6 *Dimensionality Reduction*

The complexity of any classifier or regressor depends on the number of inputs. This determines both the time and space complexity and the necessary number of training examples to train such a classifier or regressor. In this chapter, we discuss feature selection methods that choose a subset of important features pruning the rest and feature extraction methods that form fewer, new features from the original inputs.

6.1 Introduction

In an application, whether it is classification or regression, observation data that we believe contain information are taken as inputs and fed to the system for decision making. Ideally, we should not need feature selection or extraction as a separate process; the classifier (or regressor) should be able to use whichever features are necessary, discarding the irrelevant. However, there are several reasons why we are interested in reducing dimensionality as a separate preprocessing step:

- In most learning algorithms, the complexity depends on the number of input dimensions, d, as well as on the size of the data sample, N, and for reduced memory and computation, we are interested in reducing the dimensionality of the problem. Decreasing d also decreases the complexity of the inference algorithm during testing.

- When an input is decided to be unnecessary, we save the cost of extracting it.

- Simpler models are more robust on small datasets. Simpler models have less variance, that is, they vary less depending on the particulars of a sample, including noise, outliers, and so forth.

- When data can be explained with fewer features, we get a better idea about the process that underlies the data and this allows knowledge extraction.

- When data can be represented in a few dimensions without loss of information, it can be plotted and analyzed visually for structure and outliers.

FEATURE SELECTION

There are two main methods for reducing dimensionality: feature selection and feature extraction. In *feature selection*, we are interested in finding k of the d dimensions that give us the most information and we discard the other $(d - k)$ dimensions. We are going to discuss *subset selection* as a feature selection method.

FEATURE EXTRACTION

In *feature extraction*, we are interested in finding a new set of k dimensions that are combinations of the original d dimensions. These methods may be supervised or unsupervised depending on whether or not they use the output information. The best known and most widely used feature extraction methods are *Principal Components Analysis* (PCA) and *Linear Discriminant Analysis* (LDA), which are both linear projection methods, unsupervised and supervised respectively. PCA bears much similarity to two other unsupervised linear projection methods, which we also discuss—namely, *Factor Analysis* (FA) and *Multidimensional Scaling* (MDS). As examples of *nonlinear* dimensionality reduction, we are going to see *Isometric feature mapping* (Isomap) and *Locally Linear Embedding* (LLE).

6.2 Subset Selection

SUBSET SELECTION

In *subset selection*, we are interested in finding the best subset of the set of features. The best subset contains the least number of dimensions that most contribute to accuracy. We discard the remaining, unimportant dimensions. Using a suitable error function, this can be used in both regression and classification problems. There are 2^d possible subsets of d variables, but we cannot test for all of them unless d is small and we employ heuristics to get a reasonable (but not optimal) solution in reasonable (polynomial) time.

FORWARD SELECTION

There are two approaches: In *forward selection*, we start with no variables and add them one by one, at each step adding the one that decreases the error the most, until any further addition does not decrease the error (or decreases it only sightly). In *backward selection*, we start with all variables and remove them one by one, at each step removing the one that decreases the error the most (or increases it only slightly), until any further removal increases the error significantly. In either case, checking the error should be done on a validation set distinct from the training set because we want to test the generalization accuracy. With more features, generally we have lower training error, but not necessarily lower validation error.

BACKWARD SELECTION

Let us denote by F, a feature set of input dimensions, $x_i, i = 1, \ldots, d$. $E(F)$ denotes the error incurred on the validation sample when only the inputs in F are used. Depending on the application, the error is either the mean square error or misclassification error.

In *sequential forward selection*, we start with no features: $F = \emptyset$. At each step, for all possible x_i, we train our model on the training set and calculate $E(F \cup x_i)$ on the validation set. Then, we choose that input x_j that causes the least error

$$j = \arg\min_i E(F \cup x_i) \tag{6.1}$$

and we

$$\text{add } x_j \text{ to } F \text{ if } E(F \cup x_j) < E(F) \tag{6.2}$$

We stop if adding any feature does not decrease E. We may even decide to stop earlier if the decrease in error is too small, where there is a user-defined threshold that depends on the application constraints, trading off the importance of error and complexity. Adding another feature introduces the cost of observing the feature, as well as making the classifier/regressor more complex.

This process may be costly because to decrease the dimensions from d to k, we need to train and test the system $d + (d-1) + (d-2) + \cdots + (d-k)$ times, which is $\mathcal{O}(d^2)$. This is a local search procedure and does not guarantee finding the optimal subset, namely, the minimal subset causing the smallest error. For example, x_i and x_j by themselves may not be good but together may decrease the error a lot, but because this algorithm is greedy and adds attributes one by one, it may not be able to detect this. It is possible to generalize and add multiple features at a time, instead of a single one, at the expense of more computation. We can also backtrack and check which previously added feature can be removed after a current addition, thereby increasing the search space, but this increases complexity. In *floating search* methods (Pudil, Novovičová, and Kittler 1994), the number of added features and removed features can also change at each step.

FLOATING SEARCH

In *sequential backward selection*, we start with F containing all features and do a similar process except that we remove one attribute from F as opposed to adding to it, and we remove the one that causes the least error

$$j = \arg\min_i E(F - x_i) \tag{6.3}$$

and we

$$\text{remove } x_j \text{ from } F \text{ if } E(F - x_j) < E(F) \tag{6.4}$$

We stop if removing a feature does not decrease the error. To decrease complexity, we may decide to remove a feature if its removal causes only a slight increase in error.

All the variants possible for forward search are also possible for backward search. The complexity of backward search has the same order of complexity as forward search, except that training a system with more features is more costly than training a system with fewer features, and forward search may be preferable especially if we expect many useless features.

Subset selection is supervised in that outputs are used by the regressor or classifier to calculate the error, but it can be used with any regression or classification method. In the particular case of multivariate normals for classification, remember that if the original d-dimensional class densities are multivariate normal, then any subset is also multivariate normal and parametric classification can still be used with the advantage of $k \times k$ covariance matrices instead of $d \times d$.

In an application like face recognition, feature selection is not a good method for dimensionality reduction because individual pixels by themselves do not carry much discriminative information; it is the combination of values of several pixels together that carry information about the face identity. This is done by feature extraction methods that we discuss next.

6.3 Principal Components Analysis

In projection methods, we are interested in finding a mapping from the inputs in the original d-dimensional space to a new $(k < d)$-dimensional space, with minimum loss of information. The projection of $\boldsymbol{x}$ on the direction of $\boldsymbol{w}$ is

(6.5) $$z = \boldsymbol{w}^T \boldsymbol{x}$$

PRINCIPAL COMPONENTS ANALYSIS

Principal components analysis (PCA) is an unsupervised method in that it does not use the output information; the criterion to be maximized is the variance. The principal component is $\boldsymbol{w}_1$ such that the sample, after projection on to $\boldsymbol{w}_1$, is most spread out so that the difference between the sample points becomes most apparent. For a unique solution and to make the direction the important factor, we require $\|\boldsymbol{w}_1\| = 1$. We know from equation 5.14 that if $z_1 = \boldsymbol{w}_1^T \boldsymbol{x}$ with $\text{Cov}(\boldsymbol{x}) = \boldsymbol{\Sigma}$, then

$$\text{Var}(z_1) = \boldsymbol{w}_1^T \boldsymbol{\Sigma} \boldsymbol{w}_1$$

We seek $\boldsymbol{w}_1$ such that $\text{Var}(z_1)$ is maximized subject to the constraint that $\boldsymbol{w}_1^T \boldsymbol{w}_1 = 1$. Writing this as a Lagrange problem, we have

(6.6) $$\max_{\boldsymbol{w}_1} \boldsymbol{w}_1^T \boldsymbol{\Sigma} \boldsymbol{w}_1 - \alpha(\boldsymbol{w}_1^T \boldsymbol{w}_1 - 1)$$

Taking the derivative with respect to $\boldsymbol{w}_1$ and setting it equal to 0, we have

$$2\boldsymbol{\Sigma} \boldsymbol{w}_1 - 2\alpha \boldsymbol{w}_1 = 0, \text{ and therefore } \boldsymbol{\Sigma} \boldsymbol{w}_1 = \alpha \boldsymbol{w}_1$$

which holds if $\boldsymbol{w}_1$ is an eigenvector of $\boldsymbol{\Sigma}$ and α the corresponding eigenvalue. Because we want to maximize

$$\boldsymbol{w}_1^T \boldsymbol{\Sigma} \boldsymbol{w}_1 = \alpha \boldsymbol{w}_1^T \boldsymbol{w}_1 = \alpha$$

we choose the eigenvector with the largest eigenvalue for the variance to be maximum. Therefore the principal component is the eigenvector of the covariance matrix of the input sample with the largest eigenvalue, $\lambda_1 = \alpha$.

The second principal component, $\boldsymbol{w}_2$, should also maximize variance, be of unit length, and be orthogonal to $\boldsymbol{w}_1$. This latter requirement is so that after projection $z_2 = \boldsymbol{w}_2^T \boldsymbol{x}$ is uncorrelated with z_1. For the second principal component, we have

(6.7) $$\max_{\boldsymbol{w}_2} \boldsymbol{w}_2^T \boldsymbol{\Sigma} \boldsymbol{w}_2 - \alpha(\boldsymbol{w}_2^T \boldsymbol{w}_2 - 1) - \beta(\boldsymbol{w}_2^T \boldsymbol{w}_1 - 0)$$

Taking the derivative with respect to $\boldsymbol{w}_2$ and setting it equal to 0, we have

$$2\boldsymbol{\Sigma}\boldsymbol{w}_2 - 2\alpha\boldsymbol{w}_2 - \beta\boldsymbol{w}_1 = 0 \tag{6.8}$$

Premultiply by $\boldsymbol{w}_1^T$ and we get

$$2\boldsymbol{w}_1^T\boldsymbol{\Sigma}\boldsymbol{w}_2 - 2\alpha\boldsymbol{w}_1^T\boldsymbol{w}_2 - \beta\boldsymbol{w}_1^T\boldsymbol{w}_1 = 0$$

Note that $\boldsymbol{w}_1^T\boldsymbol{w}_2 = 0$. $\boldsymbol{w}_1^T\boldsymbol{\Sigma}\boldsymbol{w}_2$ is a scalar, equal to its transpose $\boldsymbol{w}_2^T\boldsymbol{\Sigma}\boldsymbol{w}_1$ where, because $\boldsymbol{w}_1$ is the leading eigenvector of $\boldsymbol{\Sigma}$, $\boldsymbol{\Sigma}\boldsymbol{w}_1 = \lambda_1\boldsymbol{w}_1$. Therefore

$$\boldsymbol{w}_1^T\boldsymbol{\Sigma}\boldsymbol{w}_2 = \boldsymbol{w}_2^T\boldsymbol{\Sigma}\boldsymbol{w}_1 = \lambda_1\boldsymbol{w}_2^T\boldsymbol{w}_1 = 0$$

Then $\beta = 0$ and equation 6.8 reduces to

$$\boldsymbol{\Sigma}\boldsymbol{w}_2 = \alpha\boldsymbol{w}_2$$

which implies that $\boldsymbol{w}_2$ should be the eigenvector of $\boldsymbol{\Sigma}$ with the second largest eigenvalue, $\lambda_2 = \alpha$. Similarly, we can show that the other dimensions are given by the eigenvectors with decreasing eigenvalues.

