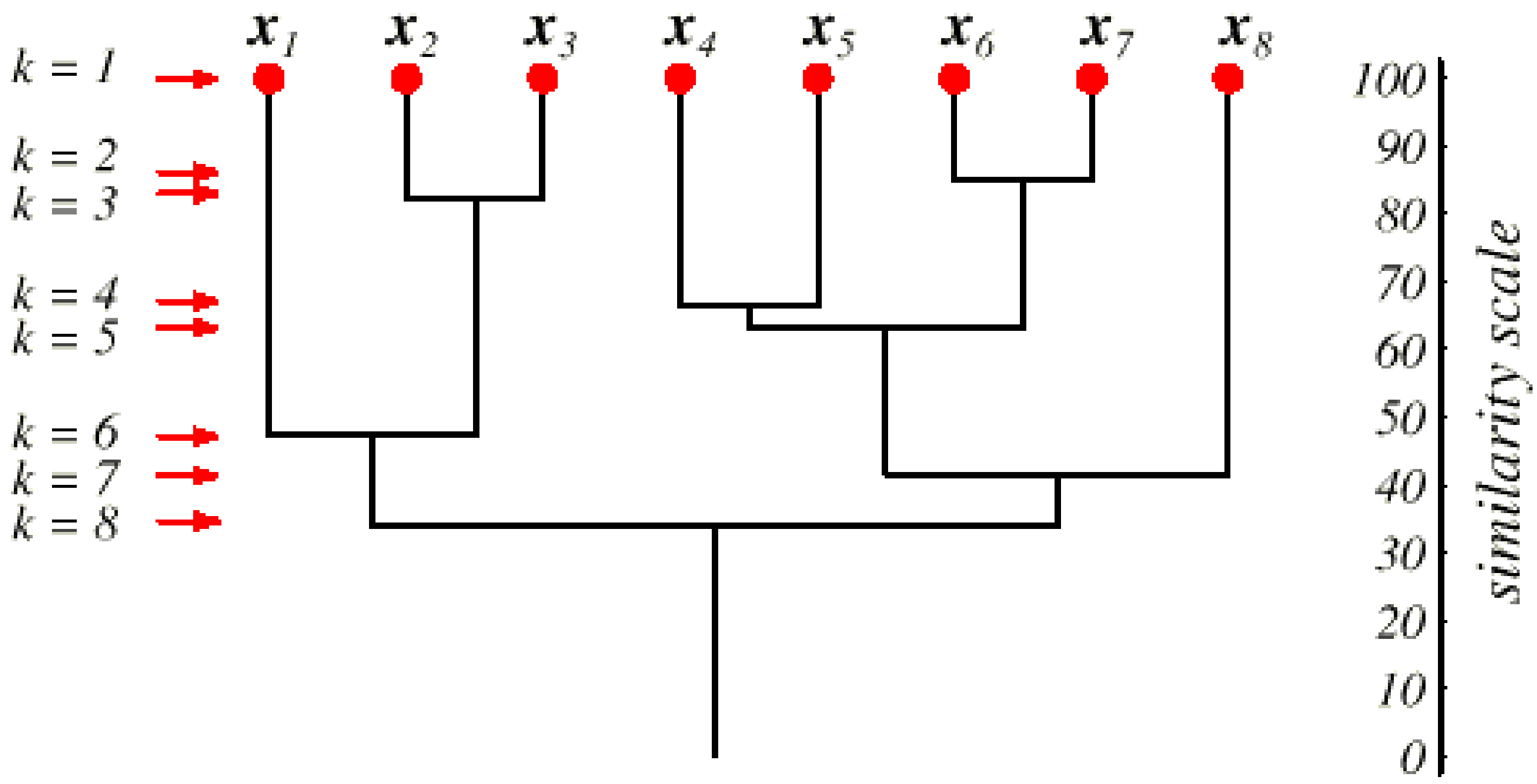
```
begin initialize $n, c, \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$
    do randomly select a sample $\hat{\mathbf{x}}$;
        $i \leftarrow \arg\min_{i'} \|\mathbf{m}_{i'} - \hat{\mathbf{x}}\|$    (classify $\hat{\mathbf{x}}$)
        if $n_i \neq 1$ then compute
            $\rho_j = \begin{cases}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

- This procedure is a sequential version of the k-*means* algorithm, with the difference that k-means waits until 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 $(c\text{-}1)$-cluster problem plus the sample farthest from the nearer cluster center

Hierarchical Clustering

- 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, $c = n\text{-}k+1$.

