Matrix Methods for Geometric Data Analysis and Pattern Recognition

Jen-Mei Chang

Department of Mathematics and Statistics, California State University, Long Beach, 1250 Bellflower Blvd., Long Beach, California 90840-1001
E-mail address: jen-mei.chang@csulb.edu. Last updated: 12/30/2011

Contents

Preface

I wish to acknowledge Dr. Michael Kirby at Colorado State University for enlightening me with his wisdom and philosophy in the wonderful field of geometric data analysis. Without a proper training with him, these lecture notes would not have been possible.

Although majority of the figures have been either recreated or generated by the author from codes in MATLAB, you will find that the first three chapters of the notes are nearly taken verbatim from Kirby's book entitled, "*Geometric Data Analysis — An Empirical Approach to Dimensionality Reduction and the Study of Patterns.*" Pieces of the notes are presented in a way that is mirrored through my personal encounter of the materials in research. With the help of appropriate citations, readers should find it easy to locate the origin of the materials. For example, Chapter 4 is inspired by Mr. Justin Marks' Master Thesis titled "Discriminative Canonical Correlations: An Offspring of Fisher's Discriminant Analysis." Chapters 5, 6, and 7 follow rather closely with the book "Digital Image Processing" by Rafael Gonzalez and Richard Woods with a hint of "Discrete Wavelet Transformations" by Patrick Van Fleet in Chapter 6.

These notes are intended as the primary source of information for *MATH 521: Matrix Methods for Data Analysis and Pattern Recognition* as taught in California State University, Long Beach during the 2012-2013 academic year. Seeing how widespread the techniques of data analysis and pattern recognition have become, it was seemingly impossible to identify a single textbook that will serve the need for this course. Furthermore, while geometric perspective in learning patterns in real data sets is essential and almost inevitable, it is often overlooked in practice. I want to make a strong point in the importance of utilizing geometric tools for analyzing large data sets encountered in real life applications.

The notes are indefinitely incomplete and almost certainly contain errors. Any other use aside from classroom purposes and personal research, please contact me at jen-mei.chang@csulb.edu. Any suggestion, comment, and error should also be sent to me. I wish you find these notes helpful in your discovery of the wonderful realm of data analysis and pattern recognition.

CHAPTER 1

Vector Spaces and Linear Transformation

This chapter starts by reviewing some basic vector and matrix manipulations in Section 1 and then goes on to cover materials concerning vector spaces, linear transformations, and matrix algebra from linear algebra and numerical linear algebra. Representing data in terms of good coordinates requires a change of basis. The ideas behind such linear transformations are described in Section 2. Basic operations on important subspaces are visited in Section 3. The fundamental notion of a projection matrix is defined and developed in Section 4 along with a detail discussion on the one of the most important type of projection matrices, orthogonal projection matrices. To place this material in context, an application to discovering novelty in patterns is given. In Section 5, we will review the basic computations and facts concerning eigenvalues and eigenvectors that leads naturally to the discussion on the theory of singular value decomposition in Section 6, where numerous applications are presented. We will follow the discussions in [**31**] and [**17**] closely throughout this chapter.

1. Vectors, Matrices, Angles, and Distances

In this section, we review some of the most fundamental tools used in geometric data analysis.

1.1. Vectors and Matrices. Pretending we know the definition of a **vector**, then one interpretation of a **matrix** is simply a collection of vectors. Another definition of a matrix that is more commonly adapted treats a matrix as a rectangular array of data. For example,

$$X = \begin{bmatrix} \clubsuit & \spadesuit & \heartsuit \\ \spadesuit & \heartsuit & \diamondsuit \\ \heartsuit & \diamondsuit & \clubsuit \\ \diamondsuit & \clubsuit & \spadesuit \end{bmatrix}$$

is a 4-by-3 (4×3) matrix with four rows and three columns. In general, we label the entries in a matrix by its relative positive in the matrix as shown in Figure 1 The entry $a_{3,2}$ (or a_{32}) encodes the

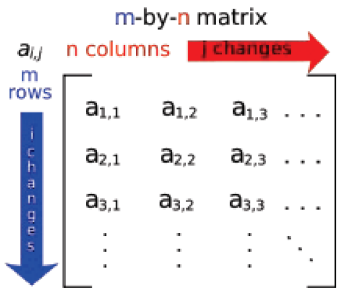
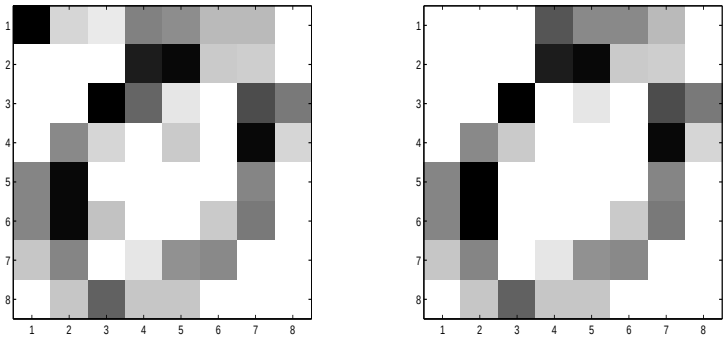


FIGURE 1. An illustration of a m-by-n matrix.

value in the 3^{rd} row and 2^{nd} column, i.e., row first, column second.
In order to develop mathematical properties of these collections of data, we often represent these data with numbers. For example, in digital photography a 8-bit black and white image of size m-by-n



(a) blurred zero (b) true zero

FIGURE 2. An illustration of matrix representation of black and white images.

is represented by a matrix of size $m \times n$ where entries in the matrix are integer numbers between 0 and 255 (8 bit $= 2^8 = 256$ possible intensity values, called grayscale.). The entries with value 0 are completely black while the entries with value 255 are completely white. Color images can be represented using various formats. For example, the RGB format stores images in three channels, which represent their intensities on the red (R), green (G), and blue scales (B), respectively. Mathematical manipulations of matrices give visual changes to images. For example, in a signal processing application (image de-blurring), we are given a "mess-up" image of the true image and asked to recover the original image. The matrix representation of the "blurred" image is given below and looks like the one given in Figure 2(a).

$$\tilde{Z} = \begin{bmatrix} 2 & 215 & 234 & 89 & 131 & 141 & 186 & 255 \\ 255 & 255 & 225 & 29 & 10 & 200 & 206 & 255 \\ 255 & 255 & 1 & 102 & 229 & 255 & 78 & 122 \\ 255 & 136 & 214 & 255 & 201 & 255 & 10 & 214 \\ 135 & 10 & 255 & 255 & 255 & 255 & 135 & 255 \\ 135 & 10 & 192 & 255 & 255 & 201 & 122 & 255 \\ 198 & 135 & 255 & 229 & 145 & 136 & 255 & 255 \\ 255 & 198 & 100 & 198 & 197 & 255 & 255 & 255 \end{bmatrix},$$

The matrix representation of the true image is given here and it looks like the image given in Figure 2(b).

$$Z = \begin{bmatrix} 255 & 255 & 255 & 87 & 136 & 136 & 186 & 255 \\ 255 & 255 & 225 & 29 & 10 & 200 & 206 & 255 \\ 255 & 255 & 1 & 102 & 229 & 255 & 78 & 122 \\ 255 & 136 & 214 & 255 & 255 & 255 & 10 & 214 \\ 135 & 1 & 255 & 255 & 255 & 255 & 135 & 255 \\ 135 & 1 & 255 & 255 & 255 & 201 & 122 & 255 \\ 198 & 135 & 255 & 229 & 145 & 136 & 255 & 255 \\ 255 & 198 & 100 & 198 & 197 & 255 & 255 & 255 \end{bmatrix}$$

The mathematics that goes into recovering the true images depends heavily on the properties of the image matrix and results from (numerical) linear algebra. This is why we are interested in the analysis of matrices.

A simple way to compare two vectors is to compare their norms. The most common **vector norms** are

$$\begin{aligned} ||x||_1 &= \sum_{i=1}^{n} |x_i| \quad \text{1-norm,} \\ ||x||_2 &= \sqrt{\sum_{i=1}^{n} x_i^2} \quad \text{2-norm or Euclidean norm,} \\ ||x||_\infty &= \max_{1\le i\le n} |x_i| \quad \text{max-norm.} \end{aligned}$$

The 2-norm is the generalization of the standard Euclidean distance in $\mathbb{R}^3$ to $\mathbb{R}^n$. All three norms defined here are special cases of the $p-$norm:

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}.$$

Associated with the Euclidean vector norm is the *inner product* between two vectors x and y in $\mathbb{R}^n$, which is defined

$$(x,y) = x^T y.$$

Generally, a *vector norm* is a mapping $\mathbb{R}^n \to \mathbb{R}$ with the properties

PROPERTY 1.1. For vectors $x, y \in \mathbb{R}^n$ and α a constant,

(1) $||x|| \ge 0$ for all x
(2) $||x|| = 0$ if and only if $x = 0$
(3) $||\alpha x|| = |\alpha| ||x||$, $\alpha \in \mathbb{R}$
(4) $||x+y|| \ge ||x|| + ||y||$, the triangle inequality

In data mining applications, it is common to use the *cosine of the angle* between two *real* vectors as a distance measure:

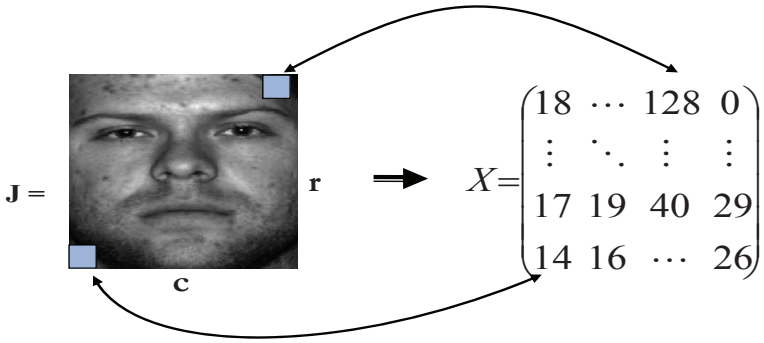
$$\cos\theta(x,y) = \frac{x^T y}{||x||_2 ||y||_2}.$$

With this measure two vectors are close if the cosine is close to one and x and y are *orthogonal* if $x^T y = 0$, i.e., angle between them is $\pi/2$. If x and y are two nonzero vectors in $\mathbb{C}^n$, then the angle between x and y is defined to be

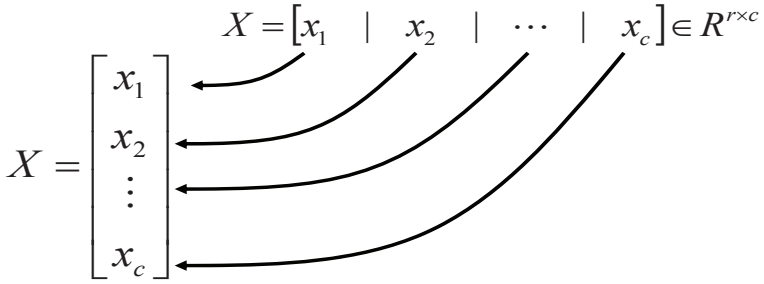
$$\angle(x,y) := \cos^{-1} \frac{|y^H x|}{||x||_2 ||y||_2}.$$

What if we want to compare a collection of data to another collection of data? This is considered as a many-to-many (set-to-set) classification paradigm. See [**1, 10**] for a detailed discussion on this classification structure.

1.2. Angles Between Subspaces. An r-by-c gray scale digital image corresponds to an r-by-c matrix where each entry enumerates one of the 256 possible gray levels of the corresponding pixel.



Now, realize X by its columns and concatenate columns into a single column vector:



Thus, an image J can be realized as a column vector of length equal to the product of J's resolutions. Now, for a subject i, we collect k distinct images (which corresponds to k column vectors) and stack them into a single data matrix so that

$$X^{(i)} = \left[\begin{array}{c|c|c|c} x_1^{(i)} & x_2^{(i)} & \cdots & x_k^{(i)} \end{array} \right]$$

with $rank(X^{(i)}) = k$. Then the column space of $X^{(i)}$ gives a k-dimensional subspace in $\mathbb{R}^n$.
For subspaces, we have a similar but recursive way to measure pairwise distances.

DEFINITION 1.1. **Principal Angles.** (Real case) If $\mathscr{X}$ and $\mathscr{Y}$ are two vector subspaces of $\mathbb{R}^n$ such that $\mathscr{R}(X) = \mathscr{X}$, $\mathscr{R}(Y) = \mathscr{Y}$, and $p = \dim(X) \geq \dim(Y) = q \geq 1$, then the **principal angles** $\theta_k \in [0, \frac{\pi}{2}], 1 \leq k \leq q$ between $\mathscr{X}$ and $\mathscr{Y}$ are defined recursively by

$$\cos(\theta_k) = \max_{u \in \mathscr{X}} \max_{v \in \mathscr{Y}} |u^T v| = |u_k^T v_k| \tag{1}$$

subject to $||u||_2 = ||v||_2 = 1$, $u^T u_i = 0$ and $v^T v_i = 0$ for $i = 1, 2, \ldots, k-1$. The vectors $(u_1, u_2, \ldots, u_q)$ and $(v_1, v_2, \ldots, v_q)$ are called the left and right principal vectors of the pair of subspaces, respectively.

To visualize this recursive definition, consider Figure 3. The first (minimum) principal angle between subspaces $\mathscr{X}$ and $\mathscr{Y}$ is found by finding a direction u in the span of $\mathscr{R}(X)$ and a direction v in the span of $\mathscr{R}(Y)$ whose angle is the smallest compared to angles between any other combination. These direction vectors that give rise to the minimum principal angle are called the first left and right *principal vectors*, respectively. Once the first angle is found, the second principal angle is computed in the orthogonal complements of the spaces spanned by u and v found previously. The smallest angle arises from all possible linear combinations of the vectors in $(\mathscr{X} - u)^{\perp}$ and $(\mathscr{Y} - v)^{\perp}$ is called the second principal angle. The process continues until it runs out of dimension to search.
A numerically stable algorithm that computes the principal angles between subspaces X and Y is given in the following Theorem [**6**]. This algorithm is accurate for large principal angles ($> 10^{-8}$).

THEOREM 1.1. *Assume that the columns of $Q_X \in \mathbb{R}^{n \times p}$ and $Q_Y \in \mathbb{R}^{n \times q}$ form orthonormal bases for two subspaces $\mathscr{X}$ and $\mathscr{Y}$ of $\mathbb{R}^n$ with $q \leq p$. Let $M = Q_X^T Q_Y$ and the SVD of this $p \times q$ matrix be*

$$M = UCV^T, \quad C = diag(\sigma_1, \ldots, \sigma_q),$$

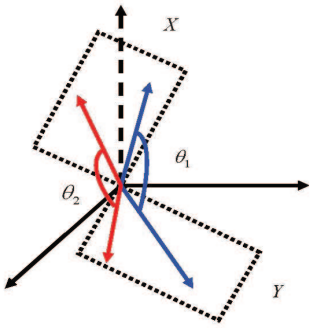


FIGURE 3. An illustration of principal vectors between two subspaces.

where $U^TU = V^TV = VV^T = I_q$. *If we assume that* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q$, *then the principal angles and the principal vectors associated with this pair of subspaces are given by*

$$\cos\theta_k = \sigma_k(M), \quad L = Q_X U, \quad R = Q_Y V. \tag{2}$$

PROOF. [**6**]. Since $M = UCV^T$, we have $U^TMV = C$. It is known that the singular values and singular vectors of the matrix M can be characterized by

$$\sigma_k(M) = \max_{||u||_2=1} \max_{||v||_2=1} u^TMv = u_k^TMv_k,$$

subject to $u^Tu_j = v^Tv_j = 0$ for $j = 1,\ldots,k-1$. If we let $l = Q_Xu \in \mathscr{R}(Q_X)$, $r = Q_Yv \in \mathscr{R}(Q_Y)$, then it follows that

$$||l||_2 = ||u||_2, \quad ||r||_2 = ||v||_2, \quad u^Tu_j = l^Tl_j, \quad v^Tv_j = r^Tf_j.$$

Now, since $u^TMv = u^TQ_XQ_Yv = (Q_Xu)^T(Q_Yv) = l^Tr$, so

$$\sigma_k(M) = \max_{||l||_2=1} \max_{||r||_2=1} l^Tr = l_k^Tr_k,$$

subject to $l^Tl_j = r^Tr_j = 0$ for $j = 1,\ldots,k-1$. Equation 2 follows directly from the definition of principal angles and vectors. □

An example is overdue at this point.

EXAMPLE 1.1. Let $\mathscr{X}$ be the *xy*-plane and $\mathscr{Y}$ the *yz*-plane in $\mathbb{R}^3$. Notice that $\mathscr{X} = \text{span}\{i,j\}$ and $\mathscr{Y} = \text{span}\{j,k\}$. Let

$$Q_X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad Q_Y = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

It is easy to see that $\mathscr{R}(Q_X) = \mathscr{X}, \mathscr{R}(Q_Y) = \mathscr{Y}$, and $Q_X^TQ_X = Q_Y^TQ_Y = I$. Now, let

$$M = Q_X^TQ_Y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}.$$

SVD of M gives

$$M = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Thus, $\cos\theta_1 = 1$ and $\cos\theta_2 = 0$, which then gives $\theta_1 = 0$ and $\theta_2 = \pi/2$. The ordered left principal vectors are the column vectors of

$$L = Q_X U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \\ 0 & 0 \end{bmatrix},$$

and the ordered right principal vectors are the column vectors of

$$R = Q_Y V = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ -1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Notice that the principal angles make sense, since two spaces share a dimension (the y-axis), which explains the minimal principal θ_1 being 0.

A sine-based algorithm for calculating small principal angles is available in [**35**]. This algorithm, presented in Algorithm 1, is often used to ensure precision of the minimum principal angles.

Algorithm 1 Small and Large Principal Angles [**35**]

This algorithm computes the principal angles between two subspaces given by the real matrices X and Y, where X is in $\mathbb{R}^{n\times p}$ and Y is in $\mathbb{R}^{n\times q}$. Principal angles are defined to be between 0 and $\pi/2$ and listed in ascending order.

Input: matrices X (n-by-p) and Y (n-by-q).

Output: principal angles θ between subspaces $\mathscr{R}(X) = \mathscr{X}$ and $\mathscr{R}(Y) = \mathscr{Y}$.

(1) Find orthonormal bases Q_x and Q_y for $\mathscr{X}$ and $\mathscr{Y}$ such that

$$Q_x^T Q_x = Q_y^T Q_y = I \quad \text{and} \quad \mathscr{R}(Q_x) = \mathscr{X}, \mathscr{R}(Q_y) = \mathscr{Y}.$$

(2) Compute SVD for cosine: $Q_x^T Q_y = H\Sigma Z^T$, where $\Sigma = diag(\sigma_1, \ldots, \sigma_q)$.

(3) Compute matrix

$$Y = \begin{cases} Q_y - Q_x(Q_x^T Q_y) & \text{if } rank(Q_x) \geq rank(Q_y); \\ Q_x - Q_y(Q_y^T Q_x) & \text{otherwise.} \end{cases}$$

(4) SVD for sine: $[H, diag(\mu_1, \ldots, \mu_q), Z] = \text{svd}(Y)$.

(5) Compute the principal angles, for $k = 1, \ldots, q$:

$$\theta_k = \begin{cases} \arccos(\sigma_k) & \text{if} \quad \sigma_k^2 < \frac{1}{2}; \\ \arcsin(\mu_k) & \text{if} \quad \mu_k^2 \leq \frac{1}{2}. \end{cases}$$

If you are a little rusty about the definitions of linear independence of vectors and rank of a matrix, please refer to Appendix A.

1.3. Distances Between Subspaces. Now that we have a way to compute angles between subspaces, we introduce a class of metrics that are used to calculate distances between them.
Recall that the (real) *Grassmann manifold* or *Grassmannian* (of k-planes in n-space) is the set $G(k,n)$ of k-dimensional vector subspaces of $\mathbb{R}^n$ (for fixed $k \leq n$). The (differential) topology on $G(k,n)$ can be described in several ways: First, as a quotient (homogeneous space) of the orthogonal group,

$$G(k,n) = O(n)/O(k) \times O(n-k). \tag{3}$$

Next, as a submanifold of projective space,

$$G(k,n) \subset \mathbb{P}(\Lambda^k \mathbb{R}^n) = \mathbb{P}^{\binom{n}{k}-1}(\mathbb{R}) \tag{4}$$

via the Plücker embedding. Finally, as a submanifold of Euclidean space,

$$G(k,n) \subset \mathbb{R}^{(n^2+n-2)/2} \tag{5}$$

via a projection embedding described recently in [**11**].