Because $\boldsymbol{\Sigma}$ is symmetric, for two different eigenvalues, the eigenvectors are orthogonal. If $\boldsymbol{\Sigma}$ is positive definite ($\boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x} > 0$, for all nonnull $\boldsymbol{x}$), then all its eigenvalues are positive. If $\boldsymbol{\Sigma}$ is singular, then its rank, the effective dimensionality, is k with $k < d$ and $\lambda_i, i = k+1, \ldots, d$ are 0 (λ_i are sorted in descending order). The k eigenvectors with nonzero eigenvalues are the dimensions of the reduced space. The first eigenvector (the one with the largest eigenvalue), $\boldsymbol{w}_1$, namely, the principal component, explains the largest part of the variance; the second explains the second largest; and so on.

We define

$$\boldsymbol{z} = \mathbf{W}^T(\boldsymbol{x} - \boldsymbol{m}) \tag{6.9}$$

where the k columns of $\mathbf{W}$ are the k leading eigenvectors of $\mathbf{S}$, the estimator to $\boldsymbol{\Sigma}$. We subtract the sample mean $\boldsymbol{m}$ from $\boldsymbol{x}$ before projection to center the data on the origin. After this linear transformation, we get to a k-dimensional space whose dimensions are the eigenvectors, and the variances over these new dimensions are equal to the eigenvalues (see figure 6.1). To normalize variances, we can divide by the square roots of the eigenvalues.

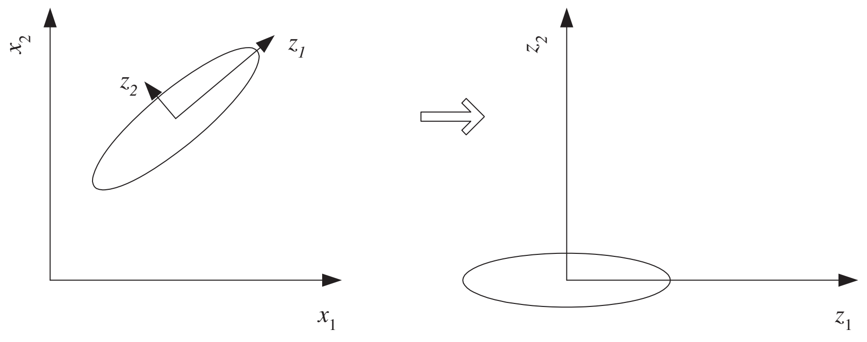

Figure 6.1 Principal components analysis centers the sample and then rotates the axes to line up with the directions of highest variance. If the variance on z_2 is too small, it can be ignored and we have dimensionality reduction from two to one.

Let us see another derivation: We want to find a matrix $\mathbf{W}$ such that when we have $\boldsymbol{z} = \mathbf{W}^T\boldsymbol{x}$ (assume without loss of generality that $\boldsymbol{x}$ are already centered), we will get $\text{Cov}(\boldsymbol{z}) = \mathbf{D}'$ where $\mathbf{D}'$ is any diagonal matrix; that is, we would like to get uncorrelated z_i.

If we form a $(d \times d)$ matrix $\mathbf{C}$ whose ith column is the normalized eigenvector $\boldsymbol{c}_i$ of $\mathbf{S}$, then $\mathbf{C}^T\mathbf{C} = \mathbf{I}$ and

$$\begin{aligned}
\mathbf{S} &= \mathbf{SCC}^T \\
&= \mathbf{S}(\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_d)\mathbf{C}^T \\
&= (\mathbf{S}\boldsymbol{c}_1, \mathbf{S}\boldsymbol{c}_2, \ldots, \mathbf{S}\boldsymbol{c}_d)C^T \\
&= (\lambda_1\boldsymbol{c}_1, \lambda_2\boldsymbol{c}_2, \ldots, \lambda_d\boldsymbol{c}_d)\mathbf{C}^T \\
&= \lambda_1\boldsymbol{c}_1\boldsymbol{c}_1^T + \cdots + \lambda_d\boldsymbol{c}_d\boldsymbol{c}_d^T \\
&= \mathbf{CDC}^T \qquad (6.10)
\end{aligned}$$

SPECTRAL DECOMPOSITION

where $\mathbf{D}$ is a diagonal matrix whose diagonal elements are the eigenvalues, $\lambda_1, \ldots, \lambda_d$. This is called the *spectral decomposition* of $\mathbf{S}$. Since $\mathbf{C}$ is orthogonal and $\mathbf{CC}^T = \mathbf{C}^T\mathbf{C} = \mathbf{I}$, we can multiply on the left by $\mathbf{C}^T$ and on the right by $\mathbf{C}$ to obtain

$$\mathbf{C}^T\mathbf{SC} = \mathbf{D} \qquad (6.11)$$

We know that if $\boldsymbol{z} = \mathbf{W}^T\boldsymbol{x}$, then $\text{Cov}(\boldsymbol{z}) = \mathbf{W}^T\mathbf{SW}$, which we would like to be equal to a diagonal matrix. Then from equation 6.11, we see that we can set $\mathbf{W} = \mathbf{C}$.

Let us see an example to get some intuition (Rencher 1995): Assume we are given a class of students with grades on five courses and we want to order these students. That is, we want to project the data onto one dimension, such that the difference between the data points become most apparent. We can use PCA. The eigenvector with the highest eigenvalue is the direction that has the highest variance, that is, the direction on which the students are most spread out. This works better than taking the average because we take into account correlations and differences in variances.

In practice even if all eigenvalues are greater than 0, if $|\mathbf{S}|$ is small, remembering that $|\mathbf{S}| = \prod_{i=1}^{d} \lambda_i$, we understand that some eigenvalues have little contribution to variance and may be discarded. Then, we take into account the leading k components that explain more than, for example, 90 percent, of the variance. When λ_i are sorted in descending order, the *proportion of variance* explained by the k principal components is

PROPORTION OF VARIANCE

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}$$

If the dimensions are highly correlated, there will be a small number of eigenvectors with large eigenvalues and k will be much smaller than d and a large reduction in dimensionality may be attained. This is typically the case in many image and speech processing tasks where nearby inputs (in space or time) are highly correlated. If the dimensions are not correlated, k will be as large as d and there is no gain through PCA.

SCREE GRAPH

Scree graph is the plot of variance explained as a function of the number of eigenvectors kept (see figure 6.2). By visually analyzing it, one can also decide on k. At the "elbow," adding another eigenvector does not significantly increase the variance explained.

Another possibility is to ignore the eigenvectors whose eigenvalues are less than the average input variance. Given that $\sum_i \lambda_i = \sum_i s_i^2$ (equal to the *trace* of $\mathbf{S}$, denoted as $\text{tr}(\mathbf{S})$), the average eigenvalue is equal to the average input variance. When we keep only the eigenvectors with eigenvalues greater than the average eigenvalue, we keep only those that have variance higher than the average input variance.

If the variances of the original x_i dimensions vary considerably, they affect the direction of the principal components more than the correlations, so a common procedure is to preprocess the data so that each dimension has mean 0 and unit variance, before using PCA. Or, one may

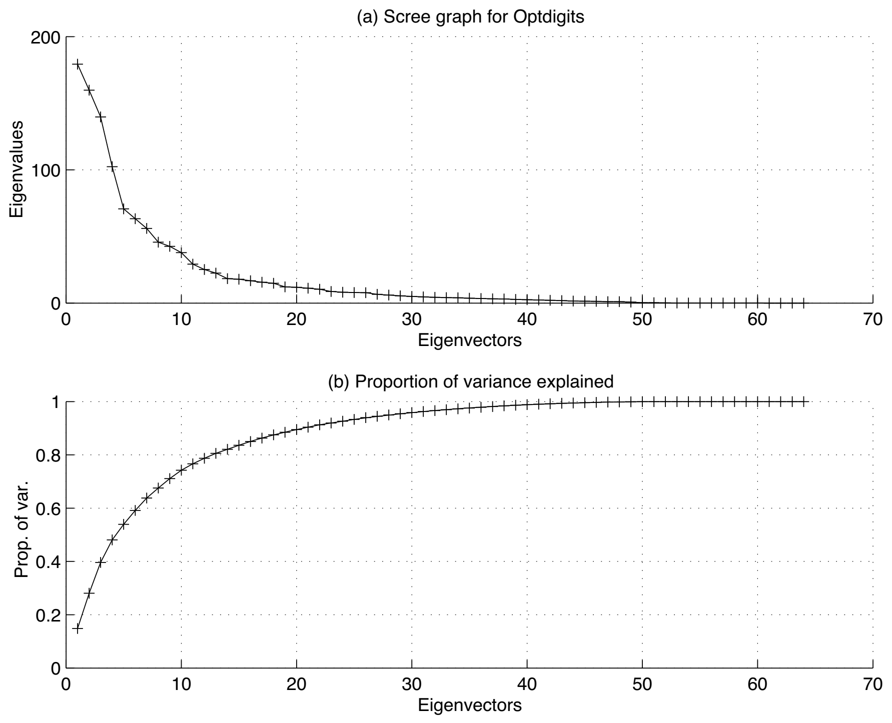

Figure 6.2 (a) Scree graph. (b) Proportion of variance explained is given for the Optdigits dataset from the UCI Repository. This is a handwritten digit dataset with ten classes and sixty-four dimensional inputs. The first twenty eigenvectors explain 90 percent of the variance.

use the eigenvectors of the correlation matrix, **R**, instead of the covariance matrix, **S**, for the correlations to be effective and not the individual variances.

