CHAPTER

LINEAR DIMENSIONALITY REDUCTION 35

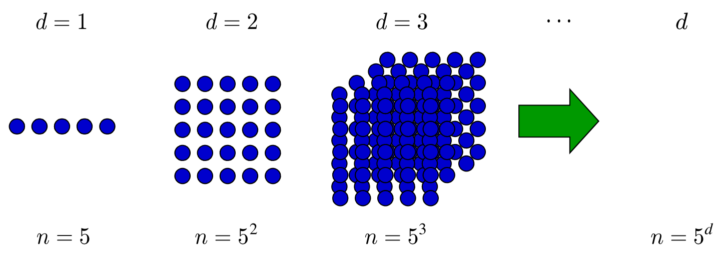
CHAPTER CONTENTS

Handling high-dimensional data is often cumbersome in practical data analysis. In this chapter, supervised and unsupervised methods of *linear dimensionality reduction* are introduced for reducing the dimensionality of data while preserving intrinsic information contained in the data.

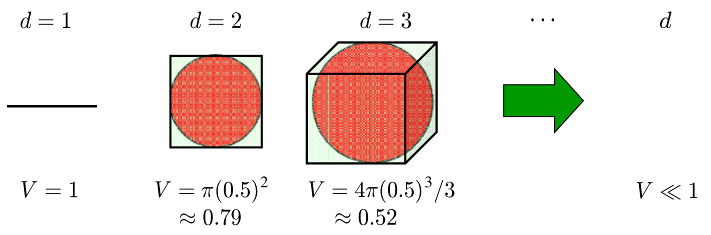
35.1 CURSE OF DIMENSIONALITY

As the dimensionality of input $\boldsymbol{x}$ grows, any learning problem significantly gets harder and harder. For example, let us sample 5 points from $[0,1]$ at regular intervals. Collecting samples in the same way in d-dimensional space requires 5^d points (Fig. 35.1(a)), which grows exponentially with respect to d. Since collecting 5^d points when d is large is not possible in practice (e.g. $5^{100} \approx 10^{70}$), samples are always scarce in high-dimensional problems.

Another trouble in high-dimensional problems is that our geometric intuition can be misleading. For example, let us consider the *inscribed hypersphere* of the unit hypercube in d-dimensional space (Fig. 35.1(b)). When $d = 1$, the volume of the inscribed hypersphere is 1, which is the same as the hypercube. When d is increased to 2 and 3, the volume of the hypercube is still 1, but the volume of the inscribed



(a) Grid sampling in high-dimensional space. The required number of samples grows exponentially with respect to the dimensionality. Thus, samples are always scarce in high-dimensional space



(b) Volume of inscribed hypersphere in high-dimensional space. While the volume of the unit hypercube is always 1, the volume of the inscribed hypersphere tends to 0 as the dimensionality grows. Thus, the inscribed hypersphere can be ignored in high-dimensional space

FIGURE 35.1

Curse of dimensionality.

hypersphere is decreased to approximately 0.79 and 0.52, respectively. Our geometric intuition is that, even though the inscribed hypersphere is smaller than the hypercube, the inscribed hypersphere is not extremely small. However, as d is further increased, the volume of the hypercube is still 1, but the volume of the inscribed hypersphere tends to 0. This means that, when d is large, the inscribed hypersphere is almost negligible.

As illustrated above, handling high-dimensional data is cumbersome in practice, which is often referred to as the *curse of dimensionality*. In the following sections, various methods of dimensionality reduction are introduced.

Linear dimensionality reduction transforms the original samples $\{\boldsymbol{x}_i\}_{i=1}^n$ to low-dimensional expressions $\{\boldsymbol{z}_i\}_{i=1}^n$ by a linear transformation $\boldsymbol{T} \in \mathbb{R}^{m \times d}$ (Fig. 35.2):

$$\boldsymbol{z}_i = \boldsymbol{T}\boldsymbol{x}_i,$$

$\boldsymbol{T}$ is called an *embedding matrix*. This chapter is devoted to introducing various linear dimensionality reduction methods. Nonlinear methods will be covered in Chapter 36.

In the rest of this chapter, training input samples $\{\boldsymbol{x}_i\}_{i=1}^n$ are assumed to be *centralized* as Fig. 35.3:

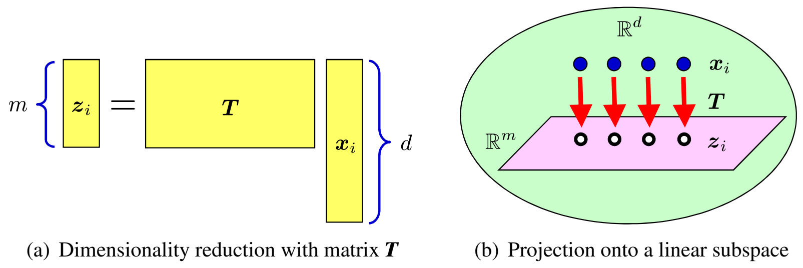

m
z_i
$=$
T
x_i
d
$\mathbb{R}^d$
x_i
T
$\mathbb{R}^m$
z_i
(a) Dimensionality reduction with matrix T
(b) Projection onto a linear subspace

FIGURE 35.2

Linear dimensionality reduction. Transformation by a fat matrix $\boldsymbol{T}$ corresponds to projection onto a subspace.

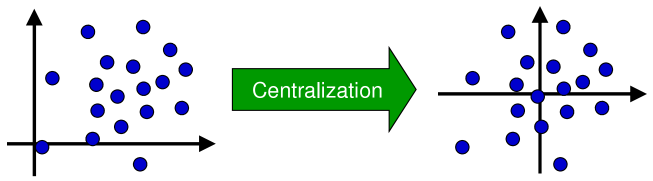


FIGURE 35.3

Data centering.

$$\boldsymbol{x}_i \longleftarrow \boldsymbol{x}_i - \frac{1}{n}\sum_{i'=1}^{n} \boldsymbol{x}_{i'}.$$

35.2 UNSUPERVISED DIMENSIONALITY REDUCTION

This section covers *unsupervised dimensionality reduction*, which is aimed at decreasing the dimensionality of input samples $\{\boldsymbol{x}_i\}_{i=1}^n$ without losing their intrinsic information.

35.2.1 PCA

First, a classical unsupervised linear dimensionality reduction method called PCA [59] is introduced.

PCA tries to keep the position of original samples when the dimensionality is reduced (Fig. 35.4). More specifically, under the constraint that $\boldsymbol{z}_i$ is an orthogonal projection of $\boldsymbol{x}_i$, embedding matrix $\boldsymbol{T}$ that keeps $\boldsymbol{z}_i$ as close to $\boldsymbol{x}_i$ as possible is found. The constraint that $\boldsymbol{z}_i$ is an orthogonal projection of $\boldsymbol{x}_i$ is equivalent to the fact that embedding matrix $\boldsymbol{T}$ satisfies $\boldsymbol{T}\boldsymbol{T}^\top = \boldsymbol{I}_m$, where $\boldsymbol{I}_m$ is the $m \times m$ identity matrix.

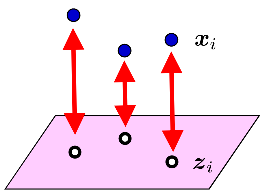


FIGURE 35.4

PCA, which tries to keep the position of original samples when the dimensionality is reduced.

