CHAPTER

Feature Generation I: Data Transformation and Dimensionality Reduction

6

6.1 INTRODUCTION

Feature generation is of paramount importance in any pattern recognition task. Given a set of measurements, the goal is to discover compact and informative representations of the obtained data. A similar process is also taking place in the human perception apparatus. Our mental representation of the world is based on a relatively small number of perceptually relevant features. These are generated after processing a large amount of sensory data, such as the intensity and the color of the pixels of the images sensed by our eyes, and the power spectra of the sound signals sensed by our ears.

The basic approach followed in this chapter is to transform a given set of measurements to a new set of features. If the transform is suitably chosen, transform domain features can exhibit high *information packing* properties compared with the original input samples. This means that most of the classification-related information is "squeezed" in a relatively small number of features, leading to a reduction of the necessary feature space dimension. Sometimes we refer to such processing tasks as *dimensionality reduction* techniques.

The basic reasoning behind transform-based features is that an appropriately chosen transform can exploit and remove information redundancies, which usually exist in the set of samples obtained by the measuring devices. Let us take for example an image resulting from a measuring device, for example, X-rays or a camera. The pixels (i.e., the input samples) at the various positions in the image have a large degree of correlation, due to the internal morphological consistencies of real-world images that distinguish them from noise. Thus, if one uses the pixels as features, there will be a large degree of redundant information. Alternatively, if one obtains the Fourier transform, for example, of a typical real-world image, it turns out that most of the energy lies in the low-frequency components, due to the high correlation between the pixels' gray levels. Hence, using the Fourier coefficients as features seems a reasonable choice, because the low-energy, high-frequency coefficients can be neglected, with little loss of information. In this chapter we will

see that the Fourier transform is just one of the tools from a palette of possible transforms.

6.2 BASIS VECTORS AND IMAGES

Let $x(0), x(1), \ldots, x(N-1)$ be a set of input samples and $\boldsymbol{x}$ be the $N \times 1$ corresponding vector,

$$\boldsymbol{x}^T = [x(0), \ldots, x(N-1)]$$

Given a unitary $N \times N$ matrix A,[1] we define the transformed vector $\boldsymbol{y}$ of $\boldsymbol{x}$ as

$$\boldsymbol{y} = A^H \boldsymbol{x} \equiv \begin{bmatrix} \boldsymbol{a}_0^H \\ \vdots \\ \boldsymbol{a}_{N-1}^H \end{bmatrix} \boldsymbol{x} \tag{6.1}$$

where H denotes the Hermitian operation, that is, complex conjugation and transposition. From (6.1) and the definition of unitary matrices we have

$$\boldsymbol{x} = A\boldsymbol{y} = \sum_{i=0}^{N-1} y(i)\boldsymbol{a}_i \tag{6.2}$$

The columns of A, $\boldsymbol{a}_i, i = 0, 1, \ldots, N-1$, are called the *basis vectors* of the transform. The elements $y(i)$ of $\boldsymbol{y}$ are nothing but the projections of $\boldsymbol{x}$ onto these basis vectors. Indeed, taking the inner product of $\boldsymbol{x}$ with $\boldsymbol{a}_j$ we have

$$\langle \boldsymbol{a}_j, \boldsymbol{x} \rangle \equiv \boldsymbol{a}_j^H \boldsymbol{x} = \sum_{i=0}^{N-1} y(i) \langle \boldsymbol{a}_j, \boldsymbol{a}_i \rangle = \sum_{i=0}^{N-1} y(i)\delta_{ij} = y(j) \tag{6.3}$$

This is due to the unitary property of A, that is, $A^H A = I$ or $\langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle = \boldsymbol{a}_i^H \boldsymbol{a}_j = \delta_{ij}$.

In many problems, such as in image analysis, the input set of samples is a two-dimensional sequence $X(i,j), i,j = 0, 1, \ldots, N-1$, defining an $N \times N$ matrix X instead of a vector. In such cases, one can define an equivalent N^2 vector $\boldsymbol{x}$, for example, by ordering the rows of the matrix one after the other (*lexicographic ordering*)

$$\boldsymbol{x}^T = [X(0,0), \ldots, X(0,N-1), \ldots, X(N-1,0), \ldots, X(N-1,N-1)]$$

and then transform this equivalent vector. However, this is not the most efficient way to work. The number of operations required to multiply an $N^2 \times N^2$ square matrix (A) with an $N^2 \times 1$ vector $\boldsymbol{x}$ is of the order of $O(N^4)$, which is prohibitive

[1] A complex matrix is called unitary if $A^{-1} = A^H$. Real matrices are equivalently called *orthogonal* if $A^{-1} = A^T$.

for many applications. An alternative possibility is to transform matrix X via a set of *basis matrices* or *basis images*. Let U and V be unitary $N \times N$ matrices. Define the transformed matrix Y of X as

$$Y = U^H X V \tag{6.4}$$

or

$$X = U Y V^H \tag{6.5}$$

The number of operations is now reduced to $O(N^3)$. Equation (6.5) can alternatively be written (Problem 6.1) as

$$X = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} Y(i,j) \boldsymbol{u}_i \boldsymbol{v}_j^H \tag{6.6}$$

where $\boldsymbol{u}_i$ are the column vectors of U and $\boldsymbol{v}_j$ the column vectors of V. Each of the outer products $\boldsymbol{u}_i \boldsymbol{v}_j^H$ is an $N \times N$ matrix

$$\boldsymbol{u}_i \boldsymbol{v}_j^H = \begin{bmatrix} u_{i0} v_{j0}^* & \cdots & u_{i0} v_{jN-1}^* \\ \vdots & \vdots & \vdots \\ u_{iN-1} v_{j0}^* & \cdots & u_{iN-1} v_{jN-1}^* \end{bmatrix} \equiv \mathcal{A}_{ij}$$

and (6.6) is an expansion of matrix X in terms of these N^2 basis images (matrices). The $*$ denotes complex conjugation. Furthermore, if Y turns out to be diagonal, then (6.6) becomes

$$X = \sum_{i=0}^{N-1} Y(i,i) \boldsymbol{u}_i \boldsymbol{v}_i^H$$

and the number of basis images is reduced to N. An interpretation similar to (6.3) is also possible. To this end, let us define the inner product between two matrices as

$$\langle A, B \rangle \equiv \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} A^*(m,n) B(m,n) \tag{6.7}$$

Then it is not difficult to show that (Problem 6.1)

$$Y(i,j) = \langle \mathcal{A}_{ij}, X \rangle \tag{6.8}$$

In words, the (i,j) element of the transformed matrix results from multiplying each element of X by the conjugate of the corresponding element of $\mathcal{A}_{ij}$ and summing up all products.

Transformations of the type (6.4) are also known as *separable* (Problem 6.2). The reason is that one can look at them as a succession of one-dimensional transforms, first applied on column vectors and then on row vectors. For example, the intermediate result in (6.4), $Z = U^H X$, is equivalent to N transforms applied to the column

vectors of X, and $(U^H X)V = (V^H Z^H)^H$ is equivalent to a second sequence of N transforms acting upon the rows of Z. All the two-dimensional transforms that we will deal with in this chapter are separable ones.

Example 6.1

Given the image X and the orthogonal transform matrix U

$$X = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}, \quad U = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

the transformed image $Y = U^T X U$ is

$$Y = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 4 & -1 \\ -1 & 0 \end{bmatrix}$$

The corresponding basis images are

$$\mathcal{A}_{00} = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} [1,1] = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

$$\mathcal{A}_{11} = \frac{1}{2} \begin{bmatrix} 1 \\ -1 \end{bmatrix} [1,-1] = \frac{1}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

and similarly

$$\mathcal{A}_{01} = \mathcal{A}_{10}^T = \frac{1}{2} \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

Now verify that the elements of Y are obtained via the matrix inner products $\langle \mathcal{A}_{ij}, X \rangle$.

6.3 THE KARHUNEN–LOÈVE TRANSFORM

In Section 5.8 the problem of the linear transformation of a feature vector was considered in the spirit of linear discriminant analysis (LDA). The class labels of the feature vectors were assumed known, and this information was optimally exploited to compute the transformation matrix. The linear transform task will also be considered in this section but from a different perspective. Here, the computation of the transformation matrix will exploit the statistical information describing the data, and it will take place in an unsupervised mode. The Karhunen–Loève transform or principal component analysis (PCA), as it is also known, is one of the most popular methods for feature generation and dimensionality reduction in pattern recognition. Though an old technique, it is still in use, and it forms the basis for a number of more advanced approaches.

Let $\boldsymbol{x}$ be the vector of input samples. In the case of an image array, $\boldsymbol{x}$ may be formed by lexicographic ordering of the array elements. In order to simplify the

presentation, we will assume that the data samples have zero mean. If this is not the case, we can always subtract the mean value. We have already mentioned that a desirable property of the generated features is to be mutually uncorrelated in an effort to avoid information redundancies. We begin this section by first developing a method that generates mutually uncorrelated features, that is, $E[y(i)y(j)] = 0$, $i \neq j$. Let[2]

$$\boldsymbol{y} = A^T \boldsymbol{x} \tag{6.9}$$

Since we have assumed that $E[\boldsymbol{x}] = \boldsymbol{0}$, it is readily seen that $E[\boldsymbol{y}] = \boldsymbol{0}$. From the definition of the correlation matrix we have

$$R_y \equiv E[\boldsymbol{y}\boldsymbol{y}^T] = E[A^T \boldsymbol{x}\boldsymbol{x}^T A] = A^T R_x A \tag{6.10}$$

In practice, R_x is estimated as an average over the given set of training vectors. For example, if we are given n data vectors $\boldsymbol{x}_k$, $k = 1, 2, \ldots, n$, then

$$R_x \approx \frac{1}{n} \sum_{k=1}^{n} \boldsymbol{x}_k \boldsymbol{x}_k^T \tag{6.11}$$

Note that R_x is a symmetric matrix, and hence its eigenvectors are mutually orthogonal (Appendix B). Thus, if matrix A is chosen so that its columns are the orthonormal eigenvectors $\boldsymbol{a}_i$, $i = 0, 1, \ldots, N-1$, of R_x, then R_y is diagonal (Appendix B)

$$R_y = A^T R_x A = \Lambda \tag{6.12}$$

where Λ is the diagonal matrix having as elements on its diagonal the respective eigenvalues λ_i, $i = 0, 1, \ldots, N-1$, of R_x. (Recall that in Section 5.8 a linear transform of the form in (6.9) was also considered, but there the elements of matrix A were computed so that a class separability criterion could be optimized.) Furthermore, assuming R_x to be positive definite (Appendix B) the eigenvalues are positive. The resulting transform is known as the *Karhunen–Loève (KL)* transform, and it achieves our original goal of generating mutually uncorrelated features. The KL transform was introduced in [Karh 46] in the context of representing a random process in terms of orthogonal functions and in the discrete form used in this section in [Hote 33]. Other classical references of the topic are [Diam 96, Joll 86].

It has to be emphasized that the solution provided by the KL transform is not a unique one, and it was obtained by imposing an orthogonal structure on matrix A ($A^T A = I$). Also, note that for zero mean variables the correlation matrix R coincides with the covariance matrix Σ. As a matter of fact, a direct consequence of the respective definitions is that

$$\Sigma_x = R_x - E[\boldsymbol{x}]E[\boldsymbol{x}]^T$$

[2] We deal with real data. The complex case is a straightforward extension.

In case the zero mean assumption is not valid, the condition for uncorrelated variables becomes $E[(y(i) - E[y(i)])(y(j) - E[y(j)])] = 0,\ i \neq j$, and the problem results in the eigendecomposition of the covariance matrix, that is,

$$\Sigma_y = A^T \Sigma_x A = \Lambda \tag{6.13}$$

Although our starting point was to generate mutually uncorrelated features, the KL transform turns out to have a number of other important properties, which provide different ways for its interpretation and also the secret for its popularity.

Mean Square Error Approximation

From Eqs. (6.2) and (6.3) we have

$$\boldsymbol{x} = \sum_{i=0}^{N-1} y(i)\boldsymbol{a}_i \quad \text{and} \quad y(i) = \boldsymbol{a}_i^T \boldsymbol{x} \tag{6.14}$$

Let us now define a new vector in the m-dimensional subspace

$$\hat{\boldsymbol{x}} = \sum_{i=0}^{m-1} y(i)\boldsymbol{a}_i \tag{6.15}$$

where only m of the basis vectors are involved. Obviously, this is nothing but the projection of $\boldsymbol{x}$ onto the subspace spanned by the m (orthonormal) eigenvectors involved in the summation. If we try to approximate $\boldsymbol{x}$ by its projection $\hat{\boldsymbol{x}}$, the resulting mean square error is given by

$$E\left[\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2\right] = E\left[\left\|\sum_{i=m}^{N-1} y(i)\boldsymbol{a}_i\right\|^2\right] \tag{6.16}$$

Our goal now is to choose the eigenvectors that result in the minimum MSE. From (6.16) and taking into account the orthonormality property of the eigenvectors, we have

$$E\left[\left\|\sum_{i=m}^{N-1} y(i)\boldsymbol{a}_i\right\|^2\right] = E\left[\sum_i \sum_j (y(i)\boldsymbol{a}_i^T)(y(j)\boldsymbol{a}_j)\right] \tag{6.17}$$

$$= \sum_{i=m}^{N-1} E[y^2(i)] = \sum_{i=m}^{N-1} \boldsymbol{a}_i^T E[\boldsymbol{x}\boldsymbol{x}^T]\boldsymbol{a}_i \tag{6.18}$$

Combining this with (6.16) and the eigenvector definition, we finally get

$$E\left[\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2\right] = \sum_{i=m}^{N-1} \boldsymbol{a}_i^T \lambda_i \boldsymbol{a}_i = \sum_{i=m}^{N-1} \lambda_i \tag{6.19}$$

Thus, if we choose in (6.15) the eigenvectors corresponding to the *m largest* eigenvalues of the *correlation matrix*, then the error in (6.19) is *minimized, being the sum of the $N - m$ smallest eigenvalues. Furthermore, it can be shown*

(Problem 6.3) that this is also the minimum MSE, compared with any other approximation of $\boldsymbol{x}$ *by an m-dimensional vector*. This is the reason that the KL transform is also known as *principal component analysis* (PCA).

A difficulty in practice is how to choose the m principal components. One way is to rank the eigenvalues in descending order, $\lambda_0 \geq \lambda_1 \geq \ldots \geq \lambda_{m-1} \geq \lambda_m \geq \ldots \geq \lambda_{N-1}$, and determine m so that the gap between the values λ_{m-1} and λ_m is "large." For more on this issue, see [Jack 91].

Note that the previous analysis concerning the MSE property of the KL transform, after projecting onto the m principal components of R_x, is still valid even if the mean of the data is not zero. However, in this case, although a minimum MSE solution is obtained, the approximation is not, in general, a good one (why?) In such cases, one tries to find the optimum m-dimensional subspace, so that the MSE between $\boldsymbol{x}$ and its following approximation

$$\hat{\boldsymbol{x}} = \sum_{i=0}^{m-1} y(i)\hat{\boldsymbol{a}}_i + \sum_{i=m}^{N-1} b_i \hat{\boldsymbol{a}}_i, \quad y(i) \equiv \hat{\boldsymbol{a}}_i^T \boldsymbol{x} \tag{6.20}$$

to be minimum, where b_i, $i = m, \ldots, N-1$, are constants independent of $\boldsymbol{x}$. It turns out (Problem 6.4) that the resulting orthonormal basis consists of the eigenvectors of the covariance matrix, Σ_x, where $\hat{\boldsymbol{a}}_i$, $i = 0, 2, \ldots, m-1$, correspond to the principal eigenvalues of Σ_x, and the constants are equal to

$$b_i = E[y(i)] = \hat{\boldsymbol{a}}_i^T E[\boldsymbol{x}], \quad i = m, \ldots, N-1$$

In other words, $\boldsymbol{x}$ is projected onto the subspace spanned by the m principal components of Σ_x and the rest $N - m$ components are frozen to the respective mean values, in order to bring the estimate closer to its mean. Note, however, that the number of free parameters remains equal to m. The optimality of the KL transform, with respect to the MSE approximation, leads to excellent information packing properties and offers us a tool to select the m dominant features out of N measurement samples. However, although this may be a good criterion, in many cases it does not necessarily lead to maximum class separability in the lower dimensional subspace. This is reasonable, since the dimensionality reduction is not optimized with respect to class separability, as was, for example, the case with the scattering matrix criteria of the previous chapter. This is demonstrated via the example of Figure 6.1. The feature vectors in the two classes follow the Gaussian distribution with the same covariance matrix. The ellipses show the curves of constant pdf values. We have computed the eigenvectors of the overall correlation matrix, and the resulting eigenvectors are shown in the figure. Eigenvector $\boldsymbol{a}_0$ is the one that corresponds to the largest eigenvalue. It does not take time for someone to realize that projection on $\boldsymbol{a}_0$ makes the two classes almost coincide. However, projecting on $\boldsymbol{a}_1$ keeps the two class separable.

Total Variance

Let $E[\boldsymbol{x}]$ be zero. Let $\boldsymbol{y}$ be the KL transformed vector of $\boldsymbol{x}$. From the respective definitions we have that $\sigma^2_{y(i)} \equiv E[y^2(i)] = \lambda_i$. That is, *the eigenvalues of the input*

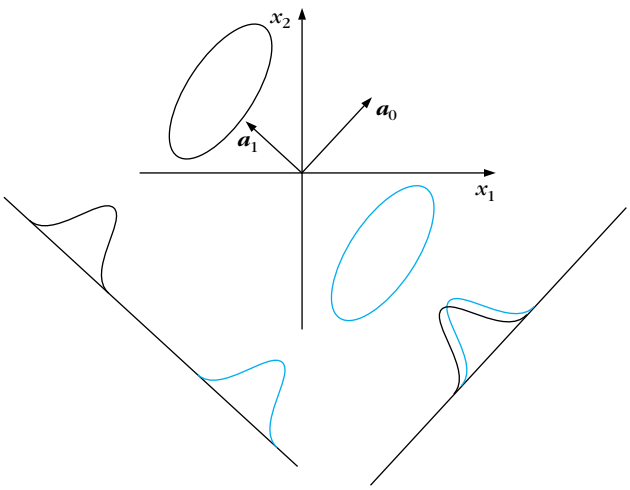


FIGURE 6.1

The KL transform is not always best for pattern recognition. In this example, projection on the eigenvector with the larger eigenvalue makes the two classes coincide. On the other hand, projection on the other eigenvector keeps the classes separated.

correlation matrix are equal to the variances of the transformed features. Thus, selecting those features, $y(i) \equiv \boldsymbol{a}_i^T \boldsymbol{x}$, corresponding to the m largest eigenvalues makes their sum variance $\sum_i \lambda_i$ maximum. In other words, the selected m features retain most of the total variance associated with the original random variables $x(i)$. Indeed, the latter is equal to the trace of R_x, which we know from linear algebra to be equal to the sum of the eigenvalues $\sum_{i=0}^{N-1} \lambda_i$ [Stra 80]. It can be shown that this is a more general property. That is, from all possible sets of m features, obtained via any orthogonal linear transformation on $\boldsymbol{x}$, the ones resulting from the KL transform have the largest sum variance (Problem 6.3). If the mean value is not zero, to maximize the sum variance, we use Σ_x in place of R_x.

Entropy

We know from Chapter 2 that the entropy of a process is defined as

$$H_y = -E[\ln p_y(\boldsymbol{y})]$$

and it is a measure of the randomness of the process. For a zero mean Gaussian multivariable m-dimensional process, the entropy becomes

$$H_y = \frac{1}{2} E[\boldsymbol{y}^T R_y^{-1} \boldsymbol{y}] + \frac{1}{2} \ln |R_y| + \frac{m}{2} \ln(2\pi) \tag{6.21}$$

However,

$$E[\boldsymbol{y}^T R_y^{-1} \boldsymbol{y}] = E[\text{trace}\{\boldsymbol{y}^T R_y^{-1} \boldsymbol{y}\}] = E[\text{trace}\{R_y^{-1} \boldsymbol{y}\boldsymbol{y}^T\}] = \text{trace}(I) = m$$

and using the known property from linear algebra the determinant is

$$\ln|R_y| = \ln(\lambda_0\lambda_1 \ldots \lambda_{m-1})$$

In words, selection of the m features that correspond to the m largest eigenvalues maximizes the entropy of the process. This is expected because variance and randomness are directly related.

Dimensionality Reduction

PCA achieves a *linear transformation* of a high-dimensional input vector into a low-dimensional one whose components are uncorrelated. As already stated, we can assume that $E[\boldsymbol{x}]$ is zero, without loss of generality. Assuming that the $N - m$ smallest eigenvalues of the correlation matrix are zero, then Eq. (6.19) suggests that $\boldsymbol{x} = \hat{\boldsymbol{x}}$. In other words, vector $\boldsymbol{x}$ lies in an m-dimensional subspace ([Eq. (6.15)]) of the original N-dimensional space. This brings us to the notion of *intrinsic dimensionality*.

A data set $X \subset \mathcal{R}^N$ is said to have intrinsic dimensionality (ID) $m < N$, if X can be described in terms of m free parameters. For example, if X consists of vectors whose components are functions of m random variables, $x_i = g_i(u_1, u_2, \ldots, u_m)$, $i = 1, 2, \ldots, N, u_i \in \mathcal{R}$, then the intrinsic dimensionality of X is m. The geometric interpretation of this is that the entire data set lies on a m-dimensional hypersurface (manifold) in $\mathcal{R}^N$. Take as an example the case of a random variable θ and the functions

$$x_1 = r\cos\theta,\ x_2 = r\sin\theta$$

It does not take time to see that $\boldsymbol{x} = [x_1, x_2]^T$ lies on the perimeter of the circle with radius equal to r. This is a one-dimensional surface (manifold) since one parameter suffices to describe the data (the length across the circumference from a point, origin, on the perimeter of the circle). From a statistical point of view, the fact that the intrinsic or "effective" dimension is smaller than the "apparent" one means that the features in the data set are correlated.

The PCA method has been used extensively for dimensionality reduction and for estimation of the ID of a data set. If ID $= m < N$, then in theory there will be $N - m$ zero eigenvalues. In practice, one has to ignore the eigenvalues with small values, and thus an approximation of the ID is obtained. PCA, being a linear projection method, works well if the data points are distributed, more or less, throughout a hyperplane. The eigenvalue–eigenvector decomposition of the correlation (covariance) matrix reveals the dimensionality of this hyperplane across which data are spread; in other words, dimensionality is a measure of the number of underlying modes of variability. Recall that PCA projects across the directions of maximum variance.

For more general cases, however where the generation mechanism of the data is highly nonlinear and they lie on more complicated manifolds, PCA fails, and it tends to overestimate the true value of the ID. For example, for the case considered before,

where all the data points lie on the perimeter of a circle in the two-dimensional space, PCA would result in ID = 2, although the true value is ID = 1. For such cases, nonlinear dimensionality reduction techniques have been developed and used. For example, in [Karh 94], a special type of neural network with three hidden layers is proposed to perform nonlinear PCA. We will return to this issue in Section 6.7.

As we have seen in Chapter 5, LDA is another linear method that has been used for dimensionality reduction. However, in the case of LDA, this is achieved in a supervised manner; that is, the lower dimensional space is chosen in order to preserve a class separability measure. Another linear technique used to project in a lower dimensional space, while respecting certain constraints, is the *metric multidimensional scaling* (MDS). Given the set $X \subset \mathcal{R}^N$, the goal is to project into a lower dimensional space, $Y \subset \mathcal{R}^m$, so that inner products are optimally preserved, that is,

$$E = \sum_i \sum_j \left(\boldsymbol{x}_i^T \boldsymbol{x}_j - \boldsymbol{y}_i^T \boldsymbol{y}_j \right)^2$$

is minimized, where $\boldsymbol{y}_i$ is the image of $\boldsymbol{x}_i$ and the sum runs over all the training points in X. The problem is similar to the PCA, and it can be shown that the solution is given by the eigendecomposition of the Gram matrix, whose elements are defined as

$$\mathcal{K}(i,j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$$

Another side of the same coin is to require the Euclidean distances, instead of the inner products, to be optimally preserved. A Gram matrix, consistent with the squared Euclidean distances can then be formed, leading to the same solution as before. It turns out that the solutions obtained by PCA and MDS are equivalent. We can see this by the following simple reasoning. PCA performs the eigen decomposition of the correlation matrix R_x, which is approximated by

$$R_x = E[\boldsymbol{x}\boldsymbol{x}^T] \approx \frac{1}{n} \sum_{k=1}^{n} \boldsymbol{x}_k \boldsymbol{x}_k^T = \frac{1}{n} X^T X \tag{6.22}$$

where, as we have defined in (3.44),

$$X^T = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$$

On the other hand, the Gram matrix can also be written as

$$\mathcal{K} = XX^T$$

However, as we will see in more detail in Section 6.4, the two matrices $X^T X$ and XX^T are of the same rank and have the same eigenvalues. Their eigenvectors, although different, are related.

More on these issues can be found in, for example, [Cox 94, Burg 04]. As we will see in Section 6.6, the main idea behind MDS of preserving the distances is used, in one way or another, in a number of more recently developed nonlinear dimensionality reduction techniques.

Remarks

- The concept of principal eigenvector subspace has also been exploited as a classifier. First, the sample mean of the whole training set is subtracted from the feature vectors. For each class, ω_i, the correlation matrix R_i is estimated and the principal m eigenvectors (corresponding to the m largest eigenvalues) are computed. A matrix A_i is then formed using the respective eigenvectors as columns. An unknown feature vector $\boldsymbol{x}$ is then classified in the class ω_j for which

$$\|A_j^T \boldsymbol{x}\| > \|A_i^T \boldsymbol{x}\|, \quad \forall i \neq j \tag{6.23}$$

that is, the class corresponding to the maximum norm subspace projection of $\boldsymbol{x}$ [Wata 73]. From the Pythagoras theorem this is equivalent to classifying a vector in its *nearest class subspace*. The decision surfaces are hyperplanes if all the subspaces have the same dimension or quadric surfaces in the more general case. *Subspace classification integrates the stages of feature generation/selection and classifier design*.

If this approach results in a relatively high classification error, the performance may be improved by suitable modifications known as *learning subspace methods*. For example, one can iteratively rotate the subspaces to adjust the lengths of the projections of the training vectors. The basic idea is to increase the length of a projection in the subspace of the correct class and decrease it for the rest. Such techniques have been applied successfully in a number of applications, such as speech recognition, texture classification, and character recognition. The interested reader may consult, for example, [Oja 83, Koho 89, Prak 97].

- For the computation of the correlation matrix eigenvectors, a number of iterative schemes have been developed. The computation is performed working directly with the vectors, without having to estimate the corresponding correlation matrix, using neural network concepts [Oja 83, Diam 96].

Example 6.2

The correlation matrix of a vector $\boldsymbol{x}$ is given by

$$R_x = \begin{bmatrix} 0.3 & 0.1 & 0.1 \\ 0.1 & 0.3 & -0.1 \\ 0.1 & -0.1 & 0.3 \end{bmatrix}$$

Compute the KL transform of the input vector.