PCA explains variance and is sensitive to outliers: A few points distant from the center would have a large effect on the variances and thus the eigenvectors. *Robust estimation* methods allow calculating parameters in the presence of outliers. A simple method is to calculate the Mahalanobis distance of the data points, discarding the isolated data points that are far away.

If the first two principal components explain a large percentage of the variance, we can do *visual analysis*: We can plot the data in this two di-

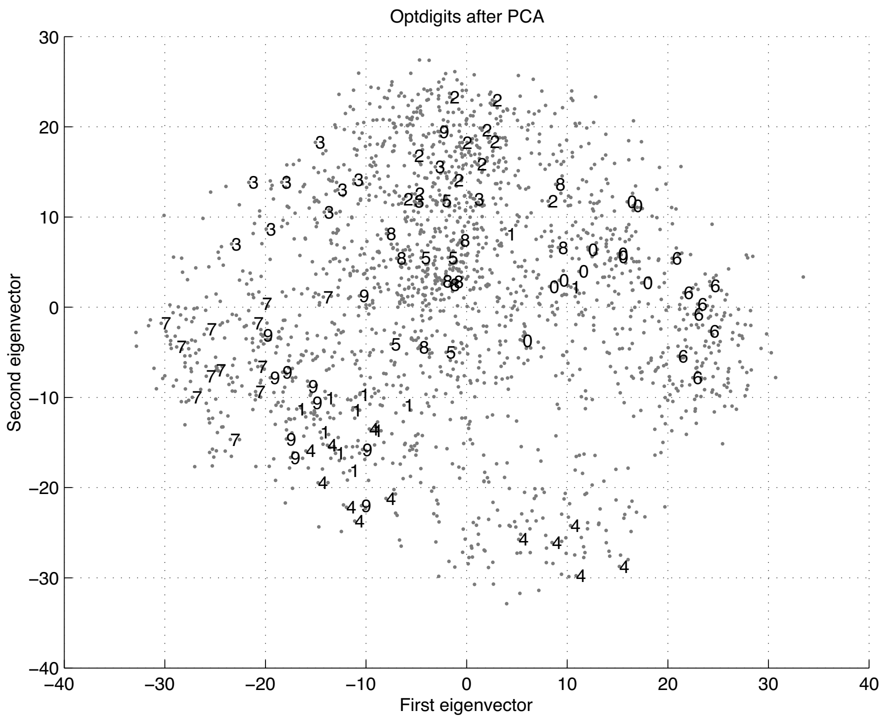

Figure 6.3 Optdigits data plotted in the space of two principal components. Only the labels of a hundred data points are shown to minimize the ink-to-noise ratio.

mensional space (figure 6.3) and search visually for structure, groups, outliers, normality, and so forth. This plot gives a better pictorial description of the sample than a plot of any two of the original variables. By looking at the dimensions of the principal components, we can also try to recover meaningful underlying variables that describe the data. For example, in image applications where the inputs are images, the eigenvectors can also be displayed as images and can be seen as templates for important features; they are typically named "*eigenfaces*," "*eigendigits*," and so forth (Turk and Pentland 1991).

EIGENFACES
EIGENDIGITS

When d is large, calculating, storing, and processing $\mathbf{S}$ may be tedious. It is possible to calculate the eigenvectors and eigenvalues directly from data without explicitly calculating the covariance matrix (Chatfield and

Collins 1980).

We know from equation 5.15 that if $\boldsymbol{x} \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then after projection $\mathbf{W}^T\boldsymbol{x} \sim \mathcal{N}_k(\mathbf{W}^T\boldsymbol{\mu}, \mathbf{W}^T\boldsymbol{\Sigma}\mathbf{W})$. If the sample contains d-variate normals, then it projects to k-variate normals allowing us to do parametric discrimination in this hopefully much lower dimensional space. Because z_j are uncorrelated, the new covariance matrices will be diagonal, and if they are normalized to have unit variance, Euclidean distance can be used in this new space, leading to a simple classifier.

Instance $\boldsymbol{x}^t$ is projected to the z-space as

$$\boldsymbol{z}^t = \mathbf{W}^T(\boldsymbol{x}^t - \boldsymbol{\mu})$$

When $\mathbf{W}$ is an orthogonal matrix such that $\mathbf{W}\mathbf{W}^T = \mathbf{I}$, it can be backprojected to the original space as

$$\hat{\boldsymbol{x}}^t = \mathbf{W}\boldsymbol{z}^t + \boldsymbol{\mu}$$

$\hat{\boldsymbol{x}}^t$ is the reconstruction of $\boldsymbol{x}^t$ from its representation in the z-space. It is known that among all orthogonal linear projections, PCA minimizes the *reconstruction error*, which is the distance between the instance and its reconstruction from the lower dimensional space:

RECONSTRUCTION ERROR

$$\sum_t \|\hat{\boldsymbol{x}}^t - \boldsymbol{x}\|^2 \tag{6.12}$$

The reconstruction error depends on how many of the leading components are taken into account. In a visual recognition application—for example, face recognition—displaying $\hat{\boldsymbol{x}}^t$ allows a visual check for information loss during PCA.

PCA is unsupervised and does not use output information. It is a one-group procedure. However, in the case of classification, there are multiple groups. *Karhunen-Loève expansion* allows using class information; for example, instead of using the covariance matrix of the whole sample, we can estimate separate class covariance matrices, take their average (weighted by the priors) as the covariance matrix, and use its eigenvectors.

KARHUNEN-LOÈVE EXPANSION

In *common principal components* (Flury 1988), we assume that the principal components are the same for each class whereas the variances of these components differ for different classes:

COMMON PRINCIPAL COMPONENTS

$$\mathbf{S}_i = \mathbf{C}\mathbf{D}_i\mathbf{C}^T$$

This allows pooling data and is a regularization method whose complexity is less than that of a common covariance matrix for all classes,

FLEXIBLE DISCRIMINANT ANALYSIS

while still allowing differentiation of $\mathbf{S}_i$. A related approach is *flexible discriminant analysis* (Hastie, Tibshirani, and Buja 1994), which does a linear projection to a lower-dimensional space where all features are uncorrelated and then uses a minimum distance classifier.

6.4 Factor Analysis

In PCA, from the original dimensions $x_i, i = 1, \ldots, d$, we form a new set of variables $\boldsymbol{z}$ that are linear combinations of x_i:

$$\boldsymbol{z} = \mathbf{W}^T(\boldsymbol{x} - \boldsymbol{\mu})$$

FACTOR ANALYSIS

LATENT FACTORS

In *factor analysis* (FA), we assume that there is a set of unobservable, *latent factors* $z_j, j = 1, \ldots, k$, which when acting in combination *generate* $\boldsymbol{x}$. Thus the direction is opposite that of PCA (see figure 6.4). The goal is to characterize the dependency among the observed variables by means of a smaller number of factors.

Suppose there is a group of variables that have high correlation among themselves and low correlation with all the other variables. Then there may be a single underlying factor that gave rise to these variables. If the other variables can be similarly grouped into subsets, then a few factors can represent these groups of variables. Though factor analysis always partitions the variables into factor clusters, whether the factors mean anything, or really exist, is open to question.

FA, like PCA, is a one-group procedure and is unsupervised. The aim is to model the data in a smaller dimensional space without loss of information. In FA, this is measured as the correlation between variables.

As in PCA, we have a sample $\mathcal{X} = \{\boldsymbol{x}^t\}_t$ drawn from some unknown probability density with $E[\boldsymbol{x}] = \boldsymbol{\mu}$ and $\mathrm{Cov}(\boldsymbol{x}) = \boldsymbol{\Sigma}$. We assume that the factors are unit normals, $E[z_j] = 0, \mathrm{Var}(z_j) = 1$, and are uncorrelated, $\mathrm{Cov}(z_i, z_j) = 0, i \neq j$. To explain what is not explained by the factors, there is an added source for each input which we denote by ϵ_i. It is assumed to be zero-mean, $E[\epsilon_i] = 0$, and have some unknown variance, $\mathrm{Var}(\epsilon_i) = \psi_i$. These specific sources are uncorrelated among themselves, $\mathrm{Cov}(\epsilon_i, \epsilon_j) = 0, i \neq j$, and are also uncorrelated with the factors, $\mathrm{Cov}(\epsilon_i, z_j) = 0, \forall i, j$.

FA assumes that each input dimension, $x_i, i = 1, \ldots, d$, can be written as a weighted sum of the $k < d$ factors, $z_j, j = 1, \ldots, k$, plus the residual

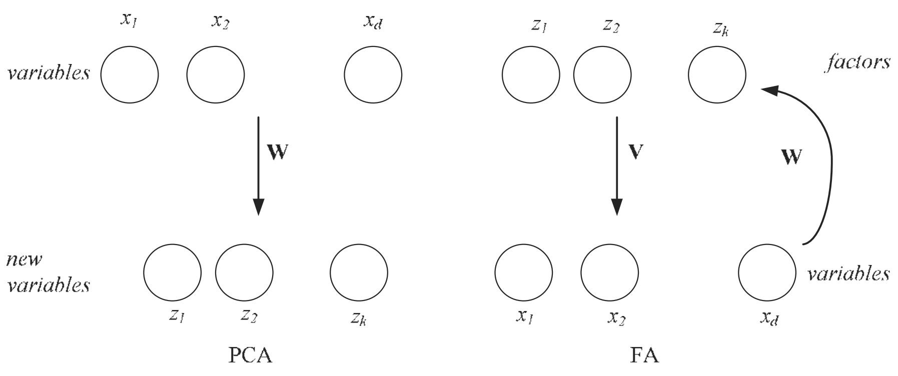

Figure 6.4 Principal components analysis generates new variables that are linear combinations of the original input variables. In factor analysis, however, we posit that there are factors that when linearly combined generate the input variables.

term (see figure 6.5):

$$x_i - \mu_i = v_{i1}z_1 + v_{i2}z_2 + \cdots + v_{ik}z_k + \epsilon_i, \forall i = 1, \ldots, d$$

$$x_i - \mu_i = \sum_{j=1}^{k} v_{ij}z_j + \epsilon_i \tag{6.13}$$

This can be written in vector-matrix form as

$$\boldsymbol{x} - \boldsymbol{\mu} = \mathbf{V}\boldsymbol{z} + \boldsymbol{\epsilon} \tag{6.14}$$

where $\mathbf{V}$ is the $d \times k$ matrix of weights, called *factor loadings*. From now on, we are going to assume that $\boldsymbol{\mu} = \mathbf{0}$ without loss of generality; we can always add $\boldsymbol{\mu}$ after projection. Given that $\text{Var}(z_j) = 1$ and $\text{Var}(\epsilon_i) = \psi_i$

$$\text{Var}(x_i) = v_{i1}^2 + v_{i2}^2 + \cdots + v_{ik}^2 + \psi_i \tag{6.15}$$

$\sum_{j=1}^{k} v_{ij}^2$ is the part of the variance explained by the common factors and ψ_i is the variance specific to x_i.

