CHAPTER

CLUSTERING 37

CHAPTER CONTENTS

In this chapter, the problem of unsupervised classification of input-only samples $\{\boldsymbol{x}_i\}_{i=1}^n$ called *clustering* is discussed, which is aimed at grouping data samples based on their similarity.

37.1 *k*-MEANS CLUSTERING

The method of *k-means clustering* is one of the most fundamental algorithms for clustering, which aims at finding cluster labels,

$$\{y_i \mid y_i \in \{1,\ldots,c\}\}_{i=1}^n,$$

so that the sum of *within-cluster scatters* is minimized:

$$\min_{y_1,\ldots,y_n \in \{1,\ldots,c\}} \sum_{y=1}^{c} \sum_{i:y_i=y} \|\boldsymbol{x}_i - \boldsymbol{\mu}_y\|^2,$$

where $\sum_{i:y_i=y}$ denotes the sum over i such that $y_i = y$,

$$\boldsymbol{\mu}_y = \frac{1}{n_y} \sum_{i:y_i=y} \boldsymbol{x}_i$$

denotes the center of center y, and n_y denotes the number of samples in cluster y.

However, solving this optimization problem will take exponential computation time with respect to the number of samples n [3], and thus is computationally infeasible if n is large. In practice, a local optimal solution is found by iteratively assigning a single sample to the cluster with the nearest center in a greedy manner:

$$y_i \longleftarrow \operatorname*{argmin}_{y \in \{1,\ldots,c\}} \|\boldsymbol{x}_i - \boldsymbol{\mu}_y\|^2, \quad i = 1,\ldots,n. \tag{37.1}$$

The algorithm of *k*-means clustering is summarized in Fig. 37.1.

An Introduction to Statistical Machine Learning. DOI: 10.1016/B978-0-12-802121-7.00048-0

1. Initialize cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c$.
2. Update cluster labels $y_1, \ldots, y_n$ for samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$:

$$y_i \longleftarrow \underset{y \in \{1,\ldots,c\}}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{\mu}_y\|^2, \ \ i = 1, \ldots, n.$$

3. Update cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c$:

$$\boldsymbol{\mu}_y \longleftarrow \frac{1}{n_y} \sum_{i:y_i=y} \boldsymbol{x}_i, \ \ y = 1, \ldots, c,$$

   where n_y denotes the number of samples in cluster y.
4. Iterate 2 and 3 until convergence.

FIGURE 37.1

k-means clustering algorithm.

A MATLAB code for k-means clustering is provided in Fig. 37.2, and its behavior is illustrated in Fig. 37.3. This shows that the k-means clustering algorithm gives a reasonable solution for this data set.

37.2 KERNEL k-MEANS CLUSTERING

Since k-means clustering uses the Euclidean distance $\|\boldsymbol{x} - \boldsymbol{\mu}_y\|$ for determining cluster assignment, it can only produce *linearly separable* clusters, as illustrated in Fig. 37.3. Here, let us apply the *kernel trick* introduced in Section 27.4 to obtain a nonlinear version of k-means clustering, called *kernel k-means clustering* [48].

More specifically, let us express the squared Euclidean distance $\|\boldsymbol{x} - \boldsymbol{\mu}_y\|^2$ in Eq. (37.1) by using the inner product $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle$ as

$$\begin{aligned}\|\boldsymbol{x} - \boldsymbol{\mu}_y\|^2 &= \left\| \boldsymbol{x} - \frac{1}{n_y} \sum_{i:y_i=y} \boldsymbol{x}_i \right\|^2 \\ &= \langle \boldsymbol{x}, \boldsymbol{x} \rangle - \frac{2}{n_y} \sum_{i:y_i=y} \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle + \frac{1}{n_y^2} \sum_{i,i':y_i=y_{i'}=y} \langle \boldsymbol{x}_i, \boldsymbol{x}_{i'} \rangle.\end{aligned}$$