However, as illustrated in Fig. 35.2(b), the dimensionality of $\boldsymbol{x}_i$ is different from that of $\boldsymbol{z}_i$, and thus the distance between $\boldsymbol{x}_i$ and $\boldsymbol{z}_i$ cannot be directly measured. Here, m-dimensional expression $\boldsymbol{z}_i$ is transformed back to d-dimensional space by $\boldsymbol{T}^\top$ and then the Euclidean distance to $\boldsymbol{x}_i$ is computed:

$$\sum_{i=1}^{n} \|\boldsymbol{T}^\top \boldsymbol{T} \boldsymbol{x}_i - \boldsymbol{x}_i\|^2 = -\mathrm{tr}\left(\boldsymbol{T}\boldsymbol{C}\boldsymbol{T}^\top\right) + \mathrm{tr}\left(\boldsymbol{C}\right),$$

where $\boldsymbol{C}$ is the *total scatter matrix*:

$$\boldsymbol{C} = \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top.$$

Summarizing the above discussion, the optimization problem of PCA is given by

$$\max_{\boldsymbol{T} \in \mathbb{R}^{m\times d}} \mathrm{tr}\left(\boldsymbol{T}\boldsymbol{C}\boldsymbol{T}^\top\right) \quad \text{subject to } \boldsymbol{T}\boldsymbol{T}^\top = \boldsymbol{I}_m.$$

Although this is a nonconvex optimization problem, a global optimal solution is given analytically by

$$\boldsymbol{T} = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_m)^\top,$$

where $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_d$ are eigenvectors of matrix $\boldsymbol{C}$ associated with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$ (Fig. 6.2):

$$\boldsymbol{C}\boldsymbol{\xi} = \lambda\boldsymbol{\xi}.$$

Note that other global optimal solutions are given by

$$\boldsymbol{T} = \boldsymbol{G}(\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_m)^\top,$$

where $\boldsymbol{G}$ is any *orthogonal matrix* such that $\boldsymbol{G}^{-1} = \boldsymbol{G}^\top$.

The embedding matrix of PCA is an orthogonal projection onto the subspace spanned by eigenvectors associated with large eigenvalues. In other words, by

```
n=100; x=[2*randn(n,1) randn(n,1)];
%x=[2*randn(n,1) 2*round(rand(n,1))-1+randn(n,1)/3];
x=x-repmat(mean(x),[n,1]);
[t,v]=eigs(x'*x,1);

figure(1); clf; hold on; axis([-6 6 -6 6]);
plot(x(:,1),x(:,2),'rx')
plot(9*[-t(1) t(1)],9*[-t(2) t(2)]);
```

FIGURE 35.5

MATLAB code for PCA.

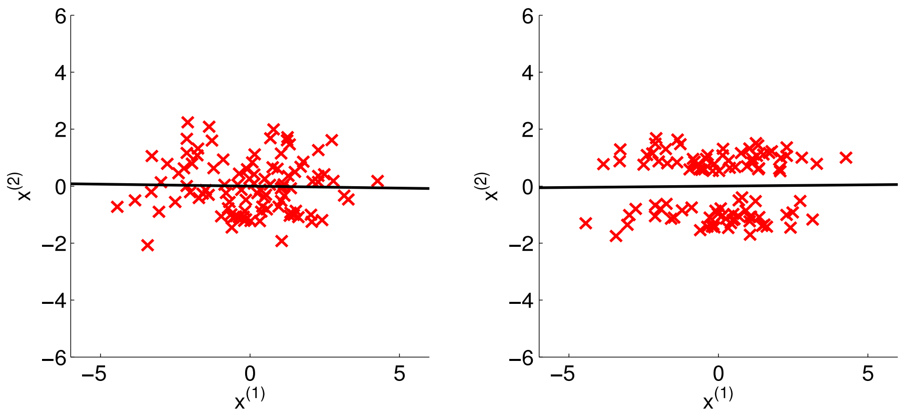


FIGURE 35.6

Example of PCA. The solid line denotes the one-dimensional embedding subspace found by PCA.

removing eigenvectors associated with small eigenvalues, the gap from the original samples is kept minimum. Note that PCA makes data samples *uncorrelated*, i.e. the sample variance-covariance matrix of $\{z_i\}_{i=1}^n$ is *diagonal*:

$$\sum_{i=1}^{n} z_i z_i^\top = \operatorname{diag}(\lambda_1, \ldots, \lambda_m).$$

A MATLAB code for PCA is provided in Fig. 35.5, and its behavior is illustrated in Fig. 35.6. This shows that samples are embedded into a one-dimensional subspace so that they do not change a lot. However, PCA does not necessarily maintain useful structure of data such as *clusters*.

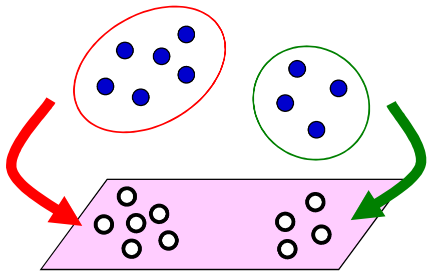


FIGURE 35.7

Locality preserving projection, which tries to keep the cluster structure of original samples when the dimensionality is reduced.

In Section 23.1, subspace-constrained LS was introduced, which requires a subspace in the input space. If PCA is used in subspace-constrained LS, the corresponding regression method is called *principal component regression*.

35.2.2 LOCALITY PRESERVING PROJECTION

To preserving cluster structure of data when dimensionality is reduced, *locality preserving projection* [53] is useful (Fig. 35.7).

In locality preserving projection, *similarity* between samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$, denoted by $0 \leq W_{i,i'} \leq 1$, is utilized. $W_{i,i'}$ takes a large value (i.e. close to one) if $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$ are "similar," while $W_{i,i'}$ takes a small value (i.e. close to zero) if $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$ are "dissimilar." The similarity is assumed to be symmetric, i.e. $W_{i,i'} = W_{i',i}$. Popular choices of the similarity measure are summarized in Fig. 35.8.

Locality preserving projection determines embedding matrix $\boldsymbol{T}$ so that sample pairs with high similarity are close to each other in the embedding space, i.e. the following criterion is minimized:

$$\frac{1}{2}\sum_{i,i'=1}^{n} W_{i,i'}\|\boldsymbol{T}\boldsymbol{x}_i - \boldsymbol{T}\boldsymbol{x}_{i'}\|^2. \tag{35.1}$$

However, this minimization problem has a trivial solution $\boldsymbol{T} = \boldsymbol{O}$, which is meaningless. To avoid such a degenerated solution, an appropriate constraint such as

$$\boldsymbol{T}\boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^\top\boldsymbol{T}^\top = \boldsymbol{I}_m$$

is imposed, where

$$\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d\times n},$$

- Gaussian similarity:

$$W_{i,i'} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{i'}\|^2}{2t^2}\right),$$

  where $t > 0$ is a tuning parameter that controls the decay of Gaussian tails.
- k-nearest-neighbor similarity:

$$W_{i,i'} = \begin{cases} 1 & (\boldsymbol{x}_i \in \mathcal{N}_k(\boldsymbol{x}_{i'}) \text{ or } \boldsymbol{x}_{i'} \in \mathcal{N}_k(\boldsymbol{x}_i)), \\ 0 & (\text{otherwise}), \end{cases}$$

  where $\mathcal{N}_k(\boldsymbol{x})$ denotes the set of k-nearest-neighbor samples of $\boldsymbol{x}$ in $\{\boldsymbol{x}_i\}_{i=1}^n$, and $k \in \{1, \ldots, n\}$ is a tuning parameter that controls the locality. Note that k-nearest-neighbor similarity produces a *sparse* similarity matrix $\boldsymbol{W}$, which is often advantageous in practice.
- Local scaling similarity [123]:

$$W_{i,i'} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{i'}\|^2}{2t_i t_{i'}}\right),$$

  where t_i is the local scaling defined by

$$t_i = \|\boldsymbol{x}_i - \boldsymbol{x}_i^{(k)}\|,$$

  and $\boldsymbol{x}_i^{(k)}$ denotes the kth nearest neighbor of $\boldsymbol{x}_i$ in $\{\boldsymbol{x}_i\}_{i=1}^n$. It is also practical to combine k-nearest-neighbor similarity and local scaling similarity.

FIGURE 35.8

Popular choices of similarity measure.

$$D_{i,i'} = \begin{cases} \sum_{i''=1}^{n} W_{i,i''} & (i = i'), \\ 0 & (i \neq i'). \end{cases}$$

For

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}, \tag{35.2}$$

```
n=100; x=[2*randn(n,1) randn(n,1)];
%x=[2*randn(n,1) 2*round(rand(n,1))-1+randn(n,1)/3];
x=x-repmat(mean(x),[n,1]); x2=sum(x.^2,2);
W=exp(-(repmat(x2,1,n)+repmat(x2',n,1)-2*x*x'));
D=diag(sum(W,2)); L=D-W; z=x'*D*x; z=(z+z')/2;
[t,v]=eigs(x'*L*x,z,1,'sm');

figure(1); clf; hold on; axis([-6 6 -6 6]);
plot(x(:,1),x(:,2),'rx')
plot(9*[-t(1) t(1)],9*[-t(2) t(2)]);
```

FIGURE 35.9

MATLAB code for locality preserving projection.