The eigenvalues of R_x are $\lambda_0 = \lambda_1 = 0.4$, $\lambda_2 = 0.1$. Since the matrix R_x is symmetric, we can always construct orthonormal eigenvectors. For this case we have

$$\boldsymbol{a}_0 = \frac{1}{\sqrt{6}} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{a}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, \quad \boldsymbol{a}_2 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

The KL transform is then given by

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \end{bmatrix} = \begin{bmatrix} 2/\sqrt{6} & 1/\sqrt{6} & 1/\sqrt{6} \\ 0 & 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{3} & -1/\sqrt{3} & -1/\sqrt{3} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \end{bmatrix}$$

where $y(0)$, $y(1)$ correspond to the two largest eigenvalues.

Example 6.3

Figure 6.2 shows 100 points in the two-dimensional space. The points spread around the $x_2 = x_1$ line, and they have been generated by the model $x_2 = x_1 + \epsilon$, where ϵ is a noise source following the uniform distribution in $[-0.5, 0.5]$.

We first compute the covariance matrix and perform an eigendecomposition. The resulting eigenvectors are

$$\boldsymbol{a}_0 = [0.7045, 0.7097]^T, \quad \boldsymbol{a}_1 = [-0.7097, 0.7045]^T$$

corresponding to the eigenvalues

$$\lambda_0 = 17.26, \quad \lambda_1 = 0.04$$

respectively. Observe that $\lambda_0 >> \lambda_1$. Figure 6.2 shows the two eigenvectors. $\boldsymbol{a}_0$, which correspond to the largest eigenvalue, points in the direction where data show maximum variability. Projecting along this direction retains most of the variance. Moreover, according to PCA, the dimensionality of the set is approximately one, due to the large gap between λ_0 and λ_1, which is the correct answer. Also, note, that $\boldsymbol{a}_0$, is (approximately) parallel to the line $x_2 = x_1$.

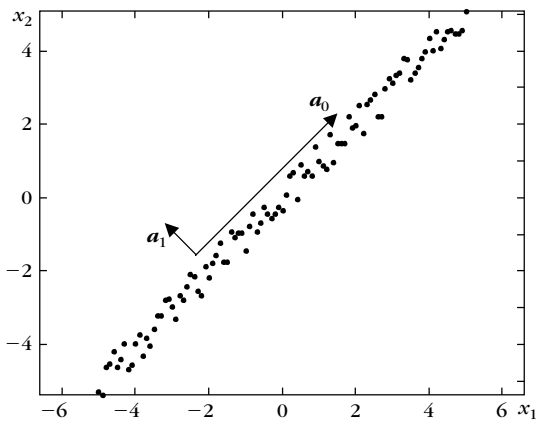


FIGURE 6.2

Points around the $x_2 = x_1$ line. The eigenvectors of the associated covariance matrix are $\boldsymbol{a}_0$ and $\boldsymbol{a}_1$. The principal eigenvector $\boldsymbol{a}_0$ points in the direction of maximum variance.

6.4 THE SINGULAR VALUE DECOMPOSITION

The singular value decomposition of a matrix is one of the most elegant and powerful algorithms in linear algebra, and it has been extensively used for rank and dimension reduction in pattern recognition and information retrieval applications. Given a $l \times n$ matrix X of rank r (obviously $r \leq \min\{l, n\}$), we will show that there exist unitary matrices U and V of dimensions $l \times l$ and $n \times n$, respectively, so that

$$X = U \begin{bmatrix} \Lambda^{\frac{1}{2}} & \mathrm{O} \\ \mathrm{O} & 0 \end{bmatrix} V^H \quad \text{or} \quad Y \equiv \begin{bmatrix} \Lambda^{\frac{1}{2}} & \mathrm{O} \\ \mathrm{O} & 0 \end{bmatrix} = U^H X V \tag{6.24}$$

where $\Lambda^{\frac{1}{2}}$ is the $r \times r$ diagonal matrix with elements $\sqrt{\lambda_i}$, and λ_i are the r nonzero eigenvalues of the associated matrix $X^H X$. O denotes a zero element matrix. *In other words, there exist unitary matrices U and V that transform X into the special diagonal structure of Y.* If $\boldsymbol{u}_i$, $\boldsymbol{v}_i$ denote the column vectors of matrices U and V, respectively, then Eq. (6.24) is written as

$$X = [\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{r-1}] \begin{bmatrix} \sqrt{\lambda_0} & & & \\ & \sqrt{\lambda_1} & & \\ & & \ddots & \\ & & & \sqrt{\lambda_{r-1}} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_0^H \\ \boldsymbol{v}_1^H \\ \vdots \\ \boldsymbol{v}_{r-1}^H \end{bmatrix} \tag{6.25}$$

or

$$X = \sum_{i=0}^{r-1} \sqrt{\lambda_i} \boldsymbol{u}_i \boldsymbol{v}_i^H \tag{6.26}$$

Sometimes, the above is also written as

$$X = U_r \Lambda^{\frac{1}{2}} V_r^H \tag{6.27}$$

where U_r denotes the $l \times r$ matrix that consists of the first r columns of U and V_r the $r \times n$ matrix formed by using the first r columns of V. More precisely, $\boldsymbol{u}_i$, $\boldsymbol{v}_i$ are the eigenvectors corresponding to the nonzero eigenvalues of the matrices XX^H and $X^H X$, respectively. The eigenvalues λ_i are known as *singular values* of X and the expansion in (6.26) as the *singular value decomposition* (SVD) of X or the *spectral representation* of X.

Proof. Given a matrix X of rank r, it is known from linear algebra [Stra 80] that the $n \times n$ matrix $X^H X$ as well as the $l \times l$ matrix XX^H are of the same rank r. Furthermore, both matrices have the same nonzero eigenvalues but different (yet related) eigenvectors (Problem 6.5),

$$XX^H \boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i \tag{6.28}$$

$$X^H X \boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \tag{6.29}$$

Since both matrices are Hermitian and nonnegative (i.e., $(XX^H)^H = XX^H$), they have nonnegative real eigenvalues and orthogonal eigenvectors (Appendix B). The eigenvectors, given in (6.28) and (6.29), can also be normalized to become orthonormal, that is, $\boldsymbol{u}_i^H \boldsymbol{u}_i = 1$ and $\boldsymbol{v}_i^H \boldsymbol{v}_i = 1$. It is straightforward to see from (6.28) and (6.29) that

$$\boldsymbol{u}_i = \frac{1}{\sqrt{\lambda_i}} X \boldsymbol{v}_i, \quad \text{for } \lambda_i \neq 0 \tag{6.30}$$

Indeed, premultiplying (6.29) by X results in

$$(XX^H) X \boldsymbol{v}_i = \lambda_i X \boldsymbol{v}_i$$

That is, $\boldsymbol{u}_i = \alpha X \boldsymbol{v}_i$, where (without loss of generality) the scaling factor α can be taken as positive and it is found from

$$\|\boldsymbol{u}_i\|^2 = 1 = \alpha^2 \boldsymbol{v}_i^H X^H X \boldsymbol{v}_i = \alpha^2 \lambda_i \|\boldsymbol{v}_i\|^2 \Rightarrow \alpha = \frac{1}{\sqrt{\lambda_i}}$$

Let us now assume that $\boldsymbol{u}_i, \boldsymbol{v}_i,\ i = 0, 1, \ldots, r-1$, are the eigenvectors corresponding to the nonzero eigenvalues and $\boldsymbol{u}_i,\ i = r, \ldots, l-1,\ \boldsymbol{v}_i,\ i = r, \ldots, n-1$, to the zero ones. Then, for the latter case we have

$$X^H X \boldsymbol{v}_i = 0 \Rightarrow \boldsymbol{v}_i^H X^H X \boldsymbol{v}_i = 0 \Rightarrow \|X \boldsymbol{v}_i\|^2 = 0$$

Hence

$$X \boldsymbol{v}_i = \boldsymbol{0}, \quad i = r, \ldots, n-1 \tag{6.31}$$

In a similar way one can show that

$$X^H \boldsymbol{u}_i = \boldsymbol{0}, \quad i = r, \ldots, l-1 \tag{6.32}$$

Combining (6.30) and (6.31), we show that the right-hand side of (6.26) is

$$\sum_{i=0}^{r-1} \sqrt{\lambda_i} \boldsymbol{u}_i \boldsymbol{v}_i^H = X \sum_{i=0}^{r-1} \sqrt{\lambda_i} \frac{1}{\sqrt{\lambda_i}} \boldsymbol{v}_i \boldsymbol{v}_i^H = X \sum_{i=0}^{n-1} \boldsymbol{v}_i \boldsymbol{v}_i^H \tag{6.33}$$

Let us now define a matrix V that has as columns the orthonormal eigenvectors $\boldsymbol{v}_i$,

$$V = [\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n-1}]$$

Orthonormality of the columns results in $V^H V = I$; that is, V is unitary and hence $VV^H = I$. Thus, it turns out that

$$I = VV^H = [\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n-1}] \begin{bmatrix} \boldsymbol{v}_0^H \\ \vdots \\ \boldsymbol{v}_{n-1}^H \end{bmatrix} = \sum_{i=0}^{n-1} \boldsymbol{v}_i \boldsymbol{v}_i^H \tag{6.34}$$

From (6.33) and (6.34) we obtain

$$X = \sum_{i=0}^{r-1} \sqrt{\lambda_i} \boldsymbol{u}_i \boldsymbol{v}_i^H \tag{6.35}$$

and X can be written as

$$X = U \begin{bmatrix} \Lambda^{\frac{1}{2}} & \mathrm{O} \\ \mathrm{O} & 0 \end{bmatrix} V^H \tag{6.36}$$

where U is the unitary matrix with columns the orthonormal eigenvectors $\boldsymbol{u}_i$. □

Low Rank Approximation

The expansion in (6.26) is an exact representation of matrix X. A very interesting implication occurs if one uses less than r (the rank of X) terms in the summation. Let X be approximated by

$$X \simeq \hat{X} = \sum_{i=0}^{k-1} \sqrt{\lambda_i} \boldsymbol{u}_i \boldsymbol{v}_i^H, \quad k \leq r \tag{6.37}$$

Matrix $\hat{X}$, being the sum of $k \leq r$ rank-one independent $l \times n$ matrices, is of rank k. If the k largest eigenvalues are involved, it can be shown that the squared error

$$\epsilon^2 = \sum_{i=0}^{l-1} \sum_{j=0}^{n-1} |X(i,j) - \hat{X}(i,j)|^2 \tag{6.38}$$

is the minimum one with respect to all rank-k $l \times n$ matrices. The square root of the right-hand side in (6.38) is also known as the Frobenius norm $\|X - \hat{X}\|_F$ of the difference matrix $X - \hat{X}$. The error in the approximation turns out to be (Problem 6.6)

$$\epsilon^2 = \sum_{i=k}^{r-1} \lambda_i \tag{6.39}$$

Hence, if we order the eigenvalues in descending order, $\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{r-1}$, then for a given number of k terms in the expansion, the SVD leads to the minimum square error. *Thus, $\hat{X}$ is the best rank-k approximation of X in the Frobenius norm sense*. This reminds us of the Karhunen–Loève expansion. However, in the latter case the optimality was with respect to the mean square error. This is a major difference in philosophy between SVD and KL. *The former is related to a single set of samples and the latter to an ensemble of them*.

Dimensionality Reduction

SVD has been used extensively for dimension reduction in pattern recognition and information retrieval, and it forms the basis of what is known as *latent semantics indexing*, see, for example, [Berr 95]. Adopting the notation used in (6.25) and

(6.27), Eq. (6.37) can be written as

$$X \simeq \hat{X} = [\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{k-1}] \begin{bmatrix} \sqrt{\lambda_0}\boldsymbol{v}_0^H \\ \sqrt{\lambda_1}\boldsymbol{v}_1^H \\ \vdots \\ \sqrt{\lambda_{k-1}}\boldsymbol{v}_{k-1}^H \end{bmatrix}$$

$$= U_k[\boldsymbol{a}_0, \ \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{n-1}] \tag{6.40}$$

where U_k consists of the first k columns of U and the k-dimensional vectors $\boldsymbol{a}_i$, $i = 0, 1, \ldots, n-1$, are the column vectors of the $k \times n$ product matrix $\Lambda_k^{\frac{1}{2}} V_k^H$, where V_k^H consists of the first k rows of V^H and $\Lambda_k^{\frac{1}{2}}$ is the diagonal matrix having elements the square roots of the respective k singular values. Figure 6.3 gives a diagrammatic interpretation of the matrix products involved in SVD. The formulation given in (6.40) suggests that each column vector, $\boldsymbol{x}_i$ of X, is approximated as

$$\boldsymbol{x}_i \simeq U_k \boldsymbol{a}_i = \sum_{m=0}^{k-1} \boldsymbol{u}_m a_i(m), \quad i = 0, 2, \ldots, n-1 \tag{6.41}$$

where $a_i(m)$, $m = 0, 1, \ldots, k-1$, denote the elements of the respective vector $\boldsymbol{a}_i$. In words, the l-dimensional vector $\boldsymbol{x}_i$ is approximated by the k-dimensional vector $\boldsymbol{a}_i$, lying in the subspace spanned by $\boldsymbol{u}_i$, $i = 0, 1, \ldots, k-1$ ($\boldsymbol{a}_i$ is the projection of $\boldsymbol{x}_i$ on this subspace; Problem 6.6.) Furthermore, due to the orthonormality of the columns $\boldsymbol{u}_i$, $i = 0, 1, \ldots, k-1$, of U_k it is straightforward to see that

$$||\boldsymbol{x}_i - \boldsymbol{x}_j|| \simeq ||U_k(\boldsymbol{a}_i - \boldsymbol{a}_j)|| = || \sum_{m=0}^{k-1} \boldsymbol{u}_m (a_i(m) - a_j(m))||$$

$$= ||\boldsymbol{a}_i - \boldsymbol{a}_j||, \quad i,j = 0, 1, \ldots, n-1 \tag{6.42}$$

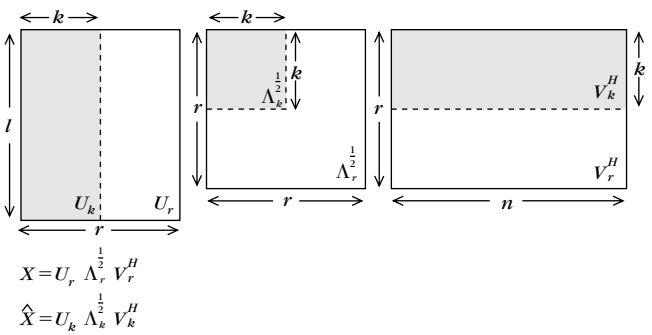


FIGURE 6.3

Diagrammatic interpretation of the matrix products involved in SVD. In the approximation of X by $\hat{X}$, the first k columns of U_r and the first k rows of V_r^H are involved.

where $||\cdot||$ represents the Euclidean norm of a vector. That is, using the previous projection and assuming the approximation to be reasonably good, the Euclidean distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the high l-dimensional space is (approximately) preserved under the projection in the lower k-dimensional subspace.

The previous observation has important implications in applications such as information retrieval. Let us take as an example the simple case where we are given a set of n patterns each represented by a l-dimensional feature vector. These patterns constitute the available database. Given an unknown pattern, the goal is to search for and recover from the database the pattern that is most similar to the unknown one, by computing its Euclidean distance from each vector in the database. When l and n are large numbers this can be a very time-consuming task. A procedure to simplify computations is the following. We form the $l \times n$ data matrix, X,[3] having as columns the n feature vectors. Perform a SVD on X and represent each feature vector, $\boldsymbol{x}_i$, by its lower dimensional projection, $\boldsymbol{a}_i$, as described before. Given the unknown vector, one projects it on the subspace spanned by the columns of U_k and performs Euclidean distance computations in the k-dimensional space. Since Euclidean distances are approximately preserved, one can decide about the proximity of vectors by working in a lower dimensional space. If $k << l$ substantial computational savings are obtained (see, e.g., [Berr 95, Deer 90, Sebr 03]).

SVD builds upon *global* information spread over all the data vectors in X. Indeed, a crucial part of the algorithm is the computation of the eigenvalues of $X^H X$ or XX^H, which, for zero mean data, is directly related to the respective covariance matrix (Eq. 6.22). Hence, the performance of the SVD, as a dimensionality reduction technique, is most effective for cases where data can sufficiently be described in terms of the covariance matrix, for example, to be Gaussian-like distributed. In [Cast 03] a modification of the simple SVD is suggested to account for data with a clustered structure. In Section 6.7, nonlinear dimensionality techniques will be reviewed, where more than a simple linear projection on a subspace is required to reduce dimensionality.

Remarks

- Due to its optimal approximation properties, the SVD transform also has excellent "information packing" properties, and an image array can be represented efficiently by a few of its singular values. Thus, SVD is a natural candidate as a tool for feature generation/selection in classification.
- Performing SVD of large matrices is a computationally expensive task. In order to overcome this drawback, a number of computationally efficient schemes have been developed, see, for example, [Ye 04, Achl 01].

[3] Note that X is defined here as the transpose of the data matrix in (3.44), to comply with the notation used in latent semantics indexing.

Example 6.4

Consider the matrix

$$X = \begin{bmatrix} 6 & 6 \\ 0 & 1 \\ 4 & 0 \\ 0 & 6 \end{bmatrix}$$

The goal is to compute its singular value decomposition.

- Step 1: Find the eigenvalues and eigenvectors of

$$X^T X = \begin{bmatrix} 52 & 36 \\ 36 & 73 \end{bmatrix}$$

  These are $\lambda_0 = 100,\ \lambda_1 = 25$, and the corresponding eigenvectors are $\boldsymbol{v}_0 = [0.6, 0.8]^T$, $\boldsymbol{v}_1 = [0.8, -0.6]^T$.

- Step 2: Compute the eigenvectors of XX^T. This is a 4×4 matrix of rank 2. The eigenvectors corresponding to the nonzero eigenvalues λ_0, λ_1 are computed via (6.30), that is, $\boldsymbol{u}_0 = 0.1X\boldsymbol{v}_0, \boldsymbol{u}_1 = 0.2X\boldsymbol{v}_1$ or $[0.84, 0.08, 0.24, 0.48]^T$ and $[0.24, -0.12, 0.64, -0.72]^T$ respectively.

- Step 3: Compute the SVD of X

$$\begin{aligned} X = \ & 10[0.84, 0.08, 0.24, 0.48]^T [0.6, 0.8] \\ & + 5[0.24, -0.12, 0.64, -0.72]^T [0.8, -0.6] \end{aligned}$$

  If we keep the first of the two terms, then the resulting approximation is the best, in the Frobenius sense, rank-1 approximation of X.

Example 6.5

The goal of this example is to demonstrate the power of the SVD as a dimensionality reduction tool, in the context used in latent semantics indexing in information retrieval.

(a) Let our data set consist of 1000 three-dimensional vectors

$$\boldsymbol{x}_i = [x_1(i), x_2(i), x_3(i)]^T, \quad i = 1, 2, \ldots, 1000$$

This set of points comprises our database. We form the 3×1000 matrix X having these data vectors as columns. For the needs of this example, the components $x_1(i)$, $x_2(i)$ are randomly generated using the uniform distribution in $[-10, 10]$. The value of the third dimension of each point is given by $x_3(i) = -x_1(i) - x_2(i) + \varepsilon$, where ε is a noise source following the uniform distribution in $[-1, 1]$. In other words, our data are crowded around the plane

$$H : x_1 + x_2 + x_3 = 0 \tag{6.43}$$

Performing SVD analysis, it turns out that the singular values are

$$\lambda_0 = 158.43,\ \lambda_1 = 89.01,\ \lambda_2 = 10.50.$$

The relatively small value of λ_2 is the consequence of the fact that our data are approximately two-dimensional. Recall that the singular values are eigenvalues of XX^T, which is the same (within a scaling factor) with the estimate of the correlation matrix used in PCA (Eq. (6.11)). The corresponding (orthonormal) eigenvectors, which are also the column vectors of the U matrix, are (after rounding to the second decimal point)

$$\boldsymbol{u}_0 = \begin{bmatrix} -0.39 \\ -0.42 \\ 0.82 \end{bmatrix},\ \boldsymbol{u}_1 = \begin{bmatrix} 0.71 \\ -0.70 \\ -0.01 \end{bmatrix},\ \boldsymbol{u}_2 = \begin{bmatrix} 0.58 \\ 0.58 \\ 0.57 \end{bmatrix}$$

It is not difficult to verify that the plane (subspace) formed by the two principal eigenvectors is

$$H_1:\ 14.26x_1 + 14.10x_2 + 13.95x_3 = 0$$

which is very close to the hyperplane H in (6.43), around which our data cluster.

(b) We now select randomly six of the points in the data set X and project them along the H_1 plane. The 6×6 distance matrix D whose (i,j) element is the squared Euclidean distance between the ith and jth points for $i,j = 1,2,\ldots,6$, is

$$D = \begin{bmatrix} 0 & 26.17 & 24.70 & 112.25 & 11.92 & 4.81 \\ 26.17 & 0 & 61.46 & 43.96 & 38.33 & 49.25 \\ 24.70 & 61.46 & 0 & 107.97 & 4.34 & 14.51 \\ 112.25 & 43.96 & 107.97 & 0 & 88.72 & 140.18 \\ 11.92 & 38.33 & 4.34 & 88.72 & 0 & 9.95 \\ 4.81 & 49.25 & 14.51 & 140.18 & 9.95 & 0 \end{bmatrix}$$

The corresponding distance matrix, D', for the respective projections on H_1 is

$$D' = \begin{bmatrix} 0 & 25.85 & 24.32 & 112.21 & 11.72 & 4.57 \\ 25.85 & 0 & 61.46 & 43.83 & 37.29 & 49.24 \\ 24.32 & 61.46 & 0 & 107.80 & 3.20 & 14.49 \\ 112.21 & 43.83 & 107.80 & 0 & 88.29 & 140.10 \\ 11.72 & 37.29 & 3.20 & 88.29 & 0 & 9.06 \\ 4.57 & 49.24 & 14.49 & 140.10 & 9.06 & 0 \end{bmatrix}$$

which is in close agreement with D. Note that this good agreement is a consequence of the fact that the true dimensionality of the data is very close to 2. Increasing the variance of the noise source ε, the spread of the data around the plane H would increase and the data would become more and more "three-dimensional." In other words, the higher the variance of ε, the less the agreement between D and D' that one expects to get.

6.5 INDEPENDENT COMPONENT ANALYSIS

As we have already seen, the principal component analysis (PCA) performed by the Karhunen–Loève transform produces features $y(i), i = 0, 1, \ldots, N-1$, that are mutually uncorrelated. The solution obtained by the KL transform solution is optimal when dimensionality reduction is the goal and one wishes to minimize the approximation mean square error. However, for certain applications, such as the one illustrated in Figure 6.1, the obtained solution falls short of the expectations. In contrast, the more recently developed *independent component analysis* (ICA) theory, for example, [Hyva 01, Como 94, Jutt 91, Hayk 00, Lee 98], tries to achieve much more than simple decorrelation of the data. The ICA task is casted as follows: Given the set of input samples $\boldsymbol{x}$, determine an $N \times N$ invertible matrix W such that the entries $y(i), i = 0, 1, \ldots, N-1$, of the transformed vector

$$\boldsymbol{y} = W\boldsymbol{x} \tag{6.44}$$

are mutually independent. The goal of statistical independence is a stronger condition than the uncorrelatedness required by the PCA. The two conditions are equivalent *only* for Gaussian random variables.

Searching for independent rather than uncorrelated features gives us the means of exploiting a lot more information, hidden in the higher order statistics of the data. As the example of Figure 6.1 suggests, constraining the search by digging information in the second-order statistics only results in the least interesting, for our problem, projection direction, that is, that of $\boldsymbol{a}_0$. However, $\boldsymbol{a}_1$ is, no doubt, the most interesting direction from the class separation point of view. In contrast, employing ICA can unveil from the higher order statistics of the data the piece of information that points $\boldsymbol{a}_1$ as the most interesting one. Furthermore, searching for statistically independent features is in line with the way nature builds up the "cognitive" maps of the outside world in the brain, by processing the (input) sensory data. Barlow [Barl 89], in the so-called Barlow's hypothesis, suggests that the outcome of the early processing performed in our visual cortical feature detectors might be the result of a *redundancy reduction* process. In other words, the neural outputs are mutually as statistically independent as possible, conditioned, of course, on the received sensory messages. The interested reader can find more on issues related to redundancy reduction and also to a number of methodologies inspired by it in [Atti 92, Fiel 94, Deco 95, Bell 00]. The potential of the ICA as an optimal feature generator technique, in the context of pattern recognition, has been demonstrated in [Cao 03, Bell 97, Hoy 00, Jang 99, Bart 02, Kwon 04].

Before we proceed to develop techniques for performing ICA, we need to be sure that such a problem is well defined and has a solution and under what conditions. To this end, let us assume that our input random data vector $\boldsymbol{x}$ is indeed generated by a linear combination of statistically independent and *stationary in the strict sense* components (sources), that is,

$$\boldsymbol{x} = \mathcal{A}\boldsymbol{y} \tag{6.45}$$

The task now is under what conditions a matrix, W, can be computed so as to recover the components of $\boldsymbol{y}$ from Eq. (6.44), by exploiting information hidden in $\boldsymbol{x}$. Usually $\mathcal{A}$ is known as the mixing and W as the demixing matrix, respectively. The following condition is proved in [Como 94].

Identifiability Condition of the ICA Model

All independent components $y(i)$, $i = 1, 2, \ldots, N$, with the possible exception of one, must be non-Gaussian. A second condition is that matrix $\mathcal{A}$ must be invertible. In the more general case where $\mathcal{A}$ is a nonsquare $l \times N$ matrix, then l must be greater than N and $\mathcal{A}$ must be of full column rank.

In other words, in contrast to PCA which can always be performed, ICA is meaningful only if the involved random variables are non-Gaussian. Indeed, as has already been stated, for Gaussian random variables independence is equivalent to uncorrelatedness and PCA suffices. From a mathematical point of view, the ICA problem is ill-posed for Gaussian processes. Indeed, if we assume that the obtained independent components $y(i)$, $i = 0, 1, \ldots, N-1$, are all Gaussian, then a linear transformation of them by *any* unitary matrix will also be a solution (see Problem 5.4). PCA achieves a unique solution by imposing a *specific orthogonal structure* onto the transformation matrix.

Under the above stated conditions, it can be shown that each one of the resulting independent components is *uniquely* estimated up to a multiplicative constant, which is a rather insignificant indeterminacy associated with the method. This is the reason that many times the components are considered to be of unit variance. Finally, it is interesting that the independent components result in no specific ordering, in contrast to the PCA, where a specific ordering is associated with the values of the corresponding eigenvalues. However, in practice, some form of ordering can be adopted. For example, the components can be ordered according to the degree of "non-Gaussianity," measured by an appropriate index, for example, the fourth-order cumulant (see Appendix A). Although such an index may seem a bit strange to a newcomer, its physical interpretation will become clearer as we go on. After all, from a common-sense point of view, a Gaussian pdf must be the least interesting one. Recall from Chapter 2 that maximizing the entropy, constraining the solution to be within the family of random variables with given mean and variance, the result is a Gaussian pdf. That is, the Gaussian is the most "random" of all the pdfs describing this family of random variables and from this point of view the least informative one with respect to the underlying structure of the data. In contrast, distributions that have the "least resemblance" to the Gaussian are more interesting since they display some structure associated with the data. This observation is at the heart of a closely related, to ICA, family of techniques known as *projection pursuit*; see also Section 4.12. The essence behind such techniques is to search for directions (subspaces) in the feature space so that the corresponding data vector projections are described by "interesting" non-Gaussian distributions. For a more rigorous discussion on such issues the reader may refer to, for example, [Hube 85, Jone 87].