Then, replacing the inner product with the kernel function $K(\boldsymbol{x}, \boldsymbol{x}')$ immediately gives the kernel k-means clustering algorithm:

$$y \longleftarrow \underset{y \in \{1,\ldots,c\}}{\operatorname{argmin}} \left[-\frac{2}{n_y} \sum_{i:y_i=y} K(\boldsymbol{x}, \boldsymbol{x}_i) + \frac{1}{n_y^2} \sum_{i,i':y_i=y_{i'}=y} K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) \right],$$

where irrelevant constant $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = K(\boldsymbol{x}, \boldsymbol{x})$ is ignored.

```
n=300; c=3; t=randperm(n);
x=[randn(1,n/3)-2 randn(1,n/3) randn(1,n/3)+2;
   randn(1,n/3) randn(1,n/3)+4 randn(1,n/3)]';
m=x(t(1:c),:); x2=sum(x.^2,2); s0(1:c,1)=inf;

for o=1:1000
  m2=sum(m.^2,2);
  [d,y]=min(repmat(m2,1,n)+repmat(x2',c,1)-2*m*x');
  for t=1:c
    m(t,:)=mean(x(y==t,:)); s(t,1)=mean(d(y==t));
  end
  if norm(s-s0)<0.001, break, end
  s0=s;
end

figure(1); clf; hold on;
plot(x(y==1,1),x(y==1,2),'bo');
plot(x(y==2,1),x(y==2,2),'rx');
plot(x(y==3,1),x(y==3,2),'gv');
```

FIGURE 37.2

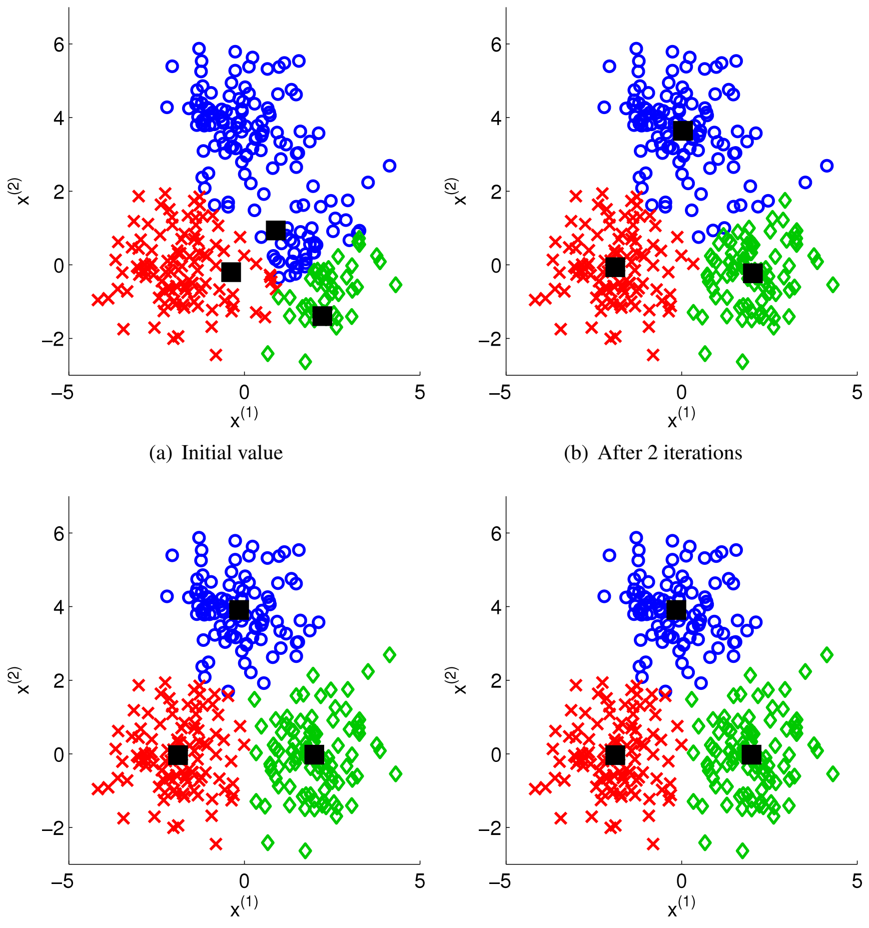
MATLAB code for k-means clustering.

Since ordinary k-means clustering in the kernel feature space corresponds to nonlinear clustering in the original input space, it can give clustering solutions with nonlinear boundaries. However, due to high nonlinearity brought by the kernel function, optimization of kernel k-means clustering is much harder then that of ordinary k-means clustering and the solution depends even strongly on initialization. For this reason, the use of kernel k-means clustering is not straightforward in practice.