Eq. (35.1) can be compactly expressed as $\mathrm{tr}\,(\boldsymbol{T}\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^\top\boldsymbol{T}^\top)$. The matrix $\boldsymbol{L}$ is called the *graph Laplacian matrix*, which plays an important role in *spectral graph theory* (see Section 33.1).

Summarizing the above discussion, the optimization problem of locality preserving projection is given as

$$\min_{\boldsymbol{T}\in\mathbb{R}^{m\times d}} \mathrm{tr}\left(\boldsymbol{T}\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^\top\boldsymbol{T}^\top\right) \ \text{ subject to } \boldsymbol{T}\boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^\top\boldsymbol{T}^\top = \boldsymbol{I}_m.$$

Although this is a nonconvex optimization problem, a global optimal solution is given analytically by

$$\boldsymbol{T} = (\boldsymbol{\xi}_d, \boldsymbol{\xi}_{d-1} \ldots, \boldsymbol{\xi}_{d-m+1})^\top,$$

where $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_d$ are generalized eigenvectors of $(\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^\top, \boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^\top)$ associated with generalized eigenvalues $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$ (Fig. 6.2):

$$\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^\top\boldsymbol{\xi} = \lambda\boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^\top\boldsymbol{\xi}.$$

Thus, the embedding matrix of locality preserving projection is given by the minor generalized eigenvectors of $(\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^\top, \boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^\top)$.

A MATLAB code for locality preserving projection is provided in Fig. 35.9, and its behavior is illustrated in Fig. 35.10. This shows that cluster structure is nicely preserved.

35.3 LINEAR DISCRIMINANT ANALYSES FOR CLASSIFICATION

In this section, methods of *supervised linear dimensionality reduction* based on input-output paired training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ for classification (i.e. $y \in \{1, \ldots, c\}$) are introduced.

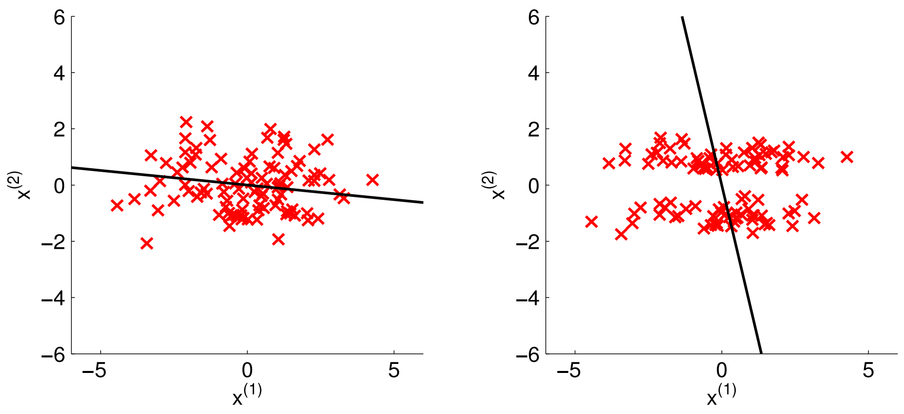


FIGURE 35.10

Example of locality preserving projection. The solid line denotes the one-dimensional embedding subspace found by locality preserving projection.

35.3.1 FISHER DISCRIMINANT ANALYSIS

First, let us introduce a classical supervised linear dimensionality reduction method called *Fisher discriminant analysis* [42].

The basic idea of Fisher discriminant analysis is to find a transformation matrix $\boldsymbol{T}$ so that sample pairs in the same class get closer to each other and sample pairs in different classes are far apart. More specifically, let $\boldsymbol{S}^{(\mathrm{w})}$ be the *within-class scatter matrix* and $\boldsymbol{S}^{(\mathrm{b})}$ be the *between-class scatter matrix* defined as

$$\boldsymbol{S}^{(\mathrm{w})} = \sum_{y=1}^{c} \sum_{i:y_i=y} (\boldsymbol{x}_i - \boldsymbol{\mu}_y)(\boldsymbol{x}_i - \boldsymbol{\mu}_y)^\top \in \mathbb{R}^{d\times d}, \tag{35.3}$$

$$\boldsymbol{S}^{(\mathrm{b})} = \sum_{y=1}^{c} n_y \boldsymbol{\mu}_y \boldsymbol{\mu}_y^\top \in \mathbb{R}^{d\times d}, \tag{35.4}$$

where $\sum_{i:y_i=y}$ denotes the summation over i such that $y_i = y$. $\boldsymbol{\mu}_y$ denotes the mean of training samples in class y:

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

where n_y denotes the number of training samples in class y. Then the optimization problem of Fisher discriminant analysis is given as

$$\max_{\boldsymbol{T}\in\mathbb{R}^{m\times d}} \mathrm{tr}\Big((\boldsymbol{T}\boldsymbol{S}^{(\mathrm{w})}\boldsymbol{T}^\top)^{-1}\boldsymbol{T}\boldsymbol{S}^{(\mathrm{b})}\boldsymbol{T}^\top\Big),$$

where $\boldsymbol{T}\boldsymbol{S}^{(\mathrm{w})}\boldsymbol{T}^\top$ denotes the within-class scatter matrix after dimensionality reduction and $\boldsymbol{T}\boldsymbol{S}^{(\mathrm{b})}\boldsymbol{T}^\top$ denotes the between-class scatter matrix after dimensionality reduction.

```
n=100; x=randn(n,2);
x(1:n/2,1)=x(1:n/2,1)-4; x(n/2+1:end,1)=x(n/2+1:end,1)+4;
%x(1:n/4,1)=x(1:n/4,1)-4; x(n/4+1:n/2,1)=x(n/4+1:n/2,1)+4;
x=x-repmat(mean(x),[n,1]);
y=[ones(n/2,1); 2*ones(n/2,1)];

m1=mean(x(y==1,:));
x1=x(y==1,:)-repmat(m1,[n/2,1]);
m2=mean(x(y==2,:));
x2=x(y==2,:)-repmat(m2,[n/2,1]);
[t,v]=eigs(n/2*m1'*m1+n/2*m2'*m2,x1'*x1+x2'*x2,1);

figure(1); clf; hold on; axis([-8 8 -6 6]);
plot(x(y==1,1),x(y==1,2),'bo')
plot(x(y==2,1),x(y==2,2),'rx')
plot(99*[-t(1) t(1)],99*[-t(2) t(2)], 'k-')
```

FIGURE 35.11

MATLAB code for Fisher discriminant analysis.

Thus, Fisher discriminant analysis finds a transformation matrix $\boldsymbol{T}$ that decreases the within-class scatter and increases the between-class scatter.

Let $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$ and $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_d$ be the *generalized eigenvalues* and *generalized eigenvectors* of $(\boldsymbol{S}^{(\mathrm{b})}, \boldsymbol{S}^{(\mathrm{w})})$:

$$\boldsymbol{S}^{(\mathrm{b})}\boldsymbol{\xi} = \lambda \boldsymbol{S}^{(\mathrm{w})}\boldsymbol{\xi}.$$

Then the solution $\widehat{\boldsymbol{T}}$ of Fisher discriminant analysis is given analytically as

$$\widehat{\boldsymbol{T}} = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_m)^\top.$$

A MATLAB code for Fisher discriminant analysis is provided in Fig. 35.11, and its behavior is illustrated in Fig. 35.12. In these examples, two-dimensional samples are projected onto one-dimensional subspaces, and a subspace that nicely separates samples in two classes can be found in Fig. 35.12(a). However, in Fig. 35.12(b), samples in two classes are mixed up due to the within-class *multimodality* in class "∘."

35.3.2 LOCAL FISHER DISCRIMINANT ANALYSIS

As illustrated in Fig. 35.12, Fisher discriminant analysis can give an undesired solution if within-class multimodality exists. Another limitation of Fisher discriminant analysis is that the between-class scatter matrix $\boldsymbol{S}^{(\mathrm{b})}$ has rank at most $c - 1$.