6.5.1 ICA Based on Second- and Fourth-Order Cumulants

This approach in performing ICA is a direct generalization of the PCA technique. The Karhunen–Loève transform focuses on the second-order statistics and demands the cross-correlations $E[y(i)y(j)]$ to be zero. Since in ICA we demand that the components of $\boldsymbol{y}$ be statistically independent, this is equivalent to demanding that all the higher order cross-cumulants to be zero (see Appendix A). In [Como 94] it is suggested that restricting ourselves up to the fourth-order cumulants is sufficient for many applications. Appendix A shows the first three cumulants are equal to the first three moments, that is,

$$\kappa_1(y(i)) = E[y(i)] = 0$$

$$\kappa_2(y(i)y(j)) = E[y(i)y(j)]$$

$$\kappa_3(y(i)y(j)y(k)) = E[y(i)y(j)y(k)]$$

and the fourth-order cumulants are given by

$$\begin{aligned}\kappa_4(y(i)y(j)y(k)y(r)) = {} & E[y(i)y(j)y(k)y(r)] - E[y(i)y(j)]E[y(k)y(r)] \\ & - E[y(i)y(k)]E[y(j)y(r)] \\ & - E[y(i)y(r)]E[y(j)y(k)]\end{aligned}$$

where zero mean processes have been assumed. Another assumption that is usually encountered in practice, and will be adopted here, is that the associated pdfs are symmetric. This makes all odd order cumulants zero. Thus the problem has now been reduced to finding a matrix, W, so that the second-order (cross-correlations) and fourth-order cross-cumulants of the transformed variables are zero. In [Como 94] this is achieved by the following steps:

Step 1: Perform a PCA on the input data, that is,

$$\hat{\boldsymbol{y}} = A^T \boldsymbol{x} \tag{6.46}$$

A is our familiar unitary transformation matrix of the Karhunen–Loève transform. The components of the transformed random vector $\hat{\boldsymbol{y}}$ are thus uncorrelated.

Step 2: Compute another unitary matrix, $\hat{A}$, so that the fourth-order cross-cumulants of the components of the transformed random vector

$$\boldsymbol{y} = \hat{A}^T \hat{\boldsymbol{y}} \tag{6.47}$$

are zero. This is equivalent to searching for a matrix $\hat{A}$ that makes the sum of the squares of the fourth-order auto-cumulants maximum, that is,

$$\max_{\hat{A}\hat{A}^T = I} \Psi(\hat{A}) \equiv \sum_{i=0}^{N-1} \kappa_4(y(i))^2 \tag{6.48}$$

Step 2 is justified as follows. It can be shown [Como 94] that the sum of the squares of the fourth-order cumulants is invariant under a linear transformation by a unitary matrix. Therefore, since the sum of squares of the fourth-order cumulants is fixed for $\hat{\boldsymbol{y}}$, maximizing the sum of squares of the auto-cumulants of $\boldsymbol{y}$ will force the corresponding cross-cumulants to zero. Observe that this is basically a diagonalization problem of the fourth-order cumulant multidimensional array. In practice, this is achieved by generalizing the method of Givens rotations, used for matrix diagonalization [Como 94]. Note that the right hand side in Eq. (6.48) is a function of (a) the elements of the unknown matrix $\hat{A}$, (b) the elements of the known (for this step) matrix A, and (c) the cumulants of the random components of the input data vector $\boldsymbol{x}$, which have to be estimated prior to the application of the method. In practice, it may turn out that the nulling of cross-cumulants is only approximately achieved. This is because (a) the input data may not obey the linear model of Eq. (6.45); (b) the input data are corrupted by noise, which has not been taken into account; and (c) the cumulants of the input are only approximately known, since they are estimated by the available input data set.

Once the two steps have been completed, the final feature vector with (approximately) independent components is given by the combined transform

$$\boldsymbol{y} = (A\hat{A})^T \boldsymbol{x} \equiv W\boldsymbol{x} \tag{6.49}$$

Notice that since $\hat{A}$ is unitary, the uncorrelatedness achieved in the first step is inherited by the elements of $\boldsymbol{y}$, which now has its second- and fourth-order cross-cumulants (at least approximately) zero.

6.5.2 ICA Based on Mutual Information

The approach based on nulling the second- and fourth-order cross-cumulants, though one of the most widely used in practice, somehow lacks in generality and also imposes, externally, a structure in the transformation matrix. An alternative, theoretically more pleasing approach is estimating W by minimizing the *mutual information* between the transformed random variables. The mutual information, $I(\boldsymbol{y})$, between the components of $\boldsymbol{y}$ is defined as

$$I(\boldsymbol{y}) = -H(\boldsymbol{y}) + \sum_{i=0}^{N-1} H(y(i)) \tag{6.50}$$

where $H(y(i))$ is the associated entropy of $y(i)$, defined as ([Papo 91])

$$H(y(i)) = -\int p_i(y(i)) \ln p_i(y(i))\, dy(i) \tag{6.51}$$

where $p_i(y(i))$ is the marginal pdf of $y(i)$. In Appendix A, it is shown that $I(\boldsymbol{y})$ is equal to the Kullback–Leibler probability distance between the joint pdf $p(\boldsymbol{y})$ and the product of the respective marginal probability densities $\prod_{i=0}^{N-1} p_i(y(i))$. This distance (and hence the associated mutual information $I(\boldsymbol{y})$) is zero if the components $y(i)$ are statistically independent. This is because only in this case is the

joint pdf equal to the product of the corresponding marginal pdfs and the Kullback–Leibler distance becomes zero. Hence, what is more natural than trying to compute W so as to force $I(\boldsymbol{y})$ to be minimum, since this will make the components of $\boldsymbol{y}$ *as independent as possible*? Combining Eqs. (6.44), (6.50), and (6.51) and taking into account the formula that relates the two pdfs associated with $\boldsymbol{x}$ and $\boldsymbol{y}$ ($\boldsymbol{y}$ is a function of $\boldsymbol{x}$), e.g., [Papo 91], we end up with

$$I(\boldsymbol{y}) = -H(\boldsymbol{x}) - \ln|det(W)| - \sum_{i=0}^{N-1} \int p_i(y(i)) \ln p_i(y(i))\, dy(i) \tag{6.52}$$

where $det(W)$ denotes the determinant of W. The elements of the unknown matrix W are hidden in the marginal pdfs of the transformed variables, $y(i)$. However, it is not easy to express this dependence explicitly. An approach currently used is to expand each of the marginal probabilities around the Gaussian pdf, $g(y)$, following Edgeworth's expansion (Appendix A), and truncate the series to a reasonable approximation. For example, keeping the first two terms in the Edgeworth expansion, we have

$$p(y) = g(y)\left(1 + \frac{1}{3!}\kappa_3(y)H_3(y) + \frac{1}{4!}\kappa_4(y)H_4(y)\right) \tag{6.53}$$

where $H_k(y)$ is the Hermite polynomial of order k (Appendix A). To obtain an approximate expression for $I(\boldsymbol{y})$ in terms of cumulants of $y(i)$ and W, we can (a) insert in Eq. (6.52) the pdf approximation in Eq. (6.53), (b) adopt the approximation $\ln(1+y) \simeq y - y^2$, and (c) perform the integrations. This is no doubt a rather painful task! For the case of Eq. (6.53) and constraining W to be unitary, the following is obtained ([Hyva 01]):

$$I(\boldsymbol{y}) \simeq C - \sum_{i=0}^{N-1}\left(\frac{1}{12}\kappa_3^2(y(i)) + \frac{1}{48}\kappa_4^2(y(i)) + \frac{7}{48}\kappa_4^4(y(i)) - \frac{1}{8}\kappa_3^2(y(i))\kappa_4(y(i))\right) \tag{6.54}$$

where C is a variable independent of W. Under the assumption that the pdfs are symmetric (thus, third-order cumulants are zero,) it can be shown that minimizing the approximate expression of the mutual information in Eq. (6.54) is equivalent to maximizing the sum of the squares of the fourth-order cumulants. Of course, the unitary W constraint is not necessary, and in this case other approximate expressions for $I(\boldsymbol{y})$ result, e.g., [Hayk 99].

Minimization of $I(\boldsymbol{y})$ in Eq. (6.54) can be carried out by a gradient descent technique (Appendix C), where the involved expectations (associated with the cumulants) are replaced by the respective instantaneous values. Although a detailed treatment of the optimization procedure is beyond the scope of this book, it is worth pointing out some of its aspects.

Before we apply the approximations, let us go back to Eq. (6.52). Since $H(\boldsymbol{x})$ does not depend on W, minimizing $I(\boldsymbol{y})$ is equivalent to maximization of

$$J(W) = \ln|det(W)| + E\left[\sum_{i=0}^{N-1} \ln p_i(y(i))\right] \tag{6.55}$$

Taking the gradient of the cost function with respect to W results in

$$\frac{\partial J(W)}{\partial W} = W^{-T} - E[\phi(\boldsymbol{y})\boldsymbol{x}^T] \tag{6.56}$$

where

$$\phi(\boldsymbol{y}) \equiv \left[-\frac{p'_0(y(0))}{p_0(y(0))}, \ldots, -\frac{p'_{N-1}(y(N-1))}{p_{N-1}(y(N-1))} \right]^T \tag{6.57}$$

and

$$p'_i(y(i)) \equiv \frac{dp_i(y(i))}{dy(i)} \tag{6.58}$$

Obviously, the derivatives of the marginal probability densities depend on the type of approximation adopted in each case. The general gradient ascent scheme at the tth iteration step can now be written as

$$\begin{aligned} W(t) &= W(t-1) + \mu(t) \left(W^{-T}(t-1) - E[\phi(\boldsymbol{y})\boldsymbol{x}^T] \right) \\ W(t) &= W(t-1) + \mu(t) \left(I - E[\phi(\boldsymbol{y})\boldsymbol{y}^T] \right) W^{-T}(t-1) \end{aligned} \tag{6.59}$$

In practice, the expectation operator is neglected, in the spirit of the stochastic approximation rationale (Section 3.4.2).

Remarks

- From the gradient in Eq. (6.56) it is easy to see that at a stationary point the following is true:

$$\frac{\partial J(W)}{\partial W} W^T = E[I - \phi(\boldsymbol{y})\boldsymbol{y}^T] = 0 \tag{6.60}$$

  In other words, what we achieve with ICA is a nonlinear generalization of PCA. Recall that for the latter, the uncorrelatedness condition can be written as

$$E[I - \boldsymbol{y}\boldsymbol{y}^T] = 0 \tag{6.61}$$

  The presence of the nonlinear function $\phi(\cdot)$ takes us beyond simple uncorrelatedness, and brings the cumulants into the scene. In fact, Eq. (6.60) was the one that inspired the early pioneering work on ICA, as a direct nonlinear generalization of PCA [Jutt 91].

- The updated equation in Eq. (6.59) involves the inversion of the transpose of the current estimate of W. Besides the computational complexity issues, there is no guarantee of invertibility in the process of adaptation. Use of the so-called natural gradient [Doug 00], instead of the gradient in Eq. (6.56), results in

$$W(t) = W(t-1) + \mu(t) \left(I - E[\phi(\boldsymbol{y})\boldsymbol{y}^T] \right) W(t-1) \tag{6.62}$$

which does not involve matrix inversion and at the same time improves convergence. A more detailed treatment of this issue is beyond the scope of the present book. Just to give an incentive to the mathematically inclined reader for indulging more deeply this field, it suffices to say that our familiar gradient, that is, Eq. (6.56), points to the steepest ascent direction if the space is Euclidean. However, in our case the parameter space consists of all the nonsingular $N \times N$ matrices, which is a multiplicative group. The space is Riemannian, and it turns out that the natural gradient, pointing to the steepest ascent direction, results if we multiply the gradient in Eq. (6.56) by $W^T W$, which is the corresponding Riemannian metric tensor [Doug 00].

6.5.3 An ICA Simulation Example

The example is a realization of the case shown in Figure 6.4. A total of 1024 samples of a two-dimensional normal distribution were generated.

The mean and covariance matrix of the normal pdf were

$$\boldsymbol{\mu} = [-2.6042, 2.5]^T, \quad \Sigma = \begin{bmatrix} 10.5246 & 9.6313 \\ 9.6313 & 11.3203 \end{bmatrix}$$

Similarly, 1024 samples from a second normal pdf were generated with the same covariance and mean $-\boldsymbol{\mu}$. For the ICA, the method based on the second- and

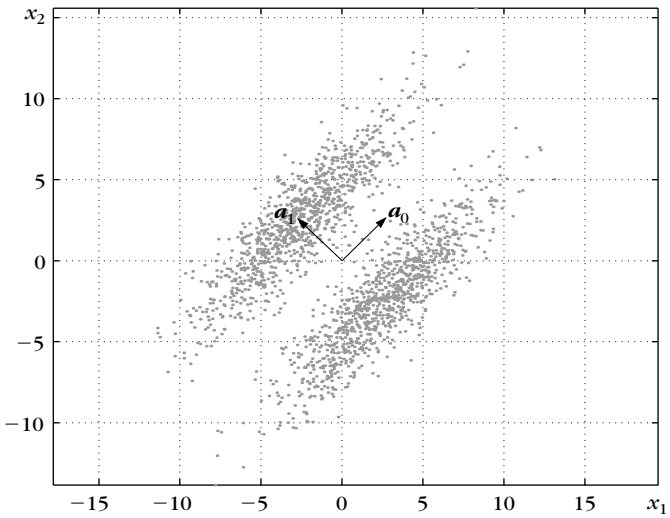


FIGURE 6.4

The setup for the ICA simulation example. The two vectors point to the projection directions resulting from the analysis. The optimal direction for projection, resulting from the ICA analysis, is that of $\boldsymbol{a}_1$.

fourth-order cumulants, presented in this section, was used. The resulting transformation matrix W is

$$W = \begin{bmatrix} -0.7088 & 0.7054 \\ 0.7054 & 0.7088 \end{bmatrix} \equiv \begin{bmatrix} \boldsymbol{a}_1^T \\ \boldsymbol{a}_0^T \end{bmatrix}$$

The vectors $\boldsymbol{a}_0$ and $\boldsymbol{a}_1$ point in the principal and minor axis directions, respectively, obtained from the PCA analysis. However, the most interesting direction for projection, according to the ICA analysis, is that of $\boldsymbol{a}_1$ and not of $\boldsymbol{a}_0$. Indeed, the kurtosis of the obtained transformed variables $[y_1, y_2]^T = W\boldsymbol{x}$ is

$$\kappa_4(y_1) = -1.7$$

$$\kappa_4(y_2) = 0.1$$

Thus, projection in the principal axis direction results in a variable with a pdf close to a Gaussian. The projection to the minor axis direction results in a variable with a pdf that deviates from the Gaussian (Figures 6.1, 6.4) and is more appropriate from the classification point of view.

6.6 NONNEGATIVE MATRIX FACTORIZATION

In the PCA as well as the SVD analysis, the underlying constraints were the orthogonality of the involved basis vectors, in the PCA, and of the column vectors in the U and V matrices in the SVD. This becomes crystal clear from the problem 6.3. PCA is formulated as a task minimizing the mean square error subject to the orthogonality constraint of the basis vectors. Although, in some cases, the resulting expansion is useful, for some other cases such a constraint turns out to be very "weak" in representing the data. More recently, a new matrix factorization was suggested in [Paat 91, Paat 94], which guarantees the nonnegativity of the elements of the resulting matrix factors. Such a constraint is enforced in certain applications since negative elements contradict physical reality. For example, in image analysis the intensity values of the pixels cannot be negative. Also, probability values cannot be negative. The resulting factorization is known as *nonnegative matrix factorization* (NMF) and it has been used successfully in a number of applications including document clustering ([Xu 03]), molecular pattern discovery ([Brun 04]), image analysis ([Lee 01]), clustering ([Szym 06]), music transcription and music instrument classification ([Smar 03, Benn 06]) and face verification ([Zafe 06]).

Given a $l \times n$ matrix X, the task of NMF consists of finding an approximate factorization of X, that is,

$$X \approx WH \tag{6.63}$$

where W and H are $l \times r$ and $r \times n$ matrices, respectively, $r < \min(n, l)$ and all the matrix elements are nonnegative, that is, $W(i,k) \geq 0$, $H(k,j) \geq 0$,

$i = 1, 2, \ldots, l$, $k = 1, 2, \ldots, r$, $j = 1, 2, \ldots, n$. Clearly, matrices W and H are of rank at most r and their product is a low rank, at most r, approximation of X. The significance of the above is that every column vector in X is represented by the expansion

$$\boldsymbol{x}_i = \sum_{k=1}^{r} H(k,i)\boldsymbol{w}_k, \quad i = 1, 2, \ldots, n$$

where $\boldsymbol{w}_k$, $k = 1, 2, \ldots, r$, are the column vectors of W and constitute the basis of the expansion. The number of vectors in the basis is less than the dimensionality of the vector itself. Hence, NMF can also be seen as a method for *dimensionality reduction*.

To get a good approximation in (6.63) one can adopt different costs. The most conventional cost is the Frobenius norm of the error matrix. In such a setting, the NMF task is casted as follows:

$$\text{minimize} \quad ||X - WH||_F \equiv \sum_{i=1}^{l}\sum_{j=1}^{n} \left(X(i,j) + [WH](i,j)\right)^2 \tag{6.64}$$

$$\text{subject to} \quad W(i,k) \geq 0,\ H(k,j) \geq 0. \quad H(k,j) \geq 0\ \forall i,k,j \tag{6.65}$$

where $[WH](i,j)$ is the (i,j) element of matrix WH. Minimization is performed with respect to W and H.

Another cost function has also been suggested, which is a close relative of the Kullback–Leibler distance (see Appendix A) and the task now becomes

$$\text{minimize} \quad \sum_{i=1}^{l}\sum_{j=1}^{n} \left(X(i,j)\ln\frac{X(i,j)}{[WH](i,j)} - X(i,j) + [WH](i,j)\right) \tag{6.66}$$

$$\text{subject to} \quad W(i,k) \geq 0,\ H(k,j) \geq 0\ \forall i,k,j \tag{6.67}$$

It is readily seen that if $X = WH$ the previous cost becomes zero. Also, observe that if $\sum_{i,j} X(i,j) = \sum_{i,j}[WH](i,j) = 1$ then the cost becomes identical to the Kullback–Leibler (KL) distance formulation. Note, however, that the previous KL-like cost is not well defined if either $X(i,j)$ or $[WH](i,j)$ are zero. For more information on this topic the interested reader may consult, for example, [Sra 06].

Once the problem has been formulated, the major issue rests on the solution of the optimization task. To this end, a number of algorithms have been proposed, for example, Newton-type or gradient descent type (Appendix C). Such algorithmic issues, as well as a number of related theoretic ones, are beyond the scope of the current book, and the interested reader may consult [Chu 04, Dono 04, Trop 03].

6.7 NONLINEAR DIMENSIONALITY REDUCTION

All the techniques that have been discussed so far in this chapter, as well as the LDA in the previous chapter, share a common goal: dimensionality reduction. In

other words, given a high-dimensional data set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathcal{R}^N$ of input patterns,[4] the goal is to compute n corresponding patterns, $Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n\} \subset \mathcal{R}^m$, $m < N$, that provide an "informative" representation of the input patterns. The word "informative" is interpreted in a different way for different methods; for example, PCA and ICA adopt different views on the issue. Another common characteristic of all the previous methods is that they respect linearity. Once a transformation matrix is computed, for each method, points in Y are obtained by projecting the points in X along the rows of this matrix.

The emphasis of dimensionality reduction has, so far, focussed to the domain of feature generation, in order to bypass the curse of dimensionality and cope efficiently with the generalization aspects of a classifier. However, the significance of such techniques goes much beyond and embraces a number of other applications. Data visualization is an area where dimensionality reduction techniques are employed in order to transform the original data from a high-dimensional into two or three dimensions, thereby offering additional insight into the problem at hand. As stated in [Tene 00], the human brain confronts the same problem extracting from the high-dimensional sensory system (i.e., 10^6 optic nerve fibers) a reduced number of perceptually relevant features. Data mining and information retrieval is another area where searching can be substantially facilitated if it is performed in a lower dimensional space. This is a typical area where data usually lie in a very high-dimensional space, although its intrinsic dimensionality is low. Dimensionality reduction has been used extensively in clustering and in semi-supervised learning, an area that is gaining in importance over the last years. So, in a way, this section is a bridge to the chapters dedicated to these topics.

The aim of this section is to discuss nonlinear dimensionality reduction techniques. We will discuss the main directions that are currently popular, and we will not delve into many details.

6.7.1 Kernel PCA

As its name suggests, this is a kernelized version of the classical PCA, introduced in [Scho 98]. Given the data set, X, we make an implicit mapping into a RKHS H,

$$\boldsymbol{x} \in X \mapsto \boldsymbol{\phi}(\boldsymbol{x}) \in H$$

Let $\boldsymbol{x}_i$, $i = 1, 2, \ldots, n$, be the available training points. We will work with an estimate of the correlation matrix in H obtained as an average over the known sample points[5]

$$R = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\phi}(\boldsymbol{x}_i) \boldsymbol{\phi}(\boldsymbol{x}_i)^T \tag{6.68}$$

[4] To serve the specific needs of this chapter, we have reserved the symbol N to denote dimensionality of the input space. For the rest of the book, N denotes the number of data points.

[5] If the dimensionality of H is infinite, the definition of the correlation matrix needs a special interpretation, but we will not bother about it.

The goal is to perform the eigendecomposition of $\boldsymbol{R}$, that is,

$$\boldsymbol{R}\boldsymbol{v} = \lambda \boldsymbol{v} \tag{6.69}$$

Let us make the assumption that the data are centered $\left(\sum_{i=1}^{n} \boldsymbol{\phi}(\boldsymbol{x}_i) = \boldsymbol{0}\right.$. This assumption is only to simplify the discussion, and it can be relaxed.$\big)$ By the definition of $\boldsymbol{R}$, it can be shown that $\boldsymbol{v}$ lies in the span of $\{\boldsymbol{\phi}(\boldsymbol{x}_1), \boldsymbol{\phi}(\boldsymbol{x}_2), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_n)\}$. Indeed,

$$\lambda \boldsymbol{v} = \left(\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\phi}(\boldsymbol{x}_i)\boldsymbol{\phi}(\boldsymbol{x}_i)^T\right)\boldsymbol{v} = \frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{\phi}(\boldsymbol{x}_i)^T\boldsymbol{v}\right)\boldsymbol{\phi}(\boldsymbol{x}_i)$$

and for $\lambda \neq 0$ we can write

$$\boldsymbol{v} = \sum_{i=1}^{n} a(i)\boldsymbol{\phi}(\boldsymbol{x}_i) \tag{6.70}$$

Combining (6.69) and (6.70), it turns out ([Scho 98]) that the problem is equivalent to performing an eigendecomposition of the Gram matrix

$$\mathcal{K}\boldsymbol{a} = n\lambda \boldsymbol{a} \tag{6.71}$$

where

$$\boldsymbol{a} \equiv [a(1), a(2), \ldots, a(n)]^T \tag{6.72}$$

As we already know (Section 4.19.1), the elements of the Gram matrix are $\mathcal{K}(i,j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, with $K(\cdot,\cdot)$ being the adopted kernel function. Thus, the kth eigenvector of $\boldsymbol{R}$, corresponding to the kth (nonzero) eigenvector of $\mathcal{K}$ in (6.71), is expressed as

$$\boldsymbol{v}_k = \sum_{i=1}^{n} a_k(i)\boldsymbol{\phi}(\boldsymbol{x}_i),\ k = 1, 2, \ldots, p \tag{6.73}$$

where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$ denote the respective eigenvalues in descending order and λ_p is the smallest nonzero one and $\boldsymbol{a}_k^T \equiv [a_k(1), \ldots, a_k(n)]$ is the kth eigenvector of the Gram matrix. The latter is assumed to be normalized so that $\langle \boldsymbol{v}_k, \boldsymbol{v}_k \rangle = 1,\ k = 1, 2, \ldots, p$, where $\langle \cdot, \cdot \rangle$ is the dot product in the Hilbert space H. This imposes an equivalent normalization on the respective $\boldsymbol{a}_k$'s, resulting from

$$\begin{aligned} 1 = \langle \boldsymbol{v}_k, \boldsymbol{v}_k \rangle &= \left\langle \sum_{i=1}^{n} a_k(i)\boldsymbol{\phi}(\boldsymbol{x}_i), \sum_{j=1}^{n} a_k(j)\boldsymbol{\phi}(\boldsymbol{x}_j) \right\rangle \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_k(i)a_k(j)\mathcal{K}(i,j) \\ &= \boldsymbol{a}_k^T\mathcal{K}\boldsymbol{a}_k = n\lambda_k \boldsymbol{a}_k^T\boldsymbol{a}_k, \quad k = 1, 2, \ldots, p \end{aligned} \tag{6.74}$$

We are now ready to summarize the basic steps for performing a kernel PCA. Given a vector $\boldsymbol{x} \in \mathcal{R}^N$ and a kernel function $K(\cdot,\cdot)$:

- Compute the Gram matrix $\mathcal{K}(i,j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j), i,j = 1,2,\ldots,n$.
- Compute the m dominant eigenvalues/eigenvectors $\lambda_k,\ \boldsymbol{a}_k,\ k = 1,2,\ldots,m$, of $\mathcal{K}$ (Eq. (6.71)).
- Perform the required normalization (Eq. (6.74)).
- Compute the m projections onto each one of the dominant eigenvectors,

$$y(k) \equiv \langle \boldsymbol{v}_k, \boldsymbol{\phi}(\boldsymbol{x}) \rangle = \sum_{i=1}^{n} a_k(i) K(\boldsymbol{x}_i, \boldsymbol{x}),\ k = 1,2,\ldots,m \tag{6.75}$$

The operations given in (6.75) correspond to a *nonlinear mapping* in the input space. Note that, in contrast to the linear PCA, the dominant eigenvectors $\boldsymbol{v}_k,\ k = 1,2,\ldots,m$, are not computed explicitly. All we know are the respective (nonlinear) projections, $y(k)$ along them. After all, this is what we are finally interested in.

Remarks

- Kernel PCA is equivalent to performing a standard PCA in the RKHS H. It can be shown that all the properties associated with the dominant eigenvectors, as discussed for the PCA, are still valid for the kernel PCA. That is, (a) the dominant eigenvector directions optimally retain most of the variance, (b) the MSE in approximating a point in H in terms of the m dominant eigenvectors is minimal, with respect to any other m directions, (c) projections onto the eigenvectors are uncorrelated, and (d) the entropy (under Gaussian assumption) is maximized ([Scho 98]).
- Recall from Section 6.3 that the eigendecomposition of the Gram matrix was required for the metric multidimensional scaling (MDS) method. Hence, kernel PCA can be considered to be a kernelized version of MDS, where inner products in the input space have been replaced by kernel operations in the Gram matrix.
- Note that the kernel PCA method does not explicitly consider the underlying structure of the manifold on which the data reside.