While the manifold structures on $G(q,m)$ obtained from these three constructions are equivalent (diffeomorphic), they naturally lead to different geometries on the Grassmannian. The standard invariant Riemannian metric on orthogonal matrices $O(n)$ descends via (3) to a Riemannian metric on the homogeneous space $G(k,n)$. The resulting *geodesic* distance function d_g (*arc length*) on the Grassmannian in terms of the principal angles $\theta_1,\ldots,\theta_q$ between $X, Y \in G(k,n)$, is (see, e.g., [**16**])

$$d_g(X,Y) = \left(\sum_{i=1}^{k} \theta_i^2\right)^{1/2} = \|\theta\|_2.$$

If one prefers the realization (4), then the Grassmannian inherits a Riemannian metric from the Fubini-Study metric on projective space (see, e.g., [**25**]), and the resulting *Fubini-Study* distance d_{FS} is given in terms of the principal angles by

$$d_{FS}(X,Y) = \cos^{-1}\left(\prod_{i=1}^{k} \cos\theta_i\right).$$

Finally, one can restrict the usual Euclidean distance function on $\mathbb{R}^{(n^2+n-2)/2}$ to the Grassmannian via (5) to obtain the *projection F* or *chordal* distance d_c (so called because the image of the Grassmannian under (5) lies in a sphere, so that the restricted distance is simply the distance along a straight-line chord connecting one point of that sphere to another; see [**11**]) which, in terms of the principal angles, has the expression

$$d_c(X,Y) = \left(\sum_{i=1}^{k} (\sin\theta_i)^2\right)^{1/2} = \|\sin\theta\|_2.$$

This projection F distance d_c has recently been used in the context of sphere-packing/coding theory in the Grassmannian, where it is significantly more efficient than the "standard" geodesic distance d_g [**11**], [**2**]. As a slight variation on the last formula, we may also consider the so-called *chordal Frobenius* distance d_{cF}, given in terms of the principal angles by

$$d_{cF}(X,Y) = \|2\sin\frac{1}{2}\theta\|_2.$$

See [**16**] for additional details.

Now, the set-to-set classification problem can be transformed to a problem on $G(k,n)$ if we realize the linear span of a set of k images as a k-dimensional vector subspace of the space of all possible images at a given resolution. Our objective is to match an unlabeled *set* of images by comparing its associated point with a collection of given points on $G(k,n)$. As a consequence of the encoding of sets of images as points on a Grassmann manifold we may avail ourselves of a variety of well-known distance measures between points on the manifold as discussed above.

Note that the standard distance between subspaces that is often presented in linear algebra is determined by the largest angle between the two subspaces. This ignores the geometric information associated with the smaller angles. We have observed that in many instances it is in fact the smallest (not largest) principal angle that carries the most significant information.

2. Linear Transformation and Matrices

The empirical scientific approach for the investigation of a physical system consists of the following basic steps:

- Data collection
- Model building
- Prediction

The passage from data collection to model building is a significant step, which involves extracting information from the data to permit a characterization of the process. The goal of this modeling phase is to provide knowledge that was not available in the raw data itself. This general approach permits us to learn and deepen our understanding of complicated phenomena. For example, the apparent order observed in financial markets and weather systems provides ample evidence that our ability to understand, manipulate, predict and control patterns is extremely important and potentially rewarding. The naturally question to ask then: how can massive quantities of data be distilled into a few basic facts, or laws, which serve to describe a process? It has been observed that there is a tendency in nature for physical systems to *self-organize*. In a very general sense, this tendency for self-organization is revealed by the formation of patterns, the essence of which is reflected by a coherence, or correlation, of measurable physical quantities. Furthermore, **the formation of patterns appears to reduce the dimension of the space required to characterize the system**. One premise of this course is that systems that exhibit self-organization may be investigated by exploiting the reduced dimension of resulting patterns. The primary tool for accomplishing this is an appropriate coordinate transformation, i.e., one that reveals the reduction in dimension associated with coherent structures, or patterns. Thus, this course is essentially about discovering useful transformations, i.e., transformations that help us reveal underlying processes that are hidden in large data sets. Very often data sets are large because their coordinate systems are too general. Good coordinate systems — or equivalently, good bases — may be obtained by incorporating some knowledge of the data. We will be primarily concerned with exactly how to transfer the knowledge from the data to the coordinate system. When carried out effectively, this program replaces massive data sets by manageable ones which retain the information essential to understanding the process or phenomenon.
With that, let us now review the necessary machinery that will help us in obtaining those dimensionality-reducing mappings. First, the notion of linearity and nonlinearity in mappings.

DEFINITION 2.1. A mapping (transformation) $f : U \to V$ is said to be *linear* if

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$$

for all $x, y \in U$ and $\alpha, \beta \in \mathbb{R}$. A mapping is said to be *nonlinear* if it is not linear.

The prototype linear mapping can be considered as a matrix multiplication, since if $A \in \mathbb{R}^{m\times n}$ is an m-by-n matrix, then the mapping $L_A : \mathbb{R}^n \to \mathbb{R}^m$ that is given by $L_A(u) = Au$ is a linear mapping that takes a vector u in $\mathbb{R}^n$ to $\mathbb{R}^m$. While it is not surprising that matrix multiplication is a linear mapping, it is notable that every linear transformation between finite-dimensional vector spaces may be represented as multiplication of a vector by an appropriate matrix. This representation is achieved by the introduction of a coordinate system, or basis, for the space. For example, the n column vectors

$$\begin{aligned} \mathbf{e}^{(1)} &= (1,0,\cdots,0)^T \\ \mathbf{e}^{(2)} &= (0,1,\cdots,0)^T \\ &\vdots \\ \mathbf{e}^{(n)} &= (0,0,\cdots,1)^T \end{aligned}$$

form a basis for $\mathbb{R}^n$ known as the *standard basis*.
Thus any $\mathbf{u} \in \mathbb{R}^n$ can be written as

$$\mathbf{u} = \alpha_1 \mathbf{e}^{(1)} + \alpha_2 \mathbf{e}^{(2)} + \cdots + \alpha_n \mathbf{e}^{(n)}.$$

The n-tuple $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ determines the *coordinates* of the point $\mathbf{u}$ with respect to the standard basis. We emphasize the dependence of the coordinates of $\mathbf{u}$ on the choice of basis. For example, given another basis $\mathscr{B}$ for $\mathbb{R}^n$ consisting of the vectors $\left\{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \cdots, \mathbf{v}^{(n)}, \right\}$, we may represent $\mathbf{u}$ as

$$\mathbf{u} = x_1\mathbf{v}^{(1)} + x_2\mathbf{v}^{(2)} + \cdots + x_n\mathbf{v}^{(n)}.$$

Now the n-tuple $(x_1, x_2, \ldots, x_n)$ determines the coordinates of the point $\mathbf{u}$ with respect to the new basis $\mathscr{B}$. A central issue in analyzing patterns in data is determining and utilizing the good basis for a given set of data. Indeed, a central theme of this course is the construction of empirical bases which are very effective for representing specific data sets. Motivated by this, we now develop the basic mechanics of changing coordinate systems.

Change of Basis.

To start, let $\left\{\mathbf{v}^{(i)}\right\}_{i=1}^{n}$ and $\left\{\mathbf{w}^{(i)}\right\}_{i=1}^{n}$ both be bases for $\mathbb{R}^n$, called $\mathscr{B}_1$ and $\mathscr{B}_2$, respectively. Let $\mathbf{u}$ be an arbitrary element of $\mathbb{R}^n$. Thus in terms of the basis $\mathscr{B}_1$ we write

$$\mathbf{u} = x_1\mathbf{v}^{(1)} + x_2\mathbf{v}^{(2)} + \cdots + x_n\mathbf{v}^{(n)},$$

and in terms of $\mathscr{B}_2$ we write

$$\mathbf{u} = y_1\mathbf{w}^{(1)} + y_2\mathbf{w}^{(2)} + \cdots + y_n\mathbf{w}^{(n)},$$

giving the representation, or coordinates,

$$\mathbf{u}_{\mathscr{B}_1} = (x_1 \cdots x_n)^T$$

with respect to $\mathscr{B}_1$, and

$$\mathbf{u}_{\mathscr{B}_2} = (y_1 \cdots y_n)^T$$

with respect to $\mathscr{B}_2$. Equivalently, we may write

$$\mathbf{u} = V\mathbf{u}_{\mathscr{B}_1} = W\mathbf{u}_{\mathscr{B}_2},$$

where V and W are the matrices of basis vectors for $\mathscr{B}_1$ and $\mathscr{B}_2$, respectively.

EXAMPLE 2.1. Given that the basis vectors defining $\mathscr{B}_1$ are

$$\mathbf{v}^{(1)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad \mathbf{v}^{(2)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

and that the basis vectors defining $\mathscr{B}_2$ are

$$\mathbf{w}^{(1)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{and} \quad \mathbf{w}^{(2)} = \begin{pmatrix} 1 \\ -1 \end{pmatrix},$$

find $\mathbf{u}_{\mathscr{B}_2}$ given

$$\mathbf{u}_{\mathscr{B}_1} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}.$$

Plugging into $V\mathbf{u}_{\mathscr{B}_1} = W\mathbf{u}_{\mathscr{B}_2}$, we have

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix} \mathbf{u}_{\mathscr{B}_2},$$

from which it follows that

$$\mathbf{u}_{\mathscr{B}_2} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}.$$

One of the most important types of matrices for our purposes is the orthogonal matrix.

DEFINITION 2.2. A real square matrix U is said to be *orthogonal* if

$$U^T U = I.$$

Note that the requirement that U be square means that an orthogonal matrix U is invertible with inverse

$$U^{-1} = U^T.$$

Thus, one of the obvious attractions of an orthogonal matrix is that its inverse is easy to compute. An orthonormal set of vectors may be used to construct an orthogonal matrix. An orthogonal matrix acts as a very useful change of basis in that it preserves Euclidean distances (norms). If U is an orthogonal matrix, we have

$$\begin{aligned} ||U\mathbf{x}||_2^2 &= (U\mathbf{x})^T U\mathbf{x} \\ &= \mathbf{x}^T U^T U\mathbf{x} \\ &= \mathbf{x}^T\mathbf{x} \\ &= ||\mathbf{x}||_2^2. \end{aligned}$$

Because the Euclidean norms are preserved, the mapping U is referred to as an *isometry*. Note that distances in the l_1-norm are not preserved. In addition, if the determinant of an orthogonal matrix is one, then we may view the transformation geometrically as a rigid rotation of the space.

Other Classes of Transformations and Homogeneous Coordinates.

In the 2-dimensional case, there are a few matrices we should be familiar with which are given here.

(1) Scaling matrices.

$$M_{sa} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}.$$

If $s_x = s_y$, then it's a uniform scaling. Otherwise, it is non-uniform or anisotropic.

(2) Shearing matrices. A shear parallel to the x-axis:

$$M_{sx} = \begin{bmatrix} 1 & \lambda \\ 0 & 1 \end{bmatrix}$$

and a shear parallel to the y-axis:

$$M_{sx} = \begin{bmatrix} 1 & 0 \\ \lambda & 1 \end{bmatrix}.$$

(3) Reflection matrices are computed based on the Householder transformation. Two common ones are reflection about the x-axis

$$M_{rx} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

and the reflection about the y-axis

$$M_{ry} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}.$$

In general, to reflect a vector about a line that *goes through the origin*, let $< l_x, l_y >$ be a vector in the direction of the line, then

$$M_{rl} = \frac{1}{l_x^2 + l_y^2} \begin{bmatrix} l_x^2 - l_y^2 & 2l_x l_y \\ 2l_x l_y & l_y^2 - l_x^2 \end{bmatrix}.$$

Note that this technique only works if the plan runs through the origin. If it does not, we need to use an *affine transformation*.

(4) Rotation matrix (about the origin).
$$M_{rt} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$
where θ is measured counterclockwise.
(5) Translation matrix does not have a nice matrix representation in the Euclidean coordinates. It can be given easily in the *homogeneous coordinates*.
(6) Projection matrices is the heart of our discussion in this chapter, we will, therefore, delay it until Section 4.

For example, to rotate a vector $p = [xy]^T$ in the xy-plane 45° counterclockwise about the origin, we perform the operation
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$
In practice (image processing applications), *homogenous coordinates* allow *affine transformations* to be easily represented by a matrix. Also they make calculations possible in projective space just as Cartesian coordinates do in Euclidean space. Therefore, we will describe these transformations using the homogeneous coordinates next. Before that, a few definitions are in order.

DEFINITION 2.3. A *similarity transformation* is a *conformal* (angle-preserving) mapping whose transformation matrix A can be written in the form
$$A = PBP^{-1},$$
where P is an invertible square matrix.

Examples of similarity transformation include scaling, translation, and rotation.

DEFINITION 2.4. In geometry, an *affine transformation* or *affine map* between two vector spaces consists of a linear transformation followed by a translation, i.e.,
$$x \mapsto Ax + b$$

Examples of affine transformation include similarity transformation and shearing.
Another important category of transformation is the *perspective projection (projective transformation)* that is of great importance in 3D computer graphics. Whereas parallel projections are used to project points onto the image plane along parallel lines, the perspective projection projects points onto the image plane along lines that emanate from a single point, called the center of projection. This means that an object has a smaller projection when it is far away from the center of projection and a larger projection when it is closer.

DEFINITION 2.5. The *projective space* generated from a particular vector space V is denoted
$$\mathbb{P}V = \{1 - \text{dimensional subspaces of } V\}.$$
The case when $V = \mathbb{R}^2$ or $V = \mathbb{R}^3$ are the *projective line* and the *projective plane*, respectively. Alternatively,
$$\mathbb{P}^n = \left\{1 - \text{dimensional subspaces of } K^{n+1}\right\}.$$

DEFINITION 2.6. The *homogeneous coordinates* of a projective point $p \in \mathbb{P}^2$ is $[x : y : z]$, where $v = (x, y, z)^T$ is any vector in the 1-dimensional subspace which defines p.

Note that the homogeneous coordinates are not unique: any vector on that line can be used to define the homogeneous coordinates.Thus, we have the following facts:

PROPOSITION 2.1. *Let* $p \in \mathbb{P}^2$.

(1) *Any three real numbers x, y, and z which are not all zero can be the homogeneous coordinates of a point of the projective plane.*
(2) $[x : y : z] = [x' : y' : z']$ *if and only if there is* $\lambda \neq 0$ *such that* $x' = \lambda x, y' = \lambda y$, *and* $z' = \lambda z$.

In general, the homogeneous coordinates of a point of projective space of dimension n are usually written as $[x : y : z : \cdots : w]$, a row vector of length $n+1$. Two sets of coordinates that are proportional denote the same point of projective space: for any non-zero scalar c from the underlying field K, $[cx : cy : cz : \cdots : cw]$ denotes the same point. Therefore this system of coordinates can be explained as follows: if the projective space is constructed from a vector space V of dimension $n+1$, introduce coordinates in V by choosing a basis, and use these in $\mathbb{P}(V)$, the equivalence classes of proportional non-zero vectors in V.
The idea of a projective space relates to perspective, more precisely to the way an eye or a camera projects a 3D scene to a 2D image. All points which lie on a projection line (i.e., a "line-of-sight"), intersecting with the focal point of the camera, are projected onto a common image point. In this case the vector space is $\mathbb{R}^3$ with the camera focal point at the origin and the projective space corresponds to the image points. See Figure 4 for an illustration.

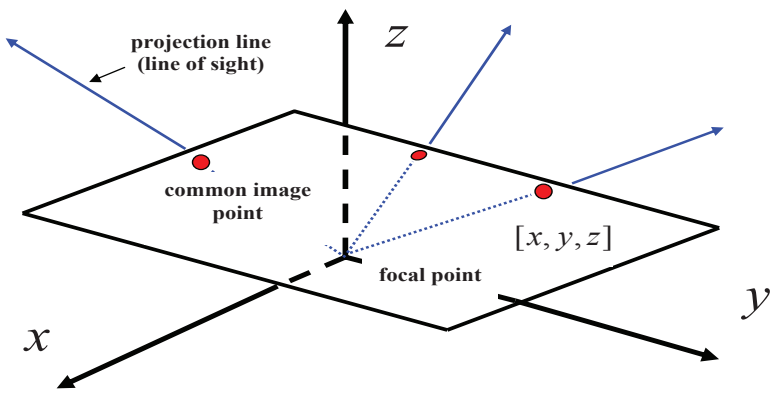


FIGURE 4. Illustration of $\mathbb{P}^2$ and perspective geometry.

Let $p = [x : y : z] \in \mathbb{P}^2$. Recall that this represents the line in $\mathbb{R}^3$ passing through the origin and the point (x, y, z). If this line is not in the xy-plane, that is, if $z \neq 0$, then using the scaling property of homogeneous coordinates we see that

$$p = [x : y : z] = \left[\frac{x}{z}, \frac{y}{z}, 1\right].$$

This gives a *normal form* for such points, in that it is a unique representation of the form $[a : b : 1]$ since

$$[a : b : 1] = [c : d : 1] \quad \text{if and only if} \quad a = b \,\text{and}\, c = d.$$

The points of $\mathbb{P}^2$ that this does not work for are exactly the points $[x : y : 0]$, where the homogeneous z coordinate is zero. In terms of the lines of $\mathbb{R}^3$, these are the horizontal lines in the xy-plane.
In this way we see that for many of the points of the projective plane $\mathbb{P}^2$, we have associated two real numbers. Geometrically, we can think of this as follows. let H be the plane in $\mathbb{R}^3$ defined by $z = 1$. Every line through the origin that is not parallel to this plane will meet H in a unique point. Conversely, every point of H may be joined to the origin by a unique line. In this way we have a correspondence between the points of the Euclidean plane H and the points of $\mathbb{P}^2$, except the $z = 0$ points.

Since the plane is naturally an $\mathbb{R}^2$, we have a natural pairing

$$\mathbb{R}^2 \hookrightarrow \mathbb{P}^2$$

which associates an ordered pair (a,b) in $\mathbb{R}^2$ to the point $[a:b:1]$ in $\mathbb{P}^2$. The inverse mapping sends a point $[x:y:z] \in \mathbb{P}^2$ to the ordered pair $(x/z, y/z) \in \mathbb{R}^2$. This inverse map is not defined at the points where $z = 0$. What about those points when $z = 0$? It is useful to "step down" one dimension to get a feeling for this concept. Suppose that instead of studying the space of lines through the origin in $\mathbb{R}^3$, we study the lines through the origin in $\mathbb{R}^2$. We all know that each of these lines may be described by a real number — its slope. If a line through the origin contains the point (x,y), then its slope is $m = y/x$. This associates a single real number to each line through the origin, *except for the vertical line*. Indeed, the slope $m = y/x$ can be viewed as the y-coordinate of the intersection of the line with the vertical line $x = 1$, in complete analogy with the situation described above. What should we associate to that one missing line, the vertical line? The slope construction clearly indicates that we should think of the vertical line as having "infinite" slope. Therefore a reasonable model for the set of lines through the origin in $\mathbb{R}^2$ is the set of real numbers (including the slope of the line) *plus one extra infinite point:*

$$\text{lines though 0 in } \mathbb{R}^2 = \mathbb{R} \cup \infty.$$

One important feature of this point of view is that the "infinite" point ∞ is approached by either letting the slope value m go to positive infinity, or by going to negative infinity. Therefore, this infinite point is somehow a "two-sided" infinity, being approached from either very large positive slope numbers and from very large negative slope numbers. Similarly, if we consider a point $[x:y:z] \in \mathbb{P}^2$ with $z \neq 0$, and let z approach 0, what do we find? If $z \neq 0$, then we have a point of the Euclidean plane $(x/z, y/z)$. As z approaches 0, keeping x and y fixed, we see that this Euclidean point has coordinates going to infinity. Therefore we conclude that the points of $\mathbb{P}^2$ with $z = 0$ are somehow "at infinity" when we think of them in relation to the Euclidean plane points where $z \neq 0$. We can also see this geometrically, using the $z = 1$ plane H: as a line in $\mathbb{R}^3$ through the origin moves and approaches a horizontal line, its intersection with H is a point which is moving away from he origin, and its coordinates are going to infinity. This convinces us that the extra points of $\mathbb{P}^2 - \mathbb{R}^2$ should be considered as being "at infinity". Now, we will give matrix representatives for the similarity transformation in homogeneous coordinates. It is strongly recommended that you verify the effects of these matrices yourselves.

- **Scaling matrix.**

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

- **Translation matrix.**

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}.$$

- **Rotation matrix.**

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

See Appendix 3 for a discussion on the registration problem in face recognition using these linear transformations.

3. Subspaces

Because data sets lie initially within large vector spaces, it is important to be able to decompose, or analyze , such spaces into smaller ones. In this section we further develop our tools for decomposing patterns into especially useful subspaces. One of the main ideas to be developed is that of the projection matrix, but first, we examine the general problem of decomposing a vector space into the sum of independent subspaces. To begin, we recall the basic definition of a subspace of a vector space.

DEFINITION 3.1. A *subspace* W of a vector space V is a subset of vectors such that

(1) (W is closed.) If $\mathbf{w}, \mathbf{w}' \in W$ and $a, b \in \mathbb{R}$ then $a\mathbf{w} + b\mathbf{w}' \in W$.
(2) (W contains the zero vector.) $\mathbf{0} \in W$.