37.3 SPECTRAL CLUSTERING

Strong dependency of kernel k-means clustering on the initial solution is more prominent if the dimensionality of the feature space is higher. *Spectral clustering* addresses this issue by combining clustering with *dimensionality reduction*.

More specifically, after transforming samples to a feature space by a kernel function, spectral clustering applies *locality preserving projection* introduced in Section 35.2.2 to reduce the dimensionality, which has the property that cluster structure of data tends to be preserved. Note that locality preserving projection in the feature space is equivalent to the *Laplacian eigenmap* introduced in Section 36.1.2.



(a) Initial value (b) After 2 iterations

(c) After 4 iterations (d) Final solution (after 6 iterations)

FIGURE 37.3

Example of k-means clustering. A filled square denotes a cluster center.

Then, after dimensionality reduction, spectral clustering employs *ordinary* (not kernel) k-means clustering to obtain cluster labels. The algorithm of spectral clustering is summarized in Fig. 37.4.

A MATLAB code for spectral clustering is provided in Fig. 37.5, and its behavior is illustrated in Fig. 37.6. The original two-dimensional samples $\{\boldsymbol{x}_i\}_{i=1}^n$ in Fig. 37.6(a) are projected onto a one-dimensional subspace in the feature space, which gives embedded samples $\{z_i\}_{i=1}^n$ degenerated into two points as illustrated in Fig. 37.6(b). Then applying ordinary k-means clustering to the degenerated samples

1. Apply Laplacian eigenmap to samples $\{\boldsymbol{x}_i\}_{i=1}^n$ and obtain $(c-1)$-dimensional expressions $\{z_i\}_{i=1}^n$, where c is the number of clusters.
2. Apply(non-kernelized) k-means clustering to the embedded samples $\{z_i\}_{i=1}^n$ and obtain cluster labels $\{y_i\}_{i=1}^n$.

FIGURE 37.4

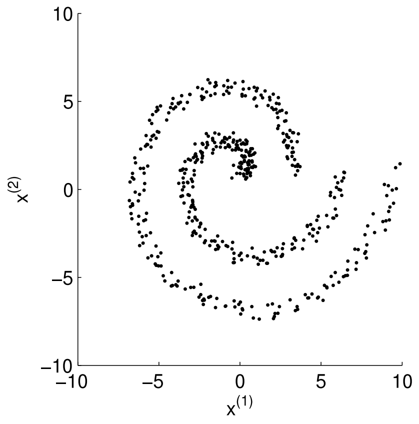
Algorithm of spectral clustering.

```
n=500; c=2; k=10; t=randperm(n); a=linspace(0,2*pi,n/2)';
x=[a.*cos(a) a.*sin(a); (a+pi).*cos(a) (a+pi).*sin(a)];
x=x+rand(n,2); x=x-repmat(mean(x),[n,1]); x2=sum(x.^2,2);
d=repmat(x2,1,n)+repmat(x2',n,1)-2*x*x'; [p,i]=sort(d);
W=sparse(d<=ones(n,1)*p(k+1,:)); W=(W+W'~=0);
D=diag(sum(W,2)); L=D-W; [z,v]=eigs(L,D,c-1,'sm');

m=z(t(1:c),:); s0(1:c,1)=inf; z2=sum(z.^2,2);
for o=1:1000
  m2=sum(m.^2,2);
  [u,y]=min(repmat(m2,1,n)+repmat(z2',c,1)-2*m*z');
  for t=1:c
    m(t,:)=mean(z(y==t,:)); s(t,1)=mean(d(y==t));
  end
  if norm(s-s0)<0.001, break, end
  s0=s;
end

figure(1); clf; hold on; axis([-10 10 -10 10]);
plot(x(y==1,1),x(y==1,2),'bo');
plot(x(y==2,1),x(y==2,2),'rx');
```

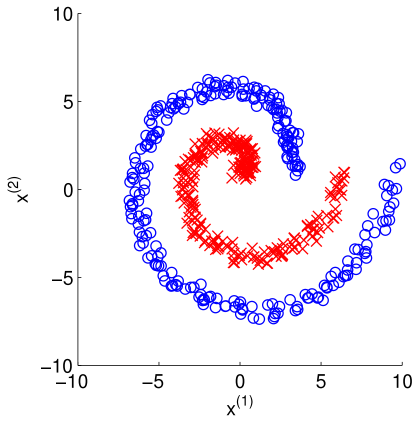
FIGURE 37.5

MATLAB code for spectral clustering.