6.7.2 Graph-Based Methods

Laplacian eigenmaps

The starting point of this method is the assumption that the data points lie on a smooth manifold (hypersurface) $\mathcal{M} \supset X$, whose intrinsic dimension is equal to $m < N$ and it is embedded in $\mathcal{R}^N$, that is, $\mathcal{M} \subset \mathcal{R}^N$. The dimension m is given as a parameter by the user. In contrast, this is not required in the kernel PCA, where m is the number of dominant components, which, in practice, is determined so that the gap between λ_m and λ_{m+1} has a "large" value.

The main philosophy behind the method is to compute the low-dimensional representation of the data so that *local neighborhood information* in $X \subset \mathcal{M}$ is optimally preserved. In this way, one attempts to get a solution that reflects the geometric structure of the manifold. To achieve this, the following steps are in order:

Step 1: Construct a graph $G = (V, E)$, where $V = \{v_i, \ i = 1, 2, \ldots, n\}$ is a set of vertices and $E = \{e_{ij}\}$ is a set of edges connecting vertices (v_i, v_j) (see also Section 13.2.5). Each node v_i of the graph corresponds to a point $\boldsymbol{x}_i$ in our data set X. We connect v_i, v_j, that is, insert the edge e_{ij} between the respective nodes, if points $\boldsymbol{x}_i, \boldsymbol{x}_j$ are "close" to each other. According to the method, there are two ways of quantifying "closeness." Vertices v_i, v_j are connected with an edge if:

1. $||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 < \epsilon$, for some user-defined parameter ϵ, where $||\cdot||$ is the Euclidean norm operation in $\mathcal{R}^N$, or
2. $\boldsymbol{x}_j$ is among the k-nearest neighbors of $\boldsymbol{x}_i$ or $\boldsymbol{x}_i$ is among the k-nearest neighbors of $\boldsymbol{x}_j$, where k is a user-defined parameter and neighbors are chosen according to the Euclidean distance in $\mathcal{R}^N$. The use of Euclidean distance is justified by the smoothness of the manifold that allows one to approximate, locally, manifold geodesics by Euclidean distances in the space where the manifold is embedded. The latter is a known result from differential geometry.

   For those who are unfamiliar with such concepts, think of a sphere embedded in three-dimensional space. If somebody is constrained to live on the surface of the sphere, the shortest path to go from one point to another is the geodesic between these two points. Obviously, this is not a straight line but rather an arc across the surface of the sphere. However, if these points are close enough, their geodesic distance can be approximated by their Euclidean distance, computed in the three-dimensional space.

Step 2: Each edge, e_{ij}, is associated with a weight, $W(i,j)$. For nodes that are not connected, the respective weights are zero. Each weight, $W(i,j)$, is a measure of the "closeness" of the respective neighbors, $\boldsymbol{x}_i, \boldsymbol{x}_j$. A typical choice is

$$W(i,j) = \begin{cases} \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{\sigma^2}\right), & \text{if } v_i, v_j \text{ correspond to neighbors} \\ 0 \quad \text{otherwise} \end{cases}$$

where σ^2 is a user-defined parameter. We form the $n \times n$ weight matrix W having as elements the weights $W(i,j)$. Note that W is symmetric, and it is *sparse* since, in practice, many of its elements turn out to be zero.

Step 3: Define the diagonal matrix D with elements $D_{ii} = \sum_j W(i,j), i = 1, 2, \ldots, n$, and also the matrix $L = D - W$. The latter is known as the *Laplacian matrix of the graph* $G(V, E)$. Perform the generalized eigendecomposition

$$L\boldsymbol{v} = \lambda D\boldsymbol{v}$$

Let $0 = \lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_m$ be the smallest $m + 1$ eigenvalues.[6] Ignore the $\boldsymbol{v}_o$ eigenvector corresponding to $\lambda_0 = 0$ and choose the next m eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m$. Then map

$$\boldsymbol{x}_i \in \mathcal{R}^N \mapsto \boldsymbol{y}_i \in \mathcal{R}^m, \quad i = 1, 2, \ldots, n$$

where

$$\boldsymbol{y}_i^T = [\boldsymbol{v}_1(i), \boldsymbol{v}_2(i), \ldots, \boldsymbol{v}_m(i)], \quad i = 1, 2, \ldots, n \tag{6.76}$$

The computational complexity of a general eigendecomposition solver amounts to $O(n^3)$ operations. However, for sparse matrices, such as L, efficient schemes, e.g., the Lanczos algorithm ([Golu 89]), can be employed to reduce complexity to subquadratic in n.

We will prove the statement of step 3 for the case of $m = 1$. That is, the low dimensional space is the real axis. Our path evolves along the lines adopted in [Belk 03]. The goal is to compute $y_i \in \mathcal{R}$, $i = 1, 2, \ldots, n$, so that connected points (in the graph, i.e., neighbors) stay as close as possible after the mapping onto the one-dimensional subspace. The criterion used to satisfy the closeness after the mapping is

$$E_L = \sum_{i=1}^{n} \sum_{j=1}^{n} (y_i - y_j)^2 W(i,j) \tag{6.77}$$

to become minimum. Observe that if $W(i,j)$ has a large value (i.e., $\boldsymbol{x}_i, \boldsymbol{x}_j$ are close in $\mathcal{R}^N$), then if the respective y_i, y_j are far apart in $\mathcal{R}$ it incurs a heavy penalty in the cost function. Also, points that are not neighbors do not affect the minimization since the respective weights are zero. For the more general case where $1 < m < N$ the cost function becomes

$$E_L = \sum_{i=1}^{n} \sum_{j=1}^{n} ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2 W(i,j)$$

[6] In contrast to the notation used for PCA, the eigenvalues here are marked in ascending order. This is because, in this subsection, we are interested in determining the smallest values, and such a choice is notationally more convenient.

Let us now reformulate (6.77). After some obvious algebra, we obtain

$$\begin{aligned} E_L &= \sum_i y_i^2 \sum_j W(i,j) + \sum_j y_j^2 \sum_i W(i,j) - 2\sum_i \sum_j y_i y_j W(i,j) \\ &= \sum_i y_i^2 D_{ii} + \sum_j y_j^2 D_{jj} - 2\sum_i \sum_j y_i y_j W(I,j) \\ &= 2\boldsymbol{y}^T L \boldsymbol{y} \end{aligned} \tag{6.78}$$

where

$$L \equiv D - W \tag{6.79}$$

and $\boldsymbol{y}^T = [y_1, y_2, \ldots, y_n]$. The Laplacian matrix L is symmetric and nonnegative definite. The latter is readily seen from the definition in (6.78), where E_L is always a nonnegative scalar. Note that the larger the value of D_{ii}, the more "important" is the sample $\boldsymbol{x}_i$. This is because it implies large values for $W(i,j)$, $j = 1, 2, \ldots, n$, and plays a dominant role in the minimization process. Obviously, the minimum of E_L is achieved by the trivial solution $y_i = 0$, $i = 1, 2, \ldots, n$. To avoid this, as it is common in such cases, we constrain the solution to a prespecified norm. Hence our problem now becomes

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{y}^T L \boldsymbol{y} \\ &\text{subject to} \quad \boldsymbol{y}^T D \boldsymbol{y} = 1 \end{aligned}$$

Although we can work directly on the previous task, we will reshape slightly it in order to use tools that are more familiar to us. Define

$$\boldsymbol{z} = D^{1/2} \boldsymbol{y} \tag{6.80}$$

and

$$\tilde{L} = D^{-1/2} L D^{-1/2} \tag{6.81}$$

which is known as the *normalized graph Laplacian* matrix. It is now readily seen that our optimization problem becomes

$$\text{minimize} \quad \boldsymbol{z}^T \tilde{L} \boldsymbol{z} \tag{6.82}$$

$$\text{subject to} \quad \boldsymbol{z}^T \boldsymbol{z} = 1 \tag{6.83}$$

Using Lagrange multipliers and equating the gradient of the Lagrangian to zero (Appendix C) it turns out that the solution is given by

$$\tilde{L} \boldsymbol{z} = \lambda \boldsymbol{z} \tag{6.84}$$

In other words, computing the solution becomes equivalent to solving an eigenvalue problem. By substituting (6.84) into the cost function (6.82) and taking into account the constraint (6.83), it turns out that the value of the cost associated with the optimal $\boldsymbol{z}$ is equal to λ. Hence, the solution is the eigenvector corresponding to

the minimum eigenvalue. However, the minimum eigenvalue of $\tilde{L}$ is zero and the corresponding eigenvector corresponds to a trivial solution. Indeed, observe that

$$\tilde{L}D^{1/2}\mathbf{1} = D^{-1/2}LD^{-1/2}D^{1/2}\mathbf{1} = D^{-1/2}(D - W)\mathbf{1} = \mathbf{0}$$

where $\mathbf{1}$ is the vector having all its elements equal 1. In words, $\boldsymbol{z} = D^{1/2}\mathbf{1}$ is an eigenvector corresponding to the zero eigenvalue, and it results to the trivial solution, $y_i = 1,\ i = 1, 2, \ldots, n$. That is, all the points are mapped onto the same point in the real line. To exclude this undesired solution, recall that $\tilde{L}$ is a nonnegative matrix, and, hence, 0 is its smallest eigenvalue (if the graph is connected, that is, at least one path (see Section 13.2.5) connects any pair of vertices, $D^{1/2}\mathbf{1}$ is the only eigenvector associated with the zero eigenvalue, λ_0, [Belk 03]. This is an assumption we adopt here.) Also, since $\tilde{L}$ is a symmetric matrix, we know (Appendix B) that its eigenvectors are orthogonal to each other. In the sequel, we impose an extra constraint, and we now require the solution to be *orthogonal* to $D^{1/2}\mathbf{1}$. Constraining the solution to be orthogonal to the eigenvector corresponding to the smallest (zero) eigenvalue drives the solution to the next eigenvector corresponding to the next smallest (nonzero) eigenvalue λ_1. Note that the eigendecomposition of $\tilde{L}$ is equivalent to what we called generalized eigendecomposition of L in step 3 earlier.

For the more general case of $m > 1$, we have to compute the m eigenvectors associated with $\lambda_1 \leq \ldots \leq \lambda_m$. For this case, the constraints prevent us from mapping into a subspace of dimension less than the desired m. For example, we do not want to project in a three-dimensional space and the points to lie on a two-dimensional plane or on an one-dimensional line. For more details, the interested reader is referred to the insightful paper [Belk 03].

Local Linear Embedding (LLE)

As was the case with the Laplacian eigenmap method, LLE assumes that our data rest on a smooth enough manifold of dimension m, which is embedded in the $\mathcal{R}^N$ subspace, with $m < N$ ([Rowe 00]). The smoothness assumption allows us to further assume that, provided there is sufficient data and the manifold is "well" sampled, nearby points lie on (or close to) a "locally" *linear* patch of the manifold. The algorithm in its simplest form is summarized in the following three steps:

Step 1: For each point $\boldsymbol{x}_i,\ i = 1, 2, \ldots, n$, search for its nearest neighbors.

Step 2: Compute the weights $W(i,j),\ i,j = 1, 2, \ldots, n$, that best reconstruct each point, $\boldsymbol{x}_i$, from its nearest neighbors, so that to minimize the cost

$$\arg\min_{W} E_W = \sum_{i=1}^{n} ||\boldsymbol{x}_i - \sum_{j=1}^{n} W(i,j)\boldsymbol{x}_{i_j}||^2 \tag{6.85}$$

where, $\boldsymbol{x}_{i_j}$ denotes the jth neighbor of the ith point. The weights are constrained: (a) to be zero for points that are not neighbors and (b) the

rows of the weight matrix add to one, that is,

$$\sum_{j=1}^{n} W(i,j) = 1 \tag{6.86}$$

That is, the sum of the weights, over all neighbors, must be equal to one.

Step 3: Use the weights obtained from the previous step to compute the corresponding points $\boldsymbol{y}_i \in \mathcal{R}^m,\ i = 1, 2, \ldots, n$, so that to minimize the cost with respect to the unknown points $Y = \{\boldsymbol{y}_i,\ i = 1, 2, \ldots, n\}$

$$\arg\min_{Y} E_Y = \sum_{i=1}^{n} ||\boldsymbol{y}_i - \sum_{j} W(i,j)\boldsymbol{y}_j||^2 \tag{6.87}$$

The above minimization takes place subject to two constraints so as to avoid degenerate results: (a) the outputs are centered, $\sum_i \boldsymbol{y}_i = \boldsymbol{0}$ and (b) the outputs have unit covariance matrix ([Saul 01]). The nearest points, in step 1, are searched in the same way as carried out for the Laplacian eigenmap method. Once again, use of the Euclidean distance is justified by the smoothness of the manifold, as long as the search is limited "locally" among neighboring points. For the second step, the method exploits the local linearity of a smooth manifold and tries to predict *linearly* each point by its neighbors using the least squares error criterion. Minimizing the cost subject to the constraint given in (6.86) results in a solution that satisfies the following three properties:

1. Rotation invariance.
2. Scale invariance.
3. Translation invariance.

The first two properties can easily be verified by the form of the cost function and the third one is the consequence of the constraint equation. The implication of this is that the computed weights encode information about the intrinsic characteristics of each neighborhood and they do not depend on the particular point.

The resulting weights, $W(i,j)$, reflect the intrinsic properties of the local geometry underlying the data. Since our goal is to retain the local information after the mapping, these weights are used to reconstruct each point in the $\mathcal{R}^m$ subspace by its neighbors. As is nicely stated in [Saul 01], it is as if we take a pair of scissors to cut small linear patches of the manifold and place them in the low-dimensional subspace.

It turns out that solving for (6.87) for the unknown points $\boldsymbol{y}_i,\ i = 1, 2, \ldots, n$, is equivalent to:

- Performing an eigendecomposition of the matrix $(I - W)^T(I - W)$.
- Discarding the eigenvector that corresponds to the smallest eigenvalue.
- Taking the eigenvectors that correspond to the next (lower) eigenvalues. These yield the low-dimensional outputs $\boldsymbol{y}_i,\ i = 1, 2, \ldots, n$.

Once again, the involved matrix W is sparse, and if this is taken into account, the eigenvalue problem scales relatively well to large data sets with complexity subquadratic in n. The complexity for step 2 scales as $O(nk^3)$, and it is contributed by the solver of the linear set of equations with k unknowns for each point. The method requires that the user provides two parameters, the number of nearest neighbors, k (or ϵ) and the dimensionality m. The interested reader can find more on the LLE method in [Saul 01].

Isometric Mapping (ISOMAP)

In contrast to the two previous methods that unravel the geometry of the manifold on a local basis, the ISOMAP algorithm adopts the view that only the geodesic distances between all pairs of the data points can reflect the true structure of the manifold. Euclidean distances between points in a manifold cannot represent it properly, since points that lie far apart, as measured by their geodesic distance, may be close when measured in terms of their Euclidean distance, (see Figure 6.5). ISOMAP is basically a variant of the MDS algorithm in which the Euclidean distances are substituted by the respective geodesic distances along the manifold. The essence of the method is to estimate geodesic distances between points that lie faraway. To this end, a two-step procedure is adopted:

Step 1: For each point, $\boldsymbol{x}_i$, $i = 1, 2, \ldots, n$, compute the nearest neighbors and construct a graph $G(V, E)$ whose vertices represent input patterns and the edges connect nearest neighbors (nearest neighbors are computed with

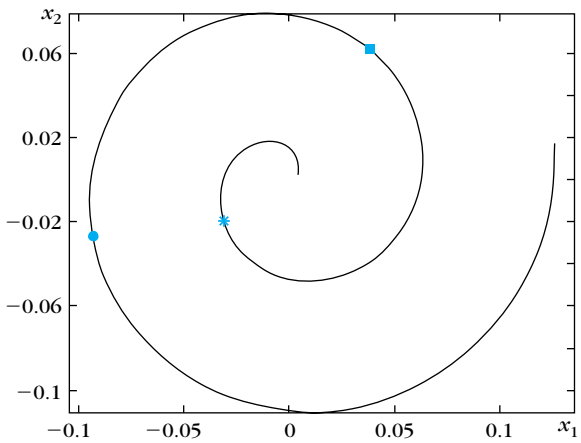


FIGURE 6.5

The point denoted by a "star" is deceptively closer to the point denoted by a "dot" than to the point denoted by a "box," if distance is measured in terms of the Euclidean distance. However, if one is constrained to travel along the spiral, the geodesic distance is the one that determines closeness and it is the "box" point that is closer to the "star."

either of the two alternatives used for the Laplacian eigenmap method. The parameters k or ϵ are user-defined parameters.) The edges are assigned weights based on the respective Euclidean distance. (For nearest neighbors this is a good approximation of the respective geodesic distance.)

Step 2: Compute the pairwise geodesic distances among all pairs (i,j), $i,j = 1, 2, \ldots, n$, along shortest paths through the graph. The key assumption is that the geodesic between any two points on the manifold can be approximated by the *shortest path* connecting the two points along the graph $G(V,E)$. To this end, efficient algorithms can be used to achieve it with complexity $O(n^2 \ln n + n^2 k)$ (e.g., Djikstar's algorithm, [Corm 01]). This cost can be prohibitive for large values of n.

Having estimated the geodesic distances between all pairs of point, the MDS method is mobilized. Thus, the problem becomes equivalent to performing the eigendecomposition of the respective Gram matrix and selecting the m most significant eigenvectors to represent the low-dimensional space. After the mapping, Euclidean distances between points in the low-dimensional subspace match the respective geodesic distances on the manifold in the original high-dimensional space. As is the case in PCA and MDS, m is estimated by the number of significant eigenvalues. It can be shown that ISOMAP is guaranteed asymptotically ($n \longrightarrow \infty$) to recover the true dimensionality of a class of nonlinear manifolds [Tene 00, Dono 04].

All three graph-based methods share a common step for computing nearest neighbors in a graph. This is a problem of complexity $O(n^2)$ but more efficient search techniques can be used by employing a special type of data structures, for example, [Beyg 06]. A notable difference between the ISOMAP on the one side and the Laplacian eigenmap and LLE methods on the other is that the latter two approaches rely on the eigendecomposition of sparse matrices as opposed to the ISOMAP that relies on the eigendecomposition of the dense Gram matrix. This gives a computational advantage to the Laplacian eigenmap and LLE techniques. Moreover, the calculation of the shortest paths in the ISOMAP is another computationally demanding task. Finally, it is of interest to note that the three graph-based techniques perform the goal of dimensionality reduction while trying to unravel, in one way or another, the geometric properties of the manifold on which the data (approximately) lie. In contrast, this is not the case with the kernel PCA, which shows no interest in any manifold learning. However, as the world is very small, in [Ham 04], it is pointed out that the graph-based techniques can be seen as special cases of the kernel PCA! This becomes possible if data-dependent kernels, derived from graphs encoding neighborhood information, are used in the place of predefined kernel functions.

The goal of this section was to present some of the most basic directions that have been suggested for nonlinear dimensionality reduction. Besides the previous basic schemes, a number of improved updates have been proposed in the literature, for example, [Desi 03, Sha 05, Beng 04]. In [Lafo 06, Qui 07], the low-dimensional embedding is achieved to preserve certain measures that reflect the connectivity

of the graph $G(V, E)$. In [He 03, Cai 05, Koki 07] the idea of preserving the local information in the manifold has been carried out to define linear transforms of the form $\boldsymbol{y} = A^T \boldsymbol{x}$, and the optimization is now carried out with respect to the elements of A. The task of incremental manifold learning for dimensionality reduction was more recently considered in [Law 06]. In [Wein 05, Sun 06], the *maximum variance unfolding* method is introduced. The variance of the outputs is maximized under the constraint that (local) distances and angles are preserved among neighbors in the graph. Like the ISOMAP, it turns out that the top eigenvectors of a Gram matrix have to be computed, albeit avoiding the computationally demanding step of estimating geodesic distances, as required by the ISOMAP. In [Shui 07] a general framework, called *graph embedding*, is presented that offers a unified view for understanding and explaining a number of known (including PCA and nonlinear PCA) dimensionality reduction techniques, and it also offers a platform for developing new ones. In [Lin 08] a manifold learning technique is adopted that constructs coordinate charts for a given Riemannian manifold. For a more detailed and insightful treatment of the topic the interested reader is referred to [Burg 04]. A review of nonlinear dimensionality reduction techniques can be found in [Cama 03].

Example 6.6

A data set consists of 30 points in the two-dimensional space. The points result from sampling the spiral of Archimedes (see Figure 6.6a), described by

$$x_1 = a\theta \cos\theta, \quad x_2 = a\theta \sin\theta$$

The points of the data set correspond to the values $\theta = 0.5\pi$, and $0.7\pi,\ 0.9\pi, \ldots, 2.05\pi$ (θ is expressed in radians) and $a = 0.1$. For illustration purposes and in order to keep track of the "neighboring" information, we have used a sequence of six symbols, that is, "$\times$", "$+$", "?", "$\square$", "$\diamond$", "$\circ$"—with black color—followed by the same sequence of symbols in red color, repeatedly.

To study the performance of PCA for this case, where data lie on a nonlinear manifold, we first performed the eigendecomposition of the covariance matrix, estimated from the data set. The resulting eigenvalues are

$$\lambda_0 = 0.089 \quad \text{and} \quad \lambda_1 = 0.049$$

Observe that, in contrast to the linear case of Example 6.3, here the eigenvalues are comparable in size. Thus, if one would trust the "verdict" coming from PCA, the answer concerning the dimensionality of the data would be that it is equal to 2. Moreover, after projecting along the direction of the principal component (the straight line in Figure 6.6b), corresponding to λ_0, neighboring information is lost since points from different locations are mixed together.

Next, the Laplacian eigenmap technique for dimensionality reduction is employed, with $\varepsilon = 0.2$ and $\sigma = \sqrt{0.5}$. The results obtained are shown in Figure 6.6c. Looking from right to left, we see that the Laplacian method nicely "unfolds" the spiral in an one-dimensional straight line. Furthermore, neighboring information is retained in this one-dimensional

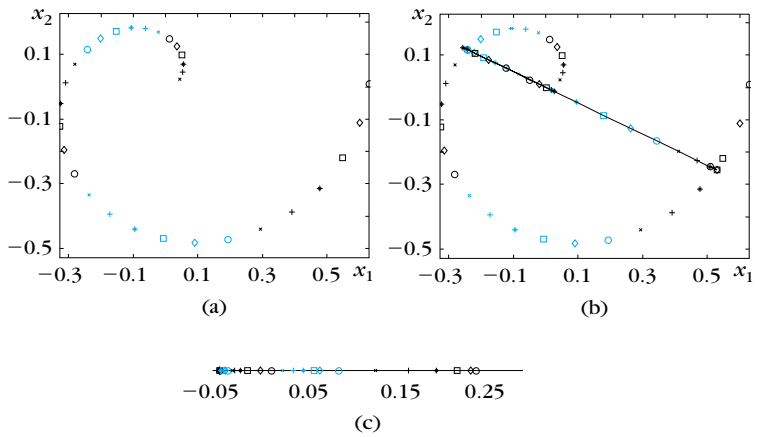


FIGURE 6.6

(a) A spiral of Archimedes in the two-dimensional space. (b) The previous spiral together with the projections of the sampled points on the direction of the first principal component, resulting from PCA. It is readily seen that neighboring information is lost after the projection and points corresponding to different parts of the spiral overlap. (c) The one-dimensional map of the spiral using the Laplacian method. In this case, the neighboring information is retained after the nonlinear projection and the spiral nicely unfold to a one-dimensional line.

representation of the data. Black and red areas are succeeding each other in the right order, and also, observing the symbols, one can see that neighbors are mapped to neighbors.

Example 6.7

Figure 6.7 shows samples from a three-dimensional spiral, parameterized as $x_1 = a\theta\cos\theta$, $x_2 = a\theta\sin\theta$, and sampled at $\theta = 0.5\pi,\ 0.7\pi,\ 0.9\pi, \ldots, 2.05\pi$ (θ is expressed in radians), $a = 0.1$ and $x_3 = -1,\ -0.8,\ -0.6, \ldots, 1$.

For illustration purposes and in order to keep track of the "identity" of each point, we have used red crosses and dots interchangeably, as we move upward in the x_3 dimension. Also, the first, the middle and the last points for each level of x_3 are denoted by black "◇," black "?" and black "□," respectively. Basically, all points at the same level lie on a two-dimensional spiral.

Figure 6.8 shows the two-dimensional mapping of the three-dimensional spiral using the Laplacian method for dimensionality reduction, with parameter values $\epsilon = 0.35$ and $\sigma = \sqrt{0.5}$. Comparing Figures 6.7 and 6.8, we see that all points corresponding to the same level x_3 are mapped across the same line, with the first point being mapped to the first one and so on. That is, as was the case in Example 6.6, the Laplacian method unfolds the three-dimensional spiral into a two-dimensional surface, while retaining neighboring information.

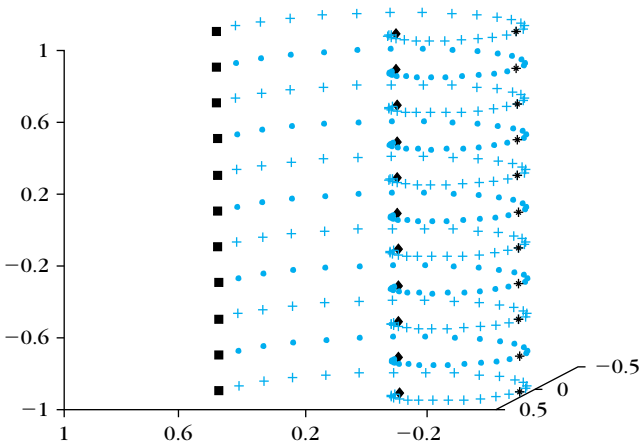


FIGURE 6.7

Samples from a three-dimensional spiral. One can think of it as a number of two-dimensional spirals one above the other. Different symbols have been used in order to track neighboring information.

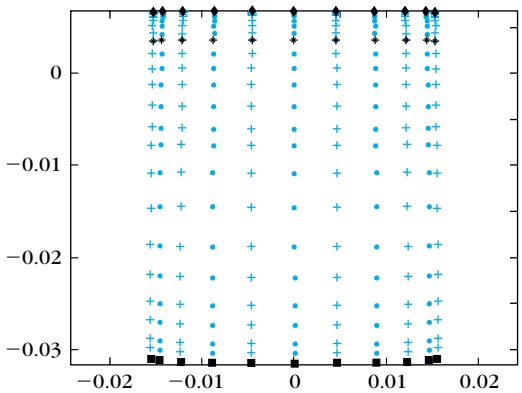


FIGURE 6.8

Two-dimensional mapping of the spiral of Figure 6.7 using the Laplacian eigenmap method. The three-dimensional structure is unfolded to the two-dimensional space by retaining the neighboring information.

6.8 THE DISCRETE FOURIER TRANSFORM (DFT)

We have already seen that the basis vectors/images for the KL and SVD expansions are not fixed but are "problem dependent" and are the result of an optimization

process. This is the reason for their optimality with respect to the decorrelation and information-packing properties. At the same time, this accounts for their major disadvantage, that is, their high computational complexity. For the rest of the chapter we will be concerned with transforms that use *fixed* basis vectors/images. Their suboptimality with respect to decorrelation and information packing properties is most often compensated by their low computational requirements.