PROPOSITION 3.1. *The set of vectors*

$$W = \left\{ w : w = \sum_i \alpha_i \mathbf{v}^{(i)} \right\}$$

that is spanned by the vectors $\left\{ \mathbf{v}^{(i)} \right\}$ *is a subspace.*

EXAMPLE 3.1. Any line through the origin of $\mathbb{R}^n$ is a one-dimensional subspace. Any space spanned by a collection of $k < n$ independent lines through the origin forms a k-dimensional subspace of $\mathbb{R}^n$.

An important parameter space whose elements are subspaces is called the Grassmann manifold.

DEFINITION 3.2. The *Grassmannian G(k,n)* or the *Grassmann manifold* is the set of k-dimensional subspaces in an n-dimensional vector space K^n for some field K, i.e.,

$$G(k,n) = \{ W \subset K^n \mid \dim(W) = k \}.$$

PROPOSITION 3.2. *If W_1 and W_2 are both subspaces, then so is their intersection $W_1 \cap W_2$.*

DEFINITION 3.3. The *sum* of the vector subspaces W_1 and W_2 is written as $W = W_1 + W_2$ and is defined to be the set

$$W_1 + W_2 = \{ \mathbf{w}_1 + \mathbf{w}_2 : \mathbf{w}_1 \in W_1, \mathbf{w}_2 \in W_2 \}.$$

PROPOSITION 3.3. *The sum of two subspaces is a subspace.*

The fact that the sum of two subspaces is a subspace provides us with a nice way to decompose a vector, viz., if $\mathbf{x} \in W$ and $W = W_1 + W_2$, we can always write $\mathbf{x} = \mathbf{w}_1 + \mathbf{w}_2$ where $\mathbf{w}_i \in W_i$. After a little bit of experimenting with this decomposition it is apparent that it is not unique. This ambiguity will generally be undesirable, but can be avoided by restricting the relationship between W_1 and W_2 as described below.

Independence of Subspaces.

To make the decomposition of a vector unique we require that the subspaces be independent.

PROPOSITION 3.4. *If W_1 and W_2 are independent subspaces and $V = W_1 + W_2$, $\mathbf{w}_1 \in W_1$, $\mathbf{w}_2 \in W_2$, then the decomposition of $\mathbf{x} \in V$ given by*

$$\mathbf{x} = \mathbf{w}_1 + \mathbf{w}_2$$

is unique.

Direct Sum Decompositions.

From above we see that the decomposition is unique if the subspaces are independent and we can distinguish the decomposition from the mere addition of subspaces by writing

$$W = W_1 \oplus W_2$$

as the *direct sum decomposition* of W. These ideas may be extended to the case of more than two subspaces. A special but important instance of independent subspaces is orthogonal subspaces.

DEFINITION 3.4. A vector $\mathbf{v} \in V$ is said to be *orthogonal* to a subspace $W \subseteq V$ if $\mathbf{v}$ is orthogonal to every $\mathbf{w} \in W$. Two subspaces W_1 and W_2 are said to be *orthogonal subspaces* if every $\mathbf{w}_1 \in W_1$ and $\mathbf{w}_2 \in W_2$ the inner product satisfies $(\mathbf{w}_1, \mathbf{w}_2) = 0$.

Given a subspace W of the vector space V, the space of all vectors orthogonal to W in V is called the *orthogonal complement* of W, written $W^{\perp}$.

EXAMPLE 3.2. Let $V = \mathbb{R}^3$. Then the x-axis and the y-axis are orthogonal subspaces of $\mathbb{R}^3$. Also, the orthogonal complement of the xy-plane is the z-axis.

An important special case of the direct sum decomposition occurs when the subspaces are orthogonal. In this situation we distinguish the notation by writing $\dot{\oplus}$.

PROPOSITION 3.5. *If two subspaces are orthogonal, then they are independent.*

Important Subspaces.

In this section, we describe the basic subspaces that will be of use in what follows. It is implicit, unless otherwise stated, that A is an $m \times n$ matrix.

DEFINITION 3.5. The *range* of A, denoted $\mathscr{R}(A)$, is the set of all vectors $\mathbf{v}$ such that $\mathbf{v} = A\mathbf{x}$, i.e.,

$$\mathscr{R}(A) = \{\mathbf{v} \in \mathbb{R}^m : \mathbf{v} = A\mathbf{x} \quad \text{for some} \quad \mathbf{x} \in \mathbb{R}^n\}.$$

The expression $\mathbf{v} = A\mathbf{x}$ may be rewritten

$$\begin{aligned} \mathbf{v} &= [\mathbf{a}^{(1)}|\mathbf{a}^{(2)}|\cdots|\mathbf{a}^{(n)}] \\ &= x_1\mathbf{a}^{(1)} + x_2\mathbf{a}^{(2)} + \cdots + x_n\mathbf{a}^{(n)}. \end{aligned}$$

This expression reveals that $\mathbf{v}$ lies in the span of the columns of A. Hence the range of A, $\mathscr{R}(A)$, is also referred to as the *column space* of A.

DEFINITION 3.6. The *null space* of A, denoted $\mathscr{N}(A)$, is the set of all vectors $\mathbf{y}$ such that $A\mathbf{y} = \mathbf{0}$, i.e.,

$$\mathscr{N}(A) = \{\mathbf{y} \in \mathbb{R}^n : A\mathbf{y} = \mathbf{0}\}.$$

DEFINITION 3.7. The *row space* of A, denoted $\mathscr{R}(A^T)$, is the set of all vectors $\mathbf{x}$ such that $\mathbf{x} = A^T\mathbf{v}$, i.e.,

$$\mathscr{R}(A^T) = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} = A^T\mathbf{v} \quad \text{for some} \quad \mathbf{v} \in \mathbb{R}^m\right\}.$$

DEFINITION 3.8. The *left null space* of A, denoted $\mathscr{N}(A^T)$, is the set of all vectors $\mathbf{v}$ such that $A^T\mathbf{v} = \mathbf{0}$, i.e.,

$$\mathscr{N}(A^T) = \left\{\mathbf{v} \in \mathbb{R}^m : A^T\mathbf{v} = \mathbf{0}\right\}.$$

PROPOSITION 3.6. *For any $m \times n$ matrix A one has*

$$\mathscr{N}(A) \perp \mathscr{R}(A^T),$$

i.e., they are orthogonal subspaces of $\mathbb{R}^n$, and

$$\mathscr{N}(A^T) \perp \mathscr{R}(A),$$

i.e., they are orthogonal subspaces of $\mathbb{R}^m$.

The range, or column space, of an $m \times n$ matrix A determines a subspace of $\mathbb{R}^m$. The number of independent vectors in this subspace, i.e., its dimension, is a very special and useful quantity for a matrix known as its *rank*.
We have a very useful counting rule:

PROPOSITION 3.7. *Let A be an $m \times n$ matrix. Then*

$$r + \dim \mathscr{N}(A) = n,$$

where r is the rank of A.

From this relationship between the dimensions it follows that the spaces $\mathscr{N}(A)$ and $\mathscr{R}(A^T)$ decompose $\mathbb{R}^n$, i.e.,

$$\mathbb{R}^n = \mathscr{N}(A) \dot{\oplus} \mathscr{R}(A^T).$$

An analogous statement holds true for the decomposition of $\mathbb{R}^m$, i.e.,

$$r + \dim \mathscr{N}(A^T) = m$$

and

$$\mathbb{R}^m = \mathscr{N}(A^T) \dot{\oplus} \mathscr{R}(A).$$

4. Projection Matrices and Orthogonal Projections

The direct sum provides a framework within which a vector space may be systematically split into subspaces that provide a unique expression for the decomposition of any vector in the space. In this section we describe a mapping, referred to as a *projector*, or *projection matrix*, which takes a vector and executes this decomposition.

DEFINITION 4.1. A matrix $\mathbb{P}$ is said to be a *projection matrix* if

$$\mathbb{P}^2 = \mathbb{P}.$$

Such matrices are said to be *idempotent*.

EXAMPLE 4.1. It is easy to verify that the matrix

$$\mathbb{P} = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} \\ 1 & \frac{1}{2} \end{pmatrix}$$

is a projection matrix. Note that it has rank 1 and that

$$\mathscr{R}(\mathbb{P}) = \left\{ \alpha \begin{pmatrix} 1 \\ 2 \end{pmatrix} : \alpha \in \mathbb{R} \right\}.$$

See Figure 5 for possible actions of a projection matrix.

Invariant Subspaces

DEFINITION 4.2. Let V be a vector space and L a linear operation on V. If W is a subspace of V, we say W is *invariant* under L for each $\mathbf{w} \in W$ we have $L\mathbf{w} \in W$. In order words, $L(W) \subseteq W$.

Furthermore, if W_1 and W_2 are subspaces invariant under A (where A is the matrix that corresponds to the linear operator L) with $V = W_1 \oplus W_2$, then we say A is *reduced* or *decomposed* by W_1 and W_2. We now show that a projection matrix naturally decomposes a vector space into a pair of invariant subspaces, i.e.,

$$V = \mathscr{R}(\mathbb{P}) \oplus \mathscr{R}(I - \mathbb{P}).$$

First, any vector $\mathbf{v} \in V$ may be decomposed into elements of $\mathscr{R}(\mathbb{P})$ and $\mathscr{R}(I - \mathbb{P})$ via

$$\mathbf{v} = \mathbb{P}\mathbf{v} + (I - \mathbb{P})\mathbf{v}.$$

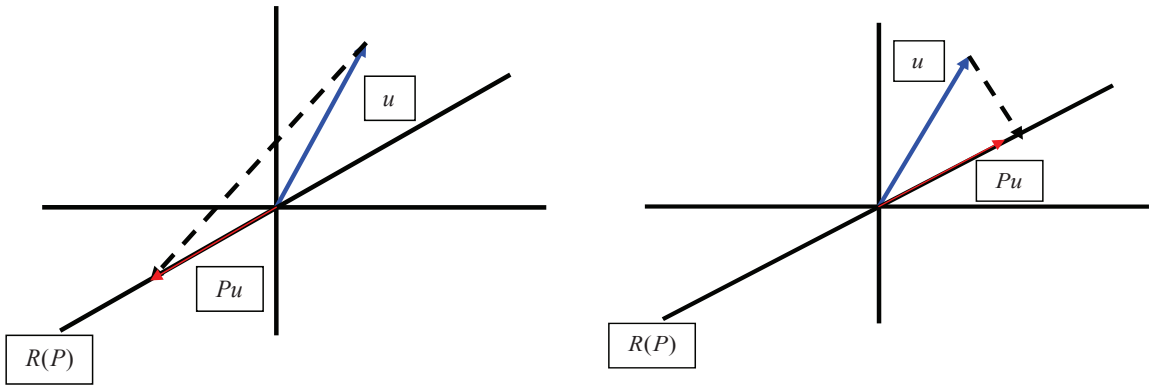


FIGURE 5. (a) A nonorthogonal, or *oblique* projection. (b) An orthogonal projection.

Note that the mapping $I-\mathbb{P}$ is also a projection matrix (known as the *complementary projection matrix*) since

$$\begin{aligned}(I-\mathbb{P})^2 &= I-2\mathbb{P}+\mathbb{P}^2\\ &= I-2\mathbb{P}+\mathbb{P}\\ &= I-\mathbb{P}.\end{aligned}$$

We may also employ the notation $\mathbb{Q}=I-\mathbb{P}$ to represent the projection matrix onto the null space. The subspace $\mathscr{R}(\mathbb{P})$ is invariant under the action of $\mathbb{P}$ may be concluded from the following proposition:

PROPOSITION 4.1. $\mathbf{v}\in\mathscr{R}(\mathbb{P})$ *if and only if* $\mathbb{P}\mathbf{v}=\mathbf{v}$.

PROOF. First assume $\mathbf{v}\in\mathscr{R}(\mathbb{P})$, i.e., $\mathbf{v}=\mathbb{P}\mathbf{x}$ for some $\mathbf{x}\in\mathbb{R}^n$. So $\mathbb{P}\mathbf{v}=\mathbb{P}^2\mathbf{x}=\mathbb{P}\mathbf{x}=\mathbf{v}$. Conversely, assume that $\mathbb{P}\mathbf{v}=\mathbf{v}$. It follows directly that $\mathbf{v}\in\mathscr{R}(\mathbb{P})$. □

Similarly, it may be argued that the space $\mathscr{R}(I-\mathbb{P})$ is invariant under the action of the projector $I-\mathbb{P}$. It is left to show that the subspaces are independent.

PROPOSITION 4.2. $\mathscr{R}(\mathbb{P})\cap\mathscr{R}(I-\mathbb{P})=\{\mathbf{0}\}$.

PROOF. Let $\mathbf{v}\in\mathscr{R}(I-\mathbb{P})$, i.e., $\mathbf{v}=(I-\mathbb{P})\mathbf{x}$ for some $\mathbf{x}$. So $\mathbb{P}\mathbf{v}=\mathbf{0}$. But, by Proposition 4.1, $\mathbf{v}\in\mathscr{R}(\mathbb{P})$ if and only if $\mathbb{P}\mathbf{v}=\mathbf{v}$; hence we conclude that $\mathbf{v}=\mathbf{0}$ is the only element common to both $\mathscr{R}(\mathbb{P})$ and $\mathscr{R}(I-\mathbb{P})$. □

Lastly, we make a connection between the range of a complementary projector and the null space of the associated projector.

PROPOSITION 4.3. *The range of the complementary projector is the same as the null space of the projector, i.e.,*

$$\mathscr{R}(I-\mathbb{P})=\mathscr{N}(\mathbb{P}).$$

PROOF. Let $\mathbf{r}\in\mathscr{R}(\mathbb{P})$, i.e.,

$$\mathbf{r}=(I-\mathbb{P})\mathbf{v}.$$

Projecting this vector gives

$$\mathbb{P}\mathbf{r}=\mathbb{P}\mathbf{v}-\mathbb{P}^2\mathbf{v}=\mathbf{0}.$$

Thus, if $\mathbf{r}\in\mathscr{R}(I-\mathbb{P})$, then $\mathbf{r}\in\mathscr{N}(\mathbb{P})$. Since this is true for an arbitrary $\mathbf{r}$, it follows that $\mathscr{R}(I-\mathbb{P})\subset\mathscr{N}(\mathbb{P})$. Conversely, if $\mathbf{r}\in\mathscr{N}(\mathbb{P})$, then $\mathbb{P}\mathbf{r}=\mathbf{0}$, i.e., $(I-\mathbb{P})\mathbf{r}=\mathbf{r}$. So $\mathbf{r}\in\mathscr{R}(I-\mathbb{P})$. Hence the result. □

From these results it is now clear that a projection matrix separates a space into the sum of two independent subspaces

$$V = \mathscr{R}(\mathbb{P}) \oplus \mathscr{N}(\mathbb{P}).$$

It follows from this discussion that for every splitting

$$V = W_1 \oplus W_2$$

there exists a projection operator $\mathbb{P}$ such that

$$\mathscr{R}(\mathbb{P}) = W_1$$

and

$$\mathscr{N}(\mathbb{P}) = W_2.$$

All that is required is to determine $\mathbb{P}$ given W_1. We will describe a method for doing this in the following discussions.

Orthogonal Projection Matrices

We have seen that projection matrices permit the decomposition of a space into invariant subspaces. The most useful application of this idea is when the resulting subspaces are orthogonal, i.e., when the projection matrix and its complement produce orthogonal vectors. We begin with a basic definition.

DEFINITION 4.3. Let $\mathbf{x} = \mathbf{w}_1 + \mathbf{w}_2$ and $\mathbf{w}_1 \in W_1, \mathbf{w}_2 \in W_2$ with $W_1 \perp W_2$. The vector $\mathbf{w}_1$ is called the *orthogonal projection* of $\mathbf{x}$ *onto* W_1, and $\mathbf{w}_2$ is called the *orthogonal projection* of $\mathbf{x}$ onto W_2.

Associated with an orthogonal projection is the operator, which we now refer to as an orthogonal projection matrix, which performs the projection described in the definition above. (Note that the orthogonal projection matrix should not be confused with an orthogonal matrix.)

DEFINITION 4.4. If the subspaces $\mathscr{R}(\mathbb{P})$ and $\mathscr{N}(\mathbb{P})$ are orthogonal, then the projection matrix $\mathbb{P}$ is said to be an *orthogonal projection matrix*.

If $\mathbb{P}$ is an orthogonal projection matrix, then we may write the direct sum decomposition of the space as

$$V = \mathscr{R}(\mathbb{P}) \dot{\oplus} \mathscr{N}(\mathbb{P}).$$

Best Approximations and the Projection Theorem

Suppose W_1 and W_2 are subspaces of an inner product space V such that $V = W_1 + W_2$, and let $\mathbf{x} \in V$ be an arbitrary vector. The notion of *best approximation* to $\mathbf{x}$ by a vector in W_1 is made explicit as follows:

DEFINITION 4.5. A best approximation to $\mathbf{x}$ by vectors in W_1 is a vector $\mathbf{w}_1 \in W_1$ such that

$$||\mathbf{x} - \mathbf{w}_1|| \leq ||\mathbf{x} - \mathbf{w}_1'||$$

for all $\mathbf{w}_1' \in W_1$.

In other words, for each $\mathbf{x} \in V$, we seek a vector $\mathbf{w}_1 \in W_1$ such that the norm $||\mathbf{x} - \mathbf{w}_1||$ is a minimum. We shall assume, unless otherwise stated, that the Euclidean norm is to be employed. The theory is of course valid for norms in general.

THEOREM 4.1. **The Projection Theorem.** *Of all decompositions of the form*

$$\mathbf{x} = \mathbf{w}_1' + \mathbf{w}_2'$$

with $\mathbf{w}_1' \in W_1$, *the orthogonal projection provides the best approximation to* $\mathbf{x}$. *Equivalently, the orthogonal projection minimizes the norm* $||\mathbf{w}_2'||$. *(See Figure 6)*

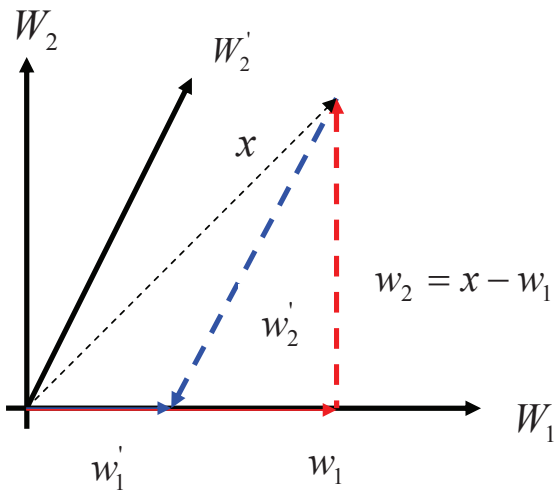


FIGURE 6. The best approximation to a point $\mathbf{x}$ is the orthogonal projection $\mathbf{w}_1$. Every other projection $\mathbf{w}_1'$ has a larger residual $\mathbf{w}_2'$.

PROOF. We rewrite

$$\begin{aligned}
||\mathbf{x}-\mathbf{w_1}'||^2 &= ||\mathbf{x}-\mathbf{w_1}+\mathbf{w_1}-\mathbf{w_1}'||^2 \\
&= \left(\mathbf{x}-\mathbf{w_1}+\mathbf{w_1}-\mathbf{w_1}', \mathbf{x}-\mathbf{w_1}+\mathbf{w_1}-\mathbf{w_1}'\right) \\
&= \left(\mathbf{x}-\mathbf{w_1}, \mathbf{x}-\mathbf{w_1}+\mathbf{w_1}-\mathbf{w_1}'\right)+\left(\mathbf{w_1}-\mathbf{w_1}', \mathbf{x}-\mathbf{w_1}+\mathbf{w_1}-\mathbf{w_1}'\right) \\
&= (\mathbf{x}-\mathbf{w_1}, \mathbf{x}-\mathbf{w_1})+\left(\mathbf{w_1}-\mathbf{w_1}', \mathbf{w_1}-\mathbf{w_1}'\right)+2\left(\mathbf{x}-\mathbf{w_1}, \mathbf{w_1}-\mathbf{w_1}'\right) \\
&= ||\mathbf{x}-\mathbf{w_1}||^2+||\mathbf{w_1}-\mathbf{w_1}'||^2+2\left(\mathbf{x}-\mathbf{w_1}, \mathbf{w_1}-\mathbf{w_1}'\right).
\end{aligned}$$

Observe that $\mathbf{x}-\mathbf{w_1}=\mathbf{w_2}\in W_2$ and that $\mathbf{w_1}-\mathbf{w_1}'\in W_1$. If $W_1\perp W_2$, i.e., the projection is orthogonal, then it follows that

$$\left(\mathbf{x}-\mathbf{w_1}, \mathbf{w_1}-\mathbf{w_1}'\right)=0.$$

From this we have

$$||\mathbf{x}-\mathbf{w_1}'||^2 \geq ||\mathbf{x}-\mathbf{w_1}||^2;$$

in other words, $\mathbf{w_1}'=\mathbf{w_1}$ is a *best approximation* to $\mathbf{x}$. Note that $||\mathbf{w_2}'||=||\mathbf{x}-\mathbf{w_1}'||$ is a minimum for $\mathbf{w_1}'$, and since $\mathbf{w_2}=\mathbf{x}-\mathbf{w_1}$, it follows that $\mathbf{w_2}'=\mathbf{w_2}$ in the case of the best approximation. □

Furthermore, it can be shown that this best approximation is unique. In addition, these results be extended to the general setting of metric spaces. See alternative texts for the details.
Note that this theorem says nothing about how to select W_1 itself. In other words, given a fixed W_1, the theorem indicates that the orthogonal projection will minimize the error for each vector in V. However, selecting W_1 for a given data set is an entirely different and interesting issue which will be pursued in the sequel.