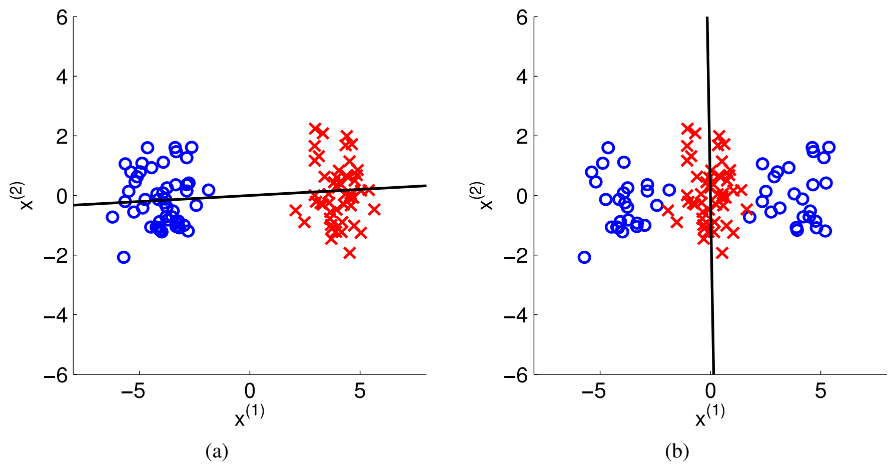


FIGURE 35.12

Examples of Fisher discriminant analysis. The solid lines denote the found subspaces to which training samples are projected.

This means that generalized eigenvectors $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_d$ used for computing the solution make sense only up to the $(c-1)$th generalized eigenvector. Thus, in practice, the reduced dimensionality m should be less than the number of classes, c. This is highly restrictive when c is small. Indeed, in binary classification where $c = 2$, m should be 1. To overcome these limitations, *local Fisher discriminant analysis* [98] introduced here.

The within-class scatter matrix $\boldsymbol{S}^{(\mathrm{w})}$ and the between-class scatter matrix $\boldsymbol{S}^{(\mathrm{b})}$ used in Fisher discriminant analysis defined by Eqs (35.3) and (35.4) can be expressed as

$$\boldsymbol{S}^{(\mathrm{w})} = \frac{1}{2}\sum_{i,i'=1}^{n} Q^{(\mathrm{w})}_{i,i'}(\boldsymbol{x}_i - \boldsymbol{x}_{i'})(\boldsymbol{x}_i - \boldsymbol{x}_{i'})^\top,$$

$$\boldsymbol{S}^{(\mathrm{b})} = \frac{1}{2}\sum_{i,i'=1}^{n} Q^{(\mathrm{b})}_{i,i'}(\boldsymbol{x}_i - \boldsymbol{x}_{i'})(\boldsymbol{x}_i - \boldsymbol{x}_{i'})^\top,$$

where

$$Q^{(\mathrm{w})}_{i,i'} = \begin{cases} 1/n_y > 0 & (y_i = y_{i'} = y), \\ 0 & (y_i \neq y_{i'}), \end{cases}$$

$$Q^{(\mathrm{b})}_{i,i'} = \begin{cases} 1/n - 1/n_y < 0 & (y_i = y_{i'} = y), \\ 1/n > 0 & (y_i \neq y_{i'}). \end{cases}$$

These pairwise expressions allow intuitive understanding of the behavior of Fisher discriminant analysis, i.e. sample pairs in the same class get close to each other and sample pairs in different classes are far apart. At the same time, failure of Fisher discriminant analysis in the presence of within-class multimodality can be explained by the fact that *all* samples in the same class are gotten close to each other even if they form multiple clusters.

To cope with this problem, local Fisher discriminant analysis uses the *local* within-class scatter matrix $\boldsymbol{S}^{(\mathrm{lw})}$ and the *local* between-class scatter matrix $\boldsymbol{S}^{(\mathrm{lb})}$ defined as

$$\boldsymbol{S}^{(\mathrm{lw})} = \frac{1}{2}\sum_{i,i'=1}^{n} Q_{i,i'}^{(\mathrm{lw})}(\boldsymbol{x}_i - \boldsymbol{x}_{i'})(\boldsymbol{x}_i - \boldsymbol{x}_{i'})^\top,$$

$$\boldsymbol{S}^{(\mathrm{lb})} = \frac{1}{2}\sum_{i,i'=1}^{n} Q_{i,i'}^{(\mathrm{lb})}(\boldsymbol{x}_i - \boldsymbol{x}_{i'})(\boldsymbol{x}_i - \boldsymbol{x}_{i'})^\top,$$

where

$$Q_{i,i'}^{(\mathrm{lw})} = \begin{cases} W_{i,i'}/n_y & (y_i = y_{i'} = y), \\ 0 & (y_i \neq y_{i'}), \end{cases}$$

$$Q_{i,i'}^{(\mathrm{lb})} = \begin{cases} W_{i,i'}(1/n - 1/n_y) & (y_i = y_{i'} = y), \\ 1/n & (y_i \neq y_{i'}). \end{cases}$$

$0 \leq W_{i,i'} \leq 1$ denotes a similarity between sample $\boldsymbol{x}_i$ and sample $\boldsymbol{x}_{i'}$ (see Fig. 35.8). In the above local scatter matrices, similarity $W_{i,i'}$ is applied to sample pairs in the same class, which mitigates faraway sample pairs in the same class to be gotten close to each other strongly and thus within-class cluster structure tends to be preserved. Local Fisher discriminant analysis can be interpreted as applying *locality preserving projection* introduced in Section 35.2.2 in a classwise manner on top of Fisher discriminant analysis. Note that similarity $W_{i,i'}$ is *not* applied to sample pairs in different classes, since they should be faraway even if they belong to different clusters.

The optimization problem of local Fisher discriminant analysis is given by

$$\max_{\boldsymbol{T}\in\mathbb{R}^{m\times d}} \mathrm{tr}\Big((\boldsymbol{T}\boldsymbol{S}^{(\mathrm{lw})}\boldsymbol{T}^\top)^{-1}\boldsymbol{T}\boldsymbol{S}^{(\mathrm{lb})}\boldsymbol{T}^\top\Big),$$

which has exactly the same form as the original Fisher discriminant analysis. Thus, the solution $\widehat{\boldsymbol{T}}$ of local Fisher discriminant analysis can be obtained analytically in the same way by

$$\widehat{\boldsymbol{T}} = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_m)^\top,$$

```
n=100; x=randn(n,2);
x(1:n/2,1)=x(1:n/2,1)-4; x(n/2+1:end,1)=x(n/2+1:end,1)+4;
%x(1:n/4,1)=x(1:n/4,1)-4; x(n/4+1:n/2,1)=x(n/4+1:n/2,1)+4;
x=x-repmat(mean(x),[n,1]); y=[ones(n/2,1); 2*ones(n/2,1)];

Sw=zeros(2,2); Sb=zeros(2,2);
for j=1:2
  p=x(y==j,:); p1=sum(p); p2=sum(p.^2,2); nj=sum(y==j);
  W=exp(-(repmat(p2,1,nj)+repmat(p2',nj,1)-2*p*p'));
  G=p'*(repmat(sum(W,2),[1 2]).*p)-p'*W*p;
  Sb=Sb+G/n+p'*p*(1-nj/n)+p1'*p1/n; Sw=Sw+G/nj;
end
[t,v]=eigs((Sb+Sb')/2,(Sw+Sw')/2,1);

figure(1); clf; hold on; axis([-8 8 -6 6]);
plot(x(y==1,1),x(y==1,2),'bo')
plot(x(y==2,1),x(y==2,2),'rx')
plot(99*[-t(1) t(1)],99*[-t(2) t(2)], 'k-')
```

FIGURE 35.13

MATLAB code for local Fisher discriminant analysis.

where $\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_d$ are generalized eigenvectors associated with generalized eigenvalues $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$ of $(\boldsymbol{S}^{(\mathrm{lb})},\boldsymbol{S}^{(\mathrm{lw})})$:

$$\boldsymbol{S}^{(\mathrm{lb})}\boldsymbol{\xi} = \lambda \boldsymbol{S}^{(\mathrm{lw})}\boldsymbol{\xi}.$$

Another important advantage of local Fisher discriminant analysis is that the low-rank problem of $\boldsymbol{S}^{(\mathrm{b})}$ can also be avoided. More specifically, since $\boldsymbol{S}^{(\mathrm{lb})}$ contains the similarity $W_{i,i'}$, $\boldsymbol{S}^{(\mathrm{lb})}$ usually has full rank. Then all generalized eigenvectors $\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_d$ make sense and thus the reduced dimensionality m can be arbitrarily large.