6.8.1 One-Dimensional DFT

Given N input samples $x(0), x(1), \ldots, x(N-1)$, their DFT is defined as

$$y(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) \exp\left(-j\frac{2\pi}{N}kn\right), \quad k = 0, 1, \ldots, N-1 \tag{6.88}$$

and the inverse DFT as

$$x(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} y(k) \exp\left(j\frac{2\pi}{N}kn\right), \quad n = 0, 1, \ldots, N-1 \tag{6.89}$$

where $j \equiv \sqrt{-1}$. Sometimes, (6.88) is given without the normalizing factor $\frac{1}{\sqrt{N}}$. In such cases the normalizing factor in (6.89) becomes $\frac{1}{N}$. Collecting all $x(n)$ and $y(k)$ together into two $N \times 1$ vectors and defining

$$W_N \equiv \exp\left(-j\frac{2\pi}{N}\right) \tag{6.90}$$

(6.88) and (6.89) are written in a matrix form as

$$\boldsymbol{y} = W^H \boldsymbol{x}, \; \boldsymbol{x} = W\boldsymbol{y} \tag{6.91}$$

where

$$W^H = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & W_N & W_N^2 & \ldots & W_N^{N-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \ldots & W_N^{(N-1)(N-1)} \end{bmatrix} \tag{6.92}$$

It is not difficult to see that W is a unitary and symmetric matrix

$$W^{-1} = W^H = W^*$$

The basis vectors are the columns of W. For example, for $N = 4$

$$W = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}$$

and the basis vectors are

$$\boldsymbol{w}_0 = \frac{1}{2}[1, 1, 1, 1]^T$$
$$\boldsymbol{w}_1 = \frac{1}{2}[1, j, -1, -j]^T$$
$$\boldsymbol{w}_2 = \frac{1}{2}[1, -1, 1, -1]^T$$
$$\boldsymbol{w}_3 = \frac{1}{2}[1, -j, -1, j]^T$$

and

$$\boldsymbol{x} = \sum_{i=0}^{3} y(i)\boldsymbol{w}_i$$

The direct computation of (6.91) requires $O(N^2)$ computations. However, taking advantage of the specific structure of the matrix W, a substantial saving in computations is possible via the celebrated Fast Fourien Transform (FFT) algorithm, which computes each equation of (6.91) in $O(N \log_2 N)$ operations [Proa 92].

So far, the DFT has been introduced as a special type of a linear unitary transform of one vector to another. Another point of view, which will be useful later in this chapter, is to see the DFT as a means of expanding a sequence $x(n)$ into a set of N *basis sequences* $h_k(n)$

$$x(n) = \sum_{k=0}^{N-1} y(k)h_k(n)$$

where

$$h_k(n) = \begin{cases} \frac{1}{\sqrt{N}} \exp(j\frac{2\pi}{N}kn), & n = 0, 1, \ldots, N-1 \\ 0, & \text{otherwise} \end{cases}$$

and $y(k)$ are the coefficients of the expansion. The DFT basis sequences belong to a more general class of sequences known as orthonormal, that is,

$$\langle h_l(n), h_k(n)\rangle \equiv \sum_n h_k(n)h_l^*(n) = \delta_{kl} \tag{6.93}$$

where $\langle .\,, .\rangle$ is known as the inner product of the sequences $h_k(n), h_l(n)$. For the DFT expansion, we have

$$\langle h_k(n), h_l(n)\rangle = \frac{1}{N}\sum_{n=0}^{N-1} \exp\Big(j\frac{2\pi}{N}kn\Big)\exp\Big(-j\frac{2\pi}{N}ln\Big)$$

$$= \frac{1}{N}\sum_{n=0}^{N-1} \exp\Big(j\frac{2\pi}{N}(k-l)n\Big), \quad k, l = 0, 1, \ldots, N-1$$

However, it can easily be shown (Problem 6.8) that

$$\frac{1}{N}\sum_{n=0}^{N-1}\exp\left(j\frac{2\pi}{N}(k-l)n\right)=\begin{cases}1, & l=k+rN, \quad r=0,\pm1,\pm2,\ldots\\ 0, & \text{otherwise}\end{cases} \tag{6.94}$$

Hence

$$\langle h_k(n), h_l(n)\rangle = \delta_{kl}$$

6.8.2 Two-Dimensional DFT

Given an $N\times N$ matrix/image, its two-dimensional DFT is defined as

$$Y(k,l)=\frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}X(m,n)W_N^{km}W_N^{ln} \tag{6.95}$$

and the inverse DFT as

$$X(m,n)=\frac{1}{N}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}Y(k,l)W_N^{-km}W_N^{-ln} \tag{6.96}$$

It is readily seen that this can be written in a compact form as

$$Y=W^HXW^H, \quad X=WYW \tag{6.97}$$

Thus, the two-dimensional DFT is a separable transform with basis images $\boldsymbol{w}_i\boldsymbol{w}_j^T$, $i,j=0,1,\ldots,N-1$. It is apparent from (6.97) that the number of operations required for the respective computations is $O(N^2\log_2 N)$, that is, the number of additions and multiplications needed for $2N$ one-dimensional DFTs, via the FFT algorithm.

6.9 THE DISCRETE COSINE AND SINE TRANSFORMS

Given N input samples $x(0),x(1),\ldots,x(N-1)$ their *discrete cosine transform* (DCT) is defined as

$$y(k)=\alpha(k)\sum_{n=0}^{N-1}x(n)\cos\left(\frac{\pi(2n+1)k}{2N}\right), \quad k=0,1,\ldots,N-1 \tag{6.98}$$

and the inverse DCT is given by

$$x(n)=\sum_{k=0}^{N-1}\alpha(k)y(k)\cos\left(\frac{\pi(2n+1)k}{2N}\right), \quad n=0,1,\ldots,N-1 \tag{6.99}$$

where

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}}, & k = 0 \\ \sqrt{\frac{2}{N}}, & k \neq 0 \end{cases}$$

In vector form the transform is written as

$$\boldsymbol{y} = C^T \boldsymbol{x}$$

where the elements of the matrix C are given by

$$C(n,k) = \frac{1}{\sqrt{N}}, \quad k = 0,\ 0 \leq n \leq N-1$$

$$C(n,k) = \sqrt{\frac{2}{N}} \cos\Big(\frac{\pi(2n+1)k}{2N}\Big), \quad 1 \leq k \leq N-1, 0 \leq n \leq N-1$$

Matrix C has real elements, and it is easy to see that it is orthogonal,

$$C^{-1} = C^T$$

The two-dimensional DCT is the separable transform defined as

$$Y = C^T X C, \quad X = CYC^T \tag{6.100}$$

The *discrete sine transform* (DST) is defined via the transform matrix

$$S(k,n) = \sqrt{\frac{2}{N+1}} \sin\left(\frac{\pi(k+1)(n+1)}{N+1}\right), \quad k,n = 0,1,\ldots,N-1$$

and it is also an orthogonal transform. The DCT and DST belong to the family of transforms that can be computed via a fast method in $O(N \log_2 N)$ operations [Jain 89, Lim 90].

Remark

- The DCT and DST have very good information-packing properties for most of the images, in the sense that they concentrate most of the energy in a few coefficients. An explanation for this property is that both offer a close approximation to the (KL) transform for a class of random signals, known as first-order Markov processes, which can approximately model a number of real-world images [Jain 89]. Figure 6.9 shows an image and the resulting DFT (magnitude), DST, and DCT transforms. It is apparent from the figure that the high-intensity coefficients of the transforms (dark) are concentrated in a small area, whose size depends on the energy compaction properties of the respective transform. This area is smaller for DCT and DST than for DFT.

FIGURE 6.9

Example of an image and its magnitude DFT, DST, and DCT transforms, shown from top left to bottom left in the clockwise sense.

6.10 THE HADAMARD TRANSFORM

The Hadamard transform and the Haar transform, to be considered in the next section, share a serious computational advantage over the previously considered DFT, DCT, and DST transforms. Their unitary matrices consist of ± 1, and the transforms are computed via additions and subtractions only, with no multiplications being involved. Hence, for processors for which multiplication is a time-consuming operation, a substantial saving is obtained.

The Hadamard unitary matrix of order n is the $N \times N$ matrix, $N = 2^n$, generated by the following iteration rule:

$$H_n = H_1 \otimes H_{n-1} \tag{6.101}$$

where

$$H_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{6.102}$$

and $\otimes$ denotes the Kronecker product of two matrices

$$A \otimes B = \begin{bmatrix} A(1,1)B & A(1,2)B & \dots & A(1,N)B \\ \vdots & \vdots & \vdots & \vdots \\ A(N,1)B & A(N,2)B & \dots & A(N,N)B \end{bmatrix}$$

where $A(i,j)$ is the (i,j) element of A, $i,j = 1,2,\dots,N$. Thus, according to (6.101), (6.102), it is

$$H_2 = H_1 \otimes H_1 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

and for $n = 3$

$$H_3 = H_1 \otimes H_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix}$$

It is not difficult to show the orthogonality of H_n, $n = 1,2,\dots$, that is,

$$H_n^{-1} = H_n^T = H_n$$

For a vector $\boldsymbol{x}$ of N samples and $N = 2^n$ the transform pair is

$$\boldsymbol{y} = H_n\boldsymbol{x}, \quad \boldsymbol{x} = H_n\boldsymbol{y} \tag{6.103}$$

The two-dimensional Hadamard transform is given by

$$Y = H_nXH_n, \quad X = H_nYH_n \tag{6.104}$$

The Hadamard transform has good to very good energy packing properties. Fast algorithms for its computation in $O(N \log_2 N)$ subtractions and/or additions are also available [Jain 89].

Remark

- Experimental results using the DCT, DST, and Hadamard transforms for texture discrimination have shown that the performance obtained was close to that of the optimal KL transform [Unse 86, Unse 89]. At the same time, this near-optimal performance is obtained at substantially reduced complexity, due to the availability of fast computational schemes as reported before.

6.11 THE HAAR TRANSFORM

The starting point for the definition of the Haar transform is the Haar functions $h_k(z)$, which are defined in the closed interval $[0,1]$. The order k of the function is

Table 6.1 Parameters for the Haar functions

k	0	1	2	3	4	5	6	7
p	0	0	1	1	2	2	2	2
q	0	1	1	2	1	2	3	4

uniquely decomposed into two integers p, q

$$k = 2^p + q - 1, \quad k = 0, 1, \ldots, L-1, \text{ and } L = 2^n$$

where

$$0 \le p \le n-1,\ 0 \le q \le 2^p \quad \text{for} \quad p \ne 0 \text{ and } q = 0 \text{ or } 1 \text{ for } p = 0$$

Table 6.1 summarizes the respective values for $L = 8$. The Haar functions are

$$h_0(z) \equiv h_{00}(z) = \frac{1}{\sqrt{L}}, \quad z \in [0, 1] \tag{6.105}$$

$$h_k(z) \equiv h_{pq}(z) = \frac{1}{\sqrt{L}} \begin{cases} 2^{\frac{p}{2}} & \frac{q-1}{2^p} \le z < \frac{q-\frac{1}{2}}{2^p} \\ -2^{\frac{p}{2}} & \frac{q-\frac{1}{2}}{2^p} \le z < \frac{q}{2^p} \\ 0 & \text{otherwise in } [0, 1] \end{cases} \tag{6.106}$$

The Haar transform matrix of order L consists of rows resulting from the preceding functions computed at the points $z = \frac{m}{L}, m = 0, 1, 2, \ldots, L-1$. For example, the 8×8 transform matrix is

$$H = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix} \tag{6.107}$$

It is not difficult to see that

$$H^{-1} = H^T$$

that is, H is orthogonal. The energy packing properties of the Haar transform are not very good. However, its importance for us lies beyond that. We will use it as the vehicle to take us from the world of unitary transforms to that of multiresolution analysis. To this end, let us look carefully at the Haar transform matrix. We readily

observe its sparse nature with a number of zeros, whose location reveals an underlying cyclic shift mechanism. To satisfy our curiosity as to why this happens, let us look at the Haar transform from a different perspective.

6.12 THE HAAR EXPANSION REVISITED

Let us split our original set of N input samples (N even) $x(0), x(1), \ldots, x(N-1)$ into successive blocks of two, that is, $(x(2k), x(2k+1)), k = 0, 1, \ldots, \frac{N}{2} - 1$, and apply the Haar transform of order $L = 2$. For each pair of input samples, a pair of transformed samples is obtained,

$$\begin{bmatrix} y_1(k) \\ y_0(k) \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x(2k) \\ x(2k+1) \end{bmatrix}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{6.108}$$

That is,

$$y_1(k) = \frac{1}{\sqrt{2}}(x(2k) + x(2k+1)) \tag{6.109}$$

$$y_0(k) = \frac{1}{\sqrt{2}}(x(2k) - x(2k+1)), \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{6.110}$$

This can be interpreted as the action—on the sequence of N input samples—of two (noncausal) filters with impulse responses ($h_1(0) = \frac{1}{\sqrt{2}}, h_1(-1) = \frac{1}{\sqrt{2}}$) and ($h_0(0) = \frac{1}{\sqrt{2}}, h_0(-1) = -\frac{1}{\sqrt{2}}$), respectively. The corresponding transfer functions (Appendix D) are

$$H_1(z) = \frac{1}{\sqrt{2}}(1 + z) \tag{6.111}$$

$$H_0(z) = \frac{1}{\sqrt{2}}(1 - z) \tag{6.112}$$

In other words, the order $L = 2$ Haar transform computes the output samples of the two filters when they are fed with the input sequence $x(n), n = 0, 1, 2, \ldots, N-1$. *Furthermore, the output sequence samples are computed for every other sample of the input sequence, at even time instants* $0, 2, 4, \ldots,$ *as* (6.109) and (6.110) suggest. This operation is portrayed in Figure 6.10b. The operation at the output of the two filters is known as *subsampling by* M, in this case $M = 2$, and it is defined in Figure 6.10a. In other words, from the samples generated at the filter output we keep one every $M(= 2)$. In the time domain and for an input sequence consisting of eight samples, the output, $y_0(k)$, of the H_0 branch of Figure 6.10b will consist of

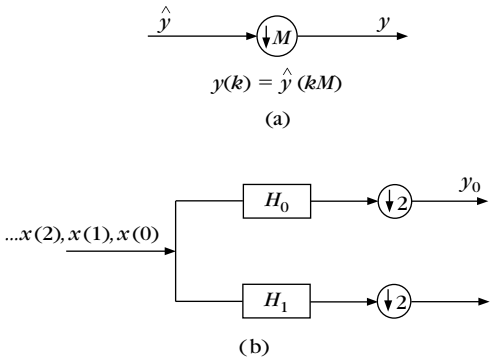


FIGURE 6.10
(a) Subsampling operation and (b) filtering interpretation of the Haar transform.

four samples given by

$$\begin{bmatrix} y_0(0) \\ y_0(1) \\ y_0(2) \\ y_0(3) \end{bmatrix} = \left[\begin{array}{cc|cc|cc|cc} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{array}\right] \begin{bmatrix} x(0) \\ x(1) \\ - - \\ x(2) \\ x(3) \\ - - \\ x(4) \\ x(5) \\ - - \\ x(6) \\ x(7) \end{bmatrix} \tag{6.113}$$

This is nothing other than the action of the last four rows of the 8×8 Haar transform in (6.107)! What about the rest? Let us carry on the splitting of Figure 6.10b one step further, as shown in Figure 6.11. Using the easily shown Noble identity illustrated in Figure 6.12a (Problem 6.17), the structure of Figure 6.11 turns out to be equivalent to that of Figure 6.12b. Taking into account the subsampling operation of the lower branch after the filters H_0 and H_1, the Noble identity leads to

$$\hat{F}_1(z) = \frac{1}{2}(1+z)(1-z^2) = \frac{1}{2}(1+z-z^2-z^3) \tag{6.114}$$

$$\hat{F}_2(z) = \frac{1}{2}(1+z)(1+z^2) = \frac{1}{2}(1+z+z^2+z^3) \tag{6.115}$$

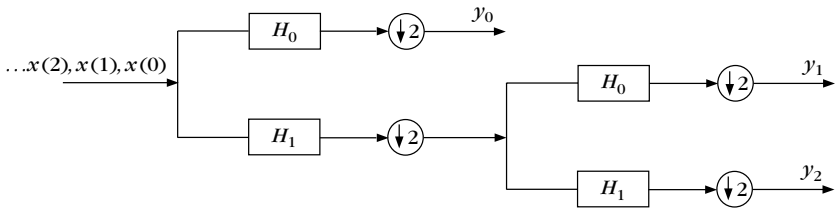


FIGURE 6.11

Two-stage filtering followed by subsampling operation.

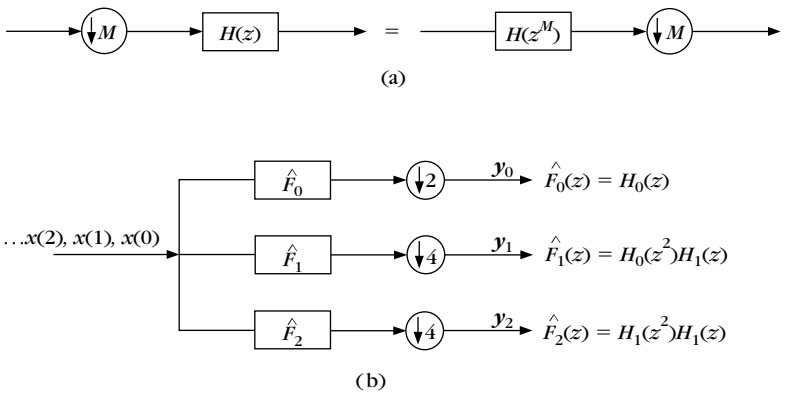


FIGURE 6.12

(a) Noble identity I and (b) equivalent filter bank of Figure 6.11.

From the transfer function $\hat{F}_1(z)$ and taking into account the subsampling by 4 (2×2) operation, the first two samples of the $y_1(k)$ sequence are given by

$$\begin{bmatrix} y_1(0) \\ y_1(1) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(7) \end{bmatrix} \tag{6.116}$$

This is nothing but the action of the third and fourth rows of the 8×8 Haar transform on the input vector. If we now carry on the splitting one step further, as in Figure 6.13, it is straightforward to show by repeating the preceding arguments that

$$y_2(0) = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(7) \end{bmatrix} \tag{6.117}$$

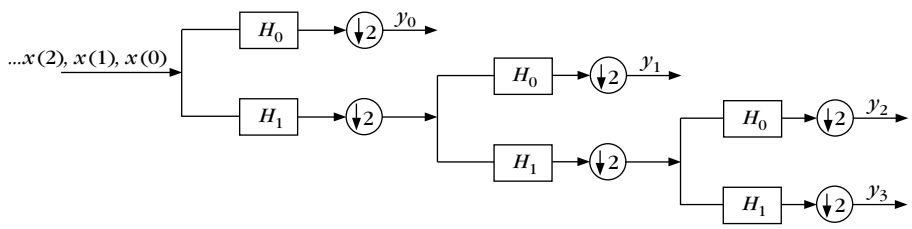


FIGURE 6.13

Tree-structured filter bank.

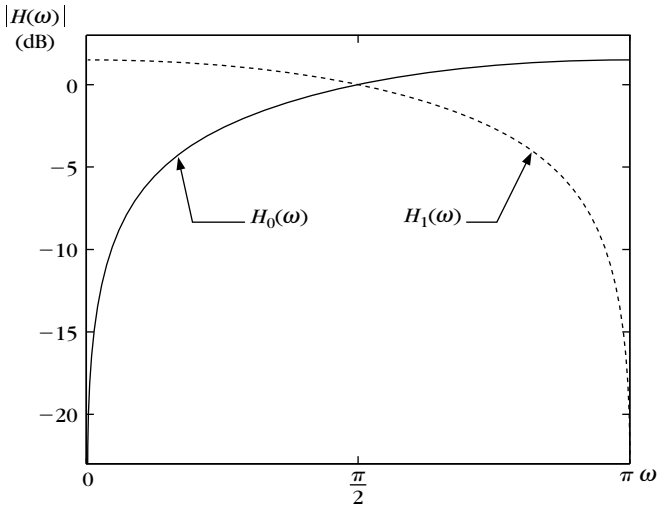


FIGURE 6.14

Magnitude of the frequency response for the two Haar filters. H_1 is a low-pass and H_0 high-pass filter.

and

$$y_3(0) = \frac{1}{\sqrt{8}}\begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}\begin{bmatrix}x(0)\\ x(1)\\ \vdots \\ x(7)\end{bmatrix} \tag{6.118}$$

These equations are the actions of the second and first rows of the Haar transform on the input vector. The structure of Figure 6.13 is known as a (three-level) *tree-structured filter bank* generated by the filters $H_0(z)$ and $H_1(z)$. Figure 6.14 shows the frequency responses of these two filters. One ($H_0(z)$) is a high-pass and the other a low-pass filter. Herein lies the importance of the filter bank interpretation of the Haar transform. The input sequence $x(n)$ is first split into two

versions *of lower resolution* with respect to the original one: a low-pass (average) *coarser* resolution version and a high-pass (difference) *detailed* resolution one. In the sequel the coarser resolution version is further split into two versions, and so on. This leads to a number of versions with a hierarchy of resolutions. This decomposition is known as *multiresolution decomposition*.

The idea of multiresolution decomposition has been around for some time [Burt 83, Akan 93] and has been exploited in various applications and for a number of reasons. Its popularity as a tool in pattern recognition is mainly due to the information compaction capabilities associated with such a decomposition, *provided filters H_0, H_1 are properly designed*. For many types of signals, such as speech and images, most of the information is localized in certain resolution levels. Thus, most of the energy is concentrated in a (relatively) small number of samples, which carry most of the necessary information [Este 77, Mall 89]. Some fundamental issues related to the multiresolution decomposition and the design of filter banks will be highlighted next.

6.13 DISCRETE TIME WAVELET TRANSFORM (DTWT)

The goal of this section is twofold. We first free ourselves from the Haar functions, and we seek the possibility of using other filters in place of H_0, H_1. There is more than one reason for this generalization. An obvious reason is that the frequency responses of the Haar filters are far from ideal. If our aim is to split the original sequence into a hierarchy of "coarse" and "detailed" versions, we should require the filters that perform the splitting to be as close as possible to the ideal low/high-pass ones (Figure 6.15). Our next concern in this section is the inversion problem.

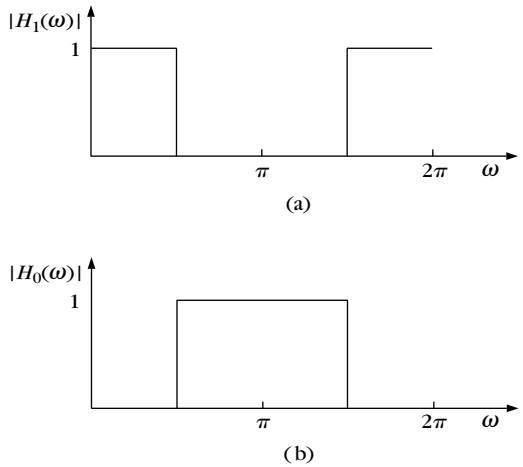


FIGURE 6.15

Ideal frequency responses for (a) low-pass and (b) high-pass filters.

That is, if we know the lower resolution versions, can we obtain the original sequence, $x(n)$, as was the case with the unitary transforms? We will show that under certain constraints in the design of H_0 and H_1 this is indeed possible.

The Two-Band Case

Let us start with the simple two-band case of Figure 6.10b, where we now assume that the filters are not the Haar ones. If $h_0(k), h_1(k)$ are the respective impulse responses, then we can write

$$y_0(k) = \sum_l x(l)h_0(n-l)|_{n=2k}$$

$$y_1(k) = \sum_l x(l)h_1(n-l)|_{n=2k}$$

where $y_1(k)$ is the output of the lower branch of Figure 6.10b. Collecting all $y_0(k), y_1(k), k = 0, 1, 2, \ldots$, together in a vector, we have

$$\begin{bmatrix} \vdots \\ y_0(0) \\ y_1(0) \\ y_0(1) \\ y_1(1) \\ y_0(2) \\ y_1(2) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \ldots & h_0(2) & h_0(1) & h_0(0) & h_0(-1) & h_0(-2) & \ldots \\ \ldots & h_1(2) & h_1(1) & h_1(0) & h_1(-1) & h_1(-2) & \ldots \\ \ldots & \ldots & \ldots & h_0(2) & h_0(1) & h_0(0) & \ldots \\ \ldots & \ldots & \ldots & h_1(2) & h_1(1) & h_1(0) & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & h_0(2) & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & h_1(2) & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ x(0) \\ x(1) \\ x(2) \\ \vdots \end{bmatrix}$$

or

$$\boldsymbol{y} = T_i \boldsymbol{x} \tag{6.119}$$

Here, we have assumed that the filters can be noncausal, but they are of *finite impulse response* (FIR) (the impulse response has a finite nonzero number of terms). The latter assumption is imposed here in order to avoid issues of convergence of infinite series. Observe the structure of T_i. It basically consists of two rows, one with the impulse response of H_0 and the other with that of H_1, which are then shifted each time by two to the right, to form the rest of the rows. This is the result of the subsampling, by two, operation. Figure 6.16b shows a structure that combines $y_0(k), y_1(k)$, through the filters G_0, G_1, to form a sequence $\hat{x}$. The symbol at the input of the filters denotes the *upsampling* by M operation, which is defined in Figure 6.16a. In this case $M = 2$. In other words, this is equivalent to stuffing $M - 1$ zeros between every two samples. That is, the input sequences of the filters G_0, G_1 will be

$$\ldots 0\, y_0(0)\, 0\, y_0(1)\, 0\, y_0(2)\, 0 \ldots$$

$$\ldots 0\, y_1(0)\, 0\, y_1(1)\, 0\, y_1(2)\, 0 \ldots$$

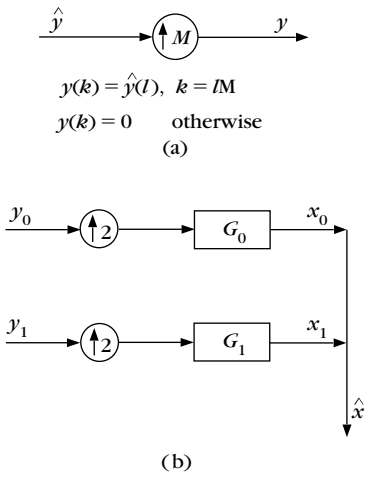


FIGURE 6.16

(a) The upsampling operation. (b) Tree-structured synthesis filter bank.

respectively. Thus, every other sample of the impulse response hits a zero and

$$x_0(n) = \sum_k y_0(k) g_0(n - 2k)$$

$$x_1(n) = \sum_k y_1(k) g_1(n - 2k)$$

$$\hat{x}(n) = x_0(n) + x_1(n)$$

Filters G_i are known as the *synthesis filters* and the corresponding H_i, of Figure 6.10b, as the *analysis filters*. Collecting all $\hat{x}(n)$ together, it is not difficult to see that

$$\begin{bmatrix} \vdots \\ \hat{x}(0) \\ \hat{x}(1) \\ \hat{x}(2) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & g_0(0) & g_1(0) & g_0(-2) & g_1(-2) & \cdots \\ \cdots & g_0(1) & g_1(1) & g_0(-1) & g_1(-1) & \cdots \\ \cdots & g_0(2) & g_1(2) & g_0(0) & g_1(0) & \cdots \\ \cdots & g_0(3) & g_1(3) & g_0(1) & g_1(1) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ y_0(0) \\ y_1(0) \\ y_0(1) \\ y_1(1) \\ \vdots \end{bmatrix}$$

or

$$\hat{\boldsymbol{x}} = T_o \boldsymbol{y} \tag{6.120}$$

In order that $\hat{\boldsymbol{x}} = \boldsymbol{x}$ we require that [Vett 92]

$$T_o T_i = I = T_i T_o \tag{6.121}$$

Multiplying rows of T_i with columns of T_o, (6.121) becomes equivalent to

$$\sum_n h_i(2k-n)g_j(n-2l) = \delta_{kl}\delta_{ij}, \quad i,j = 0,1 \tag{6.122}$$

or according to the definition of the inner product in (6.93),

$$\langle h_i(2k-n), g_j(n-2l)\rangle = \delta_{kl}\delta_{ij}$$

If (6.122) is satisfied, we say that the two-band filter bank is a *perfect reconstruction* one and $\hat{x}(n) = x(n)$. Thus,

$$x(n) = \sum_k y_0(k)g_0(n-2k) + \sum_k y_1(k)g_1(n-2k) \tag{6.123}$$

Equation (6.123) can also be seen from a different perspective. It is an expansion of $x(n)$ into a set of *basis sequences*

$$\{g_0(n-2k), g_1(n-2k)\}, \quad k \in Z$$

where Z is the set of the integer numbers. From such a point of view $y_0(k), y_1(k)$ are the respective coefficients of the expansion. This is known as the *discrete time wavelet transform (DTWT)* and the coefficients $y_0(k), y_1(k)$ as the *discrete time wavelet coefficients*. Thus, given a perfect reconstruction two-band filter bank (i.e., condition (6.122) is satisfied), the following transform pair is defined:

$$\begin{aligned} y_i(k) &= \sum_n x(n)h_i(2k-n) \ (a) \\ x(n) &= \sum_{i=0}^{1}\sum_k y_i(k)g_i(n-2k) \ (b) \end{aligned} \tag{6.124}$$

Remarks

- Two sets of basis functions are involved, namely,

  $$h_i(2k-n) \equiv \phi_{ik}(n), \ g_j(n-2l) \equiv \psi_{jl}(n) \quad i,j = 0,1 \text{ and } k,l \in Z$$

  Equation (6.122) is an orthogonality condition between $\phi_{ik}(n)$ and $\psi_{jl}(n)$, that is,

  $$\langle \phi_{ik}(n), \psi_{jl}(n)\rangle = \delta_{ij}\delta_{kl}$$

  and it is known as the *biorthogonality condition*. The discrete time wavelet transform pair in (6.124) is a *biorthogonal expansion*.