In vector-matrix form, we have

$$\begin{aligned} \mathbf{\Sigma} = \text{Cov}(\boldsymbol{x}) &= \text{Cov}(\mathbf{V}\boldsymbol{z} + \boldsymbol{\epsilon}) \quad (6.16) \\ &= \text{Cov}(\mathbf{V}\boldsymbol{z}) + \text{Cov}(\boldsymbol{\epsilon}) \\ &= \mathbf{V}\text{Cov}(\boldsymbol{z})\mathbf{V}^T + \mathbf{\Psi} \\ &= \mathbf{V}\mathbf{V}^T + \mathbf{\Psi} \quad (6.17) \end{aligned}$$

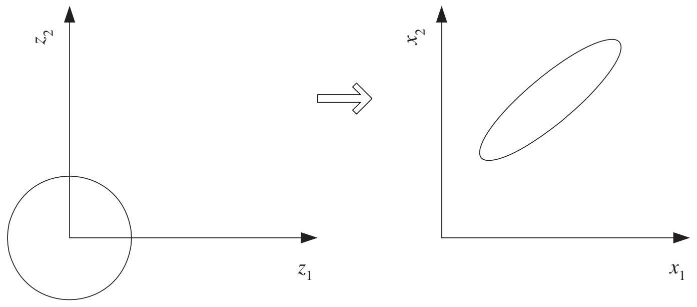

Figure 6.5 Factors are independent unit normals that are stretched, rotated, and translated to make up the inputs.

where $\mathbf{\Psi}$ is a diagonal matrix with ψ_i on the diagonals. Because the factors are uncorrelated unit normals, we have $\text{Cov}(\boldsymbol{z}) = \mathbf{I}$. With two factors, for example,

$$\text{Cov}(x_1, x_2) = v_{11}v_{21} + v_{12}v_{22}$$

If x_1 and x_2 have high covariance, then they are related through a factor. If it is the first factor, then v_{11} and v_{21} will both be high; if it is the second factor, then v_{12} and v_{22} will both be high. In either case, the sum $v_{11}v_{21} + v_{12}v_{22}$ will be high. If the covariance is low, then x_1 and x_2 depend on different factors and in the products in the sum, one term will be high and the other will be low and the sum will be low.

We see that

$$\text{Cov}(x_1, z_2) = \text{Cov}(v_{12}z_2, z_2) = v_{12}\text{Var}(z_2) = v_{12}$$

Thus $\text{Cov}(\boldsymbol{x}, \boldsymbol{z}) = \mathbf{V}$, and we see that the loadings represent the correlations of variables with the factors.

Given $\mathbf{S}$, the estimator of $\mathbf{\Sigma}$, we would like to find $\mathbf{V}$ and $\mathbf{\Psi}$ such that

$$\mathbf{S} = \mathbf{V}\mathbf{V}^T + \mathbf{\Psi}$$

If there are only a few factors, that is, if $\mathbf{V}$ has few columns, then we have a simplified structure for $\mathbf{S}$, as $\mathbf{V}$ is $d \times k$ and $\mathbf{\Psi}$ has d values, thus reducing the number of parameters from d^2 to $d \cdot k + d$.

Since $\mathbf{\Psi}$ is diagonal, covariances are represented by $\mathbf{V}$. Note that PCA does not allow a separate $\mathbf{\Psi}$ and it tries to account for both the covariances *and* the variances. When all ψ_i are equal, namely, $\mathbf{\Psi} = \psi\mathbf{I}$, we get

PROBABILISTIC PCA *probabilistic PCA* (Tipping and Bishop 1999) and the conventional PCA is when ψ_i are 0.

Let us now see how we can find the factor loadings and the specific variances: Let us first ignore $\mathbf{\Psi}$. Then, from its spectral decomposition, we know that we have

$$\mathbf{S} = \mathbf{CDC}^T = \mathbf{CD}^{1/2}\mathbf{D}^{1/2}\mathbf{C}^T = (\mathbf{CD}^{1/2})(\mathbf{CD}^{1/2})^T$$

where we take only k of the eigenvectors by looking at the proportion of variance explained so that $\mathbf{C}$ is the $d \times k$ matrix of eigenvectors and $\mathbf{D}^{1/2}$ is the $k \times k$ diagonal matrix with the square roots of the eigenvalues on its diagonals. Thus we have

(6.18) $$\mathbf{V} = \mathbf{CD}^{1/2}$$

We can find ψ_j from equation 6.15 as

(6.19) $$\psi_i = s_i^2 - \sum_{j=1}^{k} v_{ij}^2$$

Note that when $\mathbf{V}$ is multiplied with any orthogonal matrix—namely, having the property $\mathbf{TT}^T = \mathbf{I}$—that is another valid solution and thus the solution is not unique.

$$\mathbf{S} = (\mathbf{VT})(\mathbf{VT})^T = \mathbf{VTT}^T\mathbf{V}^T = \mathbf{VIV}^T = \mathbf{VV}^T$$

If $\mathbf{T}$ is an orthogonal matrix, the distance to the origin does not change. If $\boldsymbol{z} = \mathbf{T}\boldsymbol{x}$, then

$$\boldsymbol{z}^T\boldsymbol{z} = (\mathbf{T}\boldsymbol{x})^T(\mathbf{T}\boldsymbol{x}) = \boldsymbol{x}^T\mathbf{T}^T\mathbf{T}\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{x}$$

Multiplying with an orthogonal matrix has the effect of rotating the axes and allows us to choose the set of axes most interpretable (Rencher 1995). In two dimensions,

$$\mathbf{T} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}$$

rotates the axes by ϕ. There are two types of rotation: In orthogonal rotation the factors are still orthogonal after the rotation, and in oblique rotation the factors are allowed to become correlated. The factors are rotated to give the maximum loading on as few factors as possible for each variable, to make the factors interpretable. However, interpretability is subjective and should not be used to force one's prejudices on the data.

There are two uses of factor analysis: It can be used for knowledge extraction when we find the loadings and try to express the variables using fewer factors. It can also be used for dimensionality reduction when $k < d$. We already saw how the first one is done. Now, let us see how factor analysis can be used for dimensionality reduction.

When we are interested in dimensionality reduction, we need to be able to find the factor scores, z_j, from x_i. We want to find the loadings w_{ji} such that

$$z_j = \sum_{i=1}^{d} w_{ji} x_i + \epsilon_i, j = 1, \dots, k \tag{6.20}$$

where x_i are centered to have mean 0. In vector form, for observation t, this can be written as

$$\boldsymbol{z}^t = \mathbf{W}^T \boldsymbol{x}^t + \boldsymbol{\epsilon}, \forall t = 1, \dots, N$$

This is a linear model with d inputs and k outputs. Its transpose can be written as

$$(\boldsymbol{z}^t)^T = (\boldsymbol{x}^t)^T \mathbf{W} + \boldsymbol{\epsilon}^T, \forall t = 1, \dots, N$$

Given that we have a sample of N observations, we write

$$\mathbf{Z} = \mathbf{X}\mathbf{W} + \Xi \tag{6.21}$$

where $\mathbf{Z}$ is $N \times k$ of factors, $\mathbf{X}$ is $N \times d$ of (centered) observations, and Ξ is $N \times k$ of zero-mean noise. This is multivariate linear regression with multiple outputs, and we know from section 5.8 that $\mathbf{W}$ can be found as

$$\mathbf{W} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Z}$$

but we do not know $\mathbf{Z}$; it is what we would like to calculate. We multiply and divide both sides by $N - 1$ and obtain

$$\begin{aligned} \mathbf{W} &= (N-1)(\mathbf{X}^T\mathbf{X})^{-1}\frac{\mathbf{X}^T\mathbf{Z}}{N-1} \\ &= \left(\frac{\mathbf{X}^T\mathbf{X}}{N-1}\right)^{-1}\frac{\mathbf{X}^T\mathbf{Z}}{N-1} \\ &= \mathbf{S}^{-1}\mathbf{V} \end{aligned} \tag{6.22}$$

and placing equation 6.22 in equation 6.21, we write

$$\mathbf{Z} = \mathbf{X}\mathbf{W} = \mathbf{X}\mathbf{S}^{-1}\mathbf{V} \tag{6.23}$$

assuming that $\mathbf{S}$ is nonsingular. One can use $\mathbf{R}$ instead of $\mathbf{S}$ when x_i are normalized to have unit variance.

For dimensionality reduction, FA offers no advantage over PCA except the interpretability of factors allowing the identification of common causes, a simple explanation, and knowledge extraction. For example, in the context of speech recognition, $\boldsymbol{x}$ corresponds to the acoustic signal, but we know that it is the result of the (nonlinear) interaction of a small number of *articulators*, namely, jaw, tongue, velum, lips, and mouth, which are positioned appropriately to shape the air as it comes out of the lungs and generate the speech sound. If a speech signal could be transformed to this articulatory space, then recognition would be much easier. Using such generative models is one of the current research directions for speech recognition; in chapter 16, we will discuss how such models can be represented as a graphical model.