Criterion for Orthogonal Projections

PROPOSITION 4.4. *If*

$$\mathbb{P}=\mathbb{P}^T,$$

then the matrix $\mathbb{P}$ is an orthogonal projection matrix.

PROOF. Let $\mathbb{P} = \mathbb{P}^T$, $\mathbb{P}\mathbf{x} \in \mathscr{R}(\mathbb{P})$ and $(I-\mathbb{P})\mathbf{x} \in \mathscr{N}(\mathbb{P})$. Then

$$\begin{aligned}(\mathbb{P}\mathbf{x})^T(I-\mathbb{P})\mathbf{x} &= \mathbf{x}^T\mathbb{P}^T(I-\mathbb{P})\mathbf{x} \\ &= \mathbf{x}^T(\mathbf{P}-\mathbb{P}^2)\mathbf{x} \\ &= \mathbf{0}\end{aligned}$$

□

The converse is also true, i.e., if $\mathbb{P}$ is an orthogonal projection matrix, then $\mathbb{P} = \mathbb{P}^T$.

EXAMPLE 4.2. It is easy to verify that the matrix

$$\mathbb{P} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

is an orthogonal projection matrix.

EXAMPLE 4.3. Every matrix of the form $\mathbf{v}\mathbf{v}^T$ is an orthogonal projection matrix if $||\mathbf{v}|| = 1$:

$$\begin{aligned}(\mathbf{v}\mathbf{v}^T)^2 &= (\mathbf{v}\mathbf{v}^T)(\mathbf{v}\mathbf{v}^T) \\ &= \mathbf{v}(\mathbf{v}^T\mathbf{v})\mathbf{v}^T \\ &= \mathbf{v}\mathbf{v}^T.\end{aligned}$$

Note that this projection matrix has rank one and the $\mathscr{R}(\mathbf{v}\mathbf{v}^T) = \text{span}(\mathbf{v})$. From this example we observe that any vector $\mathbf{u}$ may be orthogonally projected onto a given vector $\mathbf{v}$ by defining

$$\mathbb{P}_\mathbf{v}\mathbf{u} = (\mathbf{v}\mathbf{v}^T)\mathbf{u} = \mathbf{v}(\mathbf{v}^T\mathbf{u}).$$

Also, the orthogonal complement, or *residual* $\mathbf{r}$ is then found to be

$$\mathbf{r} = \mathbb{P}_\mathbf{v}^\perp = (I-\mathbb{P}_\mathbf{v})\mathbf{u} = \mathbf{u} - (\mathbf{v}^T\mathbf{u})\mathbf{v}.$$

Orthogonal Projection Onto a Subspace

We can leverage our ability to project $\mathbf{u}$ onto a single vector $\mathbf{v}$ into a method for computing the orthogonal projection of $\mathbf{u} \in \mathbb{R}^n$ onto a subspace W. To begin, we assume that we have an orthonormal basis for the space W consisting of the vectors $\left\{\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(k)}\right\}$. We may view each of the $\mathbf{w}^{(i)}$ as spanning a one-dimensional subspace W_i. Clearly, each of these spaces is orthogonal, i.e.,

$$W_i \perp W_j, \quad i \neq j.$$

Furthermore, the sum of these subspaces span a k-dimensional subspace

$$W = W_1 + W_2 + \cdots + W_k.$$

From our previous deliberations,

$$W = W_1 \dot{\oplus} \cdots \dot{\oplus} W_k.$$

In other words, the orthonormal basis induces a direct sum decomposition of the subspace W. A projection onto W may be constructed from projections onto the individual subspaces.
The projection of $\mathbf{u}$ onto the ith subspace is given by

$$\mathbb{P}_{\mathbf{w}^{(i)}}\mathbf{u} = \mathbf{w}^{(i)}\mathbf{w}^{(i)T}\mathbf{u}.$$

If we write $\mathbb{P}_i \equiv \mathbb{P}_{\mathbf{w}^{(i)}}$, then the projection matrix onto W is given by

$$\mathbb{P} = \sum_{i=1}^{k} \mathbb{P}_i = \sum_{i=1}^{k} \mathbf{w}^{(i)}\mathbf{w}^{(i)T}. \tag{6}$$

Given the matrix $M = [\mathbf{w}^{(1)}|\cdots|\mathbf{w}^{(k)}]$, it follows that

$$\mathbb{P} = MM^T.$$

Orthogonalization

In the course of the above computations we assumed that the subspace on which we were to project was equipped with an orthonormal basis. We now review the *Gram-Schmidt* procedure for computing an orthonormal basis starting from a set of vectors $\left\{\mathbf{v}^{(i)}\right\}_{i=1}^{m}$. Take as the first element

$$\mathbf{u}^{(1)} = \frac{\mathbf{v}^{(1)}}{||\mathbf{v}^{(1)}||}.$$

The second element of this set is constructed using the same two-to-one-dimensional projection technique discussed previously. The projection of $\mathbf{v}^{(2)}$ onto $\mathbf{u}^{(1)}$ is given by

$$\mathbb{P}_{\mathbf{u}^{(1)}}\mathbf{v}^{(2)} = (\mathbf{u}^{(1)}\mathbf{u}^{(1)T})\mathbf{v}^{(2)},$$

so the vector pointing orthogonally to $\mathbf{u}^{(1)}$ is the residual

$$\mathbf{r} = (I - \mathbb{P}_{\mathbf{u}^{(1)}})\mathbf{v}^{(2)}.$$

Simplifying and normalizing this vector gives

$$\mathbf{u}^{(2)} = \frac{\mathbf{v}^{(2)} - (\mathbf{u}^{(1)T}\mathbf{v}^{(2)})\mathbf{u}^{(1)}}{||\mathbf{v}^{(2)} - (\mathbf{u}^{(1)T}\mathbf{v}^{(2)})\mathbf{u}^{(1)}||}.$$

Proceeding in the same fashion with the jth direction we have

$$\mathbf{u}^{(j)} = \frac{\mathbf{v}^{(j)} - \sum_{i=1}^{j-1}\mathbf{u}^{(i)T}\mathbf{v}^{(j)}\mathbf{u}^{(i)}}{||\mathbf{v}^{(j)} - \sum_{i=1}^{j-1}\mathbf{u}^{(i)T}\mathbf{v}^{(j)}\mathbf{u}^{(i)}||}.$$

Note that if the added direction $\mathbf{v}^{(j)}$ is dependent on the previous vectors, then $\mathbf{u}^{(j)} = \mathbf{0}$.

EXAMPLE 4.4. Consider the matrix

$$A = \begin{pmatrix} 1 & 1 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Find the orthogonal projection matrix that takes an element of $\mathbb{R}^{(4)}$ onto $\mathscr{R}(A)$. Define $\mathbf{a}^{(1)} = (1010)^T$ and $\mathbf{a}^{(2)} = (1001)^T$. Since the third column is a multiple of the first, $\mathscr{R}(A) = \text{span}(\mathbf{a}^{(1)}, \mathbf{a}^{(2)})$. To find the projection matrix $\mathbb{P}$ that maps an element of $\mathbb{R}^{(4)}$ onto $\mathscr{R}(A)$, we first determine an orthonormal basis for $\mathscr{R}(A)$. Clearly the columns $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$ are linearly independent, but they are not orthogonal. Using the Gram-Schmidt procedure we obtain

$$\mathbf{u}^{(1)} = \frac{1}{\sqrt{2}}(1010)^T$$

and

$$\mathbf{u}^{(2)} = \frac{1}{\sqrt{6}}(10-12)^T.$$

The projection matrix onto $\mathbf{u}^{(1)}$ is given by

$$\mathbb{P}_1 = \mathbf{u}^{(1)}\mathbf{u}^{(1)T} = \frac{1}{2}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

and the projection matrix onto $\mathbf{u}^{(2)}$ is given by

$$\mathbb{P}_2 = \mathbf{u}^{(2)}\mathbf{u}^{(2)T} = \frac{1}{6}\begin{pmatrix} 1 & 0 & -1 & 2 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & -2 \\ 1 & 0 & -1 & 2 \end{pmatrix}.$$

From this we have the projection matrix

$$\mathbb{P} = \mathbb{P}_1 + \mathbb{P}_2 = \frac{1}{3}\begin{pmatrix} 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & -1 \\ 1 & 0 & -1 & 2 \end{pmatrix}.$$

Application: The Novelty Filter. We have seen how a projection matrix may be constructed from an arbitrary collection of vectors which span a vector subspace. Now we consider a direct application of these ideas to a pattern processing problem.

Given a data set consisting of an ensemble of pattern vectors, e.g., digital images of human faces, we generate associated column vectors by concatenating the columns/rows. In other words, each pattern is available as an n-tuple. Further, let's assume that we are given a large number k of these images but that $k < n$, probably much less. Thus we have an ensemble $\left\{\mathbf{v}^{(i)}\right\}_{i=1}^{k}$ where $\mathbf{v}^{(i)} \in \mathbb{R}^n$ for every i.

We would like to determine a projection matrix that takes a new pattern and splits it into two components: the first component is the portion of the data that resides in the subspace spanned by the original patterns, or *training set*; the second component is orthogonal to the training set and represents the portion of the data that is *novel*.

With this in mind, we define W as the basis in which all the training patterns lie and note that $\dim W = m \leq k$ with equality if the original patterns are independent. To determine an orthonormal basis for W the Gram-Schmidt procedure is applied to the training data. This operation will take us from the set of generally nonorthogonal and possibly linearly dependent pattern vectors to an orthonormal basis for W, which we write as the set $\left\{\mathbf{u}^{(i)}\right\}_{i=1}^{m}$. In summary, $W = \text{span}(\mathbf{v}^{(1)}, \cdots, \mathbf{v}^{(k)}) = \text{span}(\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(m)})$.

Next, the orthogonal projection matrix $\mathbb{P}$ is computed via Equation (6), as well as the complementary orthogonal projection matrix $I - \mathbb{P}$. The projection of a pattern produces a point in $\mathbb{R}^m$,

$$\mathbb{P} : \mathbb{R}^n \to W,$$

$$\mathbf{x} \rightsquigarrow \mathbb{P}\mathbf{x} = \mathbf{w} \in \mathbb{R}^m,$$

and the residual sits in $\mathbb{R}^{n-m}$:

$$I - \mathbb{P} : \mathbb{R}^n \to W^{\perp},$$

$$\mathbf{x} \rightsquigarrow (I - \mathbb{P})\mathbf{x} = \mathbf{w}^{\perp} \in \mathbb{R}^{n-m}.$$

As before, this is an orthogonal decomposition of

$$\mathbf{x} = \mathbf{w} + \mathbf{w}^{\perp}.$$

According to Kohonen [**36**], we refer to this orthogonal component as the *novelty* of the pattern, and the general procedure of separating the novelty of a pattern from the non-novel component as the

novelty filter. In the face data example, novelty might correspond to a new face, or possibly a new pose of a training face.
In practice, problems may arise that make the interpretation of the novelty of a pattern more challenging. First, if the original set of patterns does not include samples of all possible normal patterns, or at least enough to span this set, then the subspace m will be too small and components of a pattern may appear novel only because the set of stored patterns is too small. In addition, the effect of noise on such a subspace representation can be significant. More details on novelty filter can be found in [**36**].

5. Eigenvalues and Eigenvectors

At the center of our discussion has been the construction of projection matrices to permit the decomposition of vector spaces. One of the most useful methods for construction projectors is to first determine a basis of eigenvectors for the space in question.
The study of eigenvalue and eigenvectors is probably one of the most important topics of linear algebra. It has applications in the study of population growth, statistical analysis, face recognition, medical imaging, control theory, vibration analysis, electric circuits, etc. And this is just a very small pool. Here, we will briefly introduce the definition of eigenvalue and eigenvectors and illustrate its use with a face recognition application.
The central question of the **eigenvalue problem** can generally be stated as follows. Given an $n \times n$ matrix A, which is usually thought of as a linear transformation, an **eigenvector** of that linear transformation (matrix A) is a nonzero vector which, when applied by that transformation, changes in length, but not direction. Mathematically, an (nonzero) eigenvector $\mathbf{x}$ of the matrix A is a $n \times 1$ matrix such that $A\mathbf{x}$ is a scalar multiple of $\mathbf{x}$. The scalar is usually denoted by λ (lambda) and is called an **eigenvalue** of A. In symbols, the eigenvalue problem can be written compactly as

$$A\mathbf{x} = \lambda\mathbf{x}. \tag{7}$$

Given an $n \times n$ matrix A, how do we find the eigenvalue and corresponding eigenvectors? Take Equation (7), rewrite it into

$$\lambda I\mathbf{x} - A\mathbf{x} = O, \tag{8}$$

where I is the $n \times n$ identity matrix and O is the $n \times n$ zero matrix. Then factor out the $\mathbf{x}$

$$(\lambda I - A)x = O. \tag{9}$$

This homogeneous system of equations (n unknowns and n equations) has nonzero solutions if and only if the determinant of $\lambda I - A$ is zero, since x cannot be a zero vector. This condition equation $\det(\lambda I - A) = 0$ is called the **characteristic equation** of A, and is a polynomial equation of degree n in λ, denoted by $\rho(\lambda) = (\lambda - \lambda^{(1)})(\lambda - \lambda^{(2)}) \cdots (\lambda - \lambda^{(n)}) = 0$. The eigenvalues of A are readily obtained by solving the characteristic equation. Once we find the eigenvalues of A, we can use Gaussian elimination to find the corresponding eigenvectors. It is possible that some of the $\lambda^{(i)}$ are the same. The number of times a particular eigenvalue is repeated is referred to as its *algebraic multiplicity*.

EXAMPLE 5.1. Find the eigenvalues and corresponding eigenvectors of the matrix

$$A = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix}.$$

Solution:

The characteristic equation of A is

$$\begin{aligned} |\lambda I - A| &= \begin{vmatrix} \lambda-2 & 12 \\ -1 & \lambda+5 \end{vmatrix} \\ &= (\lambda-2)(\lambda+5)+12 \\ &= \lambda^2+3\lambda+2 \\ &= (\lambda+1)(\lambda+2)=0 \quad \text{(set to zero)}, \end{aligned}$$

which gives $\lambda_1 = -1$ and $\lambda_2 = -2$ as the two eigenvalues of A. To obtain the corresponding eigenvectors, we use Gaussian elimination to solve the homogeneous linear system $((\lambda I - A)\mathbf{x} = O)$:
For $\lambda_1 = -1$:

$$(-1)I - A = \begin{bmatrix} -1-2 & 12 \\ -1 & -1+5 \end{bmatrix} = \begin{bmatrix} -3 & 12 \\ -1 & 4 \end{bmatrix},$$

which row reduced to

$$\begin{bmatrix} 1 & -4 \\ 0 & 0 \end{bmatrix}.$$

This gives

$$\begin{bmatrix} 1 & -4 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow x_1 - 4x_2 = 0.$$

Letting $x_2 = t$, we conclude that every eigenvector of λ_1 is of the form

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4t \\ t \end{bmatrix} = t \begin{bmatrix} 4 \\ 1 \end{bmatrix}, t \neq 0.$$

For $\lambda_2 = -2$:

$$(-2)I - A = \begin{bmatrix} -2-2 & 12 \\ -1 & -2+5 \end{bmatrix} = \begin{bmatrix} -4 & 12 \\ -1 & 3 \end{bmatrix},$$

which row reduced to

$$\begin{bmatrix} 1 & -3 \\ 0 & 0 \end{bmatrix}.$$

This gives

$$\begin{bmatrix} 1 & -3 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow x_1 - 3x_2 = 0.$$

Letting $x_2 = t$, we conclude that every eigenvector of λ_2 is of the form

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3t \\ t \end{bmatrix} = t \begin{bmatrix} 3 \\ 1 \end{bmatrix}, t \neq 0.$$

If we think of a vector as a *direction*, then the term "eigen" is commonly used as a prefix to directions that are unchanged under the influence of a linear transformation.

EXAMPLE 5.2. The matrix

$$A = \begin{pmatrix} 1 & -6 & 1 \\ 0 & -3 & -15 \\ 0 & 0 & -3 \end{pmatrix}$$

has the characteristic polynomial

$$\rho(\lambda) = (\lambda-1)(\lambda+3)^2,$$

from which we conclude that $\lambda = 1$ is an eigenvalue with (algebraic) multiplicity 1 and $\lambda = 3$ is an eigenvalue with multiplicity 2.

PROPOSITION 5.1. *Every eigenvalue has associated with it at least one eigenvector*

PROOF. Given $\det(A-\lambda I)=0$,

$$rank(A-\lambda I)<n,$$

from which it follows, using the fact $r+\dim\mathcal{N}(A)=n$, that

$$\dim\mathcal{N}(A-\lambda I)\geq 1.$$

The elements of this nontrivial space are eigenvectors. □

PROPOSITION 5.2. *The eigenvectors associated with the eigenvalue λ, and the zero vector, from an invariant subspace, referred to as the eigenspace E_λ.*

EXAMPLE 5.3. If $\mathbf{u},\mathbf{v}\in E_\lambda$, then $A\mathbf{u}=\lambda\mathbf{u}$ and $A\mathbf{v}=\lambda\mathbf{v}$. Let $\mathbf{w}=\alpha\mathbf{u}+\beta\mathbf{v}$. We have

$$\begin{aligned}A(\alpha\mathbf{u}+\beta\mathbf{v}) &= \alpha A\mathbf{u}+\beta A\mathbf{v}\\ &= \lambda(\alpha\mathbf{u}+\beta\mathbf{v})\end{aligned}$$

from which we may conclude that $\mathbf{w}\in E_\lambda$. Recall that every subspace must contain the zero vector, yet zero is not an eigenvector. The eigenspace E_λ is an invariant subspace, i.e., $\mathbf{u}\in E_\lambda$ implies $A\mathbf{u}\in E_\lambda$.

DEFINITION 5.1. The dimension of the eigenspace, i.e., $\dim E_\lambda$, is the number of independent eigenvectors associated with λ. This number is also referred to as the *geometric multiplicity* of λ.

PROPOSITION 5.3. *The algebraic multiplicity of λ is greater or equal to the geometric multiplicity.*

An eigenvalue whose geometric multiplicity is less than its algebraic multiplicity is said to be *defective*. An $n\times n$ matrix that has no defective eigenvalues must have n independent eigenvectors.

THEOREM 5.1. *Let A be an $n\times n$ matrix with n independent eigenvectors $\left\{\mathbf{v}^{(1)},\mathbf{v}^{(2)},\cdots,\mathbf{v}^{(n)}\right\}$. Define the matrix $V=[\mathbf{v}^{(1)}|\cdots|\mathbf{v}^{(n)}]$. Then*

$$V^{-1}AV=\Lambda,$$

where $\Lambda=diag(\lambda^{(1)},\ldots,\lambda^{(n)})$,

PROOF. We will show that $AV=V\Lambda$:

$$\begin{aligned}AV &= A[\mathbf{v}^{(1)},\mathbf{v}^{(2)},\cdots,\mathbf{v}^{(n)}]\\ &= [A\mathbf{v}^{(1)},A\mathbf{v}^{(2)},\cdots,A\mathbf{v}^{(n)}]\\ &= [\lambda^{(1)}\mathbf{v}^{(1)},\lambda^{(2)}\mathbf{v}^{(2)},\cdots,\lambda^{(n)}\mathbf{v}^{(n)}]\\ &= V\Lambda\end{aligned}$$

Note that the independence of the vectors of V is required so that V^{-1} exists. □

As a consequence of this theorem, a matrix that has n independent eigenvectors is said to be *diagonalizable*.

PROPOSITION 5.4. *Eigenvectors associated with distinct eigenvalues are linearly independent.*

A set of n independent eigenvectors forms a basis for $\mathbb{R}^n$ referred to as an *eigenbasis*.

COROLLARY 5.2. *An $n\times n$ matrix with n distinct eigenvalues*

$$\lambda^{(1)}>\lambda^{(2)}>\cdots>\lambda^{(n)}$$

is diagonalizable.

This follows directly from the fact that the eigenvectors must be independent.
In elementary linear algebra courses it is shown that symmetric matrices have very important properties. We bundle a few of them into a single proposition.