- Given any two samples $\mathbf{x}$ and $\mathbf{x}'$, 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 $\Rightarrow$ tree representation called *dendrogram*



- 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

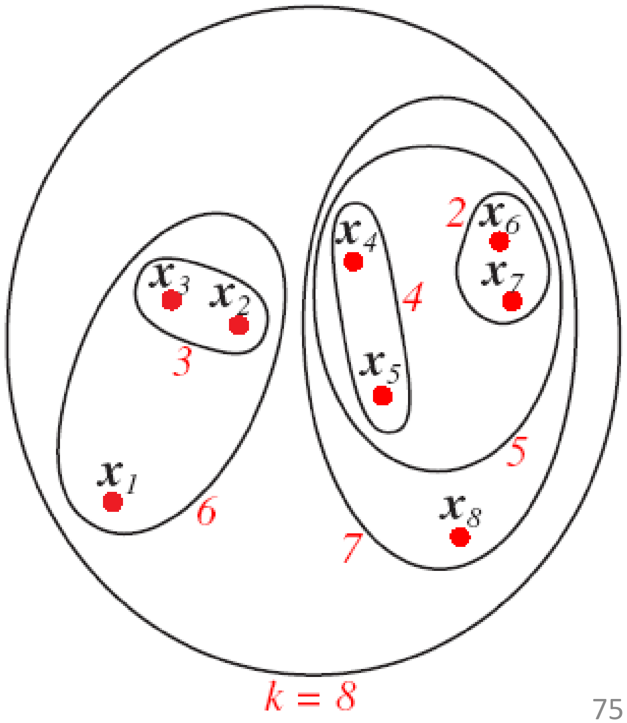


FIGURE 10.12. A set or Venn diagram representation of two-dimensional data (which was used in the dendrogram of Fig. 10.11) reveals the hierarchical structure but not the quantitative distances between clusters. The levels are numbered by k, in red.

- Hierarchical clustering can be divided in *agglomerative* (accumulative, collective) and *divisive* (creating discord or disunity).

- 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

Algorithm 4 (Agglomerative hierarchical clustering)

```
begin initialize $c, \hat{c} \leftarrow n, \mathcal{D}_i \leftarrow \{\mathbf{x}_i\}, i = 1, \ldots, n$
        do $\hat{c} \leftarrow \hat{c} - 1$
            Find nearest clusters, say, $\mathcal{D}_i$ and $\mathcal{D}_j$
            Merge $\mathcal{D}_i$ and $\mathcal{D}_j$
        until $\boxed{\hat{c} = c}$
    return $c$ clusters
end
```

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