6.5 Multidimensional Scaling

Let us say for N points, we are given the distances between pairs of points, d_{ij}, for all $i, j = 1, \dots, N$. We do not know the exact coordinates of the points, their dimensionality, or how the distances are calculated. *Multidimensional scaling* (MDS) is the method for placing these points in a low—for example, two-dimensional—space such that the Euclidean distance between them there is as close as possible to d_{ij}, the given distances in the original space. Thus it requires a projection from some unknown dimensional space to, for example, two dimensions.

MULTIDIMENSIONAL SCALING

In the archetypical example of multidimensional scaling, we take the road travel distances between cities, and after applying MDS, we get an approximation to the map. The map is distorted such that in parts of the country with geographical obstacles like mountains and lakes where the road travel distance deviates much from the direct bird-flight path (Euclidean distance), the map is stretched out to accommodate longer distances (see figure 6.6). The map is centered on the origin, but the solution is still not unique. We can get any rotated or mirror image version.

MDS can be used for dimensionality reduction by calculating pairwise Euclidean distances in the d-dimensional $\boldsymbol{x}$ space and giving this as input to MDS, which then projects it to a lower-dimensional space so as to preserve these distances.

Let us say we have a sample $\mathcal{X} = \{\boldsymbol{x}^t\}_{t=1}^N$ as usual, where $\boldsymbol{x}^t \in \Re^d$. For

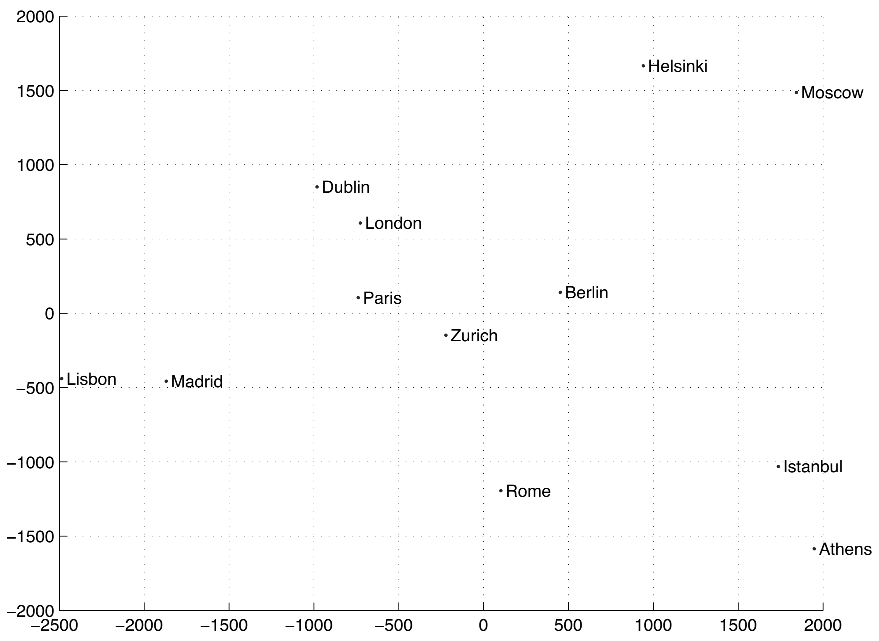

Figure 6.6 Map of Europe drawn by MDS. Pairwise road travel distances between these cities are given as input, and MDS places them in two dimensions such that these distances are preserved as well as possible.

two points r and s, the squared Euclidean distance between them is

$$\begin{aligned} d_{rs}^2 &= \|\boldsymbol{x}^r - \boldsymbol{x}^s\|^2 = \sum_{j=1}^d (x_j^r - x_j^s)^2 = \sum_{j=1}^d (x_j^r)^2 - 2\sum_{j=1}^d x_j^r x_j^s + \sum_{j=1}^d (x_j^s)^2 \\ &= b_{rr} + b_{ss} - 2b_{rs} \qquad (6.24) \end{aligned}$$

where b_{rs} is defined as

$$b_{rs} = \sum_{j=1}^d x_j^r x_j^s \qquad (6.25)$$

To constrain the solution, we center the data at the origin and assume

$$\sum_{t=1}^N x_j^t = 0, \forall j = 1, \ldots, d$$

Then, summing up equation 6.24 on r, s, and both r, s, and defining

$$T = \sum_{t=1}^{N} b_{tt} = \sum_t \sum_j (x_j^t)^2$$

we get

$$\begin{aligned}\sum_r d_{rs}^2 &= T + Nb_{ss}\\ \sum_s d_{rs}^2 &= Nb_{rr} + T\\ \sum_r \sum_s d_{rs}^2 &= 2NT\end{aligned}$$

When we define

$$d_{\bullet s}^2 = \frac{1}{N}\sum_r d_{rs}^2 \;,\; d_{r\bullet}^2 = \frac{1}{N}\sum_s d_{rs}^2 \;,\; d_{\bullet\bullet}^2 = \frac{1}{N^2}\sum_r \sum_s d_{rs}^2$$

and using equation 6.24, we get

(6.26) $$b_{rs} = \frac{1}{2}(d_{r\bullet}^2 + d_{\bullet s}^2 - d_{\bullet\bullet}^2 - d_{rs}^2)$$

Having now calculated b_{rs} and knowing that $\mathbf{B} = \mathbf{X}\mathbf{X}^T$ as defined in equation 6.25, we look for an approximation. We know from the spectral decomposition that $\mathbf{X} = \mathbf{C}\mathbf{D}^{1/2}$ can be used as an approximation for $\mathbf{X}$, where $\mathbf{C}$ is the matrix whose columns are the eigenvectors of $\mathbf{B}$ and $\mathbf{D}^{1/2}$ is a diagonal matrix with square roots of the eigenvalues on the diagonals. Looking at the eigenvalues of $\mathbf{B}$, we decide on a dimensionality k lower than d (and N), as we did in PCA and FA. Let us say $\boldsymbol{c}_j$ are the eigenvectors with λ_j as the corresponding eigenvalues. Note that $\boldsymbol{c}_j$ is N-dimensional. Then we get the new dimensions as

(6.27) $$z_j^t = \sqrt{\lambda_j} c_j^t, j = 1, \ldots, k, t = 1, \ldots, N$$

That is, the new coordinates of instance t are given by the tth elements of the eigenvectors, $\boldsymbol{c}_j, j = 1, \ldots, k$, after normalization.

It has been shown (Chatfield and Collins 1980) that the eigenvalues of $\mathbf{X}\mathbf{X}^T$ ($N \times N$) are the same as those of $\mathbf{X}^T\mathbf{X}$ ($d \times d$) and the eigenvectors are related by a simple linear transformation. This shows that PCA does the same work with MDS and does it more cheaply. PCA done on the correlation matrix rather than the covariance matrix equals doing MDS with standardized Euclidean distances where each variable has unit variance.

In the general case, we want to find a mapping $\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x}|\theta)$, where $\boldsymbol{z} \in \Re^k, \boldsymbol{x} \in \Re^d$, and $\boldsymbol{g}(\boldsymbol{x}|\theta)$ is the mapping function from d to k dimensions defined up to a set of parameters θ. Classical MDS we discussed previously corresponds to a linear transformation

$$\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x}|\mathbf{W}) = \mathbf{W}^T\boldsymbol{x} \tag{6.28}$$

SAMMON MAPPING

but in a general case, a nonlinear mapping can also be used; this is called *Sammon mapping.* The normalized error in mapping is called the *Sammon stress* and is defined as

$$\begin{aligned} E(\theta|\mathcal{X}) &= \sum_{r,s} \frac{(\|\boldsymbol{z}^r - \boldsymbol{z}^s\| - \|\boldsymbol{x}^r - \boldsymbol{x}^s\|)^2}{\|\boldsymbol{x}^r - \boldsymbol{x}^s\|^2} \\ &= \sum_{r,s} \frac{(\|\boldsymbol{g}(\boldsymbol{x}^r|\theta) - \boldsymbol{g}(\boldsymbol{x}^s|\theta)\| - \|\boldsymbol{x}^r - \boldsymbol{x}^s\|)^2}{\|\boldsymbol{x}^r - \boldsymbol{x}^s\|^2} \end{aligned} \tag{6.29}$$

One can use any regression method for $\boldsymbol{g}(\cdot|\theta)$ and estimate θ to minimize the stress on the training data $\mathcal{X}$. If $\boldsymbol{g}(\cdot)$ is nonlinear in $\boldsymbol{x}$, this will then correspond to a nonlinear dimensionality reduction.

In the case of classification, one can include class information in the distance (see Webb 1999) as

$$d'_{rs} = (1 - \alpha)d_{rs} + \alpha c_{rs}$$

where c_{rs} is the "distance" between the classes $\boldsymbol{x}^r$ and $\boldsymbol{x}^s$ belong to. This interclass distance should be supplied subjectively and α is optimized using cross-validation.

6.6 Linear Discriminant Analysis

LINEAR DISCRIMINANT ANALYSIS

Linear discriminant analysis (LDA) is a supervised method for dimensionality reduction for classification problems. We start with the case where there are two classes, then generalize to $K > 2$ classes.

Given samples from two classes C_1 and C_2, we want to find the direction, as defined by a vector $\boldsymbol{w}$, such that when the data are projected onto $\boldsymbol{w}$, the examples from the two classes are as well separated as possible. As we saw before,

$$z = \boldsymbol{w}^T\boldsymbol{x} \tag{6.30}$$

is the projection of $\boldsymbol{x}$ onto $\boldsymbol{w}$ and thus is a dimensionality reduction from d to 1.

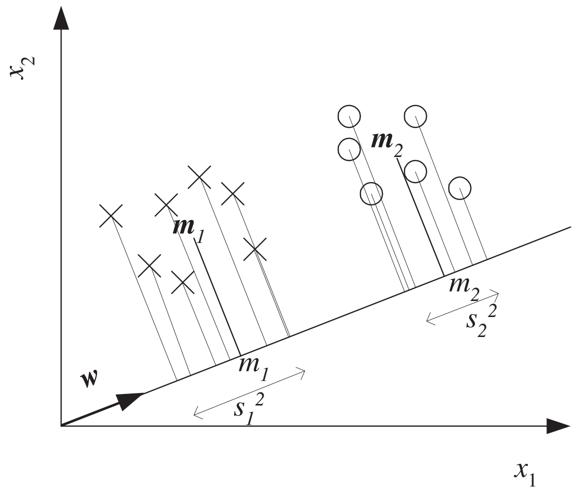

Figure 6.7 Two-dimensional, two-class data projected on $\boldsymbol{w}$.