PROPOSITION 5.5. *Let A be an $n \times n$ symmetric matrix.*

(1) *The eigenvalues of A are real.*
(2) *If $\lambda^{(i)} \neq \lambda^{(j)}$, then the eigenvectors $\mathbf{v}^{(i)}$ and $\mathbf{v}^{(j)}$ are orthogonal.*
(3) *A is not defective, i.e., it has n independent eigenvectors.*

Thus, given an $n \times n$ symmetric matrix A, an orthonormal basis for $\mathbb{R}^n$ may be constructed from its eigenvectors. Eigenvalues of algebraic multiplicity one has orthogonal eigenvectors; eigenvectors that correspond to an eigenvalue with multiplicity greater than one may be orthogonalized by applying the Gram-Schmidt procedure.

DEFINITION 5.2. We refer to A as being *orthogonally diagonalizable* if

$$V^T A V = \Lambda.$$

THEOREM 5.3. *Spectral Theorem. The matrix A is symmetric if and only if there is a real orthogonal matrix V such that*

$$V^T A V = \Lambda.$$

The equation $V^T A V = \Lambda$ may be rewritten as

$$A = V \Lambda V^T = \sum_i \lambda^{(i)} \mathbf{v}^{(i)} \mathbf{v}^{(i)T}.$$

This representation expresses a square matrix in terms of a sum of rank one matrices.

Application: Eigenfaces

Eigenfaces, as the name suggests, are a set of eigenvectors used in the computer vision problem of human face recognition. A digital image of a face can be seen as a vector whose components are the brightness of each pixel. The dimension of this vector space is the number of pixels in the image. Since human faces look relatively similar, it is reasonable to assume that there is a small set of "eigenfaces" that represent the features of the faces. The eigenfaces that are created will appear as light and dark areas that are arranged in a specific pattern. This pattern is how different features of a face are singled out to be evaluated and scored. There will be a pattern to evaluate symmetry, if there is any style of facial hair, where the hairline is, or evaluate the size of the nose or mouth. Other eigenfaces have patterns that are less simple to identify, and the image of the eigenface may look very little like a face. What this means is, given any face image, we can use this set of eigenfaces to represent it. Looking at it from a different perspective, the set of eigenfaces are "basis" vectors in the space of all faces.
These eigenfaces have provided great use in classification problems based on **Principal Component Analysis (PCA)**, which was first discovered by Karl Pearson in 1901. It is now mostly used as a tool in exploratory data analysis and for making predictive models. The main step of PCA is the extraction of eigenvalues and eigenvectors of a covariance matrix to achieve dimensionality reduction. And yes, this is why you learn eigenvalues and eigenvectors! For example, we are given a set of face images in Figure 7. Using PCA, we obtain a set of (more than 10) eigenfaces that are sorted by the magnitude of their corresponding eigenvalues. The first ten (ten largest eigenvalues) of those are shown in Figure 8. We can see that the first eigenface picks out the lighting condition of the set while the other ones pick out different features of the overall face images. The technique of using eigenvectors to perform recognition is used for handwritten digital recognition, lip reading, voice recognition, sign language/hand gestures, etc.

Application: Google Eigenvectors

The following presentation will closely follow the discussions in [**19**] and [**17**]. For a relatively more detailed mathematical discussion of the method, please see [**17**].

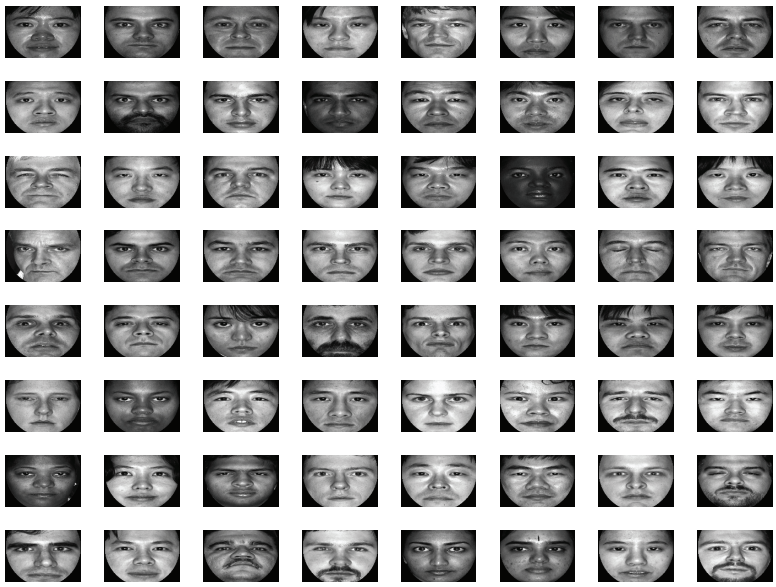


FIGURE 7. Sample face images that are used to create eigenfaces in Figure 8.

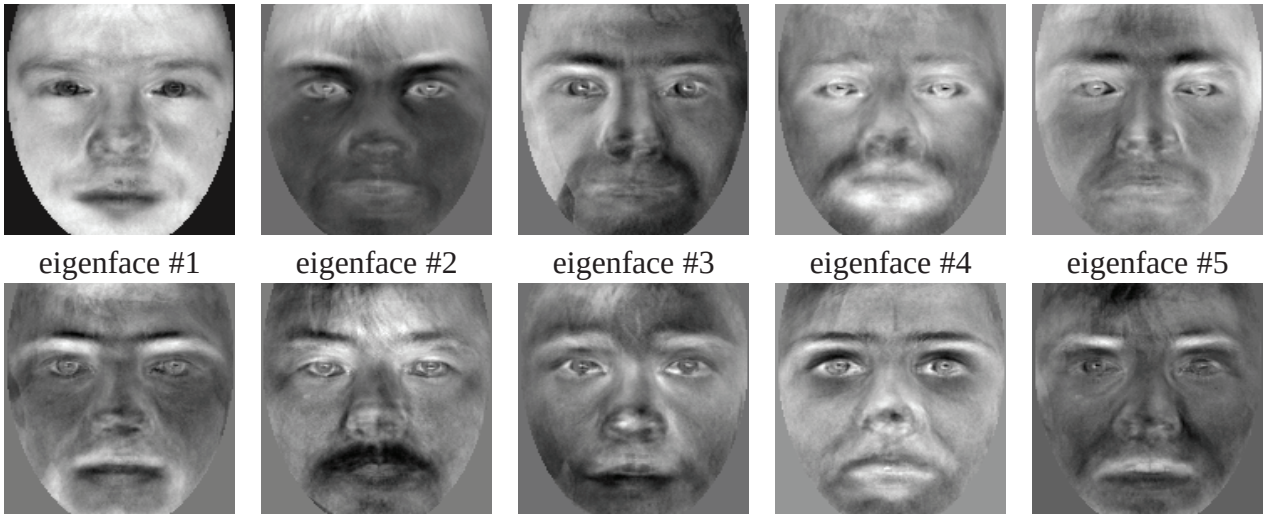


FIGURE 8. Ten eigenfaces obtained from face images in Figure 7.

Many web search engine techniques we see today follow the basic idea of web page *rankings*. The concept was first introduced by Larry Page and later developed by Sergey Brin [**44**]. The project started in 1995 and led to a functional prototype, named Google, in 1998. Shortly after, Page and Brin founded Google Inc., the company behind the Google search engine. While just one of the many factors which determine the ranking of Google search results, PageRank continues to provide the basis for all of Google's web search tools. Google uses an algorithm for ranking all the Web pages that agrees well with a common-sense quality measure.

The web (at some frozen point in time) consists of N web pages, most of them pointing to (having links to) other web pages. The importance of a page depends on the number of links to and from a page. In other words, a page which is pointed to very often would be considered important, while a

page with none or only very few other pages pointing to would be considered not important. How can we rank all web pages according to how important they are? Let's assume that all web pages are ordered in some fashion (such as lexicographic) so we can assign a number, such as i to any page. Let O_i denote the set of pages that i is linked to, the *outlinks*. The number of outlinks is denoted $N_i = |O_i|$. The *inlinks*, denoted I_i, are the pages that have an outlink to i. Note that a page is not supposed to link to itself.

In general, a page i can be considered as more important the more inlinks it has. However, a ranking system based only on the number of inlinks is easy to manipulate: when you design a Web page i that (e.g., for commercial reasons) you would like to be seen by as many users as possible, you could simply create a large number of (information-less and unimportant) pages that have outlines to i. To discourage this, one defines the rank of i so that if a highly ranked page j has an outlink to i, this adds to the importance of i. The ranking of a page i, r_i, should obey the following rules:

(1) The ranking r_i should grow with the number of page i's inlinks. (A page which is pointed to very often should deserve high ranking.)
(2) The ranking r_i should be weighted by the ranking of each of page i's inlinks, i.e., if all of those inlinks prove to be low-ranked, then their sheer number is mitigated by their low rankings. Conversely, if they are mostly high-ranked, then they should boost page i's ranking.
(3) Let page i have an inlink from page j. Then the more outlinks page j has, the less it should contribute to r_i. Namely, if page j has only one outlink, and it points to page i, then page i should be "honored" for such trust from page j. Conversely, if page j points to a large number of pages, page i among them, this does not give page i much pedigree.

Translating these into mathematics, we get

$$r_i = \sum_{j \in I_i} \frac{r_j}{N_j}. \tag{10}$$

This preliminary definition is recursive, so page ranks cannot be computed directly. Instead, a fixed-point iteration might be used. Guess an initial ranking vector $r^{(0)}$. Then iterate

$$r_i^{(k+1)} = \sum_{j \in I_i} \frac{r_j^{(k)}}{N_j}, \quad k = 0, 1, \ldots \tag{11}$$

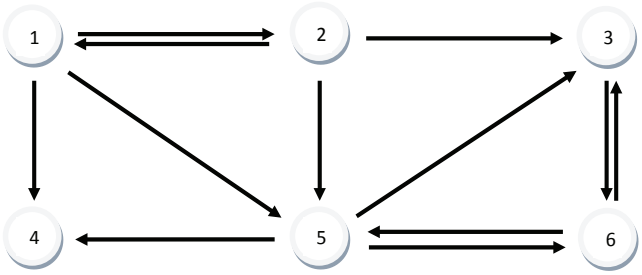
There are a few problems with such an iteration: if a page has no outlinks, then in the iteration process it accumulates rank only via its inlink, but this rank is never distributed further. Therefore it is not clear if the iteration converges. We will come back to this problem later.

More insight can be gained if we represent the connectivity structure of the web by an $n \times n$ matrix Q. Define

$$Q_{ij} = \begin{cases} \frac{1}{N_j} & \text{if there is a link from } j \text{ to } i, \\ 0 & \text{otherwise.} \end{cases}$$

This means that row i has nonzero elements in the positions that correspond to inlinks of i. Similarly, column j has nonzero elements equal to N_j in the positions that correspond to the outlinks of j, and, provided that the page has outlinks, the sum of all the elements in column j is equal to one.

EXAMPLE 5.4. The following link graph illustrates a set of Web pages with outlinks and inlinks:



The corresponding matrix becomes

$$Q = \begin{pmatrix} 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & \frac{1}{3} & 0 \end{pmatrix}.$$

Since page 4 has no outlinks, the corresponding column is equal to zero.

Obviously, the definition (10) is equivalent to the scalar product of row i and the vector r, which holds the rank of all pages. We can write the equation in matrix form,

$$\lambda r = Qr, \quad \lambda = 1, \tag{12}$$

i.e., r is an *eigenvector* of Q with *eigenvalue* $\lambda = 1$. It is now easily seen that the iteration (11) is equivalent to

$$r^{(k+1)} = Qr^{(k)}, \quad k = 0, 1, \ldots.$$

At this point it is not clear that pagerank is well defined, as we do not know if there exists an eigenvalue equal to 1. It turns out that all stochastic matrices have that property and thus no trouble arises. Interested readers are referred to [**17**] for further details. Here, we employ the *power method* to find the eigenvector, r. This method needs an initial guess for $r = [r_1, \cdots, r_n]^T$, and setting all $r_i = 1$ is not too bad for that. As the iterations converge, the solution is found. The entries of r are real, since they correspond to a real eigenvalue.
The vector r now contains the ranking — called page rank by Google — is very page. If Google retrieves a set of pages all containing a link to a term you are searching for, it presents them to you in decreasing order of the pages' ranking.

EXAMPLE 5.5. Let Q be given by

$$Q = \begin{pmatrix} 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{3} & 1 & 0 & 0 & \frac{1}{3} & 0 \end{pmatrix}.$$

The eigenvector corresponding to the eigenvalue 1 is given by

$$r^T = [0.306, 0.548, 0.278, 0.148, 0.139, 0.697].$$

Thus page 6 has the highest ranking and page 5 has the lowest ranking.

Some final remarks: Google (the company) solves Equation (12) about once a month[1]. Roughly speaking, they do not use $r_i = 1$ as the initial guess, but instead use last month's solution. In the real world, $n \approx 10^{10}$, meaning that Q has 10^{20} elements. This is world's largest matrix to be used ever. Luckily, it contains mostly zeros and thus is extremely sparse. Without taking advantage of that, Google (and other search engines) could not function.

6. The Singular Value Decomposition

The singular value decomposition (SVD) extends the spectral theorem for rectangular matrices. We shall see in Chapter 2 that it also provides the necessary mathematics for understanding an important class of optimal dimensionality-reducing mappings. We shall begin with a statement of the decomposition theorem, and it the course of proving it we will establish several important facts concerning the SVD.

Construction of the Decomposition

We begin with a statement of the decomposition theorem followed by a constructive proof.

THEOREM 6.1. *Singular Value Decomposition (SVD). Let A be a real $m \times n$ matrix and $d = \min\{m,n\}$. There exist orthogonal matrices U and V such that*

$$A = U\Sigma V^T, \tag{13}$$

where $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$, and $\Sigma = diag(\sigma^{(1)},\ldots,\sigma^{(d)}) \in \mathbb{R}^{m\times n}$.

If $m > n$, then the diagonal matrix Σ has the form

$$\Sigma = \begin{pmatrix} \sigma^{(1)} & & 0 \\ & \ddots & \\ 0 & & \sigma^{(n)} \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{pmatrix},$$

while if $m < n$, then

$$\Sigma = \begin{pmatrix} \sigma^{(1)} & & 0 & 0 & \cdots & 0 \\ & \ddots & & \vdots & & \vdots \\ 0 & & \sigma^{(m)} & 0 & & 0 \end{pmatrix}.$$

Furthermore, the entries of Σ are ordered according to

$$\sigma^{(1)} \geq \sigma^{(2)} \geq \cdots \geq \sigma^{(1)} \geq 0.$$

The case for A being a complex matrix is analogous and is treated in most linear algebra texts. For simplicity, we now assume that $m \geq n$ and that A has full rank, i.e., rank n. The rank-deficient case follows immediately from these deliberations. To establish the decomposition given by Equation (13), we first rewrite it as

$$AV = U\Sigma.$$

The ith column of this relationship is

$$A\mathbf{v}^{(i)} = \sigma^{(i)}\mathbf{u}^{(i)}, \tag{14}$$

where $i = 1,\ldots,n$. Alternatively,

$$A^T U = V\Sigma^T.$$

[1] the actual equation is a bit trickier, and is omitted here

The ith column of this relationship is

$$A^T\mathbf{u}^{(i)} = \sigma^{(i)}\mathbf{u}^{(i)}, \tag{15}$$

where again $i = 1,\ldots,n$. For our constructive proof we must establish that, for any given matrix A, solutions to equations (14) and (15) exist. In fact, as the following propositions demonstrate, the solutions to these equations occur in triples $\left\{\sigma^{(i)},\mathbf{u}^{(i)},\mathbf{v}^{(i)}\right\}$, the components of which we refer to as the *singular values* $\sigma^{(i)}$, the *left-singular* vectors $\mathbf{u}^{(i)}$ and the *right-singular* vectors $\mathbf{v}^{(i)}$.

PROPOSITION 6.1. *The n left-singular vectors of A exist and are given by the n eigenvectors of AA^T corresponding to nonzero eigenvalues. These eigenvalues correspond to the singular values squared.*

We have

$$\begin{aligned} A^T\mathbf{u} &= \sigma\mathbf{v} \\ AA^T\mathbf{u} &= \sigma A\mathbf{v} \\ AA^T\mathbf{u} &= \sigma^2\mathbf{u} \end{aligned}$$

and hence $\sqrt{\lambda} = \sigma$. Existence of the eigenvectors follows since AA^T is a symmetric matrix. Note that the size of this eigenvector problem is $m \times m$. An analogous proposition is true for the right-singular vectors.

PROPOSITION 6.2. *The n right-singular vectors of A exist and are given by the n eigenvectors of A^TA, and the associated eigenvalues correspond to the singular values squared.*

We have

$$\begin{aligned} A\mathbf{v} &= \sigma\mathbf{u} \\ A^TA\mathbf{v} &= \sigma A^T\mathbf{u} \\ A^TA\mathbf{v} &= \sigma^2\mathbf{v}. \end{aligned}$$

Again, existence of the eigenvectors follows, since A^TA is a symmetric matrix. Note that this is an $n \times n$ eigenvector problem. However, since we are assuming $n \leq m$, all of the eigenvectors in this instance are also singular vectors.

We are now in a position to provide a constructive proof of the SVD based on the existence of the left- and right-singular vectors. Again, we assume $m \geq n$, so there are n singular-vector triplets $\left\{\sigma^{(i)},\mathbf{u}^{(i)},\mathbf{v}^{(i)}\right\}$. First we will show that

$$\underbrace{A}_{m\times n}\underbrace{V}_{n\times n} = \underbrace{\hat{U}}_{m\times n}\underbrace{\hat{\Sigma}}_{n\times n}.$$

Thus,

$$\begin{aligned} AV &= A[\mathbf{v}^{(1)}|\cdots|\mathbf{v}^{(n)}] \\ &= [A\mathbf{v}^{(1)}|\cdots|A\mathbf{v}^{(n)}] \\ &= [\sigma^{(1)}\mathbf{u}^{(1)}|\cdots|\sigma^{(n)}\mathbf{u}^{(n)}] \\ &= \hat{U}\hat{\Sigma}. \end{aligned}$$

It follows that

$$A = \hat{U}\hat{\Sigma}V^T, \tag{16}$$

where $\hat{U} \in \mathbb{R}^{m\times n}$ (i.e., it is U with the last $m-n$ columns deleted), and $\hat{\Sigma} \in \mathbb{R}^{n\times n}$. This version of the SVD is referred to as the *thin* SVD, or the *reduced* SVD. The full SVD follows by including the

eigenvectors $\left\{\mathbf{u}^{(n+1)},\ldots,\mathbf{u}^{(m)}\right\}$. In this case we write

$$AV = [\mathbf{u}^{(1)}|\cdots|\mathbf{u}^{(n)}|\mathbf{u}^{(n+1)}|\cdots|\mathbf{u}^{(m)}]\begin{pmatrix}\sigma^{(1)} & & 0\\ & \ddots & \\ 0 & & \sigma^{(n)}\\ 0 & \cdots & 0\\ \vdots & & \vdots\\ 0 & \cdots & 0\end{pmatrix}. \tag{17}$$

This concludes our constructive proof of the SVD.

Corollaries of the Decomposition

The SVD can be used to establish properties of matrices as well as representations for them.

PROPOSITION 6.3. *If rank$A = r$, then there are r nonzero singular values, i.e.,*

$$\sigma^{(1)} > \cdots \sigma^{(r)} > \sigma^{(r+1)} = 0.$$

PROOF. This follows directly from the rewriting of Equation (13) as

$$A = \sum_{j=1}^{r} \sigma^{(j)}\mathbf{u}^{(j)}\mathbf{v}^{(j)T}. \tag{18}$$

To derive this expression define the matrices $\Sigma_1 = diag(\sigma^{(1)},0,\ldots,0)$, $\Sigma_2 = diag(0,\sigma^{(2)},0,\ldots,0)$, and so on. If follows that

$$\begin{aligned} A &= U\Sigma V^T \\ &= U(\Sigma_1+\Sigma_2+\Sigma_r)V^T \\ &= U\Sigma_1 V^T + U\Sigma_2 V^T + U\Sigma_r V^T \\ &= \sigma^{(1)}\mathbf{u}^{(1)}\mathbf{v}^{(1)T} + \sigma^{(2)}\mathbf{u}^{(2)}\mathbf{v}^{(2)T} + \cdots + \sigma^{(r)}\mathbf{u}^{(r)}\mathbf{v}^{(r)T}. \end{aligned}$$

□

Note that SVD decomposes, or reduces, the matrix into a sum of r rank-one matrices. Later we shall see that it does this optimally well. Proposition 6.3 may be established in another way as well. For example, the rank of a diagonal matrix is the number of nonzero diagonal elements. Furthermore, orthogonal transformations do not change the number of vectors that make up a basis. In view of $A = U\Sigma V^T$, it follows that if rank$\Sigma = r$, then rank$A = r$.
It is also true that the column covariance matrix AA^T and the row covariance matrix A^TA have the same rank. If A is a data matrix, this suggests that the arrangement of the data as column or row vector does not affect the number of relevant terms in the SVD.

PROPOSITION 6.4. *Let A be an $m \times n$ matrix of rank r. Then*

$$r = rank(AA^T) = rank(A^TA).$$

This follows directly from the correspondence of the nonzero singular values with the nonzero eigenvalues: they are exactly the same in number.
The SVD provides bases for the fundamental subspaces. Here we allow the matrix A to be *rank-deficient*, i.e., of rank $k \leq d = \min\{m,n\}$.