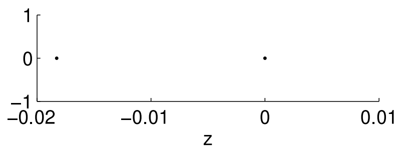
$\{z_i\}_{i=1}^n$ gives cluster labels $\{y_i\}_{i=1}^n$ illustrated in Fig. 37.6(c). This clustering solution actually corresponds to clustering two spirals in the original input space, as illustrated in Fig. 37.6(d).



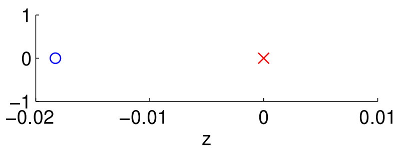
(a) Original two-dimensional samples $\{\boldsymbol{x}_i\}_{i=1}^n$



(d) Clustering results $\{y_i\}_{i=1}^n$ in the original input space



(b) One-dimensional projection $\{z_i\}_{i=1}^n$ by Laplacian eigenmap



(c) Clustering results $\{y_i\}_{i=1}^n$ obtained by ordinary k-means clustering in the feature

FIGURE 37.6

Example of spectral clustering.

37.4 TUNING PARAMETER SELECTION

Clustering solutions obtained by kernel k-means and spectral clustering depend on the choice of kernels such as the Gaussian bandwidth. In this section, the problem of tuning parameter selection is addressed.

Clustering can be regarded as *compressing* d-dimensional samples $\{\boldsymbol{x}_i\}_{i=1}^n$ into c-valued labels $\{y_i\}_{i=1}^n$ (Fig. 37.7). Based on this data compression view, let us choose tuning parameters so that the amount of *information* cluster labels $\{y_i\}_{i=1}^n$ contain on input samples $\{\boldsymbol{x}_i\}_{i=1}^n$ is maximized.

MI [92] allows us to measure the amount of information cluster labels $\{y_i\}_{i=1}^n$ contain on input samples $\{\boldsymbol{x}_i\}_{i=1}^n$:

$$\text{MI} = \int \sum_{y=1}^{c} p(\boldsymbol{x}, y) \log \frac{p(\boldsymbol{x}, y)}{p(\boldsymbol{x})p(y)} \mathrm{d}\boldsymbol{x}.$$

MI is actually the *KL divergence* (see Section 14.2) from the joint probability $p(\boldsymbol{x}, y)$ to the product of marginals $p(\boldsymbol{x})p(y)$. Therefore, it is always non-negative and is

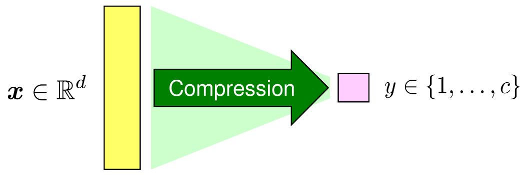


FIGURE 37.7

Clustering can be regarded as compressing d-dimensional vector $\boldsymbol{x}$ into c-valued scalar y.

zero if and only if $p(\boldsymbol{x}, y) = p(\boldsymbol{x})p(y)$, i.e. $\boldsymbol{x}$ and y are *statistically independent* (see Section 5.6). MI allows us to measure the dependency of $\boldsymbol{x}$ and y, i.e. the amount of information y contains on $\boldsymbol{x}$.

The value of MI can be approximated by the *KL density ratio estimator* described in Section 38.3. However, because the log function and the ratio of probability densities are included in MI, it tends to be sensitive to outliers. Here, let us use a variant of MI based on the *L_2-distance* called the QMI [113]:

$$\text{QMI} = \int \sum_{y=1}^{c} \Big(p(\boldsymbol{x}, y) - p(\boldsymbol{x})p(y)\Big)^2 \mathrm{d}\boldsymbol{x},$$

which is also always non-negative and takes zero if and only if $\boldsymbol{x}$ and y are statistically independent, i.e. $p(\boldsymbol{x}, y) = p(\boldsymbol{x})p(y)$.