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

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

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

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

$$d_{mean}(D_i, D_j) = \|\mathbf{m}_i - \mathbf{m}_j\|$$

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

Nearest-neighbor algorithm

- When d_{min} is used, the algorithm is called the *nearest neighbor* 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 never has any closed loop and it is a *tree*, if all subsets are linked we have a *spanning tree* (spanning tree = a tree with a path from node to any other node)

- The use of d_{min} as a distance measure and the agglomerative clustering generate a *minimal spanning tree*

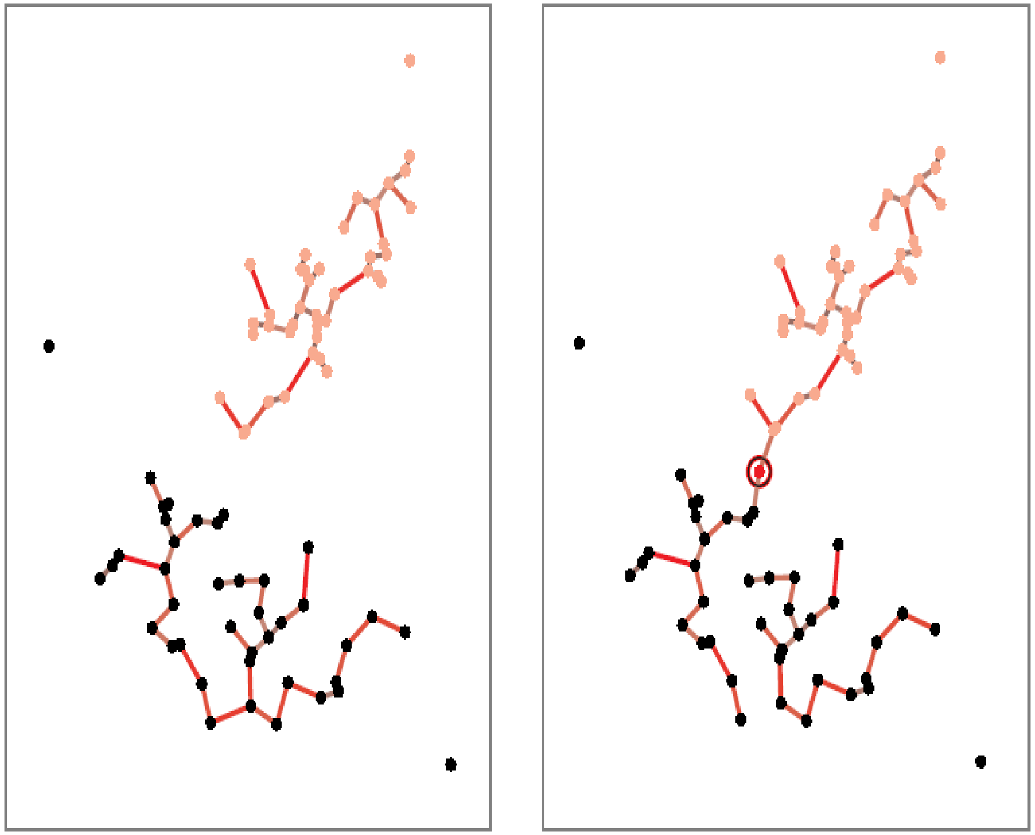


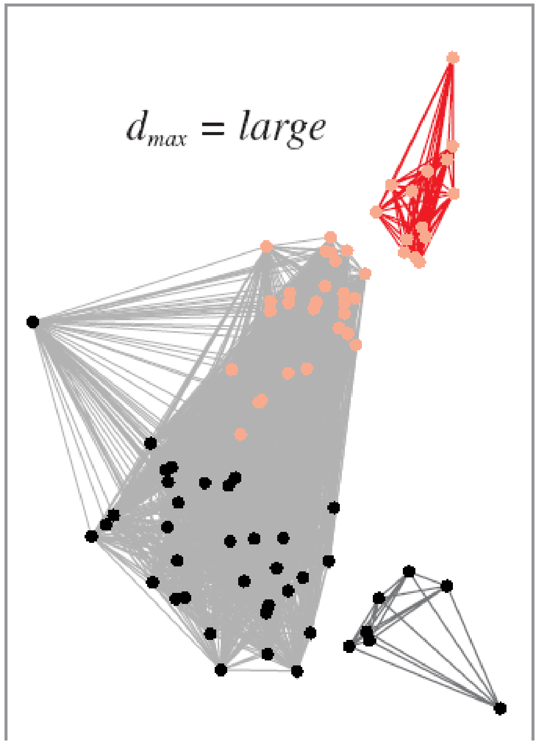
FIGURE 10.13. Two Gaussians were used to generate two-dimensional samples, shown in pink and black. The nearest-neighbor clustering algorithm gives two clusters that well approximate the generating Gaussians (left). If, however, another particular sample is generated (circled red point at the right) 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.