PROPOSITION 6.5. *Let A be an $m \times n$ matrix of rank r. Then:*

(1) *The r left-singular vectors associated with the r nonzero singular values of A, i.e., $\left\{\mathbf{u}^{(1)},\ldots,\mathbf{u}^{(r)}\right\}$, form a basis for $\mathscr{R}(A)$.*

(2) *The r right-singular vectors associated with the r nonzero singular values of A, i.e.,* $\left\{\mathbf{v}^{(1)},\ldots,\mathbf{v}^{(r)}\right\}$, *form a basis for* $\mathscr{R}(A^T)$.
(3) *The $m-r$ eigenvectors of AA^T associated with the $m-r$ zero eigenvalues, i.e.,* $\left\{\mathbf{u}^{(r+1)},\ldots,\mathbf{u}^{(m)}\right\}$, *form a basis for* $\mathscr{N}(A^T)$.
(4) *The $n-r$ eigenvectors of A^TA associated with the $n-r$ zero eigenvalues, i.e.,* $\left\{\mathbf{v}^{(r+1)},\ldots,\mathbf{u}^{(n)}\right\}$, *form a basis for* $\mathscr{N}(A)$.

PROOF. Item 1 follows from Equation (18). Item 3 follows from the fact that the vectors $\mathbf{u}^{(i)}$ sit in $\mathbb{R}^m$ and the full set forms a basis for this space (AA^T is a symmetric matrix). Thus, the $m-r$ must form a basis for $\mathscr{N}(A^T)$, since $\mathbb{R}^m=\mathscr{N}(A^T)\dot{\oplus}\mathscr{R}(A)$. Items 2 and 4 are true for similar reasons. □

EXAMPLE 6.1. Compute the SVD of the data matrix

$$A=\begin{pmatrix}1&1\\0&1\\1&0\end{pmatrix}.$$

First, we compute the right-singular vectors and singular values of A. These are exactly the eigenvectors and the square root of the eigenvalues of

$$A^TA=\begin{pmatrix}2&1\\1&2\end{pmatrix},$$

which has the characteristic equation $\rho(\lambda)=(\lambda-1)(\lambda-3)=0$. Hence $\sigma^{(1)}=\sqrt{\lambda^{(1)}}=\sqrt{3}$ has the right-singular vector

$$\mathbf{v}^{(1)}=\frac{1}{\sqrt{2}}(1,1)^T,$$

and $\sigma^{(2)}=\sqrt{\lambda^{(2)}}=1$ has the right-singular vector

$$\mathbf{v}^{(2)}=\frac{1}{\sqrt{2}}(-1,1)^T.$$

So, the matrix of right-singular vectors is

$$V=\begin{pmatrix}\frac{1}{\sqrt{2}}&-\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}&\frac{1}{\sqrt{2}}\end{pmatrix}.$$

The two left-singular vectors are given by the eigenvectors of

$$AA^T=\begin{pmatrix}2&1&1\\1&1&0\\1&0&1\end{pmatrix}$$

corresponding to the two largest eigenvalues. The characteristic equation for AA^T is given by $\rho(\lambda)=\lambda(\lambda-1)(\lambda-3)=0$. Now $\sigma^{(1)}$ (seen to be the same as above, as expected) has the left-singular vector

$$\mathbf{u}^{(1)}=\frac{1}{\sqrt{6}}(2,1,1)^T,$$

and $\sigma^{(2)}=1$ has the left-singular vector

$$\mathbf{u}^{(2)}=\frac{1}{\sqrt{2}}(0,1,-1)^T.$$

Lastly, we have the eigenvalue $\lambda^{(3)} = 0$ which corresponds to the eigenvector

$$\mathbf{u}^{(3)} = \frac{1}{\sqrt{3}}(1,-1,-1)^T.$$

This last vector completes the basis for $\mathbb{R}^3$:

$$U = \begin{pmatrix} \frac{2}{\sqrt{6}} & 0 & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}} \end{pmatrix}.$$

So the *full* SVD is given by

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} \frac{2}{\sqrt{6}} & 0 & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}} \end{pmatrix} \begin{pmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix},$$

and the *reduced* SVD is given by

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} \frac{2}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \sqrt{3} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}.$$

Reduction and Compression of Matrices

The SVD not only provides an efficient means to represent a matrix without loss, it also provides an optimal method for approximating a matrix by another matrix of reduced rank. Define a rank-$k < r$ approximation to the matrix A as

$$A_k = \sum_{i=1}^{k} \sigma^{(i)} \mathbf{u}^{(i)} \mathbf{v}^{(i)T}. \tag{19}$$

PROPOSITION 6.6. *The error of a rank-k approximation provided by A_k is given by $\sigma^{(k+1)}$, i.e.,*

$$||A - A_k||_2 = \sigma^{(k+1)}.$$

PROOF.

$$\begin{aligned} A - A_k &= \sum_{i=1}^{r} \sigma^{(i)} \mathbf{u}^{(i)} \mathbf{v}^{(i)T} - \sum_{i=1}^{k} \sigma^{(i)} \mathbf{u}^{(i)} \mathbf{v}^{(i)T} \\ &= \sum_{i=k+1}^{r} \sigma^{(i)} \mathbf{u}^{(i)} \mathbf{v}^{(i)T} \\ &= U \Sigma' V^T, \end{aligned}$$

where

$$\Sigma' = \begin{pmatrix} 0 & & & & & & & \\ & \ddots & & & & & & \\ & & \sigma^{(k+1)} & & & & & \\ & & & \ddots & & & & \\ & & & & \sigma^{(r)} & & & \\ & & & & & 0 & & \\ & & & & & & \ddots & \\ & & & & & & & 0 \end{pmatrix}.$$

It follows that

$$\begin{aligned} ||A-A_k||_2 &= ||U\Sigma' V^T||_2 \\ &= ||\Sigma'||_2 \\ &= \sigma^{(k+1)}, \end{aligned}$$

since 2-norms are invariant under multiplication by orthogonal matrices. □

If we know the correct rank of A, e.g., by inspecting the singular values, then we can *remove the noise* and *compress the data* by approximating A with a matrix of the correct rank. One way to do this is to truncate the singular value expansion:

THEOREM 6.2. *The SVD provides the best reduced-rank approximation to a given matrix A, i.e., any matrix, say B, that is not the rank-k SVD approximation has greater error. Namely, if*

$$A_k = \sum_{i=1}^{k} \sigma^{(i)} \mathbf{u}^{(i)} \mathbf{v}^{(i)T} \quad (1 \le k \le r)$$

then

$$A_k = \underset{rank(B)=k}{\arg\min} \; ||A-B||_2 \quad \textit{and} \quad A_k = \underset{rank(B)=k}{\arg\min} \; ||A-B||_F.$$

PROOF. See [**22**]. □

EXAMPLE 6.2. A rank-one approximation to the matrix

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

is given by

$$A \approx \begin{pmatrix} 1 & 1 \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

This approximation is calculated by

$$A \approx A_1 = \sigma^{(1)} \mathbf{u}^{(1)} \mathbf{v}^{(1)T}.$$

EXAMPLE 6.3. In Figure 9 we see a raw digital gray-level image of size 480×500. Treating it as a matrix, we may apply the SVD directly to the image. The effect of retaining 10, 50, and 170 terms in the expansion is shown in the subimages. The images A_{10}, A_{50}, and A_{170} filter out the smallest eigenvalues. The missing, or discarded part of the image (orthogonal complement) is shown for 10 and 170 terms in Figure 10. The images $A - A_{10}$ and $A - A_{170}$ filter out the largest eigenvalues.
Consider the compression achieved for a k-term expansion. The original image has size $m \times n = 240000$ bytes. The reconstructions require that we store k vectors of the form $\sigma^{(k)}\mathbf{u}^{(k)}$ as well as k vectors of the form $\mathbf{v}^{(k)}$. After rounding, the compressed images required $(m+n)k = 980k$ bytes. For $k = 10$ approximately 3% of the size of the original raw data is retained while for $k = 50$ and $k = 170$, approximately 15% and 51% are retained, respectively.

Geometric and Numerical Properties of SVD

Numerical properties of the SVD provide ways to detect the numerical dimensionality as well as the geometry of data sets. For example, for a data set X whose column vectors represent distinct data points, the singular values represent the distribution of the source data's energy among each of the

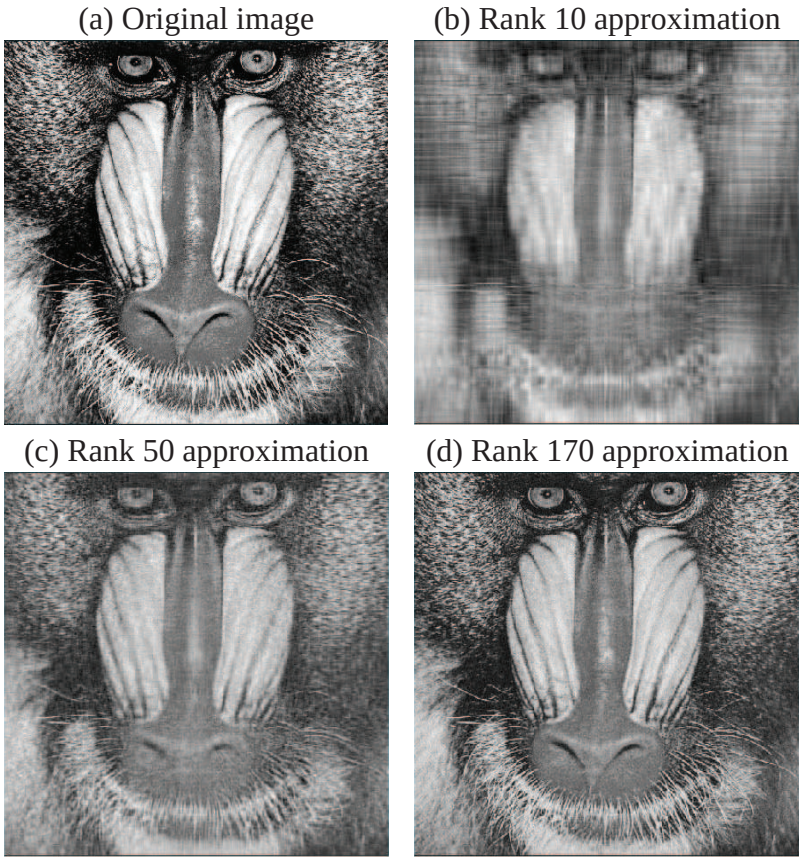


FIGURE 9. The SVD of a matrix with entries of a digital image. The other images are rank-10,-50 and -170 approximations to the original image, i.e., A_{10}, A_{50}, and A_{170}. The relative errors are 0, 0.0305, 0.0551, and 0.0126 for A, A_{10}, A_{50}, and A_{170}, respectively.

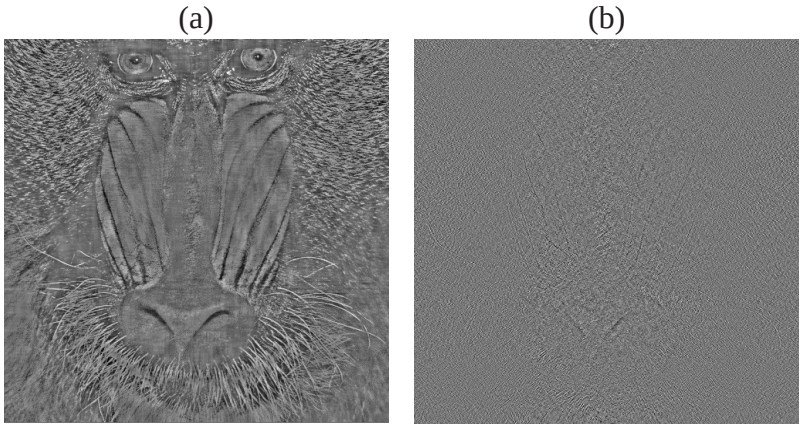


FIGURE 10. The SVD of a matrix with entries of a digital image. Top: The truncated image $A - A_{10}$. Bottom: The truncated image $A - A_{170}$.

singular vectors, where the singular vectors form a basis for the data. *Cumulative energy* of the first t

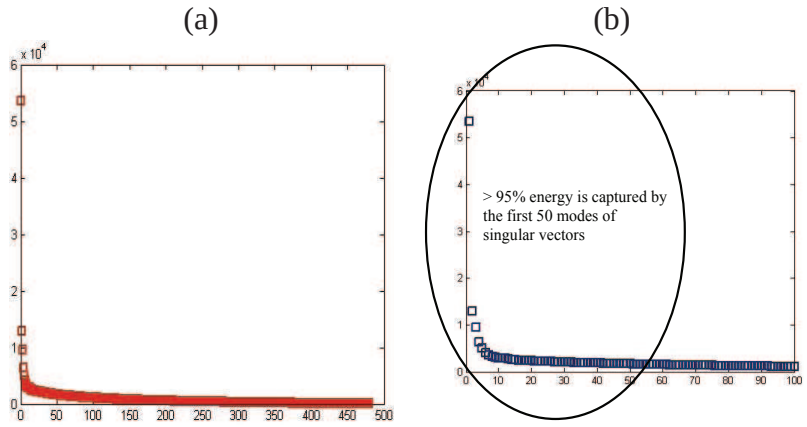


FIGURE 11. Singular value distribution for the image in Figure 9 (a).

$(1 \leq t \leq r)$ singular values is given by

$$\frac{\sum_{i=1}^{t} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}.$$

For a low rank matrix, the *number of large singular values* is often referred to as the *numerical rank* of that matrix.

Consider $A \in \mathbb{R}^{m\times n}$ as a map from $\mathbb{R}^n$ to $\mathbb{R}^m$. Let $\mathbf{S}^{n-1}$ be the unit sphere in $\mathbb{R}^n$. If the axes of $\mathbb{R}^n$ are given by the n orthogonal singular vectors $v_1, v_2, \ldots, v_n$, then A maps $\mathbf{S}^{n-1}$ into an ellipsoid in $\mathbb{R}^m$ with $r = rank(A)$ axes. The length of the axes are σ_i, $1 \leq i \leq r$ and the direction of the axes are given by u_i (or Av_i). Because this geometric property and the matrix property mentioned above, SVD is used a lot to reduce the dimensionality and detect the geometry of large data sets.

EXAMPLE 6.4. Treating the digital image in Figure 9 (a) as a matrix and applying the SVD directly to the image, we get the singular value distribution in Figure 11 (a). Using the following MATLAB code, we find that the first 50 modes of the singular vectors capture over 95% of the energy (Figure 11 (b)).

```
load mandrill.mat
[U,S,V] = svd(X,0); % X is the image matrix
D = diag(S).^2;
cum_energy = 100.*cumsum(D)./sum(D);
I = find(cum_energy > 95);
```

This suggests that the approximation A_{50} gives an error rate less than 5%.

EXAMPLE 6.5. Let

$$X = \begin{pmatrix} 15 & 113 & \cdots & 219 \\ 9 & 12 & \cdots & 52 \\ 34 & 129 & \cdots & 30 \\ \vdots & \vdots & \ddots & \vdots \\ 76 & 78 & \cdots & 198 \end{pmatrix}$$

be a size m-by-n matrix with singular value decomposition $X = USV^T$. Then $U(:,1:2)^T X = S(1:2,1:2)V(:,1:2)^T = \hat{S}\hat{V}^T$ is a matrix of size 2-by-n. $\hat{S}\hat{V}^T$ is the original data projected onto the first two dimensions. Now, imagine each column vector in X represent a data point in $\mathbb{R}^m$ and belongs to one of the three distinct classes (black, green, red). By projecting these column vectors onto the first two dimensions, we can visualize the neighborhood relationships among the three classes

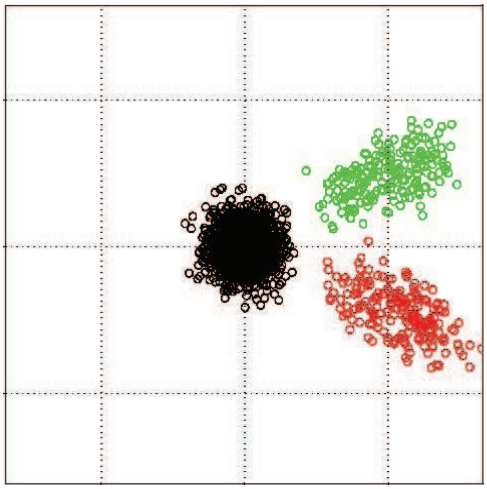


FIGURE 12. A data matrix projected onto the first 2 dimensions.

(see Figure 12). Notice that this method works relatively well for data sets that are intrinsically low dimensional, in particular, 2-dimensional.

Computation of the SVD

We have shown that the singular values may be computed by forming the covariance matrices AA^T or A^TA and computing their eigenvalues. While this approach is suitable for many applications, it is numerically unstable. An alternative to forming the covariance matrices is to calculate the left- and right-singular vectors directly from the system

$$\begin{pmatrix} \mathbf{0} & A \\ A^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u}^{(i)} \\ \mathbf{v}^{(i)} \end{pmatrix} = \sigma^{(i)} \begin{pmatrix} \mathbf{u}^{(i)} \\ \mathbf{v}^{(i)} \end{pmatrix}. \tag{20}$$

If the matrix A is perturbed by a small amount then it can be shown that the perturbed singular values $\tilde{\sigma}^i$ satisfy

$$|\tilde{\sigma}^{(i)} - \sigma^{(i)}| = O(\varepsilon||A||)$$

when computed using Equation (20); ε is the machine precision. On the other hand, if the singular values are obtained by first computing the eigenvalues of the smaller of the two matrices AA^T or A^TA, then

$$|\tilde{\sigma}^{(i)} - \sigma^{(i)}| = O(\varepsilon||A||^2/\sigma^{(i)}).$$

This squaring of the norm followed by the division by the singular value becomes significant especially for the smaller singular values.

EXAMPLE 6.6. To demonstrate the mechanics of the system calculation we revisit Example 6.1. Now we have to compute the eigenvectors and eigenvalues of

$$\begin{pmatrix} \mathbf{0} & A \\ A^T & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The eigenvectors are

$$\begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 & \frac{1}{\sqrt{3}} \\ \frac{1}{2\sqrt{3}} & \frac{1}{2\sqrt{3}} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{\sqrt{3}} \\ \frac{1}{2\sqrt{3}} & \frac{1}{2\sqrt{3}} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{\sqrt{3}} \\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & 0 \end{pmatrix},$$

where columns are ordered from left to right via the singular values $\sigma = -\sqrt{3},\ \sqrt{3},\ -1,\ 1, 0$. The eigenvectors now contain the right- and left-singular vectors as components, i.e., each column is of the form $(\mathbf{u}^{(i)}, \mathbf{v}^{(i)})$. If is interesting to note that all the singular values $\pm\sigma^{(i)}$ are present and that the associated eigenvectors $(\mathbf{u}^{(i)}, \pm\mathbf{v}^{(i)})$.

Application: Principal Component Analysis

[**17**] The approximation properties of the SVD can be used to elucidate the equivalence between the SVD and *Principal Component Analysis* (PCA). Assume that $X \in \mathbb{R}^{m\times n}$ is a data matrix, where each column is an observation of a real-valued random vector with mean zero. The matrix is assumed to be centered, i.e., the mean of each column in equal to zero. Let the SVD of X be $X = U\Sigma V^T$. The right-singular vectors $\mathbf{v}^{(i)}$ are called *principal components directions* of X. The vector

$$z^{(1)} = X\mathbf{v}^{(1)} = \sigma^{(1)}\mathbf{u}^{(1)}$$

has the largest sample variance among all normalized linear combinations of the columns of X:

$$\mathrm{Var}\,(z^{(1)}) = \frac{(\sigma^{(1)})^2}{m}.$$

Find the vector of maximal variance is equivalent, using linear algebra terminology, to maximizing the Rayleih quotient:

$$\mathbf{v}^{(1)} = \max_{\mathbf{v}\neq 0} \frac{\mathbf{v}^T X^T X\mathbf{v}}{\mathbf{v}^T\mathbf{v}}.$$

The normalized variable $\mathbf{u}^{(1)} = \frac{1}{\sigma^{(1)}} X\mathbf{v}^{(1)}$ is called the *normalized first principal component* of X. Having determined the vector of largest sample variance, we usually want to go on and find the vector of second largest sample variance that is *orthogonal* to the first. The is done by computing the vector of largest sample variance of the *deflated data matrix* $X - \sigma^{(1)}\mathbf{u}^{(1)}\mathbf{v}^{(1)}$. Continuing this process we can determine all the principal components in order, i.e., we compute the singular vectors. In the general step of the procedure, the subsequent principal component is defined as the vector of maximal variance subject to the constraint that it is orthogonal to the previous ones.