Below, an estimator of QMI from $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ called *LS QMI estimation* is introduced, which is an application of *LS density difference estimation* introduced in Section 39.1.3 to QMI. More specifically, LS QMI estimation does not involve estimation of $p(\boldsymbol{x}, y)$, $p(\boldsymbol{x})$, and $p(y)$, but it directly estimates the density difference function:

$$f(\boldsymbol{x}, y) = p(\boldsymbol{x}, y) - p(\boldsymbol{x})p(y).$$

Let us employ the Gaussian kernel model for approximating the density difference $f(\boldsymbol{x}, y)$:

$$f_{\boldsymbol{\alpha}^{(y)}}(\boldsymbol{x}, y) = \sum_{j=1}^{n_y} \alpha_j^{(y)} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_j^{(y)}\|^2}{2h^2}\right),$$

where $\boldsymbol{\alpha}^{(y)} = (\alpha_1, \ldots, \alpha_{n_y})^\top$ is a parameter vector for cluster y, n_y denotes the number of samples in cluster y, and $\{\boldsymbol{x}_i^{(y)}\}_{i=1}^{n_y}$ denote samples in cluster y among $\{\boldsymbol{x}_i\}_{i=1}^n$. The parameter vector $\{\boldsymbol{\alpha}^{(y)}\}_{y=1}^c$ is learned so that the following squared error is minimized:

$$\begin{aligned}J(\boldsymbol{\alpha}^{(1)},\ldots,\boldsymbol{\alpha}^{(c)}) &= \int \sum_{y=1}^{c} \Big(f_{\boldsymbol{\alpha}^{(y)}}(\boldsymbol{x},y) - f(\boldsymbol{x},y)\Big)^2 \mathrm{d}\boldsymbol{x} \\ &= \int \sum_{y=1}^{c} f_{\boldsymbol{\alpha}^{(y)}}(\boldsymbol{x},y)^2 \mathrm{d}\boldsymbol{x} - 2\int \sum_{y=1}^{c} f_{\boldsymbol{\alpha}^{(y)}}(\boldsymbol{x},y) f(\boldsymbol{x},y)\mathrm{d}\boldsymbol{x} + C,\end{aligned}$$

where $C = \int \sum_{y=1}^{c} f(\boldsymbol{x},y)^2 \mathrm{d}\boldsymbol{x}$ is a constant independent of parameter $\boldsymbol{\alpha}$ and thus is ignored. The expectation in the second term may be approximated by the sample average and $p(y) \approx n_y/n$ as

$$\begin{aligned}&\int f_{\boldsymbol{\alpha}^{(y)}}(\boldsymbol{x},y) f(\boldsymbol{x},y)\mathrm{d}\boldsymbol{x} \\ &= \int f_{\boldsymbol{\alpha}^{(y)}}(\boldsymbol{x},y) p(\boldsymbol{x}|y)p(y)\mathrm{d}\boldsymbol{x} - \int f_{\boldsymbol{\alpha}^{(y)}}(\boldsymbol{x},y) p(\boldsymbol{x})p(y)\mathrm{d}\boldsymbol{x} \\ &\approx \frac{1}{n_y}\sum_{i=1}^{n_y} f_{\boldsymbol{\alpha}^{(y)}}(\boldsymbol{x}_i^{(y)},y)\frac{n_y}{n} - \frac{1}{n}\sum_{i=1}^{n} f_{\boldsymbol{\alpha}^{(y)}}(\boldsymbol{x}_i,y)\frac{n_y}{n} \\ &= \frac{1}{n}\sum_{i=1}^{n_y} f_{\boldsymbol{\alpha}^{(y)}}(\boldsymbol{x}_i^{(y)},y) - \frac{n_y}{n^2}\sum_{i=1}^{n} f_{\boldsymbol{\alpha}^{(y)}}(\boldsymbol{x}_i,y).\end{aligned}$$