$\boldsymbol{m}_1$ and m_1 are the means of samples from C_1 before and after projection, respectively. Note that $\boldsymbol{m}_1 \in \Re^d$ and $m_1 \in \Re$. We are given a sample $\mathcal{X} = \{\boldsymbol{x}^t, r^t\}$ such that $r^t = 1$ if $\boldsymbol{x}^t \in C_1$ and $r^t = 0$ if $\boldsymbol{x}^t \in C_2$.

$$\begin{aligned} m_1 &= \frac{\sum_t \boldsymbol{w}^T \boldsymbol{x}^t r^t}{\sum_t r^t} = \boldsymbol{w}^T \boldsymbol{m}_1 \\ m_2 &= \frac{\sum_t \boldsymbol{w}^T \boldsymbol{x}^t (1 - r^t)}{\sum_t (1 - r^t)} = \boldsymbol{w}^T \boldsymbol{m}_2 \end{aligned} \tag{6.31}$$

SCATTER

The *scatter* of samples from C_1 and C_2 after projection are

$$\begin{aligned} s_1^2 &= \sum_t (\boldsymbol{w}^T \boldsymbol{x}^t - m_1)^2 r^t \\ s_2^2 &= \sum_t (\boldsymbol{w}^T \boldsymbol{x}^t - m_2)^2 (1 - r^t) \end{aligned} \tag{6.32}$$

FISHER'S LINEAR DISCRIMINANT

After projection, for the two classes to be well separated, we would like the means to be as far apart as possible and the examples of classes be scattered in as small a region as possible. So we want $|m_1 - m_2|$ to be large and $s_1^2 + s_2^2$ to be small (see figure 6.7). *Fisher's linear discriminant* is $\boldsymbol{w}$ that maximizes

$$J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} \tag{6.33}$$

Rewriting the numerator, we get

$$\begin{aligned}(m_1 - m_2)^2 &= (\boldsymbol{w}^T\boldsymbol{m}_1 - \boldsymbol{w}^T\boldsymbol{m}_2)^2 \\ &= \boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T\boldsymbol{w} \\ &= \boldsymbol{w}^T\mathbf{S}_B\boldsymbol{w} \qquad (6.34)\end{aligned}$$

BETWEEN-CLASS SCATTER MATRIX

where $\mathbf{S}_B = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T$ is the *between-class scatter matrix*. The denominator is the sum of scatter of examples of classes around their means after projection and can be rewritten as

$$\begin{aligned}s_1^2 &= \sum_t (\boldsymbol{w}^T\boldsymbol{x}^t - m_1)^2 r^t \\ &= \sum_t \boldsymbol{w}^T(\boldsymbol{x}^t - \boldsymbol{m}_1)(\boldsymbol{x}^t - \boldsymbol{m}_1)^T\boldsymbol{w}r^t \\ &= \boldsymbol{w}^T\mathbf{S}_1\boldsymbol{w} \qquad (6.35)\end{aligned}$$

where

$$\mathbf{S}_1 = \sum_t r^t(\boldsymbol{x}^t - \boldsymbol{m}_1)(\boldsymbol{x}^t - \boldsymbol{m}_1)^T \qquad (6.36)$$

WITHIN-CLASS SCATTER MATRIX

is the *within-class scatter matrix* for C_1. $\mathbf{S}_1/\sum_t r^t$ is the estimator of $\boldsymbol{\Sigma}_1$. Similarly, $s_2^2 = \boldsymbol{w}^T\mathbf{S}_2\boldsymbol{w}$ with $\mathbf{S}_2 = \sum_t (1 - r^t)(\boldsymbol{x}^t - \boldsymbol{m}_2)(\boldsymbol{x}^t - \boldsymbol{m}_2)^T$, and we get

$$s_1^2 + s_2^2 = \boldsymbol{w}^T\mathbf{S}_W\boldsymbol{w}$$

where $\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$ is the total within-class scatter. Note that $s_1^2 + s_2^2$ divided by the total number of samples is the variance of the pooled data. Equation 6.33 can be rewritten as

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T\mathbf{S}_B\boldsymbol{w}}{\boldsymbol{w}^T\mathbf{S}_W\boldsymbol{w}} = \frac{|\boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)|^2}{\boldsymbol{w}^T\mathbf{S}_W\boldsymbol{w}} \qquad (6.37)$$

Taking the derivative of J with respect to $\boldsymbol{w}$ and setting it equal to 0, we get

$$\frac{\boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)}{\boldsymbol{w}^T\mathbf{S}_W\boldsymbol{w}}\left(2(\boldsymbol{m}_1 - \boldsymbol{m}_2) - \frac{\boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)}{\boldsymbol{w}^T\mathbf{S}_W\boldsymbol{w}}\mathbf{S}_W\boldsymbol{w}\right) = 0$$

Given that $\boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)/\boldsymbol{w}^T\mathbf{S}_W\boldsymbol{w}$ is a constant, we have

$$\boldsymbol{w} = c\mathbf{S}_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2) \qquad (6.38)$$

where c is some constant. Because it is the direction that is important for us and not the magnitude, we can just take $c = 1$ and find $\boldsymbol{w}$.

Remember that when $p(\boldsymbol{x}|C_i) \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$, we have a linear discriminant where $\boldsymbol{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$, and we see that Fisher's linear discriminant is optimal if the classes are normally distributed. Under the same assumption, a threshold, w_0, can also be calculated to separate the two classes. But Fisher's linear discriminant can be used even when the classes are not normal. We have projected the samples from d dimensions to one, and any classification method can be used afterward.

In the case of $K > 2$ classes, we want to find the matrix $\mathbf{W}$ such that

$$\boldsymbol{z} = \mathbf{W}^T \boldsymbol{x} \tag{6.39}$$

where $\boldsymbol{z}$ is k-dimensional and $\mathbf{W}$ is $d \times k$. The within-class scatter matrix for C_i is

$$\mathbf{S}_i = \sum_t r_i^t (\boldsymbol{x}^t - \boldsymbol{m}_i)(\boldsymbol{x}^t - \boldsymbol{m}_i)^T \tag{6.40}$$

where $r_i^t = 1$ if $\boldsymbol{x}^t \in C_i$ and 0 otherwise. The total within-class scatter is

$$\mathbf{S}_W = \sum_{i=1}^{K} \mathbf{S}_i \tag{6.41}$$

When there are $K > 2$ classes, the scatter of the means is calculated as how much they are scattered around the overall mean

$$\boldsymbol{m} = \frac{1}{K} \sum_{i=1}^{K} \boldsymbol{m}_i \tag{6.42}$$

and the between-class scatter matrix is

$$\mathbf{S}_B = \sum_{i=1}^{K} N_i (\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T \tag{6.43}$$

with $N_i = \sum_t r_i^t$. The between-class scatter matrix after projection is $\mathbf{W}^T \mathbf{S}_B \mathbf{W}$ and the within-class scatter matrix after projection is $\mathbf{W}^T \mathbf{S}_W \mathbf{W}$. These are both $k \times k$ matrices. We want the first scatter to be large, that is, after the projection, in the new k-dimensional space we want class means to be as far apart from each other as possible. We want the second scatter to be small, that is, after the projection, we want samples from the same class to be as close to their mean as possible. For a scatter (or covariance) matrix, a measure of spread is the determinant, remembering that the determinant is the product of eigenvalues and that an

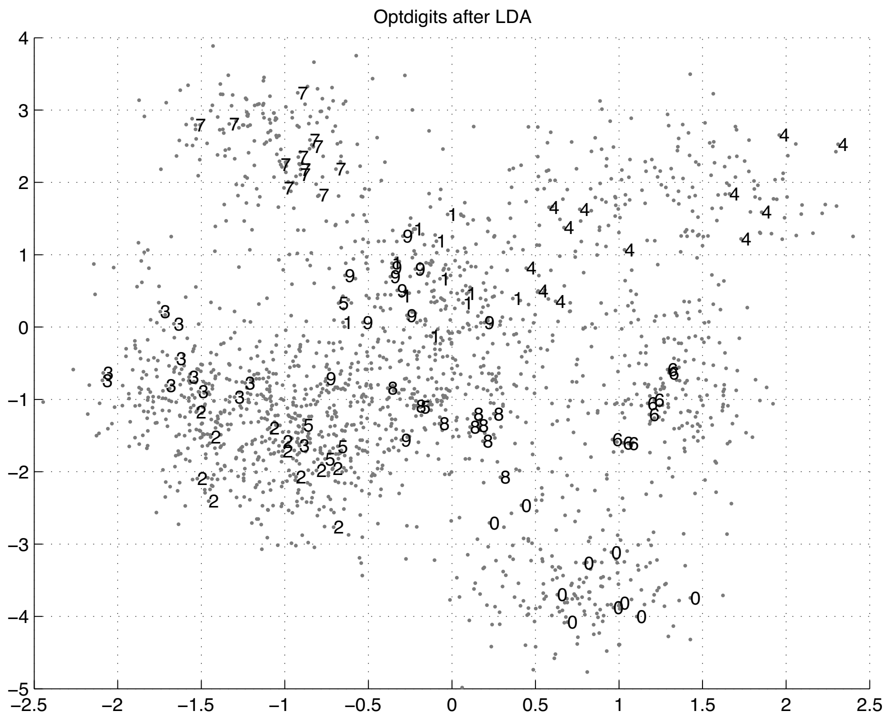

Figure 6.8 Optdigits data plotted in the space of the first two dimensions found by LDA. Comparing this with figure 6.3, we see that LDA, as expected, leads to a better separation of classes than PCA. Even in this two-dimensional space (there are nine altogether), we can discern separate clouds for different classes.

eigenvalue gives the variance along its eigenvector (component). Thus we are interested in the matrix **W** that maximizes

$$J(\mathbf{W}) = \frac{|\mathbf{W}^T\mathbf{S}_B\mathbf{W}|}{|\mathbf{W}^T\mathbf{S}_W\mathbf{W}|} \tag{6.44}$$

The largest eigenvectors of $\mathbf{S}_W^{-1}\mathbf{S}_B$ are the solution. $\mathbf{S}_B$ is the sum of K matrices of rank 1, namely, $(\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$, and only $K - 1$ of them are independent. Therefore, $\mathbf{S}_B$ has a maximum rank of $K - 1$ and we take $k = K - 1$. Thus we define a new lower, $(K - 1)$-dimensional space where the discriminant is then to be constructed (see figure 6.8). Though LDA uses class separability as its goodness criterion, any classification method can be used in this new space for estimating the discriminants.