A MATLAB code for local Fisher discriminant analysis is provided in Fig. 35.13, and its behavior is illustrated in Fig. 35.14. This shows that local Fisher discriminant analysis performs well even in the presence of within-class multimodality.

35.3.3 SEMISUPERVISED LOCAL FISHER DISCRIMINANT ANALYSIS

Supervised dimensionality reduction tends to *overfit* if the number of training samples is small. Here, *semisupervised local Fisher discriminant analysis* [100] is introduced, which utilizes unlabeled samples $\{\boldsymbol{x}_i\}_{i=n+1}^{n+n'}$ in addition to ordinary labeled training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ to mitigate overfitting (Section 33.1).

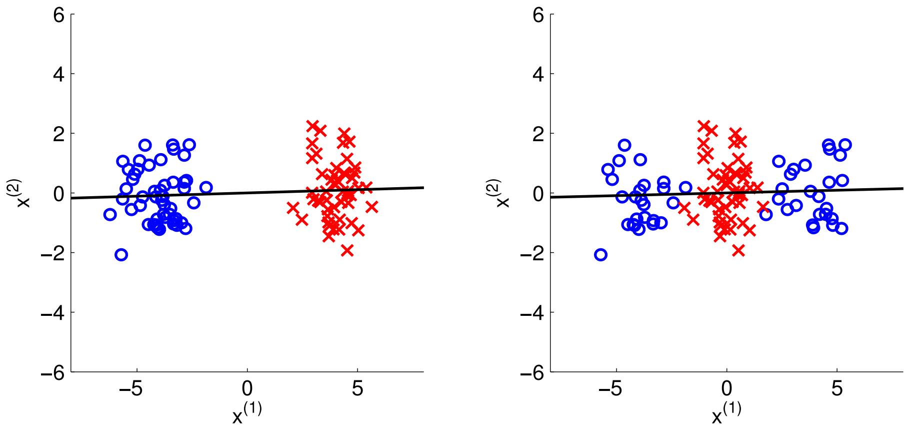


FIGURE 35.14

Examples of local Fisher discriminant analysis for the same data sets as Fig. 35.12. The solid lines denote the found subspaces to which training samples are projected.

The basic idea of semisupervised local Fisher discriminant analysis is to combine (supervised) local Fisher discriminant analysis with (unsupervised) PCA. As explained in Section 35.2.1, the solution of PCA is given by the leading eigenvectors of the *total scatter matrix*:

$$\boldsymbol{S}^{(\mathrm{t})} = \sum_{i=1}^{n+n'} (\boldsymbol{x}_i - \boldsymbol{\mu}^{(\mathrm{t})})(\boldsymbol{x}_i - \boldsymbol{\mu}^{(\mathrm{t})})^\top,$$

where $\boldsymbol{\mu}^{(\mathrm{t})}$ denotes the mean of all (i.e. labeled and unlabeled) input samples $\{\boldsymbol{x}_i\}_{i=1}^{n+n'}$:

$$\boldsymbol{\mu}^{(\mathrm{t})} = \frac{1}{n+n'} \sum_{i=1}^{n+n'} \boldsymbol{x}_i.$$

Note that $\boldsymbol{\mu}^{(\mathrm{t})}$ is not necessarily zero even though input of labeled samples, $\{\boldsymbol{x}_i\}_{i=1}^{n}$, is assumed to be centralized through this chapter (Fig. 35.3). Similarly, the solution of local Fisher discriminant analysis is given by solving a generalized eigenvalue problem, as shown in Section 35.3.2. Based on these facts, the basic idea of semisupervised local Fisher discriminant analysis is to combine these eigenvalue problems.

More specifically, instead of the local scatter matrices used in local Fisher discriminant analysis, semisupervised local Fisher discriminant analysis uses the following *semisupervised* local scatter matrices:

$$\boldsymbol{S}^{(\mathrm{slw})} = (1-\beta)\boldsymbol{S}^{(\mathrm{lw})} + \beta \boldsymbol{S}^{(\mathrm{t})},$$

$$\boldsymbol{S}^{(\mathrm{slb})} = (1-\beta)\boldsymbol{S}^{(\mathrm{lb})} + \beta \boldsymbol{I},$$

where $\boldsymbol{I}$ denotes the identity matrix and $\beta \in [0,1]$ controls the balance between labeled and unlabeled samples. Based on these scatter matrices, the optimization problem of semisupervised local Fisher discriminant analysis is given as

$$\max_{\boldsymbol{T}\in\mathbb{R}^{m\times d}} \mathrm{tr}\Big((\boldsymbol{T}\boldsymbol{S}^{(\mathrm{slw})}\boldsymbol{T}^\top)^{-1}\boldsymbol{T}\boldsymbol{S}^{(\mathrm{slb})}\boldsymbol{T}^\top\Big),$$

which has exactly the same form as the original Fisher discriminant analysis. Thus, the solution $\widehat{\boldsymbol{T}}$ of semisupervised local Fisher discriminant analysis can be obtained analytically in the same way by

$$\widehat{\boldsymbol{T}} = (\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_m)^\top,$$

where $\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_d$ are generalized eigenvectors associated with generalized eigenvalues $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$ of $(\boldsymbol{S}^{(\mathrm{slb})},\boldsymbol{S}^{(\mathrm{slw})})$:

$$\boldsymbol{S}^{(\mathrm{slb})}\boldsymbol{\xi} = \lambda\boldsymbol{S}^{(\mathrm{slw})}\boldsymbol{\xi}.$$

This solution is reduced to (supervised) local Fisher discriminant analysis if $\beta = 0$ and is reduced to (unsupervised) PCA if $\beta = 1$. For $0 < \beta < 1$, semisupervised local Fisher discriminant analysis is expected to give a solution that bridges the two extreme cases.

A MATLAB code for semisupervised local Fisher discriminant analysis is provided in Fig. 35.15, and its behavior is illustrated in Fig. 35.16. This shows that semisupervised local Fisher discriminant analysis can successfully avoid overfitting.

35.4 SUFFICIENT DIMENSIONALITY REDUCTION FOR REGRESSION

The discriminant analysis methods introduced in the previous section explicitly used the class labels and thus cannot be applied to regression. Here, another supervised dimensionality reduction method called *sufficient dimensionality reduction* [67] is introduced, which can also be applied to regression.

35.4.1 INFORMATION THEORETIC FORMULATION

The basic idea of sufficient dimensionality reduction is to find a transformation matrix $\boldsymbol{T}$ that makes $\boldsymbol{x}$ *conditionally independent* of output y given projection $\boldsymbol{z} = \boldsymbol{T}\boldsymbol{x}$:

$$p(\boldsymbol{x}, y|\boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(y|\boldsymbol{z}).$$

This means that $\boldsymbol{z}$ contains all information about output y.