EXAMPLE 6.7. The concept of PCA is illustrated in Figure 13. 35 data points were generated and collected in a data matrix $X \in \mathbb{R}^{3\times 35}$. The data points and the three principal components are illustrated in the middle plot of Figure 13. From this, we can see that the data set is approximately 2-dimensional, since the third dimension has relatively smaller variance. Having this realization gives us a way to represent data points using two singular modes only, as shown in the right plot of Figure 13.

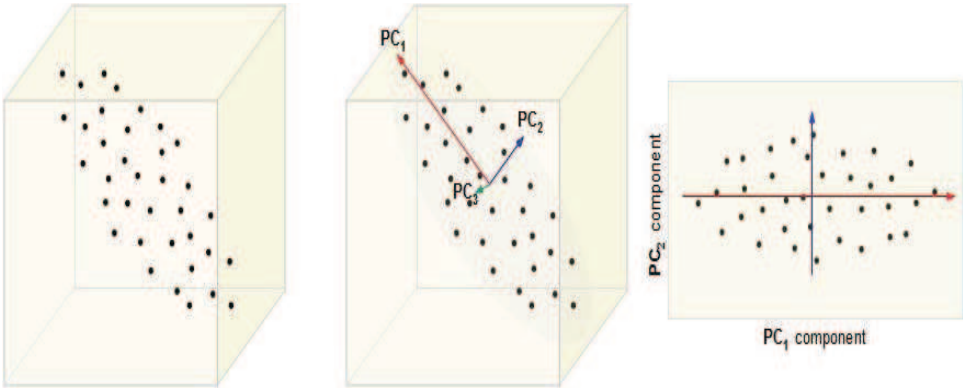


FIGURE 13. The three principal components of a data set.

CHAPTER 2

Optimal Orthogonal Pattern Representations

1. Introduction

In this chapter, we develop the idea of an optimal dimensionality-reducing mapping. Specially, we consider the optimization problem over the class of orthogonal transformations. As orthogonal transformations are linear, the optimization problem amounts to finding a best orthonormal basis, which we write as the columns of the matrix Φ, for the change of coordinates. Thus, given a data point $\mathbf{x}$, we seek the orthogonal matrix Φ (i.e., $\Phi^T\Phi = I$) such that the transformation

$$\mathbf{a} = \Phi^T\mathbf{x}$$

is optimal in a sense to be made precise. Geometrically, the goal of the optimization is to rotate the ambient coordinates of the data to reveal the subspace in which the data resides.

The main result is the well-known Karhunen-Loéve (KL) expansion. Its importance in pattern analysis is substantiated by the number of aliases under which the technique is known, which include the principal component analysis (PCA) [**28, 29**], empirical orthogonal functions (EOFs) [**38**]. It is also closely related to the well-known singular value decomposition (SVD) [**27**].

We will start this chapter by defining by defining *optimal bases* in Section 2 followed by the construction of optimal bases with KL in Section 3. The resulting approach will be referred to as the *direct method*. Section 4 presents the most important and widely used properties of KL expansion. The direct method for implementing the KL transformation cannot be applied to elements of high-dimensional vector spaces — say, dimensions above 1000. In this case, an alternative approach is used, referred to as the *snapshot method* in view of its natural application to digital images. In Section 5, we present this technique and apply it to a real problem. Section 6 reexamines the KL transformation from the perspective of the SVD described earlier. The SVD permits a deeper understanding of the relationship between the direct and snapshot methods for computing the eigenvectors.

2. What is an Optimal Basis?

The purpose of this section is to mathematically characterize the notion of an *optimal basis*. In practice, an optimal basis for V will extract, or package, the salient features and information in the data. Ideally, this setting will enhance our ability to study the data in terms of a significantly reduced number of expansion coefficients; only a small number of the entries of $\mathbf{a} = \Phi^T\mathbf{x}$ will be of interest.

Consider an N-dimensional inner product space V equipped with an ordered orthonormal basis $\mathscr{B} = \left\{\phi^{(1)}, \ldots, \phi^{(N)}\right\}$. Every point in V may be expressed without error in terms of the basis vector as

$$\mathbf{x}^{(\mu)} = a_1^{(\mu)}\phi^{(1)} + \cdots + a_N^{(\mu)}\phi^{(N)}, \tag{21}$$

where $a_i^{(\mu)} = (\mathbf{x}^{(\mu)}, \phi^{(i)})$. This inner product will generally be the usual Euclidean inner, or dot, product.

Given a data set, how should a basis $\mathscr{B}$ be constructed such that the truncation of the full N-term expansion in Equation 21 to a D-term expansion

$$\mathbf{x}_D^{(\mu)} = a_1^{(\mu)}\phi^{(1)} + \cdots + a_D^{(\mu)}\phi^{(D)} \tag{22}$$

will produce a minimum error? Since $\mathbf{x}_D^{(\mu)}$ is an approximation to $\mathbf{x}^{(\mu)}$, we write

$$\mathbf{x}^{(\mu)} \approx \mathbf{x}_D^{(\mu)},$$

and the accuracy of this expression will be at the center of our discussion.

Typically we are interested in approximating a collection, or ensemble, of P patterns $\left\{\mathbf{x}^{(\mu)}\right\}$ rather than a single pattern. The error vector for each pattern, $\varepsilon_D^{(\mu)}$, is the difference between the exact point $\mathbf{x}^{(\mu)}$ and the truncated expansion $\mathbf{x}_D^{(\mu)}$, i.e.,

$$\varepsilon_D^{(\mu)} = \mathbf{x}^{(\mu)} - \mathbf{x}_D^{(\mu)}.$$

A scalar measure of the error is then simply $\varepsilon = ||\varepsilon_D^{(\mu)}||$, where the norm is induced by the (usually Euclidean) inner product. As shown below, a closed-form formula for the best basis may be obtained for the squared error

$$\varepsilon_{\text{se}} = ||\varepsilon_D^{(\mu)}||^2.$$

Thus, our criterion for an optimal basis is that it shall minimize the mean squared error over the set of all orthonormal bases. The mean, or ensemble average, of a set of vectors $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\cdots$, $\mathbf{x}^{(P)}$ is defined as

$$\langle \mathbf{x} \rangle = \frac{1}{P} \sum_{\mu=1}^{P} \mathbf{x}^{(\mu)}. \tag{23}$$

It should be noted that the above addition is applied componentwise; it is standard practice to omit the pattern index μ for terms within the angled brackets when writing an ensemble average. It is customary to mean-subtract each pattern in the ensemble. This is geometrically equivalent to moving the center of the coordinate system of the patterns to the ensemble average (or centroid) of all the data set. Thus we define a new ensemble

$$\tilde{\mathbf{x}}^{(\mu)} = \mathbf{x}^{(\mu)} - \langle \mathbf{x} \rangle .$$

DEFINITION 2.1. The quantity $\tilde{\mathbf{x}}^{(\mu)}$ is called the *fluctuating field*, or *caricature*, of the pattern $\mathbf{x}^{(\mu)}$.

In what follows we will assume, unless otherwise stated, that all the pattern vectors have been mean-subtracted, and we drop the tild'e for convenience.

DEFINITION 2.2. The mean squared truncation error ε_{mse} of a D-term approximation to an ensemble of vectors is defined as

$$\varepsilon_{\text{mse}} = \left\langle ||\mathbf{x} - \mathbf{x}_D||^2 \right\rangle = \left\langle ||\varepsilon_D^{\mu}||^2 \right\rangle .$$

The Subspace Approach Let's reexamine our previous remarks in terms of subspaces. We may decompose $\mathbf{x}^{(\mu)}$ in two pieces as

$$\mathbf{x}^{(\mu)} = \sum_{i=1}^{D} a_i^{(\mu)} \phi^{(i)} + \sum_{i=D+1}^{N} a_i^{(\mu)} \phi^{(i)} \tag{24}$$

$$= \mathbf{x}_D^{(\mu)} + \varepsilon_D^{(\mu)}. \tag{25}$$

The basis for this vector expansion may be used to define the subspaces $W_D = \text{span}\left\{\phi^{(1)}, \ldots, \phi^{(D)}\right\}$ and $W_D^{\perp} = \text{span}\left\{\phi^{(D+1)}, \ldots, \phi^{(N)}\right\}$. These subspaces split V into two pieces as a direct sum

$$V = W_D \oplus W_D^{\perp},$$

where the truncated representations $\mathbf{x}_D^{(\mu)}$ lie in the orthogonal subspace W_D and the error vectors $\varepsilon_D^{(\mu)}$ lie in $W_D^{\perp}$.

While the orthogonal projection theorem states that the orthogonal expansion provides a "best approximation", it says nothing about how to find W_D such that we obtain the best possible approximation. Thus our task is to determine a single basis that provides the *optimal subspace* W_D for any level of truncation $1 \leq D < N$. Again, optimal here means that a well-defined error should be minimized over all possible D-dimensional subspaces.

3. Construction of the Optimal Basis

We have characterized an optimal orthogonal basis as one that minimizes the mean squared truncation error of the expansion. Equivalently, one may seek an orthonormal basis that maximizes the mean squared projection of the data. Before considering the equivalence of these optimality criteria further, we proceed with the sequential construction of a basis that maximizes the mean squared projection of the data. The basic algorithm for constructing such as basis is as follows:

- Find the best one-dimensional subspace W_1.
- Find the best one-dimensional subspace W_2 with the restriction that it must be orthogonal to W_1.
- Find the best one-dimensional subspace W_i with the restriction that $W_i \perp W_j$ for all $j < i$.

Now we define the best first eigenvector $\phi^{(1)}$ to be the one that maximizes the mean squared projection of all patterns in the ensemble onto itself. Namely, find

$$\max_{\phi^{(1)}} \left\langle (\phi^{(1)}, \mathbf{x})^2 \right\rangle$$

subject to

$$\left(\phi^{(1)}, \phi^{(1)}\right) = 1.$$

The normalization of $\phi^{(1)}$ to be of unit length is required as otherwise simply multiplying this vector by a constant would increase the projection.
This constrained optimization problem may be solved via the technique of Lagrange multipliers. To apply this method we define the *Lagrangian* g_1 to be

$$g_1(\lambda_1, \phi^{(1)}) = \left\langle (\phi^{(1)}, \mathbf{x})^2 \right\rangle - \lambda_1[(\phi^{(1)}, \phi^{(1)}) - 1].$$

The necessary condition for a maximum (or a minimum) are then given by

$$\begin{aligned} \frac{\partial g_1}{\partial \phi^{(1)}} &= 0 \\ \frac{\partial g_1}{\partial \lambda_1} &= 0 \end{aligned}$$

where the last equation is exactly the constraint.[1] Noting that

$$\begin{aligned} (\mathbf{x}, \phi)^2 &= (\phi, \mathbf{x})(\mathbf{x}, \phi) \\ &= (\phi^T x)(x^T \phi) = \phi^T (\mathbf{x}\mathbf{x}^T \phi) \\ &= (\phi, \mathbf{x}\mathbf{x}^T \phi) \end{aligned}$$

[1] In what follows we employ the notation

$$\frac{\partial F(\phi)}{\partial \phi} = \left(\frac{\partial F}{\partial \phi_1}, \cdots, \frac{\partial F}{\partial \phi_N} \right)^T.$$

If it left as an exercise for the reader to show that $\partial(\phi, \phi) = 2\phi$ and that, if C is a symmetric matrix, $\partial(\phi, C\phi) = 2C\phi$.

and defining the symmetric matrix $C = \langle \mathbf{x}\mathbf{x}^T \rangle$, the Lagrangian may be rewritten in the more useful form

$$g_1(\lambda_1, \phi^{(1)}) = (\phi^{(1)}, C\phi^{(1)}) - \lambda_1[(\phi^{(1)}, \phi^{(1)}) - 1].$$

Differentiating with respect to $\phi^{(1)}$, we obtain

$$\frac{\partial g_1}{\partial \phi^{(1)}} = 2C\phi^{(1)} - 2\lambda_1\phi^{(1)} = 0,$$

or

$$C\phi^{(1)} = \lambda_1\phi^{(1)}.$$

The resulting eigenvector problem is a necessary condition. It is associated with an extremum of the Lagrangian. Hence, it is necessary for *both* the best and worst directions to satisfy this equation.
Assuming $\phi^{(1)}$ corresponds to the best eigenvector, i.e., the one with a maximum projection, the next best basis direction should satisfy the above requirements of maximum projection, but with the added restriction that it must be orthogonal to the best direction $\phi^{(1)}$. Thus, the second eigenvector $\phi^{(2)}$ is found by requiring

$$\max_{\phi^{(2)}} \left\langle (\phi^{(1)}, \mathbf{x})^2 \right\rangle$$

subject to

$$\left(\phi^{(2)}, \phi^{(2)}\right) = 1 \quad \text{and} \quad \left(\phi^{(1)}, \phi^{(2)}\right) = 0,$$

where now $\phi^{(1)}$ is assumed to be the (now fixed) orthonormal vector found above. The associated Lagrangian is now

$$g_2(\phi^{(2)}) = (\phi^{(2)}, C\phi^{(2)}) - \lambda_2[(\phi^{(2)}, \phi^{(2)}) - 1] - \mu(\phi^{(1)}, \phi^{(2)}),$$

and the necessary conditions for an extremum have become

$$\frac{\partial g_2}{\partial \phi^{(2)}} = \frac{\partial g_2}{\partial \lambda_2} = \frac{\partial g_2}{\partial \mu} = 0.$$

we will simplify our problem by ignoring the term $\mu(\phi^{(1)}, \phi^{(2)})$, which can be shown to be zero given that C is symmetric and such matrices always have orthogonal eigenvectors. Now, differentiating with respect to $\phi^{(2)}$, we obtain

$$\frac{\partial g_2}{\partial \phi^{(2)}} = 2C\phi^{(2)} - 2\lambda_2\phi^{(2)} = 0,$$

or

$$C\phi^{(2)} = \lambda_2\phi^{(2)}.$$

The process of determining the ith best eigenvector given the first $i-1$ eigenvectors is analogous. The result is that the optimal basis vectors must come from solutions of the eigenvector problem

$$C\phi^{(i)} = \lambda_i\phi^{(i)}.$$

Computing the KL eigenvectors via the above equation is referred to as the *direct method*.

Special Cases In the above derivation it is possible that the best one-dimensional subspace is not unique. Geometrically this corresponds to having an eigenspace with a geometric multiplicity greater than one. The algebraic multiplicity of the eigenvalue must be equal to the geometric multiplicity since C is symmetric and thus has a set of N linearly independent eigenvectors.
In practice two-dimensional eigenspaces, indicated by eigenvalues of multiplicity two, arise with translational invariant data where the KL eigenvectors are Fourier modes. This special case may be viewed as data traveling periodically on the unit circle; every orthonormal basis consists of vectors

that capture exactly the same projection of the data. Of course, if the data consists of an ellipse with unequal semiaxes, then the eigenvalues will be distinct.
If the matrix C has one or more zero eigenvalues, then the data has zero projection onto this basis vector. We may conclude that no information is present and this coordinate may be safely removed.

Ordering of the Optimal Basis A natural ordering for the optimal basis $\phi^{(j)}$ is provided by the *spectrum*, or KL *spectrum* (i.e., the discrete set of eigenvalues) of C. Recall that the first eigenvector was found by requiring that

$$\left\langle \left(\phi^{(1)}, \mathbf{x}\right)^2 \right\rangle = \text{maximum}$$

or, equivalently, that $\left\langle a_1^2 \right\rangle = \lambda_1 = \text{maximum}$. Proceeding in this fashion, the second eigenvalue is defined so that $\lambda_2 = \text{maximum}$, subject to the constraint that the associated coordinate direction $\phi^{(2)}$ must be orthogonal to $\phi^{(1)}$. Hence

$$\lambda_1 \geq \lambda_2.$$

The remaining eigenvalues are defined iteratively so that each λ_i is a maximum subject to the requirement that the associated coordinate direction $\phi^{(i)}$ be orthogonal to $\left\{\phi^{(1)}, \ldots, \phi^{(i-1)}\right\}$. Hence at each step $\lambda_i \geq \lambda_{i+1}$, so we conclude

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N \geq 0.$$

Therefore the eigenvectors can be ordered naturally according to the amount of variance contained in their respective directions. The pattern $\mathbf{x}^{(\mu)}$ is then approximated by the basis vectors ϕ corresponding to the largest eigenvalues of C.

Maximum Squared Projection vs Minimum Squared Error

Now we show the connection of the minimum-mean-squared error criterion and the maximum-squared-projection criterion. Given an ensemble of vectors $\left\{\mathbf{x}^{(\mu)}\right\}_{\mu=1}^{P}$ with each $\mathbf{x}^{(\mu)} \in V$ and $\dim V = N$, we seek a set of basis vectors $\left\{\phi^{(\mu)}\right\}_{\mu=1}^{N}$ such that the error of the truncated expansion is minimized in the mean-square sense. Recall that any pattern vector $\mathbf{x}^{(\mu)}$ may be written without error as

$$\mathbf{x}^{(\mu)} = \sum_{j=1}^{N} a_j^{(\mu)} \phi^{(j)}.$$

The expansion error vector may be expressed in terms of the basis, since

$$\begin{aligned} \varepsilon_D^{(\mu)} &= \mathbf{x}^{(\mu)} - \mathbf{x}_D^{(\mu)} \\ &= \sum_{j=D+1}^{N} a_j^{(\mu)} \phi^{(j)}. \end{aligned}$$

On average we have

$$\begin{aligned}
\varepsilon_{\text{mse}} &= \left\langle ||\varepsilon_D^{(\mu)}||^2 \right\rangle \\
&= \left\langle \left(\varepsilon_D^{(\mu)}, \varepsilon_D^{(\mu)} \right) \right\rangle \\
&= \left\langle \left(\sum_{j=D+1}^{N} a_j \phi^{(j)}, \sum_{k=D+1}^{N} a_k \phi^{(k)} \right) \right\rangle \\
&= \left\langle \sum_{j,k=D+1}^{N} a_j a_k \left(\phi^{(j)}, \phi^{(k)} \right) \right\rangle,
\end{aligned}$$

which upon invoking the orthonormality relation gives

$$\varepsilon_{\text{mse}} = \left\langle \sum_{j=D+1}^{N} a_j^2 \right\rangle$$

and hence

$$\varepsilon_{\text{mse}} = \left\langle \sum_{j=D+1}^{N} \left(\mathbf{x}, \phi^{(j)} \right)^2 \right\rangle.$$

Minimizing the mean squared error can now be seen as equivalent to maximizing the sum of the first D squared projections, i.e.,

$$g_1 + \cdots + g_D = \left\langle \sum_{j=1}^{D} \left(\mathbf{x}, \phi^{(j)} \right)^2 \right\rangle.$$

4. General Properties of the KL Expansion

In this section, we outline the useful properties of the KL decomposition. It will be seen that the optimal orthogonal transformation has remarkable structure in that it may be derived from several optimality criteria.

PROPERTY 4.1. The $N \times N$ matrix C is referred as the *ensemble-averaged covariance* matrix. It is symmetric and determines an ordered set of N orthogonal eigenvectors with associated real eigenvalues.

The following property is actually true for any data set that has zero mean.

PROPERTY 4.2. For an ensemble of mean-subtracted vectors, i.e., $\langle \mathbf{x} \rangle = 0$, the coordinate values a_j also have mean zero. To see this, write

$$\begin{aligned}
\langle a_j \rangle &= \left\langle \left(\mathbf{x}, \phi^{(j)} \right) \right\rangle \\
&= \left(\langle \mathbf{x} \rangle, \phi^{(j)} \right) \\
&= \left(0, \phi^{(j)} \right) \\
&= 0.
\end{aligned}$$

PROPERTY 4.3. The KL expansion coefficients are uncorrelated on average, i.e.,

$$\langle a_j a_k \rangle = 0$$

when $j \neq k$. Indeed, we have

$$\begin{aligned}
\langle a_j a_k \rangle &= \left\langle \left(\mathbf{x}, \phi^{(j)}\right)\left(\mathbf{x}, \phi^{(k)}\right) \right\rangle \\
&= \left\langle \left(\phi^{(j)}, \mathbf{x}\mathbf{x}^T \phi^{(k)}\right) \right\rangle \\
&= \left(\phi^{(j)}, \langle \mathbf{x}\mathbf{x}^T \rangle \phi^{(k)}\right) \\
&= \left(\phi^{(j)}, C\phi^{(k)}\right) \\
&= \left(\phi^{(j)}, \lambda_k \phi^{(k)}\right) \\
&= \lambda_k \delta_{jk}.
\end{aligned}$$

In particular, $\langle a_j a_k \rangle = 0$ when $j \neq k$.

Although the data is uncorrelated *on average* in the KL coordinate system, it is possible that the data is correlated on subsets of the total ensemble. For example, in a time-series setting it is possible for the data to have short-time correlations in the KL basis coordinates. If we consider a subset of the total data, i.e., $\tilde{X} \subset X$, then we may write $\langle a_j a_k \rangle_{\tilde{X}} \neq 0$.
The above property is also equivalent to the fact that in the KL basis, the covariance matrix is diagonal.

PROPERTY 4.4. The eigenvalues of C are nonnegative:

$$\lambda_j = \langle a_j^2 \rangle \geq 0$$

for $j = 1, \ldots, N$.