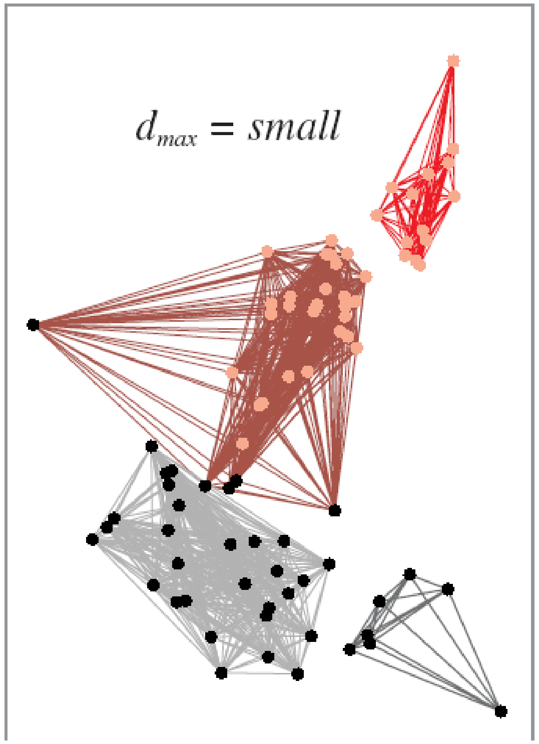
- *Chaining* effect: defect of this distance measure (right)

The farthest neighbor algorithm

- When d_{max} is used, the algorithm is called the *farthest neighbor* 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

- 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

The problem of the number of clusters

- 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.
- Another approach is to state a threshold for the creation of a new cluster; this adapts to on line 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

- The 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 matrix* $\mathbf{S} = [s_{ij}]$

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

- 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-neighbor* 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.

- 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

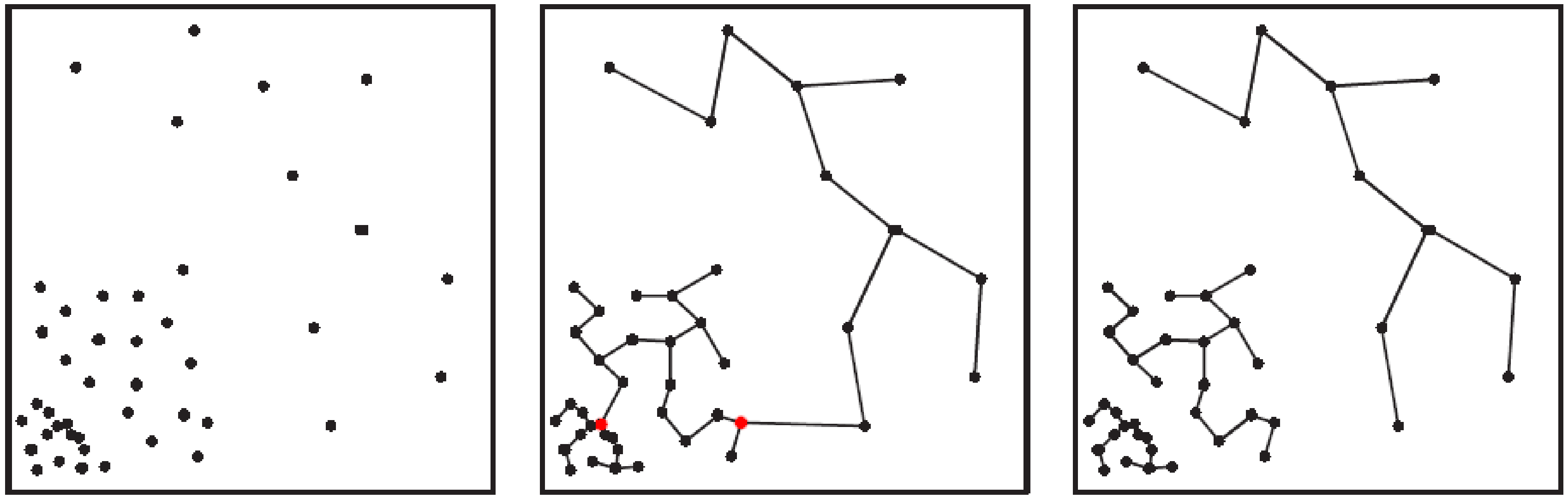


Figure 10.20: The removal of inconsistent edges—ones with length significantly larger than the average incident upon a node — may yield natural clusters. The original data is shown at the left and its minimal spanning tree is shown in the middle. At virtually every node, incident edges are of nearly 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.