Such a transformation matrix $\boldsymbol{T}$ is characterized as the maximizer of *mutual information* (MI) [92]:

$$\mathrm{MI} = \iint p(\boldsymbol{z}, y)\log\frac{p(\boldsymbol{z}, y)}{p(\boldsymbol{z})p(y)}\mathrm{d}\boldsymbol{z}\mathrm{d}y.$$

```
n=2; m=200; x=0.1*randn(n,2); b=0.9; %b=0.001; b=1;
x(:,1)=x(:,1)+[repmat(3,[n/2,1]); repmat(-3,[n/2,1])];
%x(1:n/2,2)=x(1:n/2,2)+repmat(5,[n/2,1]);
xx=randn(m,2).*repmat([1 2],[m 1]);
xx(:,1)=xx(:,1)+[repmat(-3,[m/2,1]); repmat(3,[m/2,1])];
%x(:,2)=x(:,2)*1.7; xx(:,2)=xx(:,2)*1.7;
mu=mean([x;xx]); x=x-repmat(mu,[n,1]);
xx=xx-repmat(mu,[m,1]); y=[ones(n/2,1); 2*ones(n/2,1)];

x2=sum(x.^2,2); Qlb=zeros(n,n); Qlw=zeros(n,n);
W=exp(-(repmat(x2,1,n)+repmat(x2',n,1)-2*x*x'));
for j=1:2
  Wy=W.*((y==j)/2*(y==j)'); Qlw=Qlw+Wy/sum(y==j);
  Qlb=Qlb+Wy*(1/n-1/sum(y==j))+(y==j)/n/2*(y~=j)';
end
Srlb=(1-b)*x'*(diag(sum(Qlb))-Qlb)*x+b*cov([x; xx],1);
Srlw=(1-b)*x'*(diag(sum(Qlw))-Qlw)*x+b*eye(2);
[t,v]=eigs((Srlb+Srlb')/2,(Srlw+Srlw')/2,1);

figure(1); clf; hold on; axis([-6 6 -6 6]);
plot(xx(:,1),xx(:,2),'k.');
plot(x(y==1,1),x(y==1,2),'bo');
plot(x(y==2,1),x(y==2,2),'rx');
plot(99*[-t(1) t(1)],99*[-t(2) t(2)], 'k-');
```

FIGURE 35.15

MATLAB code for semisupervised local Fisher discriminant analysis.

MI is actually the *KL divergence* (see Section 14.2) from the joint probability density $p(z, y)$ to the product of marginals $p(z)p(y)$. Therefore, it is always non-negative and is zero if and only if $p(z, y) = p(z)p(y)$, i.e. z and y are *statistically independent* (see Section 5.6). Maximizing MI with respect to $\boldsymbol{T}$ implies maximizing *statistical dependency* between z and y, which is intuitively understandable as supervised dimensionality reduction.

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 density ratio $\frac{p(z,y)}{p(z)p(y)}$ 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 *quadratic mutual information* (QMI) [113]:

$$\text{QMI} = \frac{1}{2} \iint f(z, y)^2 \mathrm{d}z\mathrm{d}y,$$

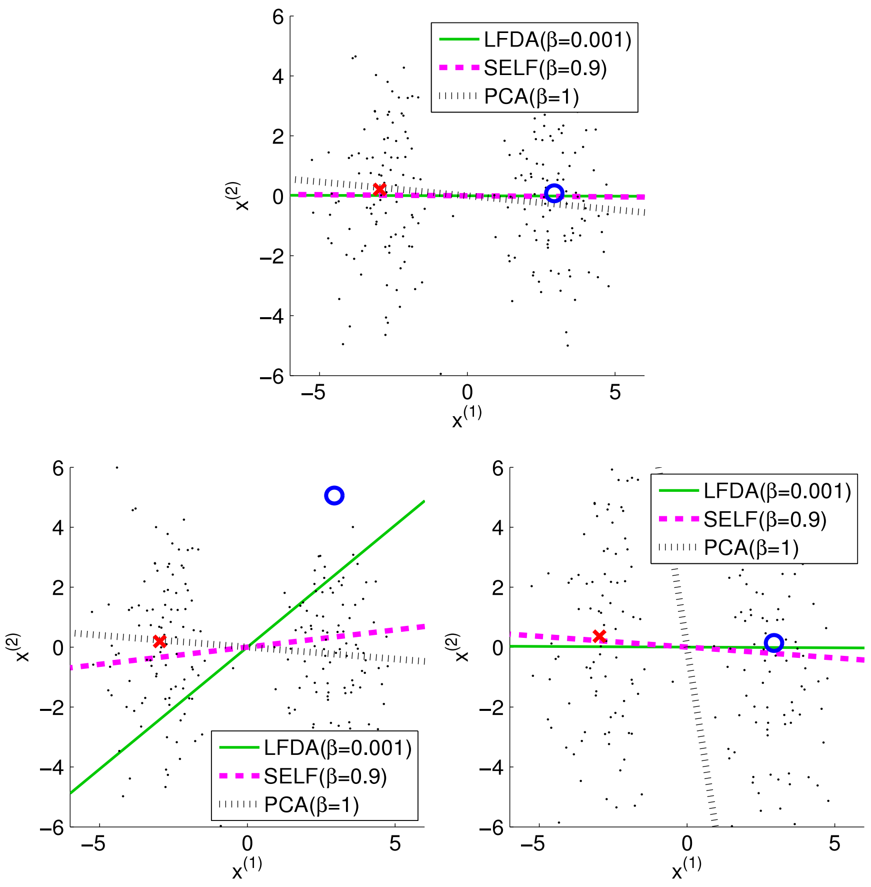


FIGURE 35.16

Examples of semisupervised local Fisher discriminant analysis. Lines denote the found subspaces to which training samples are projected. “LFDA” stands for local Fisher discriminant analysis, “SELF” stands for semisupervised LFDA, and “PCA” stands for principal component analysis.

where $f(z, y)$ is the density difference function defined as

$$f(z, y) = p(z, y) - p(z)p(y).$$

Then a maximizer of QMI with respect to transformation matrix $\boldsymbol{T}$ may be obtained by *gradient ascent*:

$$\boldsymbol{T} \longleftarrow \boldsymbol{T} + \varepsilon \frac{\partial \text{QMI}}{\partial \boldsymbol{T}},$$

where $\varepsilon > 0$ is the step size.

35.4.2 DIRECT DERIVATIVE ESTIMATION

The partial derivative of QMI with respect to $T_{\ell,k}$ can be expressed as

$$\begin{aligned}\frac{\partial \mathrm{QMI}}{\partial T_{\ell,k}} &= \frac{1}{2}\iint \frac{\partial f(z,y)^2}{\partial T_{\ell,k}}\mathrm{d}z\mathrm{d}y = \iint f(z,y)\frac{\partial f(z,y)}{\partial T_{\ell,k}}\mathrm{d}z\mathrm{d}y\\ &= \iint p(z,y)\sum_{\ell'=1}^{m}\frac{\partial f(z,y)}{\partial z^{(\ell')}}\frac{\partial z^{(\ell')}}{\partial T_{\ell,k}}\mathrm{d}z\mathrm{d}y\\ &\quad - \iint p(z)p(y)\sum_{\ell'=1}^{m}\frac{\partial f(z,y)}{\partial z^{(\ell')}}\frac{\partial z^{(\ell')}}{\partial T_{\ell,k}}\mathrm{d}z\mathrm{d}y.\end{aligned}$$

For $\boldsymbol{z} = (z^{(1)},\ldots,z^{(m)})^\top = \boldsymbol{T}\boldsymbol{x}$, the partial derivative $\frac{\partial z^{(\ell')}}{\partial T_{\ell,k}}$ is given by

$$\frac{\partial z^{(\ell')}}{\partial T_{\ell,k}} = \begin{cases} x^{(k)} & (\ell = \ell'),\\ 0 & (\ell \neq \ell').\end{cases}$$

Further approximating the expectations by the sample averages yields

$$\frac{\partial \mathrm{QMI}}{\partial T_{\ell,k}} \approx \frac{1}{n}\sum_{i=1}^{n}\frac{\partial f(z_i,y_i)}{\partial z^{(\ell)}}x_i^{(k)} - \frac{1}{n^2}\sum_{i,i'=1}^{n}\frac{\partial f(z_i,y_{i'})}{\partial z^{(\ell)}}x_i^{(k)}. \tag{35.5}$$

A naive way to approximate $\frac{\partial f(z,y)}{\partial z^{(\ell)}}$ in Eq. (35.5) is to separately estimate the densities $p(z,y)$, $p(z)$, and $p(y)$ from $\{(z_i,y_i)\}_{i=1}^n$, $\{z_i\}_{i=1}^n$, and $\{y_i\}_{i=1}^n$, to plug the estimated densities in $f(z,y)$, and to compute the partial derivative with respect to $z^{(\ell)}$. However, density estimation is a hard statistical estimation problem and thus such a plug-in derivative estimator may not be reliable. Below, a direct estimator of $\frac{\partial f(z,y)}{\partial z^{(\ell)}}$ without density estimation is introduced [108].