We see that to be able to apply LDA, $\mathbf{S}_W$ should be invertible. If this is not the case, we can first use PCA to get rid of singularity and then apply LDA to its result; however, we should make sure that PCA does not reduce dimensionality so much that LDA does not have anything left to work on.

6.7 Isomap

Principal component analysis (PCA), which we discussed in section 6.3, works when the data lies in a linear subspace. However, this may not hold in many applications. Take, for example, face recognition where a face is represented as a two-dimensional, say 100×100 image. In this case, each face is a point in $10,000$ dimensions. Now let us say that we take a series of pictures as a person slowly rotates his or her head from right to left. The sequence of face images we capture follows a trajectory in the $10,000$-dimensional space and this is not linear. Now consider the faces of many people. The trajectories of all their faces as they rotate their faces define a manifold in the $10,000$-dimensional space, and this is what we want to model. The similarity between two faces cannot simply be written in terms of the sum of the pixel differences, and hence Euclidean distance is not a good metric. It may even be the case that images of two different people with the same pose have smaller Euclidean distance between them than the images of two different poses of the same person. This is not what we want. What should count is the distance along the manifold, which is called the *geodesic distance*. *Isometric feature mapping* (Isomap) (Tenenbaum, de Silva, and Langford 2000) estimates this distance and applies multidimensional scaling (MDS) (section 6.5), using it for dimensionality reduction.

GEODESIC DISTANCE

ISOMETRIC FEATURE MAPPING

Isomap uses the geodesic distances between all pairs of data points. For neighboring points that are close in the input space, Euclidean distance can be used; for small changes in pose, the manifold is locally linear. For faraway points, geodesic distance is approximated by the sum of the distances between the points along the way over the manifold. This is done by defining a graph whose nodes correspond to the N data points and whose edges connect neighboring points (those with distance less than some ϵ or one of the n nearest) with weights corresponding to Euclidean distances. The geodesic distance between any two points is calculated as the length of the shortest path between the corresponding

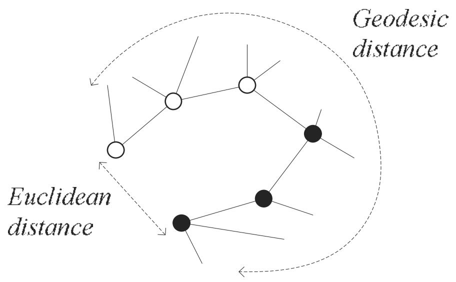

Figure 6.9 Geodesic distance is calculated along the manifold as opposed to the Euclidean distance that does not use this information. After multidimensional scaling, these two instances from two classes will be mapped to faraway positions in the new space, though they are close in the original space.

two nodes. For two points that are not close by, we need to hop over a number of intermediate points along the way, and therefore the distance will be the distance along the manifold, approximated as the sum of local Euclidean distances (see figure 6.9).

Two nodes r and s are connected if $\|\boldsymbol{x}^r - \boldsymbol{x}^s\| < \epsilon$ (while making sure that the graph is connected), or if $\boldsymbol{x}^s$ is one of the n neighbors of $\boldsymbol{x}^r$ (while making sure that the distance matrix is symmetric), and we set the edge length to $\|\boldsymbol{x}^r - \boldsymbol{x}^s\|$. For any two nodes r and s, d_{rs} is the length of the shortest path between them. We then apply MDS on d_{rs} to reduce dimensionality to k by observing the proportion of variance explained. This will have the effect of placing r and s that are far apart in the geodesic space also far in the new k-dimensional space even if they are close in terms of Euclidean distance in the original d-dimensional space.

It is clear that the graph distances provide a better approximation as the number of points increases, though there is the trade-off of longer execution time; if time is critical, one can subsample and use a subset of "landmark points" to make the algorithm faster. The parameter ϵ needs to be carefully tuned; if it is too small, there may be more than one connected component, and if it is too large, "shortcut" edges may be added that corrupt the low-dimensional embedding (Balasubramanian et al. 2002).

One problem with Isomap, as with MDS, is that it places the N points in a low-dimensional space, but it does not learn a general mapping function that will allow mapping a new test point; the new point should be added to the dataset and the whole algorithm needs to be run once more using $N + 1$ instances.

6.8 Locally Linear Embedding

LOCALLY LINEAR EMBEDDING

Locally linear embedding (LLE) recovers global nonlinear structure from locally linear fits (Roweis and Saul 2000). The idea is that each local patch of the manifold can be approximated linearly and given enough data, each point can be written as a linear, weighted sum of its neighbors (again either defined using a given number of neighbors, n, or distance threshold, ϵ). Given $\boldsymbol{x}^r$ and its neighbors $\boldsymbol{x}^s_{(r)}$ in the original space, one can find the reconstruction weights $\mathbf{W}_{rs}$ that minimize the error function

$$\mathcal{E}^w(\mathbf{W}|\mathcal{X}) = \sum_r \|\boldsymbol{x}^r - \sum_s \mathbf{W}_{rs}\boldsymbol{x}^s_{(r)}\|^2 \tag{6.45}$$

using least squares subject to $\mathbf{W}_{rr} = 0, \forall r$ and $\sum_s \mathbf{W}_{rs} = 1$.

The idea in LLE is that the reconstruction weights $\mathbf{W}_{rs}$ reflect the intrinsic geometric properties of the data that we expect to be also valid for local patches of the manifold, that is, the new space we are mapping the instances to (see figure 6.10). The second step of LLE is hence to now keep the weights $\mathbf{W}_{rs}$ fixed and let the new coordinates $\boldsymbol{z}^r$ take whatever values they need respecting the interpoint constraints given by the weights:

$$\mathcal{E}^z(\mathcal{Z}|\mathbf{W}) = \sum_r \|\boldsymbol{z}^r - \sum_s \mathbf{W}_{rs}\boldsymbol{z}^s\|^2 \tag{6.46}$$

Nearby points in the original, d-dimensional space should remain nearby and similarly colocated with respect to one another in the new, k-dimensional space. Equation 6.46 can be rewritten as

$$\mathcal{E}^z(\mathcal{Z}|\mathbf{W}) = \sum_{r,s} \mathbf{M}_{rs}(\boldsymbol{z}^r)^T\boldsymbol{z}_s \tag{6.47}$$

where

$$\mathbf{M}_{rs} = \delta_{rs} - \mathbf{W}_{rs} - \mathbf{W}_{sr} + \sum_i \mathbf{W}_{ir}\mathbf{W}_{is} \tag{6.48}$$

$\mathbf{M}$ is sparse (only a small percentage of data points are neighbors of a data point: $n \ll N$), symmetric, and positive semidefinite. As in other dimensionality reduction methods, we require that the data be centered at the origin, $E[\boldsymbol{z}] = 0$, and that the new coordinates be uncorrelated

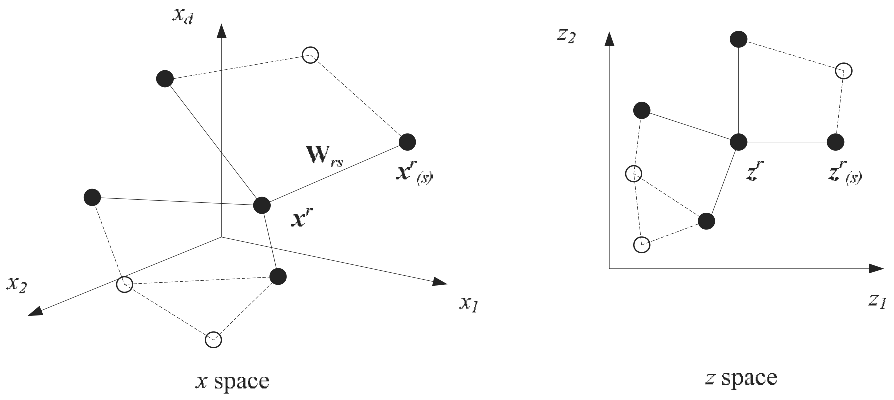

Figure 6.10 Local linear embedding first learns the constraints in the original space and next places the points in the new space respecting those constraints. The constraints are learned using the immediate neighbors (shown with continuous lines) but also propagate to second-order neighbors (shown dashed).

and unit length: $\text{Cov}(\boldsymbol{z}) = \mathbf{I}$. The solution to equation 6.47 subject to these two constraints is given by the $k+1$ eigenvectors with the smallest eigenvalues; we ignore the lowest one and the other k eigenvectors give us the new coordinates.

Because the n neighbors span a space of dimensionality $n-1$ (you need distances to three points to uniquely specify your location in two dimensions), LLE can reduce dimensionality up to $k \le n-1$. It is observed (Saul and Roweis 2003) that some margin between k and n is necessary to obtain a good embedding. Note that if n (or ϵ) is small, the graph (that is constructed by connecting each instance to its neighbors) may no longer be connected and it may be necessary to run LLE separately on separate components to find separate manifolds in different parts of the input space. On the other hand, if n (or ϵ) is taken large, some neighbors may be too far for the local linearity assumption to hold and this may corrupt the embedding. It is possible to use different n (or ϵ) in different parts of the input space based on some prior knowledge, but how this can be done is open to research (Saul and Roweis 2003).