- 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:

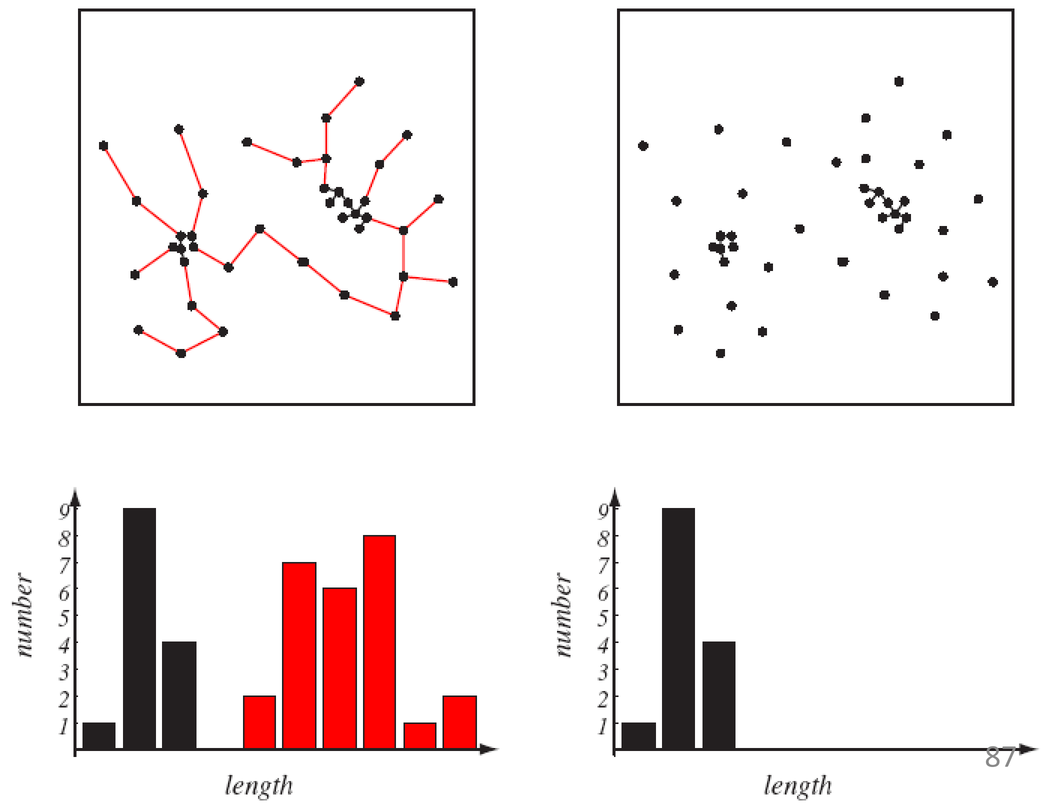


Figure 10.21: A minimal spanning tree is shown at the left; its bimodal edge length distribution is evident in the histogram below. If all links of intermediate or high length are removed (red), the two natural clusters are revealed (right).

Unsupervised Learning And Clustering

- Competitive Learning
  - Unknown number of clusters
  - Adaptive Resonance
- Component analysis
  - Principal component analysis (PCA)
  - Non-linear component analysis
  - Independent component analysis (ICA)
- Low-Dimensional Representations and Multi-dimensional Scaling (MDS)
  - Self-organizing feature maps
  - Clustering and Dimensionality Reduction