Let us employ the following Gaussian kernel model for approximating $\frac{\partial f(z,y)}{\partial z^{(\ell)}}$:

$$g_{\boldsymbol{\alpha}}(z,y) = \sum_{j=1}^{n}\alpha_j \exp\left(-\frac{\|z-z_j\|^2 + (y-y_j)^2}{2h^2}\right).$$

The parameter vector $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_n)^\top$ is learned so that the following squared error is minimized:

$$\begin{aligned}J(\boldsymbol{\alpha}) &= \iint \left(g_{\boldsymbol{\alpha}}(z,y) - \frac{\partial f(z,y)}{\partial z^{(\ell)}}\right)^2 \mathrm{d}z\mathrm{d}y\\ &= \iint g_{\boldsymbol{\alpha}}(z,y)^2\mathrm{d}z\mathrm{d}y - 2\iint g_{\boldsymbol{\alpha}}(z,y)\frac{\partial f(z,y)}{\partial z^{(\ell)}}\mathrm{d}z\mathrm{d}y + C,\end{aligned}$$

```
n=500; x=(rand(n,2)*2-1)*10; y=sin(x(:,1)/10*pi);
y2=y.^2; yy=repmat(y2,1,n)+repmat(y2',n,1)-2*y*y';
e=10; h=1; r=exp(-yy/(2*h)); rr=sum(r)'/(n^2);
t0=randn(2,1); t0=t0/norm(t0); c=pi*h;
for o=1:10000
  z=x*t0; z2=z.^2; zz=repmat(z2,1,n)+repmat(z2',n,1)-2*z*z';
  k=exp(-zz/(2*h)); kz=k.*(repmat(z',[n 1])-repmat(z,[1 n]));
  U=c*exp(-(zz+yy)/(4*h)); v=mean(kz.*r/h,2)-sum(kz,2).*rr/h;
  a=(U+0.1*eye(n))\v; g=(k.*r)*x/n-(k*x).*repmat(rr,[1 2]);
  t=t0+e*g'*a; t=t/norm(t);
  if norm(t-t0)<0.00001, break, end
  t0=t;
end

figure(1); clf; hold on; axis([-10 10 -10 10]); colormap gray
scatter3(x(:,1),x(:,2),y,100,y,'filled'); colorbar;
plot(99*[-t(1) t(1)],99*[-t(2) t(2)],'k-');
```

FIGURE 35.17

MATLAB code for supervised dimensionality reduction based on QMI.

where $C = \iint \left(\frac{\partial f(z,y)}{\partial z^{(\ell)}}\right)^2 \mathrm{d}z\mathrm{d}y$ is a constant independent of parameter $\boldsymbol{\alpha}$ and thus is ignored. Suppose that $g_{\boldsymbol{\alpha}}(z,y)f(z,y) \to 0$ as z and y tend to $\pm\infty$. Then *integration by parts* (4.4) yields

$$\iint g_{\boldsymbol{\alpha}}(z,y)\frac{\partial f(z,y)}{\partial z^{(\ell)}}\mathrm{d}z\mathrm{d}y = -\iint \frac{\partial g_{\boldsymbol{\alpha}}(z,y)}{\partial z^{(\ell)}} f(z,y)\mathrm{d}z\mathrm{d}y.$$

Plugging this into $J(\boldsymbol{\alpha})$, approximating the expectations by the sample averages, and adding the ℓ_2-regularizer result in the following optimization problem:

$$\min_{\boldsymbol{\alpha}} \left[\boldsymbol{\alpha}^\top \boldsymbol{U}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top \widehat{\boldsymbol{v}}_\ell + \lambda\|\boldsymbol{\alpha}\|^2\right],$$

where $\lambda \geq 0$ is the regularization parameter, and $\boldsymbol{U}$ and $\widehat{\boldsymbol{v}}_\ell$ are the $n \times n$ matrix and the n-dimensional vector defined as

$$U_{j,j'} = (\sqrt{\pi}h)^{m+1} \exp\left(-\frac{\|z_j - z_{j'}\|^2 + (y_j - y_{j'})^2}{4h^2}\right),$$

$$\widehat{v}_{\ell,j} = \frac{1}{nh^2}\sum_{i=1}^{n} \exp\left(-\frac{\|z_i - z_j\|^2 + (y_i - y_j)^2}{2h^2}\right)(z_i^{(\ell)} - z_j^{(\ell)})$$
$$- \frac{1}{n^2h^2}\sum_{i,i'=1}^{n} \exp\left(-\frac{\|z_i - z_j\|^2 + (y_{i'} - y_j)^2}{2h^2}\right)(z_i^{(\ell)} - z_j^{(\ell)}).$$

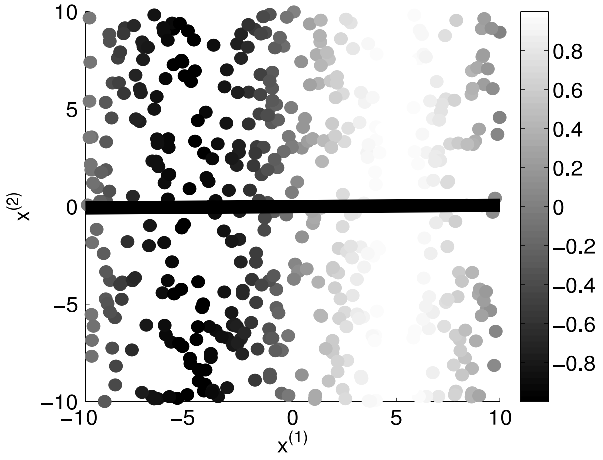


FIGURE 35.18

Example of supervised dimensionality reduction based on QMI. The solid line denotes the found subspace to which training samples are projected.

The above minimizer can be obtained analytically as

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

Then, Eq. (35.5) yields

$$\frac{\partial \mathrm{QMI}}{\partial T_{\ell,k}} \approx \frac{1}{n} \sum_{i=1}^{n} g_{\widehat{\boldsymbol{\alpha}}_\ell}(z_i, y_i) x_i^{(k)} - \frac{1}{n^2} \sum_{i,i'=1}^{n} g_{\widehat{\boldsymbol{\alpha}}_\ell}(z_i, y_{i'}) x_i^{(k)}.$$

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 (or the misclassification error if a classifier is applied after dimensionality reduction).

A MATLAB code for QMI-based supervised dimensionality reduction is provided in Fig. 35.17, and its behavior is illustrated in Fig. 35.18. This shows that a subspace in the input space that strongly depends on output is obtained.

If QMI between $\boldsymbol{x}$ and its projection $\boldsymbol{z} = \boldsymbol{T}\boldsymbol{x}$ is considered, the QMI-based dimensionality reduction method can actually be applied in unsupervised dimensionality reduction. A MATLAB code for unsupervised dimensionality reduction based on QMI is provided in Fig. 35.19, and its behavior is illustrated in Fig. 35.20. This shows that similar results to locality preserving projection are obtained *without* explicitly using sample similarity.

Note that the sufficient dimensionality reduction method introduced above can also be applied to classification.

```
n=100; x=[2*randn(n,1) randn(n,1)];
%x=[2*randn(n,1) 2*round(rand(n,1))-1+randn(n,1)/3];
x=x-repmat(mean(x),[n,1]); x2=sum(x.^2,2);
yy=repmat(x2,1,n)+repmat(x2',n,1)-2*x*x';

e=10; h=1; l=exp(-yy/(2*h)); ll=sum(l)'/(n^2);
t0=randn(2,1); t0=t0/norm(t0); c=pi*h;
for o=1:10000
  z=x*t0; z2=z.^2; zz=repmat(z2,1,n)+repmat(z2',n,1)-2*z*z';
  k=exp(-zz/(2*h)); kz=k.*(repmat(z',[n 1])-repmat(z,[1 n]));
  U=c*exp(-(zz+yy)/(4*h)); v=mean(kz.*l/h,2)-sum(kz,2).*ll/h;
  a=(U+0.1*eye(n))\v; g=(k.*l)*x/n-(k*x).*repmat(ll,[1 2]);
  t=t0+e*g'*a; t=t/norm(t);
  if norm(t-t0)<0.00001, break, end
  t0=t;
end

figure(1); clf; hold on; axis([-6 6 -6 6]);
plot(x(:,1),x(:,2),'rx','LineWidth',2,'MarkerSize',12);
plot(9*[-t(1) t(1)],9*[-t(2) t(2)],'k-');
```

FIGURE 35.19

MATLAB code for unsupervised dimensionality reduction based on QMI.