Further adding the ℓ_2-regularizer yields the following criterion:

$$\min_{\boldsymbol{\alpha}^{(1)},\ldots,\boldsymbol{\alpha}^{(c)}} \sum_{y=1}^{c} \Big[\boldsymbol{\alpha}^{(y)\top}\boldsymbol{U}^{(y)}\boldsymbol{\alpha}^{(y)} - 2\boldsymbol{\alpha}^{(y)\top}\widehat{\boldsymbol{v}}^{(y)} + \lambda\|\boldsymbol{\alpha}^{(y)}\|^2\Big],$$

where $\lambda \geq 0$ is the regularization parameter and $\boldsymbol{U}^{(y)}$ and $\widehat{\boldsymbol{v}}^{(y)}$ are the $n_y \times n_y$ matrix and the n_y-dimensional vector defined as

$$\begin{aligned}U_{j,j'}^{(y)} &= (\sqrt{\pi}h)^d \exp\left(-\frac{\|\boldsymbol{x}_j^{(y)} - \boldsymbol{x}_{j'}^{(y)}\|^2}{4h^2}\right), \\ \widehat{v}_j^{(y)} &= \frac{1}{n}\sum_{i=1}^{n_y} \exp\left(-\frac{\|\boldsymbol{x}_i^{(y)} - \boldsymbol{x}_j^{(y)}\|^2}{2h^2}\right) - \frac{n_y}{n^2}\sum_{i=1}^{n} \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j^{(y)}\|^2}{2h^2}\right).\end{aligned}$$

This optimization problem can be solved analytically and separately for each $y = 1,\ldots,c$ as

$$\widehat{\boldsymbol{\alpha}}^{(y)} = \Big(\boldsymbol{U}^{(y)} + \lambda\boldsymbol{I}\Big)^{-1}\widehat{\boldsymbol{v}}^{(y)}.$$

Then QMI can be approximated as

$$\begin{aligned}\widehat{\mathrm{QMI}} &= \sum_{y=1}^{c}\left(\frac{1}{n}\sum_{i=1}^{n_y} f_{\widehat{\boldsymbol{\alpha}}^{(y)}}(\boldsymbol{x}_i^{(y)},y) - \frac{n_y}{n^2}\sum_{i=1}^{n} f_{\widehat{\boldsymbol{\alpha}}^{(y)}}(\boldsymbol{x}_i,y)\right) \\ &= \sum_{y=1}^{c} \widehat{\boldsymbol{\alpha}}^{(y)\top}\widehat{\boldsymbol{v}}^{(y)},\end{aligned}$$

```
n=500; a=linspace(0,2*pi,n/2)'; y=[ones(1,n/2) zeros(1,n/2)];
x=[a.*cos(a) a.*sin(a); (a+pi).*cos(a) (a+pi).*sin(a)];
x=x+rand(n,2); x=x-repmat(mean(x),[n,1]); x2=sum(x.^2,2);
d=repmat(x2,1,n)+repmat(x2',n,1)-2*x*x';

hs=[0.5 1 2].^2; ls=10.^[-5 -4 -3]; m=2; r=size(x,2)/2;
u=mod(randperm(n),m)+1; hhs=2*[0.5 1 2].^2;
g=zeros(length(hhs),length(ls),m);

for hk=1:length(hs)
  h=hs(hk); k=exp(-d/(2*h));
  for j=unique(y)
    t=(y==j); U=(pi*h)^r*exp(-d(t,t)/(4*h));
    for i=1:m
      ai=(u~=i); ni=sum(a); aj=(u==i); nj=sum(aj);
      vi=sum(k(t,ai&t),2)/ni-sum(k(t,ai),2)*sum(ai&t)/(ni^2);
      vj=sum(k(t,aj&t),2)/nj-sum(k(t,aj),2)*sum(aj&t)/(nj^2);
      for lk=1:length(ls)
        l=ls(lk); a=(U+l*eye(sum(t)))\vi;
        g(hk,lk,i)=g(hk,lk,i)+a'*U*a-2*vj'*a;
end, end, end, end
g=mean(g,3); [gl,ggl]=min(g,[],2); [ghl,gghl]=min(gl);
L=ls(ggl(gghl)); H=hs(gghl); s=0;
for j=unique(y)
  t=(y==j); ny=sum(t); U=(pi*H)^r*exp(-d(t,t)/(4*H));
  k=exp(-d(t,:)/(2*H)); v=sum(k(:,t),2)/n-sum(k,2)*ny/(n^2);
  a=(U+L*eye(ny))\v; s=s+v'*a;
end
disp(sprintf('Information=%g',s));
```