As with Isomap, LLE solution is the set of new coordinates for the N points, but we do not learn a mapping and hence cannot find $\boldsymbol{z}'$ for a new $\boldsymbol{x}'$. There are two solutions to this:

1. Using the same idea, one can find the n neighbors of $\boldsymbol{x}'$ in the original d-dimensional space and first learn the reconstruction weights $\boldsymbol{w}_j$ that minimizes

$$\mathcal{E}^w(\boldsymbol{w}|\mathcal{X}) = \|\boldsymbol{x}' - \sum_s \boldsymbol{w}_s \boldsymbol{x}^s\|^2 \tag{6.49}$$

   and then use them to reconstruct $\boldsymbol{z}'$ in the new k-dimensional space:

$$\boldsymbol{z}' = \sum_s \boldsymbol{w}_s \boldsymbol{z}^s \tag{6.50}$$

   Note that this approach can also be used to interpolate from an Isomap (or MDS) solution. The drawback however is the need to store the whole set of $\{\boldsymbol{x}^t, \boldsymbol{z}^t\}_{t=1}^N$.

2. Using $\mathcal{X} = \{\boldsymbol{x}^t, \boldsymbol{z}^t\}_{t=1}^N$ as a training set, one can train any regressor, $\boldsymbol{g}(\boldsymbol{x}^t|\theta)$—for example, a multilayer perceptron (chapter 11)—as a generalizer to approximate $\boldsymbol{z}^t$ from $\boldsymbol{x}^t$, whose parameters θ is learned to minimize the regression error:

$$\mathcal{E}(\theta|\mathcal{X}) = \sum_t \|\boldsymbol{z}^t - \boldsymbol{g}(\boldsymbol{x}^t|\theta)\|^2 \tag{6.51}$$

   Once training is done, we can calculate $\boldsymbol{z}' = \boldsymbol{g}(\boldsymbol{x}'|\theta)$. The model $\boldsymbol{g}(\cdot)$ should be carefully chosen to be able to learn the mapping. There may no longer be a unique optimum and hence there are all the usual problems related to minimization, that is, initialization, local optima, convergence, and so on.

In both Isomap and LLE, there is local information that is propagated over neighbors to get a global solution. In Isomap, the geodesic distance is the sum of local distances; in LLE, the final opimization in placing $\boldsymbol{z}^t$ takes into account all local $\mathbf{W}_{rs}$ values. Let us say a and b are neighbors and b and c are neighbors. Though a and c may not be neighbors, there is dependence between a and c either through the graph, $d_{ac} = d_{ab} + d_{bc}$, or the weights $\mathbf{W}_{ab}$ and $\mathbf{W}_{bc}$. In both algorithms, the global nonlinear organization is found by integrating local linear constraints that overlap partially.

6.9 Notes

A survey of feature selection algorithms is given in Devijer and Kittler 1982. WRAPPERS Feature subset selection algorithms are also known as the *wrapper* approach, where the feature selection is thought to "wrap" around the learner it uses as a subroutine (Kohavi and John 1997). Subset selection in regression is discussed in Miller 1990. The forward and backward search procedures we discussed are local search procedures. Fukunaga and Narendra (1977) proposed a branch and bound procedure. At considerable more expense, one can use a stochastic procedure like simulated annealing or genetic algorithms to search more widely in the the search space.

There are also *filtering* algorithms for feature selection where heuristic measures are used to calculate the "relevance" of a feature in a preprocessing stage without actually using the learner. For example, in the case of classification, instead of training a classifier and testing it at each step, one can use a separability measure, like the one used in linear discriminant analysis, to measure the quality of the new space in separating classes from each other (McLachlan 1992). With the cost of computation going down, it is best to include the learner in the loop because there is no guarantee that the heuristic used by the filter will match the bias of the learner that uses the features; no heuristic can replace the actual validation accuracy. A survey of feature selection methods is given by Guyon and Elisseeff (2003).

Projection methods work with numeric inputs, and discrete variables should be represented by 0/1 dummy variables, whereas subset selection can use discrete inputs directly. Finding the eigenvectors and eigenvalues is quite straightforward; an example of a code is given in Press et al. 1992. Factor analysis was introduced by the British psychologist Charles Spearman to find the single factor for intelligence which explains the correlation between scores on various intelligence tests. The existence of such a single factor, called g, is highly disputed. More information on multidimensional scaling can be found in Cox and Cox 1994.

The projection methods we discussed are batch procedures in that they require that the whole sample be given before the projection directions are found. Mao and Jain (1995) discuss online procedures for doing PCA and LDA, where instances are given one by one and updates are done as new instances arrive. Another possibility in doing a nonlinear projection is when the estimator in Sammon mapping is taken as a nonlinear function, for example, a multilayer perceptron (section 11.11) (Mao and Jain 1995). It is also possible but much harder to do nonlinear factor analysis. When the models are nonlinear, it is difficult to come up with the right nonlinear model. One also needs to use complicated optimization and approximation methods to solve for the model parameters.

For more information, one can refer to the Isomap homepage that is at `http://web.mit.edu/cocosci/isomap/isomap.html` and the LLE homepage is at `http://www.cs.toronto.edu/~roweis/lle/`. Both contain links to related publications and example code.

Just as we implement polynomial regression by using linear regression where we consider high-order terms as additional inputs (section 5.8), another way to do nonlinear dimensionality reduction is to first map to a new space by using nonlinear basis functions and then use a linear method there. In chapter 13 where we will discuss kernel methods, we will see how this can be done efficiently.

There is a trade-off between feature extraction and decision making. If the feature extractor is good, the task of the classifier (or regressor) becomes trivial, for example, when the class code is extracted as a new feature from the existing features. On the other hand, if the classifier is good enough, then there is no need for feature extraction; it does its automatic feature selection or combination internally. We live between these two ideal worlds.

There exist algorithms that do some feature selection internally, though in a limited way. Decision trees (chapter 9) do feature selection while generating the decision tree, and multilayer perceptrons (chapter 11) do nonlinear feature extraction in the hidden nodes. We expect to see more development along this line in embedding feature extraction in the actual step of classification/regression.

6.10 Exercises

1. Assuming that the classes are normally distributed, in subset selection, when one variable is added or removed, how can the new discriminant be calculated quickly? For example, how can the new $\mathbf{S}_{new}^{-1}$ be calculated from $\mathbf{S}_{old}^{-1}$?
2. Using Optdigits from the UCI repository, implement PCA. For various number of eigenvectors, reconstruct the digit images and calculate the reconstruction error (equation 6.12).
3. Plot the map of your state/country using MDS, given the road travel distances as input.

4. In Sammon mapping, if the mapping is linear, namely, $g(\boldsymbol{x}|\mathbf{W}) = \mathbf{W}^T\boldsymbol{x}$, how can $\mathbf{W}$ that minimizes the Sammon stress be calculated?
5. Redo exercise 3, this time using Isomap where two cities are connected only if there is a direct road between them that does not pass through any other city.
6. In Isomap, instead of using Euclidean distance, we can also use Mahalanobis distance between neighboring points. What are the advantages and disadvantages of this approach, if any?
7. Draw two-class, two-dimensional data such that (a) PCA and LDA find the same direction and (b) PCA and LDA find totally different directions.
8. Multidimensional scaling can work as long as we have the pairwise distances between objects. We do not actually need to represent the objects as vectors at all as long as we have some measure of similarity. Can you give an example?
9. How can we incorporate class information into Isomap and LLE such that instances of the same class are mapped to nearby locations in the new space?
10. In factor analysis, how can we find the remaining ones if we already know some of the factors?
11. Discuss an application where there are hidden factors (not necessarily linear) and where factor analysis would be expected to work well.

6.11 References

Balasubramanian, M., E. L. Schwartz, J. B. Tenenbaum, V. de Silva, and J. C. Langford. 2002. “The Isomap Algorithm and Topological Stability.” *Science* 295: 7.

Chatfield, C., and A. J. Collins. 1980. *Introduction to Multivariate Analysis.* London: Chapman and Hall.

Cox, T. F., and M. A. A. Cox. 1994. *Multidimensional Scaling.* London: Chapman and Hall.

Devijer, P. A., and J. Kittler. 1982. *Pattern Recognition: A Statistical Approach.* New York: Prentice-Hall.

Flury, B. 1988. *Common Principal Components and Related Multivariate Models.* New York: Wiley.

Fukunaga, K., and P. M. Narendra. 1977. “A Branch and Bound Algorithm for Feature Subset Selection.” *IEEE Transactions on Computers* C-26: 917–922.

Guyon, I., and A. Elisseeff. 2003. “An Introduction to Variable and Feature Selection.” *Journal of Machine Learning Research* 3: 1157–1182.

Hastie, T. J., R. J. Tibshirani, and A. Buja. 1994. “Flexible Discriminant Analysis by Optimal Scoring.” *Journal of the American Statistical Association* 89: 1255–1270.

Kohavi, R., and G. John. 1997. “Wrappers for Feature Subset Selection.” *Artificial Intelligence* 97: 273–324.

Mao, J., and A. K. Jain. 1995. “Artificial Neural Networks for Feature Extraction and Multivariate Data Projection.” *IEEE Transactions on Neural Networks* 6: 296–317.

McLachlan, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition.* New York: Wiley.

Miller, A. J. 1990. *Subset Selection in Regression.* London: Chapman and Hall.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. 1992. *Numerical Recipes in C.* Cambridge, UK: Cambridge University Press.

Pudil, P., J. Novovičová, and J. Kittler. 1994. “Floating Search Methods in Feature Selection.” *Pattern Recognition Letters* 15: 1119–1125.

Rencher, A. C. 1995. *Methods of Multivariate Analysis.* New York: Wiley.

Roweis, S. T., and L. K. Saul. 2000. “Nonlinear Dimensionality Reduction by Locally Linear Embedding.” *Science* 290: 2323–2326.

Saul, K. K., and S. T. Roweis. 2003. “Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds.” *Journal of Machine Learning Research* 4: 119–155.

Tenenbaum, J. B., V. de Silva, and J. C. Langford. 2000. “A Global Geometric Framework for Nonlinear Dimensionality Reduction.” *Science* 290: 2319–2323.

Tipping, M. E., and C. M. Bishop. 1999. “Probabilistic Principal Components Analysis.” *Journal of the Royal Statistical Society Series B* 61: 611–622.

Turk, M., and A. Pentland. 1991. “Eigenfaces for Recognition.” *Journal of Cognitive Neuroscience* 3: 71–86.

Webb, A. 1999. *Statistical Pattern Recognition.* London: Arnold.