- The basis sequences $\phi_{ik}(n)$ and $\psi_{jl}(n)$ of the expansion are *shifts by an even number of samples* of four basic mother sequences $g_0(n)$, $g_1(n)$, $h_0(-n)$, $h_1(-n)$, which are the impulse responses of the synthesis and the time-reversed analysis filters. For the recovery of $x(n)$ from its discrete time wavelet coefficients, each coefficient $y_i(k)$ weighs and adds a copy of the mother sequences $g_i(n)$ shifted by $2k$.

- When the sequences $\phi_{ik}(n) = h_i(2k - n)$ are themselves orthogonal, that is,

$$\sum_n h_j^i(2k - n)h_j(2l - n) = \delta_{kl}\delta_{ij}, \quad i,j = 0, 1 \text{ and } k,l \in Z$$

then

$$g_i(n) = h_i(-n)$$

That is, the synthesis filters are the time reverse of the analysis ones. Such a filter bank is known as orthogonal or *paraunitary*, and we have the same set of mother sequences (h_i only) involved in both equations of the discrete time wavelet transform (6.124).

- A number of orthogonal and biorthogonal perfect reconstruction filter pairs have been proposed in the literature [Daub 90, Vett 95]. Table 6.2 gives the coefficients for the first four of Daubechies's maximally flat orthogonal filters. The low-pass version $h_1(n)$ is shown. The high-pass versions are obtained as $h_0(n) = (-1)^n h_1(-n + 2L - 1)$, where L is the length of the filters.

- Besides the case of wavelet basis sequences with predefined values, a large research effort has been devoted to constructing such sequences that are optimized to the specific problem of interest. This has also been used in pattern recognition applications. For example, in [Mall 97] it is proposed to design the filters of the bank to optimize a class discriminant criterion. A different approach is followed in [Szu 92], where an optimal linear combination of predefined bases is sought for classification of speech signals.

- When implementing filter banks in practice, noncausal filters have to be appropriately delayed to make them realizable (Appendix D). This makes it

Table 6.2 Daubechies's Low-pass Filters of Length 4, 6, 8, and 10

$h_1(0)$	0.4829629	0.33267	0.2303778	0.1601024
$h_1(1)$	0.8365163	0.806891	0.7148466	0.6038293
$h_1(2)$	0.2241439	0.459877	0.6308808	0.7243085
$h_1(3)$	−0.1294095	−0.135011	−0.0279838	0.1384281
$h_1(4)$		−0.08544	−0.1870348	−0.2422949
$h_1(5)$		0.03522	0.0308414	−0.0322449
$h_1(6)$			0.0328830	0.0775715
$h_1(7)$			−0.0105974	−0.0062415
$h_1(8)$				−0.0125807
$h_1(9)$				0.0033357

necessary to involve certain delay elements at different points, in order to safeguard the perfect reconstruction property of the analysis–synthesis bank (Problem 6.19).

- In practice, the number of input samples $x(n)$ is finite, that is, $n = 0, 1, \ldots, N-1$. Thus, for the computation of (6.124) some initial conditions are required. Zero, periodic, or symmetric extensions of the data are popular alternatives. Such implementation issues as well as algorithms for the efficient computation of the DTWT coefficients are discussed in [Vett 95, Chapter 6].

Many Bands Case

Figure 6.17 shows the synthesis part corresponding to the analysis bank of Figure 6.13, and it is a generalization of the two-band synthesis concept. Using the easily shown Noble identity given in Figure 6.18 (Problem 6.17), we end up with the equivalent structure of the synthesis part, shown in Figure 6.19. Let $f_i(n)$ be the impulse responses of the F_i filters. It is easy to see that the respective contribution of each $y_i(k)$ sequence to the output $\hat{x}(n)$ is

$$x_i(n) = \sum_k y_i(k) f_i(n - 2^{i+1}k) \quad i = 0, 1, \ldots, J-2$$

$$x_{J-1}(n) = \sum_k y_{J-1}(k) f_{J-1}(n - 2^{J-1}k)$$

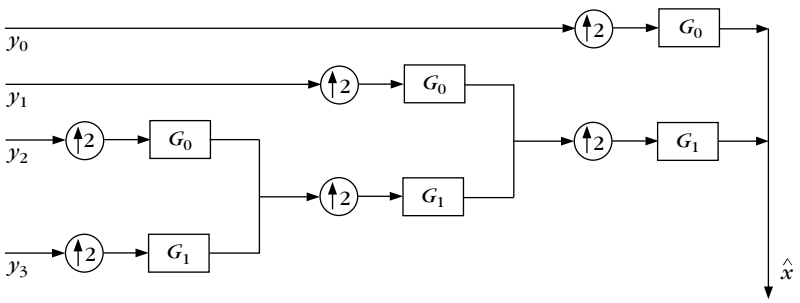


FIGURE 6.17

Tree-structured synthesis filter bank.

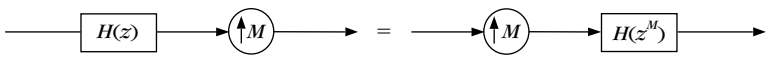


FIGURE 6.18

Noble identity II.

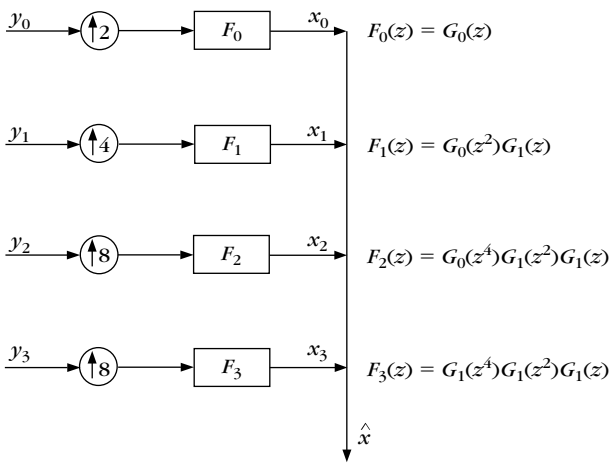


FIGURE 6.19

Equivalent of the tree-structured filter bank of Figure 6.17.

$$\hat{x}(n) = \sum_{i=0}^{J-1} x_i(n)$$

where J is the number of bands, with $J = 4$ in the case of Figures 6.17 and 6.19. It can be shown [Vaid 93] that if the mother filters G_0, G_1, which generate the synthesis part, and the mother filters H_0, H_1 of the analysis part *satisfy the biorthogonality condition* (6.122), *then the J-level analysis–synthesis bank is also a perfect reconstruction filter bank*, that is,

$$\hat{x}(n) = x(n) = \sum_{i=0}^{J-2} \sum_{k} y_i(k) f_i(n - 2^{i+1}k) + \sum_{k} y_{J-1}(k) f_{J-1}(n - 2^{J-1}k) \tag{6.125}$$

where

$$y_i(k) = \sum_{n} x(n) \hat{f}_i(2^{i+1}k - n), \quad i = 0, 1, \ldots, J-2 \tag{6.126}$$

$$y_{J-1}(k) = \sum_{n} x(n) \hat{f}_{J-1}(2^{J-1}k - n) \tag{6.127}$$

with $\hat{f}_i(k)$ being the impulse responses of the corresponding analysis band, in analogy with Figure 6.12b. To summarize our findings, let us define

$$\psi_{ik}(n) = f_i(n - 2^{i+1}k), \quad i = 0, 1, \ldots, J-2$$
$$\psi_{(J-1)k}(n) = f_{J-1}(n - 2^{J-1}k)$$

Table 6.3 The Discrete Time Wavelet Transform

$y_i(k) = \sum_n x(n)\phi_{ik}(n)$	DTWT
$x(n) = \sum_i \sum_k y_i(k)\psi_{ik}(n)$	Inverse DTWT
$\sum_n \phi_{ik}(n)\psi_{jl}(n) = \delta_{kl}\delta_{ij}$	Biorthogonal expansion
$\psi_{ik}(n) = \phi_{ik}(n)$ $\sum_n \phi_{ik}(n)\phi_{jl}(n) = \delta_{kl}\delta_{ij}$	Orthonormal expansion

$$\phi_{ik}(n) = \hat{f}_i(2^{i+1}k - n) \quad i = 0, 1, \ldots, J-2$$
$$\phi_{(J-1)k}(n) = \hat{f}_{J-1}(2^{J-1}k - n)$$

Then from (6.125), (6.126), and (6.127) we obtain Table 6.3.

Remarks

- A notable characteristic of the DTWT is that the basis sequences for each level i are the power of two shifts of a corresponding mother sequence:

$$\begin{aligned} \psi_{ik}(n) = \psi_{i0}(n - 2^r k), \quad & r = i + 1 && \text{for } i \neq J-1 \\ & \text{or } r = J - 1 && \text{for } i = J-1 \\ \phi_{ik}(n) = \phi_{i0}(n - 2^r k), \quad & r = i + 1 && \text{for } i \neq J-1 \\ & \text{or } r = J - 1 && \text{for } i = J-1 \end{aligned}$$

  In the more elegant theory of continuous wavelet transform, all analysis (synthesis) basis functions are produced from a *single* analysis (synthesis) mother function by dilations (time scaling) and shifts [Meye 93, Daub 90, Vett 95].

- The magic number 2, whose powers determine the shifts in the mother basis sequences, results from the successive splitting by two in the tree-structured filter banks, which we have adopted to introduce the DTWT. Filter banks of this type are known as octave-band filter banks. Their characteristic is that the bandwidth of each of the filters in the bank is the same in a logarithmic scale. Sometimes they are also called constant-Q filter banks to stress the fact that the ratio of the filters' bandwidth to the respective central frequency is constant. Generalizations of DTWT with another integer M in place of 2 can also be defined and used [Stef 93].

Example 6.8

The Haar Transform—The Epilogue

We have already seen that the Haar transform is equivalent to a tree-structured analysis filter bank. Let us now look at the synthesis problem. For the 8×8 Haar transform and after a row reshuffling of the corresponding Haar matrix, we have

$$\begin{bmatrix} y_0(0) \\ y_0(1) \\ y_0(2) \\ y_0(3) \\ y_1(0) \\ y_1(1) \\ y_2(0) \\ y_3(0) \end{bmatrix} = \frac{1}{\sqrt{8}} \begin{bmatrix} 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(7) \end{bmatrix}$$

or

$$\boldsymbol{y} = \hat{H}\boldsymbol{x}$$

Thus the 8×8 Haar transform gives four coefficients at the finest resolution level 0, two at level 1 and one for each of the coarsest resolution levels 2 and 3. We will now design the corresponding synthesis bank to obtain $x(n)$ from these coefficients. The impulse responses of the Haar analysis filters are

$$h_1(n) = \begin{cases} \frac{1}{\sqrt{2}} & n = 0 \text{ or } n = -1 \\ 0 & \text{otherwise} \end{cases}$$

$$h_0(n) = \begin{cases} \frac{1}{\sqrt{2}} & n = 0 \\ -\frac{1}{\sqrt{2}} & n = -1 \\ 0 & \text{otherwise} \end{cases}$$

It is readily seen that

$$\sum_n h_i(2k - n)h_j(2l - n) = \delta_{ij}\delta_{kl}, \quad i,j = 0,1$$

That is, the Haar filter bank is paraunitary. Thus, the synthesis filters can be defined as

$$g_i(n) = h_i(-n), \quad i = 0,1$$

Hence

$$G_0(z) = \frac{1}{\sqrt{2}}(1 - z^{-1})$$

$$G_1(z) = \frac{1}{\sqrt{2}}(1 + z^{-1})$$

From the equivalent structure of the synthesis bank of Figure 6.19 we have

$$F_1(z) = G_1(z)G_0\left(z^2\right) = \frac{1}{2}\left(1 + z^{-1} - z^{-2} - z^{-3}\right)$$

and the respective impulse response is

$$f_1(n) = \begin{cases} \frac{1}{2} & n = 0,1 \\ -\frac{1}{2} & n = 2,3 \\ 0 & \text{otherwise} \end{cases}$$

Following similar arguments, we have

$$f_2(n) = \begin{cases} \frac{1}{\sqrt{8}} & n = 0, 1, 2, 3 \\ -\frac{1}{\sqrt{8}} & n = 4, 5, 6, 7 \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(n) = \begin{cases} \frac{1}{\sqrt{8}} & n = 0, 1, \ldots, 7 \\ 0 & \text{otherwise} \end{cases}$$

If we now insert these values in (6.125) and collect the values of $\boldsymbol{x}(n)$ together, we get

$$\begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(7) \end{bmatrix} = \frac{1}{\sqrt{8}} \begin{bmatrix} 2 & 0 & 0 & 0 & \sqrt{2} & 0 & 1 & 1 \\ -2 & 0 & 0 & 0 & \sqrt{2} & 0 & 1 & 1 \\ 0 & 2 & 0 & 0 & -\sqrt{2} & 0 & 1 & 1 \\ 0 & -2 & 0 & 0 & -\sqrt{2} & 0 & 1 & 1 \\ 0 & 0 & 2 & 0 & 0 & \sqrt{2} & -1 & 1 \\ 0 & 0 & -2 & 0 & 0 & \sqrt{2} & -1 & 1 \\ 0 & 0 & 0 & 2 & 0 & -\sqrt{2} & -1 & 1 \\ 0 & 0 & 0 & -2 & 0 & -\sqrt{2} & -1 & 1 \end{bmatrix} \begin{bmatrix} y_0(0) \\ y_0(1) \\ y_0(2) \\ y_0(3) \\ y_1(0) \\ y_1(1) \\ y_2(0) \\ y_3(0) \end{bmatrix}$$

or

$$\boldsymbol{x} = \hat{\boldsymbol{H}}^T \boldsymbol{y}$$

That is, we reobtain the inverse (within a permutation) Haar transform. Hence, we can now state that *the Haar transform and its inverse form a DTWT pair, using the orthogonal Haar sequences as the basis for the wavelet expansion.*

6.14 THE MULTIRESOLUTION INTERPRETATION

The goal of this section is to highlight, without resorting to mathematical details, an important aspect of the wavelet transform that accounts for its success as a tool in pattern recognition as well as in numerous other applications. Let us assume for simplicity that the two filters in the analysis–synthesis bank of a paraunitary filter bank are ideal low/high pass. Figure 6.20 shows the magnitude responses of the respective filters in the equivalent of the tree-structured octave band filter bank of Figure 6.19. The width of the frequency response (bandwidth) is halved for each level of the tree (Figure 6.20d). That is, the "detail" resolution (high-pass) filters have a wide bandwidth, and the "coarse" resolution (low-pass) filters are of narrow bandwidth. Filters F_3 and F_2, the two coarser resolution ones, are of the same bandwidth. These observations are true for any octave band filter bank of any

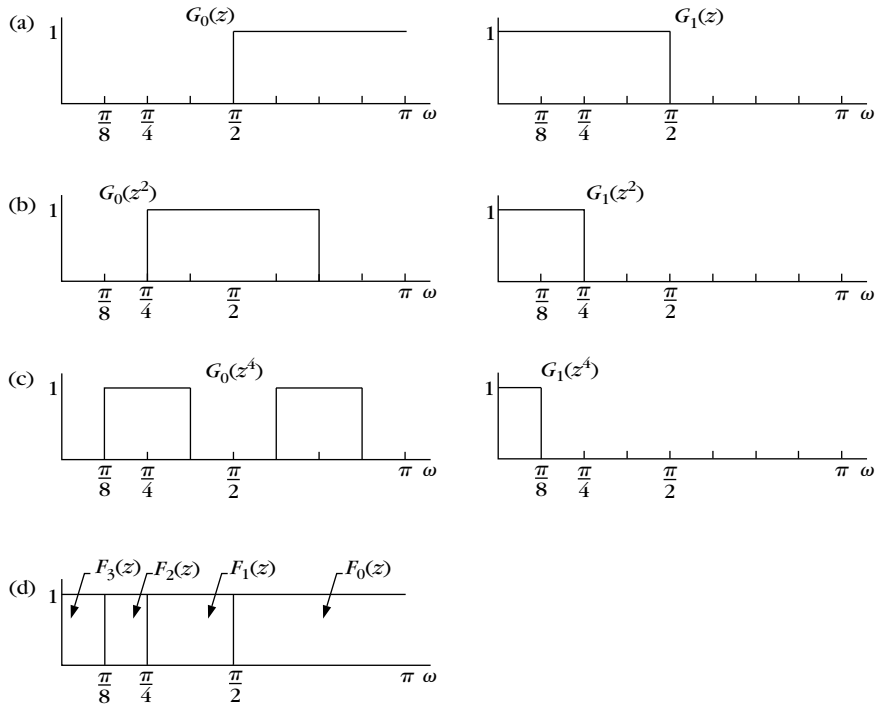


FIGURE 6.20

Filter bandwidths in an octave band filter bank.

number of levels J. That is, the width of $F_i(z)$ is half of the width of $F_{i-1}(z)$, and the widths of F_{J-2} and F_{J-1} are equal. This multiresolution viewpoint of the DTWT is a source of its power as a tool, and it is worth spending some time on it.

It is known (uncertainty principle) that filters with narrow bandwidth have long impulse responses and that filters with wide bandwidth have short impulse responses. Let us take for example the Haar filter bank. The filter impulse response at level zero, $h_0(n) = \hat{f}_0(n)$, is $(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$ and at level three [i.e., corresponding to Eq. (6.118)], $\hat{f}_3(n)$, is $\frac{1}{\sqrt{8}}(1, 1, 1, \underline{1}, 1, 1, \underline{1}, 1)$. Since the output of each filter of the analysis bank, that is, the DTWT coefficients, is the convolution of the input sequence with the respective impulse response, the filter tends to spread out the input activity. For example, a single impulse in the input of the $\hat{F}_3(z)$ filter produces a sequence of eight samples at the output. Thus, if our goal is to identify sudden (short in time) changes in an input signal, it is apparent that one should use filters of short impulse response to be able to obtain good time locality. Otherwise, the sudden activity will spread in time. Hence, for sudden changes in time (rich in high-frequency components), one needs a "detailed" analysis filter, that is, a short impulse response. For slowly time-varying activities, rich in low frequencies, one needs a "less detailed"

analysis filter, so as to be able to "see" the longer scale variations. Thus, detail is not of interest here, and long impulse responses are required. In other words, *the resolution should match the scale of the activity under investigation*.

The wavelet transform provides the means of analyzing the input signal into a number of different resolution levels in a hierarchical fashion. This is also known as *multiresolution analysis*. Thus, signal components corresponding to different physical activities can be best represented at different resolution levels: short high-frequency activities at the finer resolution and long low-frequency activities at coarser resolution levels. It turns out that this coarse-to-fine analysis strategy is appropriate for a number of pattern recognition tasks.

On the synthesis part, the signal can be reconstructed from its multiresolution components. See, for example, Figure 6.19. The sequence $x(n)$ is synthesized first by its coarser component $x_3(n)$, and then higher frequency (detailed) components are added, resulting in a *successively finer approximation*. When the component of the finest detail, $x_0(n)$, is added, the original signal is obtained. This philosophy is at the heart of a number of signal compression schemes.

Remarks

- The analysis of a signal in a number of components via a filter bank is not new and goes back to the work of Gabor in the 1940s. It is directly related to the short-time Fourier transform defined as [Gabo 46, Vett 95]

$$X_s(\omega, m) = \sum_{n=-\infty}^{\infty} x(n)\theta(n-m)\exp(-j\omega n) \tag{6.128}$$

  where $\theta(n)$ is a window sequence, whose center is successively moved to the different points m. Thus, each time, the part of the sequence $x(n)$ around m (depending on the window's effective width) is selected and Fourier transformed. It can be shown that this is equivalent to filtering the signal $x(n)$ by a bank of filters, each centered at a different frequency but all of them having the same bandwidth (Problem 6.20). This is its drawback, because low- and high-frequency signal components are "looked" at through the same window in time, resulting in poor overall localization of the events. What is really needed is a long window to analyze slowly time-varying low-frequency components and a narrow window to detect high-frequency short-time activities. As we saw, this is offered by a tree-structured octave-band filter bank, associated with the DTWT.

- All we have said about wavelet transforms and multiresolution analysis is just a glimpse of the whole story, a story that is really worth further effort; see, for example, [Daub 90].

6.15 WAVELET PACKETS

The DTWT has been introduced via an octave-band filter bank, and the wavelet coefficients result at the outputs of the bank, when its input is fed with the signal of interest. The octave-band filter bank is constructed by successively splitting by two the lowest frequency band (leaf) of the tree-structured bank (Figure 6.13). However, in many cases most of the activity is not in the low-frequency band but in the middle or high-frequency parts of the spectrum. In such cases, it may be useful to be able to allocate finer frequency bandwidths in the bands where the activity occurs. As we will see later on in the chapter, this can boost the discriminatory power of our system from a classification point of view, which is always our main interest. Figure 6.21a shows an example of a tree-structured filter bank but with the finer frequency splitting occurring at a midfrequency band. Figure 6.21b shows the resulting bandwidths for each of the (ideal) filters in the bank (f-axis) and the respective window length of the impulse responses in the time domain (n-axis). In other words, filters 2 and 3 have half the bandwidth and twice the impulse response of 4. Furthermore, they have one fourth of the bandwidth and a four times longer impulse response than that of 1. For comparison, Figure 6.22 shows

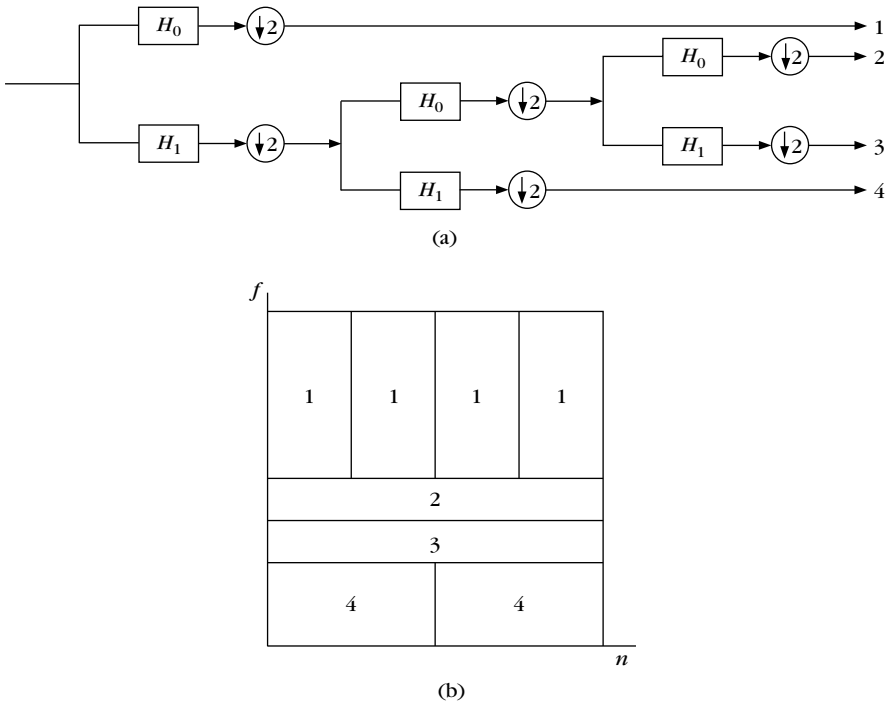


FIGURE 6.21

(a) Wavelet packet tree structure and (b) the corresponding frequency versus time resolution.

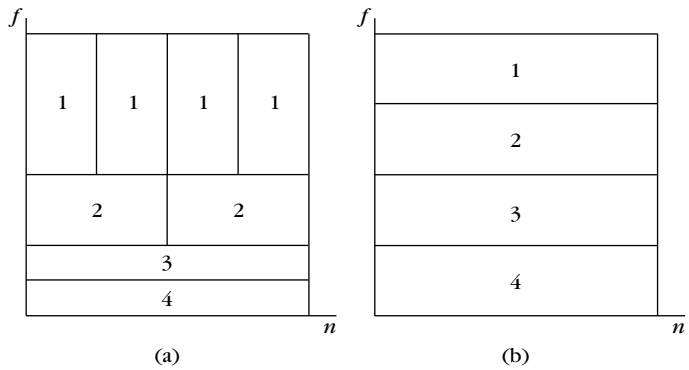


FIGURE 6.22

Frequency versus time resolution for (a) octave band and (b) equal bandwidth filter banks.

the frequency–time resolution plots for an octave-band filter bank (a) and for a bank with equal bandwidths (b), associated with the DTWT and the short-time Fourier transform, respectively. Having freed ourselves from the octave-band tree structure, filter banks can be constructed by various tree growth strategies, with that of Figure 6.21 being just one possibility. As was the case with the octave-band philosophy, these arbitrary tree structures also lead to a set of basis sequences for discrete signal expansions [Coif 92] called *wavelet packets*. Following arguments similar to those in Section 6.13 and using filters with the perfect reconstruction property, the basis sequences for the wavelet packets result from the respective impulse responses of the synthesis bank, after the appropriate, for each level, power of two time shifts.