FIGURE 37.8

MATLAB code for LS QMI estimation.

which comes from the following expression of QMI:

$$\mathrm{QMI} = \int \sum_{y=1}^{c} f(\boldsymbol{x}, y)\Big(p(\boldsymbol{x}, y) - p(\boldsymbol{x})p(y)\Big)\mathrm{d}\boldsymbol{x}.$$

All tuning parameters such as the regularization parameter λ and the Gaussian bandwidth h can be optimized by cross validation with respect to the squared error J. This is a strong advantage in unsupervised clustering.

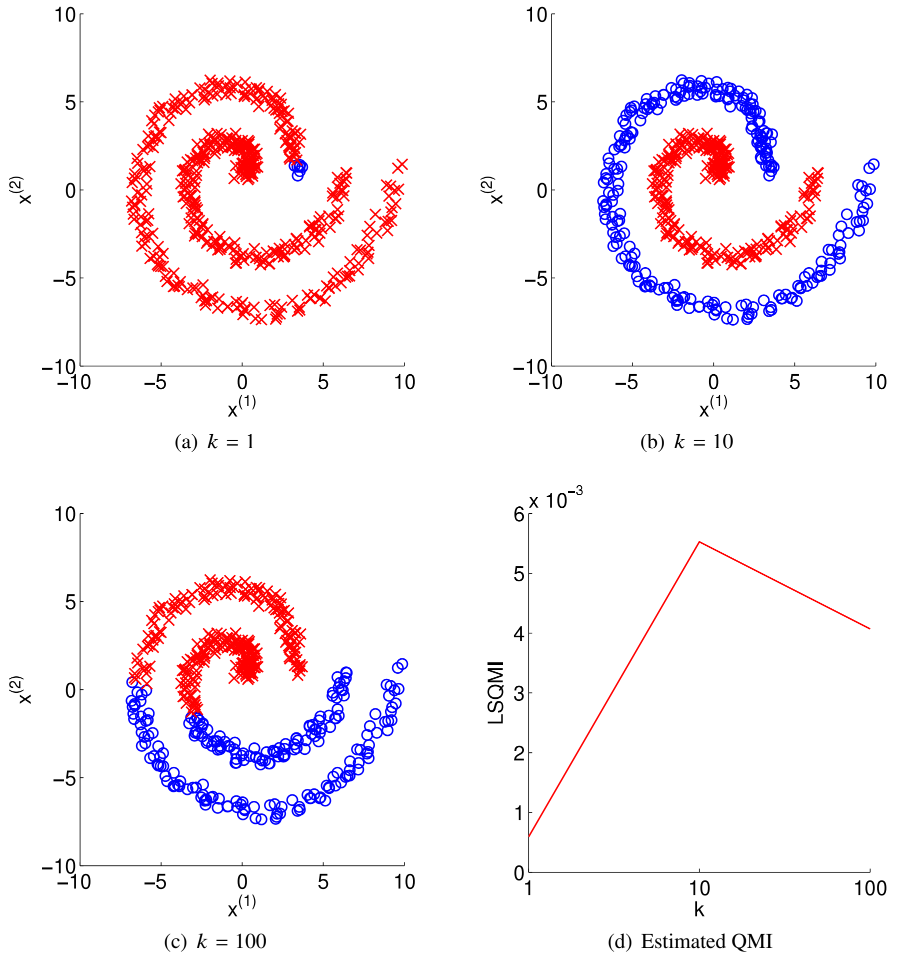


FIGURE 37.9

Example of LS QMI estimation.

A MATLAB code for LS QMI estimation is provided in Fig. 37.8, and its behavior is illustrated in Fig. 37.9. This shows that setting the parameter k included in the nearest neighbor similarity at a too small value (Fig. 37.9(a)) or too large value (Fig. 37.9(c)) does not produce a desirable clustering solution. On the other hand, if the parameter k is chosen so that the LS QMI estimator is maximized, an intermediate value is selected (Fig. 37.9(d)) and this choice works well for this data set (Fig. 37.9(b)).