35.5 MATRIX IMPUTATION

Let us consider a matrix $\boldsymbol{X} \in \mathbb{R}^{d_1 \times d_2}$ with *missing entries*, i.e. X_{k_1,k_2} is observed only for $(k_1,k_2) \in \mathcal{K}$ and X_{k_1,k_2} is missing for all $(k_1,k_2) \notin \mathcal{K}$. The objective of *matrix imputation* is to fill the missing entries from observed entries. Such an imputation problem arises, for example, in *recommender systems* [83], where entry X_{k_1,k_2} corresponds to the rating of item k_1 by user k_2. Given a subset $\mathcal{K}$ of ratings of items by users, a recommender system finds the items that are expected to be most favored by a target user.

Naive approaches to matrix imputation would be to pad the missing entries with zeros or the average of observed entries. However, such naive methods are not necessarily useful in practice. The idea for imputing the missing element introduced here is to approximate the matrix $\boldsymbol{X}$ with a *low-rank* matrix.

A naive implementation of this idea is to assume that $\boldsymbol{X}$ can be decomposed into the product of $\boldsymbol{U} \in \mathbb{R}^{d_1 \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{r \times d_2}$:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V} \in \mathbb{R}^{d_1 \times d_2},$$

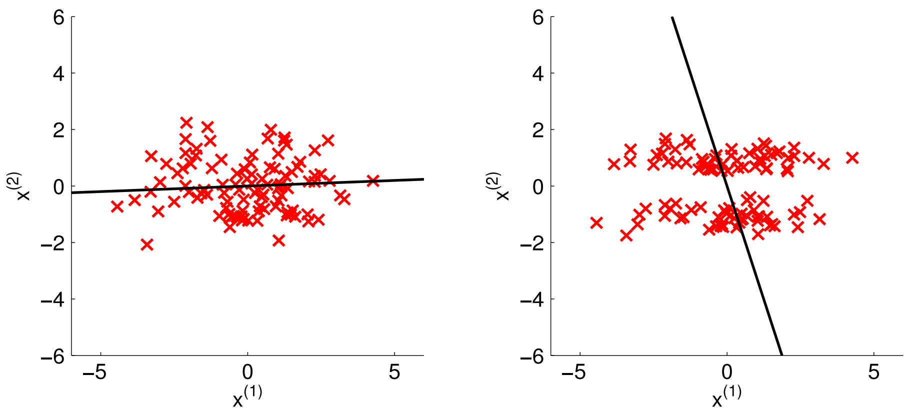


FIGURE 35.20

Example of unsupervised dimensionality reduction based on QMI. The solid line denotes the found subspace to which training samples are projected.

```
d1=20; d2=10; n=100; K=(reshape(randperm(d1*d2),[d1,d2])<=n);
r=3; X=randn(d1,r)*randn(r,d2); T0=randn(d1,d2); e=0.1; l=1;
for o=1:1000
  [U,S,V]=svd(T0-e*((T0.*K)-(X.*K)),'econ');
  S=diag(max(0,diag(S)-e*l)); T=U*S*V';
  if norm(T-T0)<0.001, break, end
  T0=T;
end
figure(1); imagesc(X.*K); figure(2); imagesc(T);
```

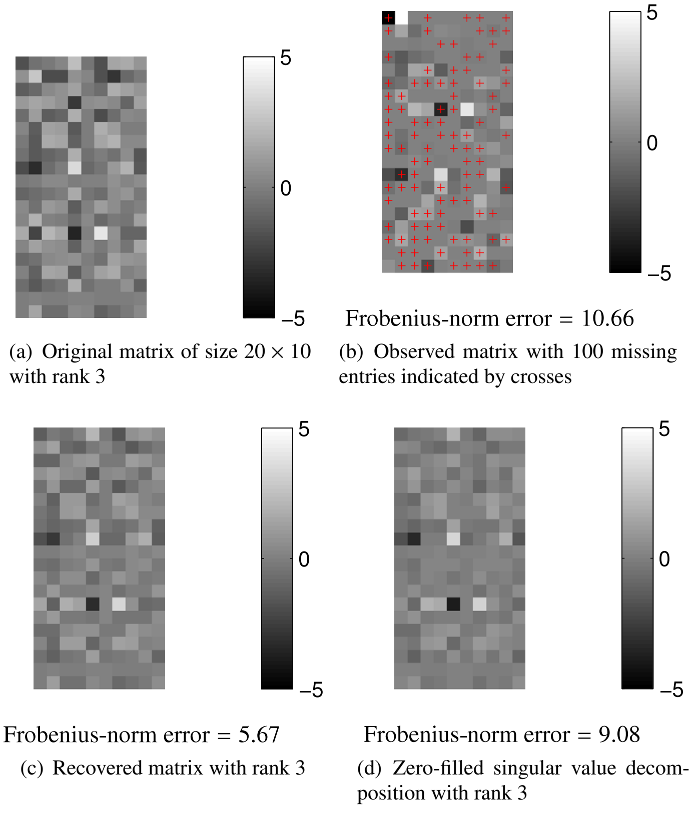
FIGURE 35.21

MATLAB code for unsupervised matrix imputation.

where r is the rank of $\boldsymbol{X}$. Then the factorized matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are estimated from the observed elements, i.e. for $\boldsymbol{\Theta} = \boldsymbol{U}\boldsymbol{V}$, the following optimization problem is solved:

$$\min_{\boldsymbol{U}\in\mathbb{R}^{d_1\times r},\boldsymbol{V}\in\mathbb{R}^{r\times d_2}} \frac{1}{2} \sum_{(k_1,k_2)\in\mathcal{K}} \left(X_{k_1,k_2} - \Theta_{k_1,k_2}\right)^2.$$

Although this minimization problem may be naively solved by gradient descent, finding the global minimizer is not straightforward in practice due to its nonconvexity.



(a) Original matrix of size 20 × 10 with rank 3

(b) Observed matrix with 100 missing entries indicated by crosses

(c) Recovered matrix with rank 3

(d) Zero-filled singular value decomposition with rank 3

FIGURE 35.22

Example of unsupervised matrix imputation. The gray level indicates the value of each entry in [−5, 5].

To overcome the nonconvexity, let us employ regularization with the *trace norm* $\|\boldsymbol{\Theta}\|_{\mathrm{tr}}$ (see Fig. 24.10), which tends to produce a low-rank solution:

$$\|\boldsymbol{\Theta}\|_{\mathrm{tr}} = \sum_{k=1}^{\min(d_1,d_2)} \sigma_k,$$

where σ_k is a *singular value* of $\boldsymbol{\Theta}$ (see Fig. 22.2). Then fitting $\boldsymbol{\Theta}$ to $\boldsymbol{X}$ with trace norm regularization allows us to recover the missing entries:

$$\min_{\boldsymbol{\Theta}\in\mathbb{R}^{d_1\times d_2}} \frac{1}{2} \sum_{(k_1,k_2)\in\mathcal{K}} \left(X_{k_1,k_2} - \Theta_{k_1,k_2}\right)^2 + \lambda\|\boldsymbol{\Theta}\|_{\mathrm{tr}},$$

where $\lambda > 0$ is the regularization parameter. Since this is a convex optimization problem, the global optimal solution can be easily obtained, for example, by the *proximal gradient method* explained in Fig. 34.8.

A MATLAB code for unsupervised matrix imputation is provided in Fig. 35.21, and its behavior is illustrated in Fig. 35.22. This shows that missing entries can be recovered by trace norm regularization better than singular value decomposition after filling the missing entries with zeros.