6.16 A LOOK AT TWO-DIMENSIONAL GENERALIZATIONS

All the concepts discussed so far can be carried over in the two-dimensional case. No doubt, the task is even more challenging now. How can one define subsampling here? The straightforward way is via the "separable" philosophy. That is, we first transform (filter) the columns of the two-dimensional sequence and then the resulting rows. This leads to the subsampling shown in Figure 6.23. In other words, we leave out every other row and every other column (for subsampling by 2). Figure 6.24 shows the filter bank structure that complies with this philosophy. The image sequence $I(m, n)$ appears in the filters of stage 1 column after column, and the respective outputs are subsampled by 2. The resulting subsampled images are in turn filtered at stage two, but now they are fed into the filters row after row.

Assuming H_0 to be the (ideal) high-pass and H_1 the low-pass filter, the four frequency bands that are formed by the previous procedure are illustrated in Figure 6.25a. The area H_1H_1 corresponds to low-pass columns and rows, H_1H_0 to low-pass columns and high-pass rows, and so forth. Figure 6.25b shows the

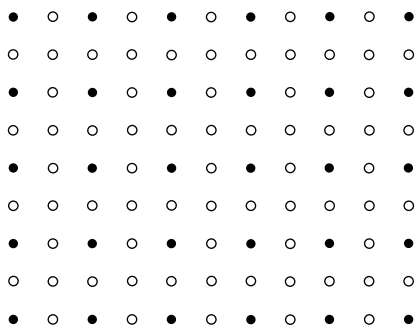


FIGURE 6.23

Separable subsampling by 2 for images.

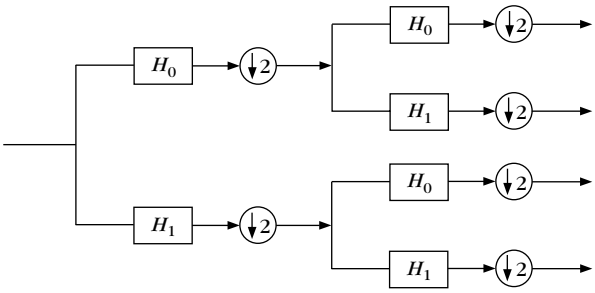


FIGURE 6.24

Basic element for a two-dimensional filter bank, leading to separable subsampling by 2.

resulting segmentation of the frequency domain when the low-pass area H_1H_1 is successively split by repeating the procedure.

Example 6.9

Figure 6.26 shows a 64×64 image of a triangle. The three "line" images are the 32×32 images that result when passing the triangle image through the structure of Figure 6.24. In the columnwise filtering of the first stage, the vertical line goes through the low-pass H_1 filter (no variation across it) and the horizontal and diagonal lines go through the high-pass H_0. This is because in a columnwise filtering, these appear as impulses in each column, and thus are rich in high frequencies. In the row scanning of the second stage, it will be the horizontal line that will go through the low-pass filter. Similar reasoning explains the position of the various parts of the triangle in the different bands.

Although this is obviously a very simplified example, it is quite instructive. It demonstrates how the original image can be obtained from its multiresolution components and also how different characteristics (directional in this case) of the whole may be isolated at different bands.

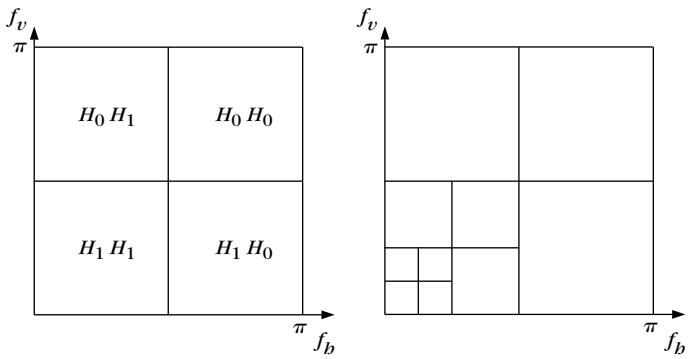


FIGURE 6.25

(a) Frequency domain division corresponding to the filter bank of Figure 6.24 and (b) the result of a successive division of the low-pass part of the spectrum.

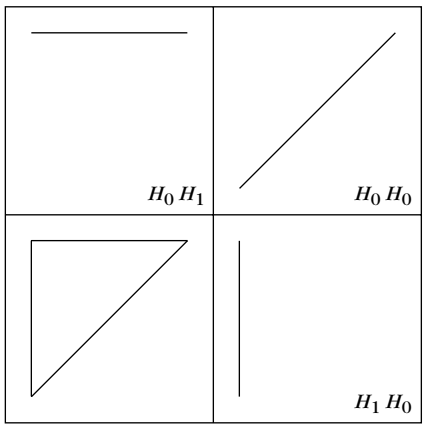


FIGURE 6.26

A triangle image and its filtered versions through the filter bank of Figure 6.24.

6.17 APPLICATIONS

All the transforms we have studied in this chapter are good candidates for feature generation, and they have been used extensively in various pattern recognition tasks. However, the wavelet transform offers an extra advantage, which in some cases can be beneficially exploited. *Its multiresolution properties conform to the way perception is achieved by humans, through their hearing and visual systems*. The human ear exhibits decreasing resolution at higher frequencies, in a

way that is uniform on a logarithmic scale (octave bands) [Flan 72]. Experiments in psychophysics and physiology show that the human visual cortex perceives by decomposing the stimuli in a number of frequency bands [Camp 68, Levi 85], which are also dependent on the spatial orientation [Camp 66]. Experimental results in [Nach 75] indicate that these frequency bands have the approximate bandwidth of an octave. The similarities between the mechanism with which human perception systems treat the respective stimuli and the processing techniques that split the signal into various (spatial) frequency bands, in a way similar to the wavelet transform, justify the use of the latter in pattern recognition tasks [Mall 89].

The following examples come from two of the most important areas of interest in today's pattern recognition applications.

Recognition of Handwritten Characters

The development of OCR systems is of particular importance in various application areas. One of the most challenging among them is the recognition of handwritten characters. Figure 6.27 shows the character "3" as well as its boundary contour after the application of a contour tracing algorithm [Pita 92]. The task now becomes one of shape recognition. As discussed in more detail in Chapter 7, the boundary can

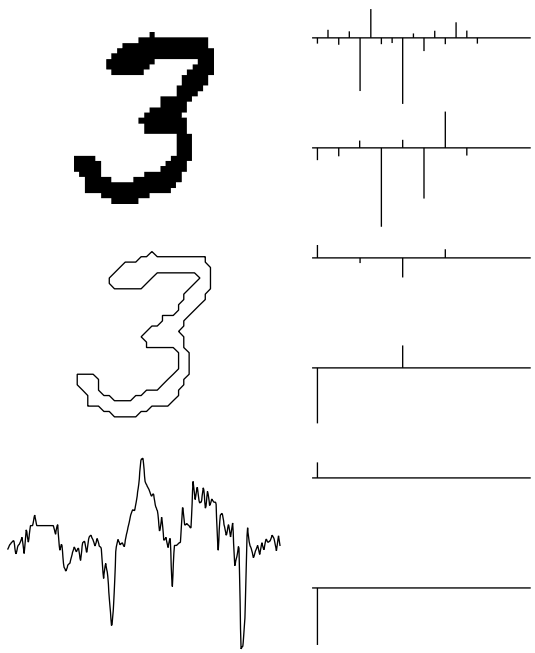


FIGURE 6.27

Wavelet coefficients corresponding to the curvature of the boundary of number "3."

be represented as a closed parametric curve in the complex plane

$$u(n) = x(n) + jy(n), \quad 0 \leq n \leq N-1 \tag{6.129}$$

with N being the number of samples (pixels) found tracing the contour and $x(n), y(n)$ the corresponding coordinates. The first point $(x(0), y(0))$ of the sequence is considered as the origin. Fourier methods have been used extensively in such classification tasks, by obtaining the DFT of $u(n)$ and keeping a sufficient number, from the total of N, of Fourier components as features. An alternative way is to extract the features from the wavelet domain. In other words, $x(n)$ and $y(n)$ are independently filtered through a tree-structured filter bank of appropriate resolution depth, and the resulting wavelet coefficients are used as features. The low-frequency components account for the basic shape of the character and are less sensitive to varying writing styles. The high-frequency components account for the details and are more sensitive to the specific handwriting style. In [Wuns 95], a comparative study was carried out using the same number of DTWT coefficients and Fourier-based features, with a neural network classifier. The experiments showed that classification based on wavelet coefficients resulted in reduced error rates. Furthermore, it was pointed out that Fourier-based features exhibited larger within-class variance and weaker between-class separation than the wavelet-based ones. A major disadvantage associated with the wavelet coefficients is that *they are not shift invariant*. That is, if we rotate/translate a character, the resulting coefficients will not be the same. This is a consequence of the subsampling process [Mall 89] (Problem 6.21). In other words, if

$$x'(n) = x(n - n_0)$$

and $y'(n)$, $y(n)$ are the sequences of wavelet coefficients of $x'(n)$ and $x(n)$, respectively, then, in general

$$y'(n) \neq y(n - n_0)$$

This has led to research in designing filter banks that seek to overcome this property [Marc 95, Hui 96]. The shift dependence obviously also makes the wavelet coefficients sensitive to the choice of the initial point from which the contour is traced, as already described. As we will see in more detail in Chapter 7, the Fourier coefficients are also dependent on shifts *but* in a deterministic way. Thus, various normalizing techniques exist that result in shift-invariant feature parameters [Crim 82, Arbt 90]. To overcome the problems associated with the choice of the initial point within the contour, in order to minimize its effects on the wavelet coefficients, a number of techniques have been suggested and used in practice. A simple method is to select a specific point resulting during the scanning process, for example, the first pixel when scanning the character from left to right. Other more "intelligent" ways have also been suggested [Wuns 95, Chua 96, Pun 03].

An alternative approach to that of Eq. (6.129) is to describe a contour in terms of the arc length between a given point and the origin, within the contour. As arc

length t at a given point, we define the number of consecutive pixels between the given point and the point considered as the origin. The contour description can now be achieved via a one-parameter real-valued function, the arc tangent angle $\theta(t)$ or the corresponding curvature $\kappa(t)$, defined as

$$\theta(t) = \tan^{-1}\left[\frac{dy(t)}{dx(t)}\right]$$

$$\kappa(t) = \frac{d\theta(t)}{dt}$$

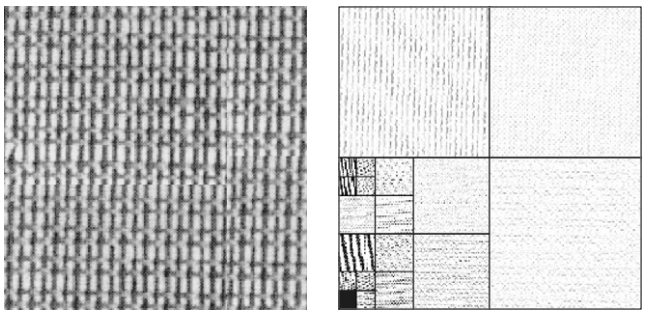
where $x(t), y(t)$ are the coordinates of the respective point as a function of length t from the origin and $dt = \sqrt{dy^2 + dx^2}$. A further discussion of this is provided in Chapter 7.

[Kapo 96] suggests wavelet transforming the curvature of the contour and using the corresponding wavelet coefficients as features. Figure 6.27 shows the coefficients resulting from the wavelet analysis of the curvature (bottom left) of the boundary contour of number 3. The wavelet basis used for the analysis was Daubechies's biorthogonal pairs (3,9) [Vett 95]. Six successive resolution levels are shown, the finest being on top and the coarsest at the bottom. Extensive experimentation with a number of different characters shows that the use of wavelet coefficients from more than six resolution levels adds no further discriminatory information to the system. Thus, each of the resulting feature vectors has 32 components. A different philosophy is followed in [Geze 00, Geze 02], in the context of an OCR system for the Greek Orthodox Byzantine music notation. The wavelet transform is applied to the vector combining the four projections (horizontal, vertical, left-diagonal, right-diagonal) of the characters. It turns out that such an approach leads to an efficient coding of the directional properties of the characters.

Texture Classification

Texture characterization in image analysis tasks is another area where the wavelet transform, as well as the other transforms discussed in this chapter, has been heavily utilized. The basic approach is similar to that in the OCR example. However, because texture is a property of the image region and not of its boundary, the two-dimensional variants of the transforms are used. The two-dimensional wavelet transform offers the advantages of spatial frequency and orientation selectivity, provided the appropriate bases are chosen. The information compaction properties result from the fact that most of the energy activity is concentrated in certain resolution levels, as was the case with the OCR example considered earlier. The wavelet coefficients of these levels are then selected as features to form the feature vectors. Sometimes a function of the features is employed, such as the energy, $\sum_i y_i^2$, or the entropy, $\sum_i y_i^2 \log y_i^2$, where y_i are the respective wavelet coefficients at each resolution level [Lain 93].

In many cases, the underlying image texture exhibits a great deal of activity in middle or high-frequency bands.



(a) (b)

FIGURE 6.28

An example of (a) a textured image and (b) its corresponding wavelet packet transform.

Figure 6.28a is an example of a textured image taken from [Brod 66]. In such cases, one must pay attention to the bands of high energy, instead of looking with fine frequency bandwidths at low-frequency bands of low energy. This leads to the adoption of wavelet packets, discussed in Section 6.15. There, we saw that there are a number of choices for splitting the frequency bands, leading to different wavelet packets. In [Chan 93] a dynamic procedure is suggested, depending on the particular texture image. A threshold C is selected prior to the analysis. If the output energy in a band is less than C, no splitting of the band is carried out. If it is higher than C, the band is split further. Splitting terminates if subimages, after filtering and subsampling, become small, for example, 16×16. It is then suggested to use as features the energy values at the J (a preselected number) most dominant (energywise) bands. In [Mojs 00] the effects of the properties of the analysis band on texture characterization are considered. It is demonstrated that the choice of the analysis filters may have a significant influence on the classification performance. In [Unse 95, Lain 96] another variation of the DTWT is adopted, called the *discrete wavelet frame*. The difference from DTWT is that the filter outputs in the bank are not subsampled. Although this leads to a redundant representation, it results in a texture description tolerant to translations.

A procedure similar in concept to the DTWT one is to employ a bank of two-dimensional Gabor filters to perform the splitting of the image into a number of frequency bands. The impulse response (point spread function, Appendix D) of the complex two-dimensional *Gabor filter* is given as the product of a Gaussian low-pass filter with a complex exponential, that is [Bovi 91],

$$h(x,y) = g'(x,y)\exp(j(\omega_x x + \omega_y y)) \tag{6.130}$$

where

$$g'(x,y) = \frac{1}{\lambda\sigma^2} g\left(\frac{x'}{\lambda\sigma}, \frac{y'}{\sigma}\right), g(x,y) = \frac{1}{2\pi}\exp\left(-\frac{x^2+y^2}{2}\right) \tag{6.131}$$

and

$$x' = x\cos\theta + y\sin\theta$$
$$y' = -x\sin\theta + y\cos\theta \tag{6.132}$$

That is, $g'(x, y)$ is a version of the Gaussian $g(x, y)$ that is spatially scaled and rotated by θ. The parameter σ is the spatial scaling, which controls the width of the filter impulse response, and λ defines the aspect ratio of the filter, which determines the directionality of the filter, which is no longer circularly symmetric. The orientation angle θ is usually chosen to be equal to the direction of the filter's center circular frequency

$$\omega = \sqrt{\omega_x^2 + \omega_y^2} \tag{6.133}$$

That is,

$$\theta = \tan^{-1}\frac{\omega_y}{\omega_x} \tag{6.134}$$

By varying the free parameters σ, λ, ω, and θ, filters of arbitrary orientation and bandwidth characteristics are obtained (Problem 6.22). Figure 6.29a shows the magnitude of the complex Gabor filter for $\sigma = 1.0, \lambda = 0.3$. Figure 6.29b shows the magnitude of the corresponding spatial frequency response. The choice of the Gabor filters is justified by the fact that these filters offer the optimal trade-off between spectral bandwidth and spatial localization. In Section 6.14 we saw that the shorter the filter's impulse response (spatial localization), the wider its frequency bandwidth and vice versa, according to the *uncertainty principle* [Papo 91],

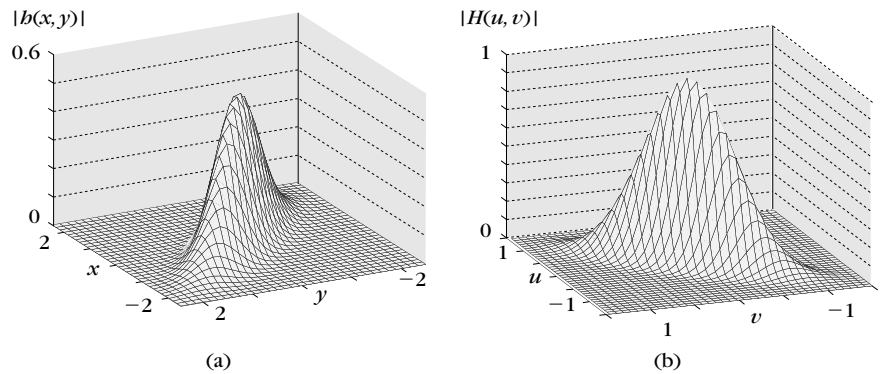


FIGURE 6.29

Plot of (a) the magnitude of point spread function of a Gabor filter and (b) the magnitude of its Fourier transform.

that is,

$$\Delta x \Delta \omega_x \geq \frac{1}{2}$$
$$\Delta y \Delta \omega_y \geq \frac{1}{2} \tag{6.135}$$

In [Daug 85] it has been shown that the two-dimensional Gabor filters attain the minimum uncertainty bound. For digital images, a sampling of the above functions has to be performed, which introduces aliasing errors regardless of the sampling interval. This happens because Gabor filters are not bandlimited, but have a Gaussian-shaped frequency response (Problem 6.23). The topic is treated in [Bovi 90]. Gabor filter banks for analyzing the image in a number of bands, in the context of texture classification, have been successfully employed in a number of cases, [Jain 91, Turn 86, Bovi 90, Hale 95, Hale 99, Weld 96]. A set of Gabor filters centered at different frequencies and having different orientations are used to cover the frequency range of interest, using various frequency and orientation bandwidths. Images are then filtered through this set of filters, and the features are generated from the resulting output samples. For example, the output energies of the Gabor filters may be chosen to be the respective features. Thus, using this strategy, the generated features encode classification information related to the spatial frequency as well as the orientation activity of the various textures. In order to grasp most of the textural information, techniques have been proposed to place the centers of the Gabor filters in the most "important" image frequencies [Pich 96]. In [Chan 93, Grig 02] a comparative study of various transform-based features is provided in the context of texture classification.

6.18 PROBLEMS

6.1 Show the equivalence (a) between (6.5) and (6.6) and (b) between (6.7) and (6.8).

6.2 Consider the separable transform $Y = UXV^T$. Then show that if Y, X are turned into the row-ordered vectors $\boldsymbol{y}, \boldsymbol{x}$, respectively, then $\boldsymbol{y} = (U \otimes V)\boldsymbol{x}$, where $\otimes$ denotes the Kronecker product of two matrices.

6.3 Let $\boldsymbol{e}_i, i = 0, 1, \ldots, N-1$, be any orthonormal basis in the N-dimensional space. Show that the MSE between an N-dimensional vector and an m-dimensional projection of it is minimized if (a) the basis consists of the eigenvectors of R_x and (b) the m-dimensional subspace is the one spanned by the eigenvectors corresponding to the m largest eigenvalues. Furthermore, the projection onto the latter subspace is the one that maximizes the sum of the variances of its components.
Hint: Minimize the mean square error $E[\|\boldsymbol{\epsilon}\|^2]$ subject to the constraint $\boldsymbol{e}_i^T \boldsymbol{e}_i = 1$.

6.4 Consider an N-dimensional random vector $\boldsymbol{x}$, which is approximated by

$$\hat{\boldsymbol{x}} = \sum_{i=0}^{m-1} y_i \boldsymbol{e}_i + \sum_{i=m}^{N-1} c_i \boldsymbol{e}_i$$

where c_i are nonrandom constants and $\boldsymbol{e}_i, i = 0, 1, 2, \ldots, N-1$, constitute an orthonormal basis. Show that the minimum mean square error $E\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2$ is achieved if (a) $c_i = E[y_i], i = m, \ldots, N-1$; (b) the orthonormal basis consists of the eigenvectors of Σ_x; and (c) $\boldsymbol{e}_i, i = m, \ldots, N-1$, correspond to the $N - m$ smallest eigenvalues.

6.5 If X is a rank r matrix, show that the two square matrices XX^H and X^HX have the same nonzero eigenvalues.

6.6 (a) Show Eq. (6.39).
(b) Show that the expansion in the right-hand side of (6.41) is the projection of $\boldsymbol{x}_i$ on the subspace spanned by the first $\boldsymbol{k}$ columns of U.

6.7 Given the matrix

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 3 \end{bmatrix}$$

compute its SVD representation.

6.8 Show the orthogonality of the DFT matrix W and also identity (6.94).

6.9 Given the image array

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \\ 2 & 1 & 2 \end{bmatrix}$$

compute its two-dimensional DFT transform.

6.10 For one of the images available at the Web site of the book, write a program to compute its DFT transform. Use a routine to implement the fast Fourier transform and plot the magnitude of the resulting Fourier transform.

6.11 Show the orthogonality of the DCT transform.

6.12 Compute the DCT of the image array of Problem 6.9.

6.13 Develop a program to compute the DCT for one of the images available from the Web site of the book.

6.14 Show the orthogonality of the Hadamard transform.

6.15 Compute the Hadamard transform for a 2×2 submatrix of the matrix of Problem 6.9.

6.16 Show the orthogonality of the Haar transform.

6.17 Show the two Noble identities of Figures 6.12a and 6.18.

6.18 Show the equivalence between the tree structure of Figure 6.11 and that of Figure 6.12b.

6.19 Consider the perfect reconstruction two-band Haar bank. Show that

a. If we make the analysis filters causal by delaying each of them by one sample, then the reconstructed sequence $\hat{x}(n)$ is delayed by one sample, that is, $\hat{x}(n) = x(n-1)$. In the more general case, if both the analysis filters have to be delayed by L, then the output of the band is also delayed by L.

b. If this is repeated with the more general N-band case, show that the output is delayed as $\hat{x}(n) = x(n-(2^{N-1}-1)L)$ and a delay must be inserted in each band to safeguard perfect reconstruction. Figure 6.30 shows the three-band case. In the general case, the delay element in each band is $z^{-(2^{N-i-1}-1)L}$, $i = 0, 1, \ldots, N-1$.

6.20 Show that the short-time Fourier transform defined in (6.128) is equal to

$$X_s(\omega, m) = \exp(-j\omega m) \sum_{n=-\infty}^{\infty} x(n)\theta(n-m)\exp(j\omega(m-n))$$

Verify that this is equivalent to filtering the sequence $x(n)$ with different filters of the same bandwidth but centered at different frequencies.

6.21 Show that the process of filtering and then subsampling is equivalent to the action of a linear but time-varying system. In fact, it is a periodically time-varying linear system. The same is true for the process of upsampling followed by linear filtering.

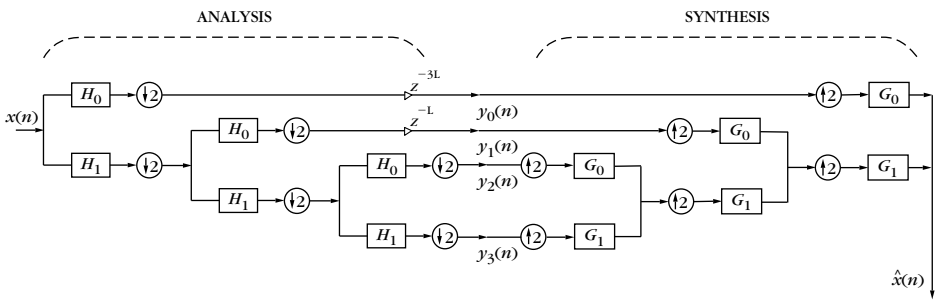


FIGURE 6.30

A three-band perfect reconstruction filter bank with causal analysis and synthesis filters.

6.22 Show that the frequency (octave) and orientation (radians) half-peak bandwidths for Gabor filters are given by B_f, B_θ, respectively, where

$$B_f = \log_2 \frac{\omega\lambda\sigma + \sqrt{2\ln 2}}{\omega\lambda\sigma - \sqrt{2\ln 2}}$$

$$B_\theta = 2\tan^{-1}\frac{\sqrt{2\ln 2}}{\omega\sigma}$$

Compute these for different values of λ, σ.

6.23 Show that the Fourier transform of the two-dimensional Gabor filter response $b(x, y)$ is given by

$$H(u, v) = \exp\left(-\frac{\sigma^2}{2}\{(u' - \omega_x')^2\lambda^2 + (v' - \omega_y')^2\}\right)$$

where

$$u' = u\cos\theta + v\sin\theta$$

$$v' = -u\sin\theta + v\cos\theta$$

and ω_x', ω_y' the corresponding versions of ω_x, ω_y rotated by θ. Draw its magnitude versus frequency (u, v) for different values of λ and θ.

MATLAB PROGRAMS AND EXERCISES

Computer Programs

6.1 *Generation of points around an* $(l-1)$*-dimensional hyperplane:* Write a MATLAB function named *generate_hyper* that generates randomly l-dimensional points $\boldsymbol{x}_i = [x_1(i), x_2(i), \ldots, x_l(i)]^T$ around an $(l-1)$-dimensional hyperplane H: $\boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$, where $\boldsymbol{w} = [w_1, w_2, \ldots, w_l]^T$. More specifically, the function takes as inputs: (a) the parameter (column) vector $\boldsymbol{w}$ for H ($w_l \neq 0$), (b) the offset w_0 for H, (c) a positive parameter a that defines the range $[-a, a]$, where each one of the first $(l-1)$ coordinates of the points is uniformly distributed, (d) the positive parameter e that defines the range $[-e, e]$ of a uniformly distributed noise source, which is added to the term $(-w_0 - \sum_{i=0}^{l-1} w_i x_i)/w_l$ to produce the lth coordinate, (e) the number N of points to be generated, and (f) the seed *sed* for the *rand* MATLAB function. It returns an $l \times N$ dimensional matrix, X, whose columns contain the generated data points. In addition, the function plots the data points for $l = 2, 3$.

Solution

```
function X=generate_hyper(w,w0,a,e,N,sed)
  l=length(w);
```

```
t=(rand(l-1,N)-.5)*2*a;
t_last=-(w(1:l-1)/w(l))'*t + 2*e*(rand(1,N)-.5)-(w0/w(l));
X=[t; t_last];
%Plots for the 2d and 3d case
if(l==2)
  figure(1), plot(X(1,:),X(2,:),'.b')
elseif(l==3)
  figure(1), plot3(X(1,:),X(2,:),X(3,:),'.b')
end
figure(1), axis equal
```

6.2 *PCA analysis:* Write MATLAB commands to compute the principal components of the covariance matrix of an $l \times N$ dimensional data matrix X as well as the corresponding variances.

Solution

Just write

```
[pc,variances]=pcacov(cov(X'))
```

In the above command (a) $cov(X')$ computes the covariance matrix of X' and (b) *pcacov* returns the principal components (eigenvectors of the covariance matrix) in the columns of *pc* as well as the corresponding variances (eigenvalues) in the column vector *variances* (note that *pcacov* assumes the the data vectors lie in the rows of corresponding data matrix).

6.3 *Distance matrix computation:* Write a MATLAB function named *compute_distances* that takes as input an $l \times N$ matrix X and returns the $N \times N$ dimensional matrix *distX* whose (i,j) entry contains the squared Euclidean distance between the ith and jth column vectors of X.

Solution

```
function distX=compute_distances(X)
  [l,N]=size(X);
  distX=zeros(N);
  for i=1:N
    for j=i+1:N
      distX(i,j)=(X(:,i)-X(:,j))'*(X(:,i)-X(:,j));
      distX(j,i)=distX(i,j);
    end
  end
```

6.4 *Singular Value Decomposition:* Write MATLAB commands to perform SVD on an $l \times N$ dimensional data matrix X whose columns are the data vectors.

Solution

Just write

```
[U,S,V]=svd(X)
```

The above command returns: (a) a diagonal matrix S of the same size with X, containing in its diagonal the singular values of X in decreasing order and (b) the unitary matrices U, V such that $U * S * V' = X$.

6.5 *Dimensionality reduction using SVD*: Write a MATLAB function named *SVD_eval* that evaluates the performance of the SVD method when applied on a data matrix X. More specifically, this function takes as inputs: (a) an $l \times N$ dimensional matrix X, whose columns contain the data vectors, (b) the dimensionality $k(<l)$ of the reduced space (k-dimensional hyperplane), h, generated by the k column vectors of the matrix U_r, which correspond to the k largest singular values of X. It returns: (a) an $l \times 1$ column vector containing the singular values of X, (b) the corresponding U_r matrix, denoted by Ur, (c) the $1 \times l$ dimensional parameter vector w of h, (d) the offset $w0$ of h, (e) the distance matrix *distX* for X, and (f) the distance matrix *distX_proj* of the projections of the vectors of X on h.

Solution

```
function [s,Ur,w,w0,distX,distX_proj]=SVD_eval(X,k)
  [l,N]=size(X);
  [Ur,S,Vr] = svd(X);
  s=diag(S(1:l,1:l));
  a=S(1:k,1:k)*Vr(:,1:k)';
  X_proj=Ur(:,1:k)*a;
  % Deterimnation of the estimated by the SVD hypeprlane
  P=X_proj(:,1:l)';
  w=[];
  for i=1:l
    w=[w (-1)^(i+1)*det([P(:,1:i-1) P(:,i+1:l) ones(l,1)])];
  end
  w0=(-1)^(l+2)*det(P(:,1:l));
  % Computation of distances
  distX=compute_distances(X);
  distX_proj=compute_distances(X_proj);
```

Computer Experiments

6.1 **a.** Generate an $l \times N$ dimensional matrix X ($l = 2$ and $N = 1000$), whose columns are two-dimensional points lying around the line h: $x_1 + x_2 = 0$

(i.e., $w = [1, 1]^T$ and $w_0 = 0$), using the *generate_hyper* function with parameters $a = 10, e = 1$ and $sed = 0$.

b. Compute the principal components of the covariance of X as well as the corresponding variances (eigenvalues). Compare the direction of the first principal component with the direction vector of h (which is perpendicular to w) and draw your conclusions.

6.2 Repeat 6.1 with $e = 5$.

6.3 Repeat 6.1 and 6.2 where now $l = 3$ and the line h is replaced by the plane H: $x_1 - 5x_2 + 2x_3 = 0$.

6.4 **a.** Generate an $l \times N$ dimensional matrix X ($l = 3$ and $N = 1000$), whose columns are two-dimensional points lying around the three dimensional hyperplane with $w = [1, 1, 1]^T, w_0 = 0$, using the *generate_hyper* function, with parameters $a = 10, e = 1$ and $sed = 0$.

b. Use the *SVD_eval* function to compute (a) the singular values of X,(b) $k = 2$ column vectors of the matrix U_r, that correspond to the two largest singular values of X. Also, compare the distances between the points of X and the distances of their corresponding projections on h.

c. Repeat (b) for $k = 1$.

6.5 Repeat 6.4 with $e = 6$. Comment on the results.

REFERENCES

[Achl 01] Achlioptas D., McSherry F. "Fast computation of low rank approximations," *Proceedings of the ACM STOC Conference*, pp. 611–618, 2001.

[Akan 93] Akansu A.N., Hadda R.A. *Multiresolution Signal Decomposition*, Academic Press, 1992.

[Arbt 90] Arbter K., Snyder W.E., Burkhardt H., Hirzinger G. "Application of affine-invariant Fourier descriptors to recognition of 3-D objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12(7), pp. 640–647, 1990.

[Atti 92] Attick J.J. "Entropy minimization: A design principle for sensory perception," *International Journal of Neural Systems*, Vol. 3, pp. 81–90, 1992.

[Barl 89] Barlow H.B. "Unsupervised learning," *Neural Computation*, Vol. 1, pp. 295–311, 1989.

[Bart 02] Bartlett M.S., Movellan J.R., Sejnowski T.J. "Face recognition by independent component analysis," *IEEE Transactions on Neural Networks*, Vol. 13(6), pp. 1450–1464, 2002.

[Bell 00] Bell A.J. "Information theory, independent component analysis, and applications," in *Unsupervised Adaptive Filtering, Part I: Blind Source Separation* (Haykin S., ed.), pp. 237–264, John Wiley & Sons, 2000.

[Bell 97] Bell A.J., Sejnowski T.J. "The independent components of natural scenes are edge filters," *Vision Research*, Vol. 37(23), pp. 3327–3338, 1997.

[Belk 03] Belikn M., Niyogi P. "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, Vol. 15(6), pp. 1373–1396, 2003.

[Beng 04] Bengio Y., Paiement J.-F., Vincent P., Delalleau O., Le Roux N., Quimet M. "Out of sample extensions for LLE, Isomap, MDS, Eigenmaps and Spectral clustering," *Advances in Neural Information Processing Systems Conference*, (Thrun S., Saul L., Schölkopf B., eds.), MIT Press, 2004.

[Benn 06] Benetos E., Kotti M., Kotropoulos C. "Applying supervised classifiers based on non-negative matrix factorization to musical instrument classification," *Proceedings IEEE Intl. Conference on Multimedia and Expo*, pp. 2105–2108, Toronto, Canada, 2006.

[Berr 95] Berry M., Dumais S., O'Brie G. "Using linear algebra for intelligent information retrieval," *SIAM Review*, Vol. 37, pp. 573–595, 1995.

[Beyg 06] Beygelzimer A., Kakade S., Langford J. "Cover trees for nearest neighbor," *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

[Bovi 91] Bovic A.C. "Analysis of multichannel narrow-band filters for image texture segmentation," *IEEE Transactions on Signal Processing*, Vol. 39(9), pp. 2025–2044, 1991.

[Bovi 90] Bovic A.C., Clark M., Geisler W.S. "Multichannel texture analysis using localized spatial filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12(1), pp. 55–73, 1990.

[Brod 66] Brodatz P. *Textures: A Photographic Album for Artists and Designers*, Dover, 1966.

[Brun 04] Brunet J.-P. Tamayo P., Golub T.R., Mesirov J.P. "Meta-genes and molecular pattern discovery using matrix factorization," *Proceedings of the National Academy of Science*, Vol. 101(2), pp. 4164–4169, 2004.

[Burg 04] Burges C.J.C. "Geometric methods for feature extraction and dimensional reduction: A guided tour," *Technical Report MSR-TR-2004-55*, Microsoft Research, 2004.

[Burt 83] Burt P.J., Adelson E.H. "The Laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, Vol. 31(4), pp. 532–540, 1983.

[Cai 05] Cai D., He X. "Orthogonal locally preserving indexing," *Proceedings 28th Annual International Conference on Research and Development in Information Retrieval*, 2005.

[Cama 03] Camastra F. "Data dimensionality estimation methods: A survey," *Pattern Recognition*, Vol. 36, pp. 2945–2954, 2003.

[Camp 66] Campell F., Kulikowski J. "Orientation selectivity of the human visual system," *Journal of Physiology*, Vol. 197, pp. 437–441, 1966.

[Camp 68] Campell F., Robson J. "Application of Fourier analysis to the visibility of gratings," *Journal of Physiology*, Vol. 197, pp. 551–566, 1968.

[Cao 03] Cao J.J., Chua K.S., Chong W.K., Lee H.P., Gu Q.M. " A comparison of PCA, KPCA and ICA for dimensionality reduction," *Neurocomputing*, Vol. 55, pp. 321–336, 2003.

[Cast 03] Casteli V., Thomasian A., Li C.-S. "CSVD: Clustering and singular value decomposition for approximate similarity searches in high-dimensional space," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15(3), pp. 671–685, 2003.

[Chan 93] Chang T., Kuo C.C.J. "Texture analysis and classification with tree structured wavelet transform," *IEEE Transactions on Image Processing*, Vol. 2(4), pp. 429–442, 1993.

[Chu 04] Chu M., Diele F., Plemmons R., Ragni S. "Optimality, computation and interpretation of the nonnegative matrix factorization," available at http://www.wfu.edu/~plemmons, 2004.

[Chua 96] Chuang G.C.H., Kuo C.C.J. "Wavelet descriptor of planar curves: Theory and applications," *IEEE Transactions on Image Processing*, Vol. 5(1), pp. 56–71, 1996.

[Coif 92] Coifman R.R., Meyer Y., Wickerhauser M.V. "Wavelet analysis and signal processing," in *Wavelets and Their Applications* (Ruskai M.B. *et al.*, eds.), pp. 153–178, Jones and Barlett, 1992.

[Como 94] Comon P. "Independent component analysis—A new concept?" *Signal Processing*, Vol. 36, pp. 287–314, 1994.

[Corm 01] Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. *Introduction to Algorithms*, Second Edition, MIT Press and McGraw-Hill, 2001.

[Cox 94] Cox T., Cox M. Multidimensional Scaling, Chapmay & Hall, London, 1994.

[Crim 82] Crimmins T.R. "A complete set of Fourier descriptors for two dimensional shapes," *IEEE Transactions on Systems, Man Cybernetics*, Vol. 12(6), pp. 848–855, 1982.

[Daub 90] Daubechies I. *Ten Lectures on Wavelets*, SIAM, Philadelphia, 1991.

[Daug 85] Daugman J.G. "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two dimensional visual cortical filters," *Journal of Optical Society of America*, Vol. 2, pp. 1160–1169, 1985.

[Deco 95] Deco G., Obradovic D. "Linear redundancy reduction learning," *Neural Networks*, Vol. 8(5), pp. 751–755, 1995.

[Deer 90] Deerwester S., Dumais S., Furnas G., Landauer T., Harshman R. "Indexing by latent semantic analysis," *Journal of the Society for Information Science*, Vol. 41, pp. 391–407, 1990.

[Desi 03] De Silva V., Tenenbaum J.B. "Global versus local methods in nonlinear dimensionality reduction," in *Advances in Neural Information Processing Systems* Becker S., Thrun S., Obermayer K. (eds.), Vol. 15, pp. 721–728, MIT Press, 2003.

[Diam 96] Diamantaras K.I., Kung S.Y. *Principal Component Neural Networks*, John Wiley Sons, 1996.

[Dono 02] Donoho D.L., Grimes C.E. "When does ISOMAP recover the natural parameterization of families of articulated images?" *Technical Report 2002-27*, Department of Statistics, Stanford University, 2002.

[Dono 04] Donoho D., Stodden V. "When does nonnegative matrix factorization give a correct decomposition into parts?" in *Advances in Neural Information Processing Systems* (Thrun S., Saul L., Schölkopf B., eds.), MIT Press, 2004.

[Doug 00] Douglas S.C., Amari S. "Natural gradient adaptation," in *Unsupervised Adaptive Filtering, Part I: Blind Source Separation* (Haykin S., ed.), pp. 13–61, John Wiley & Sons, 2000.

[Este 77] Esteban D., Galand C. "Application of quadrature mirror filters to split band voice coding schemes," *Proceedings of the IEEE Conference on Acoustics Speech and Signal Procesing*, pp. 191–195, May 1977.

[Fiel 94] Field D.J. "What is the goal of sensory coding?" *Neural Computation*, Vol. 6, pp. 559–601, 1994.

[Flan 72] Flanagan J.L. *Speech Analysis, Synthesis and Perception*, Springer-Verlag, New York, 1972.

[Fuku 90] Fukunaga K. *Introduction to Statistical Pattern Recognition*, 2nd ed., Academic Press, 1990.

[Gabo 46] Gabor D. "Theory of communications," *Journal of the Institute of Elec. Eng.*, Vol. 93, pp. 429–457, 1946.

[Geze 00] Gezerlis V., Theodoridis, S. "An optical music recognition system for the notation of Orthodox Hellenic Byzantine music," *Proceedings of the International Conference on Pattern Recognition (ICPR)*, Barcelona, 2000.

[Geze 02] Gezerlis V., Theodoridis S. "Optical character recognition of the Orthodox Hellenic Byzantine music," *Pattern Recognition*, Vol. 35(4), pp. 895–914, 2002.

[Golu 89] Golub G.H., Van Loan C.F. *Matrix Computations*, Johns Hopkins Press, 1989.

[Grig 02] Grigorescu S.E., Petkov N., Kruizinga P. "Comparison of texture features based on Gabor filters," *IEEE Transactions on Image Processing*, Vol. 11(10), pp. 1160–1167, 2002.

[Hale 95] Haley G., Manjunath B.S. "Rotation-invariant texture classification using modified Gabor filters," *IEEE International Conference on Image Processing*, pp. 262–265, 1995.

[Hale 99] Haley G., Manjunath B.S. "Rotation-invariant texture classification using complete space frequency model," *IEEE Transactions on Image Processing*, Vol. 8(2), pp. 255–269, 1999.

[Ham 04] Ham J., Lee D.D., Mika S., Schölkopf B. "A kernel view of the dimensionality reduction of manifolds," *Proceedings of the 21st International Conference on Machine Learning*, pp. 369–376, Banff, Canada, 2004.

[Hayk 99] Haykin S. *Neural Networks—A Comprehensive Foundation*, 2nd ed., Prentice Hall, 1999.

[Hayk 00] Haykin S. (ed.) *Unsupervised Adaptive Filtering, Part I: Blind Source Separation*, John Wiley & Sons, 2000.

[He 03] He X., Niyogi P. "Locally preserving projections," *Proceedings Advances in Neural Information Processing Systems Conference*, 2003.

[Hote 33] Hotelling H. "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, Vol. 24, pp. 417–441, 1933.

[Hoy 00] Hoyer P.O., Hyvärien A. "Independent component analysis applied to feature extraction from color and stereo images," *Network: Comput. Neural Systems*, Vol. 11(3), pp. 191–210, 2000.

[Hube 85] Huber P.J. "Projection pursuit," *The Annals of Statistics*, Vol. 13(2), pp. 435–475, 1985.

[Hui 96] Hui Y., Kok C.W., Nguyen T.Q. "Theory and design of shift invariant filter banks," *Proceeding of IEEE TFTS'96*, June 1996.

[Hyva 01] Hyvärien A., Karhunen J., Oja E. *Independent Component Analysis*, Wiley Interscience, 2001.

[Jack 91] Jackson J.E *A User's Guide to Principle Components*, John Wiley & Sons, 1991.

[Jain 89] Jain A.K. *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.

[Jain 91] Jain A.K., Farrokhnia F. "Unsupervised texture segmentation using Gabor filters," *Pattern Recognition*, Vol. 24(12), pp. 1167–1186, 1991.

[Jang 99] Jang G.J., Yun S.J., Hwan Y. "Feature vector transformation using independent component analysis and its application to speaker identification," *Proceedings of Eurospeech*, pp. 767–770, Hungary, 1999.

[Joll 86] Jollife I.T. *Principal Component Analysis*, Springer-Verlag, 1986.

[Jone 87] Jones M.C., Sibson R. "What is projection pursuit?" *Journal of the Royal Statistical Society, Ser. A*, Vol. 150, pp. 1–36, 1987.

[Jutt 91] Jutten C., Herault J. "Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic architecture," *Signal Processing*, Vol. 24, pp. 1–10, 1991.

[Kann 04] Kannan R., Vempala S., Vetta A. "On clustering: good, bad and spectral," *Journal of the ACM*, Vol. 51(3), pp. 497–515, 2004.

[Kapo 96] Kapogiannopoulos G., Papadakis M. "Character recognition using biorthogonal discrete wavelet transform," *Proceedings of the 41st Annual SPIE Meeting*, Vol. 2825, August 1996.

[Karh 46] Karhunen K. "Zur spektraltheorie stochastischer prozesse," *Annales Academiae Scientiarum Fennicae*, Vol. 37, 1946.

[Karh 94] Karhunen J., Joutsensalo J. "Representation and separation of signals using nonlinear PCA type learning," *Neural Networks*, Vol. 7(1), pp. 113–127, 1994.

[Koho 89] Kohonen T. *Self-Organization and Associative Memory*, 3rd ed., Springer-Verlag, 1989.

[Koki 07] Kokiopoulou E., Saad Y. "Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29(12), pp. 2143–2156, 2007.

[Kwon 04] Kwon O.W., Lee T.W. "Phoneme recognition using the ICA-based feature extraction and transformation," *Signal Processing*, Vol. 84(6), pp. 1005–1021, 2004.

[Lafo 06] Lafon S., Lee A.B. "Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning and data set parameterization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28(9), pp. 1393–1403, 2006.

[Lain 93] Laine A., Fan J. "Texture classification by wavelet packet signatures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15(11), pp. 1186–1191, 1993.

[Lain 96] Laine A., Fan J. "Frame representations for texture segmentation," *IEEE Transcaction on Image Processing*, Vol. 5(5), pp. 771–780, 1996.

[Law 06] Law M.H.C., Jain A.K. "Incremental nonlinear dimensionality reduction by manifold learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28(3), pp. 377–391, 2006.

[Lee 98] Lee T.-W. *Independent Component Analysis*, Kluwer Academic Publishers, 1998.

[Lee 01] Lee D.D., Seung S. "Learning the parts of objects by nonnegative matrix factorization," *Nature*, Vol. 401, pp. 788–791, 1999.

[Levi 85] Levine M.D. *Vision in Man and Machine*, McGraw-Hill, 1985.

[Lim 90] Lim J.S. *Two-Dimensional Signal Processing*, Prentice Hall, 1990.

[Lin 08] Lin T., Zha H. "Riemannian manifold learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30(5), pp. 796–810, 2008.

[Mall 89] Mallat S. "Multifrequency channel decompositions of images and wavelet models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37(12), pp. 2091–2110, 1989.

[Mall 97] Mallet Y., Coomans D., Kautsky J., De Vel O. "Classification using adaptive wavelets for feature extraction," *IEEE Transactions for Pattern Analysis and Machine Intelligence*, Vol. 19(10), pp. 1058–1067, 1997.

[Marc 95] Marco S.D., Heller P.N., Weiss J. "An M-band two dimensional translation-invariant wavelet transform and its applivations," *Proceedings of the IEEE Conference on Acoustics Speech and Signal Processing*, pp. 1077–1080, 1995.

[Meye 93] Meyer Y. *Wavelets, Algorithms and Applications*, SIAM, Philadelphia, 1993.

[Mojs 00] Mojsilovic A., Popovic M.V., Rackov D.M. "On the selection of an optimal wavelet basis for texture characterization," *IEEE Transactions Image Processing*, Vol. 9(12), 2000.

[Nach 75] Nachmais J., Weber A. "Discrimination of simple and complex gratings," *Vision Research*, Vol. 15, pp. 217–223, 1975.

[Oja 83] Oja E. *Subspace Methods for Pattern Recognition*, Res. Studies Press, Letchworth, U.K., 1983.

[Paat 91] Paatero P., Tapper U., aalto R., Kulmala M. "Matrix factorization methods for analysis diffusion battery data," *Journal of Aerosol Science*, Vol. 22 (Supplement 1), pp. 273–276, 1991.

[Paat 94] Paatero P., Tapper U. "Positive matrix factor model with optimal utilization of error," *Environmetrics*, Vol. 5, pp. 111–126, 1994.

[Papo 91] Papoulis A. *Probability, Random Variables, and Stochastic Processes*, 3rd ed., McGraw-Hill, 1991.

[Pich 96] Pichler O., Teuner A., Hosticka, B. "A comparison of texture feature extraction using adaptive Gabor filtering, pyramidal and tree structured wavelet transforms," *Pattern Recognition*, Vol. 29(5), pp. 733–742, 1996.

[Pita 92] Pitas I. *Digital Image Processing Algorithms*, Prentice Hall, 1992.

[Prak 97] Prakash M., Murty M.N. "Growing subspace pattern recognition methods and their neural network models," *IEEE Transactions on Neural Networks*, Vol. 8(1), pp. 161–168, 1997.

[Proa 92] Proakis J., Manolakis D. *Digital Signal Processing*, 2nd ed., Macmillan, 1992.

[Pun 03] Pun C.-M., Lee M.-C. "Log-Polar wavelet energy signatures for rotation and scale invariant texture classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25(5), pp. 590–603, 2003.

[Qui 07] Qui H., Hancock E.R. "Clustering and embedding using commute times," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29(11), pp. 1873–1890, 2007.

[Rowe 00] S.T. Roweis S.T., Saul L.K. "Nonlinear dimensionality reduction by locally linear embedding," *Science*, Vol. 290, pp. 2323–2326, 2000.

[Saul 01] Saul L.K., Roweis S.T. "An introduction to locally linear embedding," http://www.cs.toronto.edu/~ roweis/lle/papers/lleintro.pdf

[Scho 98] Schölkopf B., Smola A., Muller K.R. "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, Vol. 10, pp. 1299–1319, 1998.

[Sebr 03] Sebro N., Jaakola T. "Weighted low-rank approximations," *Proceedings of the ICML Conference*, pp. 720–727, 2003.

[Sha 05] Sha F., Saul L.K. "Analysis and extension of spectral methods for nonlinear dimensionality reduction," *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

[Shui 07] Shuicheng Y., Xu D., Zhang B., Zhang H.-J., Yang Q., Lin S. "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29(1), pp. 40–51, 2007.

[Shaw 04] Shawe-Taylor J., Cristianini N. *Kernel Methods for Pattern Analysis*, Cambridge University Press, Cambridge, MA, 2004.

[Smar 03] Smaragdis P., Brown J.C. "Nonnegative matrix factorization for polyphonic music transcription," *Proceedings IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003.

[Sra 06] Sra S., Dhillon I.S. "Non-negative matrix approximation: Algorithms and applications," Technical Report TR-06-27, University of Texas at Austin, 2006.

[Stef 93] Steffen P., Heller P.N., Gopinath R.A., Burrus C.S. "Theory of regular M-band wavelet bases," *IEEE Tansactions on Signal Processing*, Vol. 41(12), pp. 3497–3511, 1993.

[Stra 80] Strang G. *Linear Algebra and Its Applications*, 2nd ed., Harcourt Brace Jovanovich, 1980.

[Szu 92] Szu H.H., Telfer B.A., Katambe S. "Neural network adaptive wavelets for signal representation and classification," *Optical Eng*., Vol. 31, pp. 1907–1916, 1992.

[Szym 06] Szymkowiak-Have A., Girolami M.A., Larsen J. "Clustering via kernel decomposition," *IEEE Transactions on Neural Networks*, Vol. 17(1), pp. 256–264, 2006.

[Sun 06] Sun J., Boyd S., Xiao L., Diaconis P. "The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem," *SIAM Review*, Vol. 48(4), pp. 681–699, 2006.

[Tene 00] Tenenbaum J.B., De Silva V., Langford J.C. "A global geometric framework for dimensionality reduction," *Science*, Vol. 290, pp. 2319–2323, 2000.

[Trop 03] Tropp J.A. "Literature survey: Nonnegative matrix factorization," Unpublished note, http://www-personal.umich.edu/~jtropp/, 2003.

[Turn 86] Turner M.R. "Texture discrimination by Gabor functions," *Biol. Cybern*., Vol. 55, pp. 71–82, 1986.

[Unse 86] Unser M. "Local linear transforms for texture measurements," *Signal Processing*, Vol. 11(1), pp. 61–79, 1986.

[Unse 95] Unser M. "Texture classification and segmentation using wavelet frames," *IEEE Transactions on Image Processing*, Vol. 4(11), pp. 1549–1560, 1995.

[Unse 89] Unser M., Eden M. "Multiresolution feature extraction and selection for texture segmentation," *IEEE Transations on Pattern Analysis and Machine Intelligence*, Vol. 11(7), pp. 717–728, 1989.

[Vaid 93] Vaidyanathan P.P. *Multirate Systems and Filter Banks*, Prentice Hall, 1993.

[Vett 92] Vetterli M., Herley C. "Wavelets and filter banks: Theory and design," *IEEE Transactions on Signal Processing*, Vol. 40(9), pp. 2207–2232, 1992.

[Vett 95] Vetterli M., Kovacevic J. *Wavelets and Subband Coding*, Prentice Hall, 1995.

[Wata 73] Watanabe S., Pakvasa N. "Subspace method in pattern recognition," *Proceedings of the International Joint Conference on Pattern Recognition*, pp. 25–32, 1973.

[Weld 96] Weldon T., Higgins W., Dunn D. "Efficient Gabor filter design for texture segmentation," *Pattern Recognition*, Vol. 29(2), pp. 2005–2025, 1996.

[Wein 05] Weinberger K.Q., Saul L.K. "Unsupervised learning of image manifolds by semidefinite programming," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 988–995, Washington D.C., USA, 2004.

[Wuns 95] Wuncsh P., Laine A. "Wavelet descriptors for multiresolution recognition of handwritten characters," *Pattern Recognition*, Vol. 28(8), pp. 1237–1249, 1995.

[Xu 03] Xu W., Liu X., Gong Y. "Document clustering based on nonnegative matrix factorization," *Proceedings 26th Annual International ACM SIGIR Conference*, pp. 263–273, ACM Press, 2003.

[Ye 04] Ye J. "Generalized low rank approximation of matrices," *Proceedings of the 21st International Conference on Machine Learning*, pp. 887–894, Banff, Alberta, Canada, 2004.

[Zafe 06] Zafeiriou S., Tefas A., Buciu I., Pitas I. "Exploiting discriminant information in non-negative matrix factorization with application to frontal face verification," *IEEE Transactions on Neural Networks*, Vol. 17(3), pp. 683–695, 2006.