This follows directly from the previous property for the case $j = k$. Note that the number of nonzero eigenvalues is equal to the rank of the matrix C, which in turn equals the dimension of the space spanned by the data set.
We see from the next property that it is also possible to view the KL basis as the one that maximizes the variance along each coordinate direction, subject to orthogonality constraints.

PROPERTY 4.5. For mean-subtracted data, the statistical variance of the jth coordinate direction is proportional to the jth eigenvalue of C.

We write the statistical variance of the jth direction over the ensemble of patterns as

$$\begin{aligned}
\text{var}(a_j) &= \frac{1}{P-1} \sum_{\mu=1}^{P} \left(a_j^{(\mu)} - \langle a_j \rangle\right)^2 \\
&= \frac{P}{P-1} \frac{1}{P} \sum_{\mu=1}^{P} \left(a_j^{(\mu)}\right)^2 \\
&= \frac{P}{P-1} \langle a_j^2 \rangle = \frac{P}{P-1} \lambda_j.
\end{aligned}$$

That is,

$$\text{var}(a_j) \propto \lambda_j,$$

where we have used the fact $\langle a_j \rangle = 0$ and the Property 4.3.

PROPERTY 4.6. The eigenvalues of C give a measure of the truncation error:

$$\varepsilon_{\text{mse}} = \sum_{j=D+1}^{N} \lambda_j$$

Substituting the eigenvector equation $C\phi^{(j)} = \lambda_j \phi^{(j)}$ into

$$\varepsilon_{\text{mse}} = \sum_{j=D+1}^{N} \left(\phi^{(i)}, C\phi^{(i)} \right),$$

we have

$$\varepsilon_{\text{mse}} = \sum_{j=D+1}^{N} \left(\phi^{(i)}, \lambda_j \phi^{(i)} \right) = \sum_{j=D+1}^{N} \lambda_j.$$

PROPERTY 4.7. The KL basis captures more statistical variance than any other basis. Let $\left\{ \psi^{(i)} \right\}_{i=1}^{N}$ be any other basis for the inner product space V, and write the D-term expansion of an element of V as

$$\mathbf{x}_D^{(\mu)} = \sum_{j=1}^{D} b_j^{(\mu)} \psi^{(j)}.$$

Define a measure of the variance (for mean-subtracted data) with respect to the basis $\left\{ \psi^{(i)} \right\}$ as

$$\rho_j = \left\langle b_j^2 \right\rangle.$$

Then

$$\sum_{j=1}^{D} \rho_j \leq \sum_{j=1}^{D} \lambda_j$$

with equality when $\left\{ \psi^{(i)} \right\}$ is the KL basis.

DEFINITION 4.1. A data set is said to be *translationally invariant* is $\mathbf{x} \in X$ implies that any cyclic permutation of $\mathbf{x}$ is also in X.

PROPERTY 4.8. If X is a translationally invariant data set, then the optimal eigenvectors are the Fourier vectors, i.e., sinusoids.

Thus, for translationally invariant data, the discrete Fourier transform provides an analytical form for the best bases.

Shannon's Entropy A standard measure of information is provided by Shannon's entropy which is defined as

$$H = -\sum_{i=1}^{N} P_i \ln P_i,$$

where $\sum_{i=1}^{N} P_i = 1$. If we interpret the normalized eigenvalues of the covariance matrix

$$\tilde{\lambda}_i = \frac{\lambda_i}{\sum_{j=1}^{N} \lambda_j}$$

as the possibilities P_i, then it is possible to show that the KL eigenvectors are optimal in an information-theoretic sense, i.e., they minimize H [**46**].
The significance of Shannon's entropy H in this context is that it provides a measure of the distribution of the magnitude of the eigenvalues, or energy, across the coordinates of the basis. In particular, if the probabilities are all constant with

$$P_i = \frac{1}{N}$$

for all $i = 1, \ldots, N$, then

$$H = \ln N.$$

Also, if

$$P_i = \begin{cases} 1 & \text{if} \quad i = 1, \\ 0 & \text{if} 1 < i \leq N, \end{cases}$$

then $H = 0$.
In these two extreme cases we see that if all the eigenvalues are equal then there is no compression of information, i.e., there is no preferred coordinate direction and H is a maximum. On the other hand, if there is only one nonzero eigenvalue, then all the information is contained along one coordinate and H is a minimum.

PROPERTY 4.9. The Karhunen-Loéve basis minimizes Shannon's entropy.

For a proof of this property, see [**13**] or [**30**].

Truncation Criteria Several *ad hoc* criteria have been proposed for determining the number of terms D to retain in the expansion

$$\mathbf{x} = \sum_{j=1}^{D} a_j \phi^{(j)}.$$

A simple but widely used variance-based criterion is to retain the number of terms necessary to capture a specified fraction of the total variance [**21**].
From Equation (24) and the Pythagorean theorem, it follows that

$$||x^{(\mu)}||^2 = ||x_D^{(\mu)}||^2 + ||\varepsilon_D^{(\mu)}||^2.$$

Taking ensemble averages,

$$\left\langle ||x^{(\mu)}||^2 \right\rangle = \left\langle ||x_D^{(\mu)}||^2 \right\rangle + \left\langle ||\varepsilon_D^{(\mu)}||^2 \right\rangle.$$

In the KL basis, these terms area measured by the eigenvalues:

$$\left\langle ||x^{(\mu)}||^2 \right\rangle = \sum_{i=1}^{D} \lambda_i + \sum_{i=D+1}^{N} \lambda_i.$$

Since the statistical variance is a measure of the amplitudes squared, it is often referred to as *energy*. Using this terminology, the total energy in the data is denoted

$$E_N = \left\langle ||x^{(\mu)}||^2 \right\rangle = \sum_{i=1}^{N} \lambda_i.$$

The energy captured by a D-term expansion is given by

$$E_D = \sum_{i=1}^{D} \lambda_i.$$

Typically, for purposes of comparison, we shall be interested in the normalized energy

$$\tilde{E}_D = \frac{E_D}{E_N}.$$

Now one can also interpret the quantity

$$\tilde{\lambda}_i = \frac{\lambda_i}{E_N}$$

as the probability that a pattern is contained in the subspace spanned by the eigenvector $\phi^{(i)}$. Note that the normalized mean squared error is readily available as

$$\varepsilon_{\text{nmse}} = \sum_{i=D+1}^{N} \frac{\lambda_i}{E_N} = 1 - \tilde{E}_D.$$

This error is actually the relative error squared of the reconstruction of the data matrix in the Frobenius norm.

We will refer to a plot of the eigenvalues versus the eigenvalue index as a KL-spectrum plot. Often we will plot $\log \lambda_i$ vs i to enhance visualization of sharp decreases in the eigenvalues. Also, it is often useful to plot $\tilde{E}_D$ as a function of the number D of terms in the expansion. These plots are used to estimate the so-called KL dimension of the data. This dimension is generally taken as the number of terms required to ensure that some minimum quantity of energy is captured by the data.

Now, we have the normalized energy criterion

$$\tilde{E}_D > \gamma, \tag{26}$$

or equivalently, that the normalized mean squared error

$$\varepsilon_{\text{nmse}} < 1 - \gamma \tag{27}$$

should be less than some constant γ, typically taken to be $\gamma \in [0.90, 0.99]$. In addition, it is often useful to add the restriction that

$$\frac{\lambda_{D+1}}{\lambda_1} < \delta, \tag{28}$$

where $\delta = 0.01$, for example. This is a restriction on the 2-norm of the data matrix. We summarize these remarks with the following definitions:

DEFINITION 4.2. The KL *energy dimension*, written dim(KLE$_\gamma$), is defined to be the minimum number of terms D_γ required in the orthogonal expansion to ensure that $\tilde{E}_{D_\gamma} > \gamma$.

DEFINITION 4.3. The KL *stretching dimension*, written dim(KLM$_\delta$), is defined to be the minimum number D_δ required to ensure that $\lambda_{D_\delta+1}/\lambda_1 < \delta$.

In addition, it is useful to combine these definitions into a total KL dimension, written dim(KLD$_{\gamma,\delta}$), which may be defined as the maximum of KLE$_\gamma$ and KLM$_\delta$, i.e.,

$$D_T = \max\left\{D_\delta, D_\gamma\right\}.$$

Note that the utility of these global definitions of dimension is limited by the requirement of making *ad hoc* choices for γ and δ.

A number of other criteria have been proposed for determining the number of terms to retain in a best basis expansion. For example, it has been observed that the KL spectrum often can be viewed as two lines. The point these lines intersect determines the value of D. Often there is a gap in the eigenvalues, which indicates a value of D for truncation. A more rigorous approach is to apply a cross-validation scheme as outlined in [**15**], but even such methods do not always give consistent results. They do, however, have the advantage over the ad hoc energy criterion in that they actually take noise in the data into account when estimating dimension.

Finally, it should be emphasized that the utility of these measures is greatly enhanced if they can be implemented in a problem-dependent fashion. It is clear that one may require far fewer terms for a classification problem than for a reconstruction problem where more details in the pattern are required. Additionally, some problems have critical information in the lowest-energy modes, clearly indicating that the KL eigenvector selection does not have a unique solution.

Matrix Notation

In this section we would like to reinterpret the expansions as linear transformations and emphasize the dimensionality reducing properties of these transformations. We begin by constructing a matrix Φ made up of the eigenvectors of C, i.e.,

$$\Phi = \left[\phi^{(1)}|\phi^{(2)}|\cdots|\phi^{(N)}\right].$$

Thus the coefficients of the pattern vector with respect to the KL basis are now

$$\mathbf{a}^{(\mu)} = \Phi^T \mathbf{x}^{(\mu)},$$

where $\mathbf{a}^{(\mu)} = (a_1^{(\mu)}, \ldots, a_N^{(\mu)})^T$. These relations may be combined to give

$$A = \Phi^T X. \tag{29}$$

If we have determined a number D of terms to retain in our expansion, clearly it is not required to compute all the terms in the expansion. Hence, it is useful to define a dimensionality-reducing transformation based on

$$\Phi_D = \left[\phi^{(1)}|\phi^{(2)}|\cdots|\phi^{(D)}\right],$$

a matrix with D columns, namely the first D eigenvectors. Now, the D coefficients are given by

$$\hat{\mathbf{a}}^{(\mu)} = \Phi_D^T \mathbf{x}^{(\mu)},$$

where $\hat{\mathbf{a}}^{(\mu)} = (a_1^{(\mu)}, \ldots, a_D^{(\mu)})^T$. Or, in matrix notation,

$$\hat{A} = \Phi_D^T X,$$

where $\hat{A}$ is a $D \times P$ matrix. It is identical to the first D rows of A in Equation (29).

PROPERTY 4.10. The KL basis diagonalizes the ensemble average covariance matrix C:

$$\begin{aligned} \left\langle \mathbf{a}\mathbf{a}^T \right\rangle &= \left\langle (\Phi^T \mathbf{x})(\mathbf{x}^T \Phi) \right\rangle \\ &= \Phi^T C \Phi \\ &= \Lambda, \end{aligned}$$

where $\Lambda_{ii} = \lambda_i$ and all the off-diagonal elements are zero.

PROPERTY 4.11. We now have the spectral decomposition of the covariance matrix as $C = \Phi\Lambda\Phi^T$, i.e.,

$$C = \lambda_1 \phi^{(1)} \phi^{(1)} T + \lambda_2 \phi^{(2)} \phi^{(2)T} + \cdots + \lambda_1 \phi^{(N)} \phi^{(N)T}.$$

This allows us to decompose the covariance matrix in an optimal way. Note that if $P < N$, then we do not expect more than a basis of P vectors. The remaining $N - P$ vectors belong to the null space of C.

Geometrical Interpretation of KL Eigenvectors

PROPOSITION 4.1. *Let C be a nonsingular ensemble-averaged covariance matrix generated by the $N \times P$ data matrix X, i.e., $C = XX^T$. Consider the ellipsoid defined by*

$$\mathbf{x}^T C^{-1} \mathbf{x} = 1. \tag{30}$$

The eigenvectors of C are the directions of the principal axes of the ellipse, and the associated square roots of the eigenvalues are the lengths of the semi-axes.

PROOF. It suffices to transform the equation for the ellipse into the KL basis. Let U consist of the eigenvectors of C. Then

$$\mathbf{z} = U^T\mathbf{x}$$

is the appropriate change of basis. Since U is an orthogonal matrix, $\mathbf{x} = U\mathbf{z}$. Substituting this relation into Equation (30) gives

$$\mathbf{z}^T U^T C^{-1} U\mathbf{z} = 1. \tag{31}$$

Given that U diagonalizes C, we have

$$C = U\Lambda U^T.$$

Assuming C is nonsingular,

$$C^{-1} = U\Lambda^{-1}U^T.$$

Substituting this equation into Equation (31) gives

$$\mathbf{z}^T U^T U\Lambda^{-1}U^T U\mathbf{z} = 1.$$

Again, using the orthogonality of U, we have $\mathbf{z}^T\Lambda^{-1}\mathbf{z} = 1$, or

$$\sum_{i=1}^{N} \frac{z_i^2}{\lambda_i} = 1.$$

Note that the expression $\mathbf{x}^T C^{-1}\mathbf{x} = c$ represents a family of concentric ellipses. □

As an example consider Gaussian random data (x_1, x_2) where the first variable has mean zero and standard deviation 2.5 while the second variable has mean zero and standard deviation 1.0. This data is then rotated in the plane and the KL procedure applied. Calculation shows $\lambda^{(1)} = 6.3373$ and $\lambda^{(2)} = 1.0543$, so $\sigma^{(1)} = \sqrt{\lambda^{(1)}} = 2.5174$ and $\sigma^{(2)} = \sqrt{\lambda^{(2)}} = 1.0268$.[2]
The ellipsoidal neighborhoods determined in the first stage of the algorithm may be used to order the data in the ε-neighborhood matrix via the introduction of the A-norm written $||\mathbf{x}||_A^2 = \mathbf{x}^T A\mathbf{x}$, where the matrix A is taken to be C^{-1} and $C = B_\varepsilon B_\varepsilon^T$:

$$\begin{aligned} ||\mathbf{x}||_{C^{-1}}^2 &= \mathbf{x}^T C^{-1}\mathbf{x} \\ &= \mathbf{x}^T U\Lambda^{-1}U^T\mathbf{x} \\ &= (\mathbf{x}^T U\Lambda^{-1/2})(\Lambda^{-1/2})U^T\mathbf{x} \\ &= \mathbf{y}^T\mathbf{y} = ||\mathbf{y}||^2, \end{aligned}$$

where $\mathbf{y} = \Lambda^{-1/2}U^T\mathbf{x}$. Hence the ellipsoidal norm may be interpreted as the standard Euclidean norm after the change of coordinates

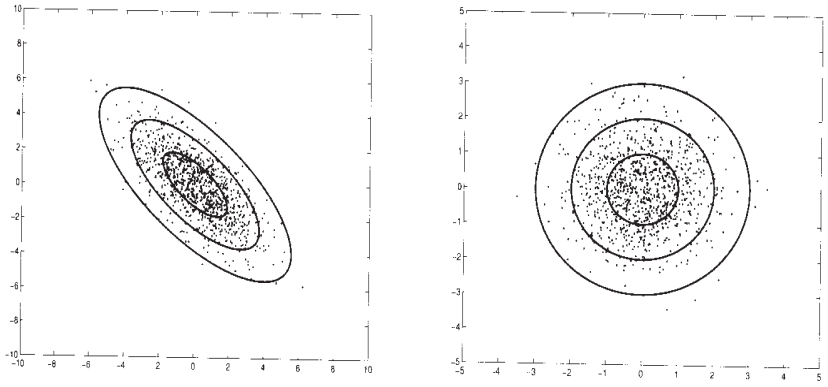
$$Y = \Lambda^{-1/2}U^T B_\varepsilon.$$

This transformation is referred to as a *whitening* transformation. The covariance matrix is the identity in the new coordinate system, i.e.,

$$YY^T = I.$$

See Figure 1 for an example of applying the whitening transformation to Gaussian data.

[2]Note that in our deviation of KL, no assumptions were made concerning the statistical distribution of the data.



(a) Gaussian noise (b) Data on the left after whitening

FIGURE 1. (Images are taken directly from [**31**]) Random data before and after whitening. The contours correspond to the loci $\mathbf{x}C^{-1}\mathbf{x} = c_i$, where $c_1 = 1$, $c_2 = 4$, and $c_3 = 9$.

5. The Snapshot Method

The construction of a *data-dependent* basis as outlined above requires solving the eigenvector problem

$$C\phi^{(j)} = \lambda_j \phi^{(j)},$$

where C is an $N \times N$ matrix formed by averaging P rank one covariance matrices. When N becomes large, e.g., 1000 to 10000, it is generally not possible to solve this problem directly. There are a variety of techniques for computing the largest eigenvalues and eigenvectors, but if C is a nonsingular matrix, then the problem may be reduced without approximation to an eigenvector problem of size $P \times P$. The technique is referred to as the *snapshot method* because of its applicability to data sets consisting of high-resolution digital snapshots [**32, 45**]. This result is very useful if the number of patterns, P, is manageable, typically $P < 1000$. The dimension N now enters in only in storage space and add/multiplies.

Reduction of Lip Motion

In this application, introduced in [**33, 34**], we are interested in characterizing the motion of lips during speech, i.e., machine lip reading. The data sets come in the form of sequences of digital images which are recorded by focusing a camera on the speaker's lips. In this setting, a word ω is viewed as a short sequence ($P = 16$) of high-dimensional (120×100) images

$$\omega = \left\{ \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(P-1)}, \mathbf{x}^{(P)} \right\}.$$

In the coordinate system of the ambient space, a word is represented by $N = 12,000$ spatially and temporally correlated time series. To reduce the dimensionality of this representation we digitally record a set of words, to be used for training, that characterizes the lip motion, i.e., we assume the data set is large enough to span the space of all relevant lip motions. From this training set we compute the eigenpictures, or *eigenlips*, displayed in Figure 2, using the snapshot method. A sample word (not from the training set) is represented by the sequence of 16 images shown on the left of Figure 3. These images were projected onto the first 20 eigenpictures. The result of the reconstruction is excellent; see the right of Figure 3. We note that plotting the relationship between two coefficients in the plane gives a characteristic curve for particular words; see [**34**] for details.

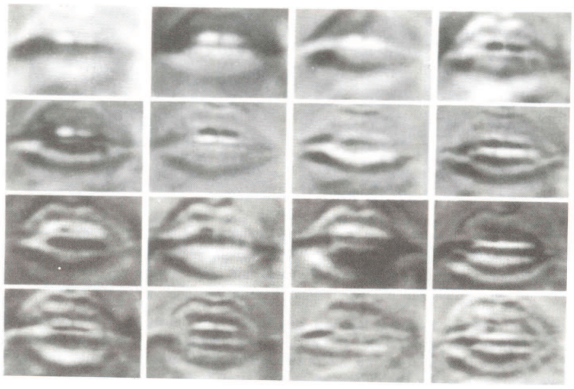


FIGURE 2. (Images are taken directly from [**31**]) The first 16 eigenlips ordered from left to right and top to bottom.

FIGURE 3. (Images are taken directly from [**31**]) Left: A sequence of snapshots of lip motion. Right: The reconstruction of lip images after projection onto a 20-dimensional optimal subspace.

6. The SVD and The KL Expansion

In this section we reexamine, via the SVD, the direct and snapshot methods for implementing the KL expansion. The SVD, as described in Section 6 is a classical and powerful tool in numerical linear algebra. Here our purpose is to demonstrate that the eigenvector problems associated with the direct and the snapshot method fit neatly into the mathematical framework of the SVD. In particular, we shall see that the *left-singular vectors are the eigenvectors computed in the direct method, while the right-singular vectors are the eigenvectors computed in the snapshot method.*

We begin by constructing an $N \times P$ data matrix X out of our ensemble $\left\{\mathbf{x}^{(\mu)}\right\}_{\mu=1}^{P}$ of pattern vectors in $\mathbb{R}^N$, where the columns of X are the pattern vectors: $X = [\mathbf{x}^{(1)}|\cdots|\mathbf{x}^{(P)}]$.

To assist in the interpretation of our results, we will assume that the ensemble consists of time-dependent vectors. For simplicity we assume that the spatial variable is one-dimensional. It should be emphasized that these assumptions are for convenience and are not a requirement of the theory. For these time-dependent observations, $(X)_{ij} = x_i^{(j)}$, the column index is the time index $j = 1, \ldots, P$, and

the row index, $i = 1, \ldots, N$ is the spatial index. Thus our *spatiotemporal* data matrix

$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(P)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(P)} \\ \vdots & \vdots & \ddots & \vdots \\ x_N^{(1)} & x_N^{(2)} & \cdots & x_N^{(P)} \end{bmatrix}$$

is indexed from left to right by time and from top to bottom by space. Note that the size of this matrix may be enormous in practice and its actual formation may not be possible due to computer memory limitations. However, this does not prevent us from applying the mathematics of the SVD.
The transpose of X is given by

$$X^T = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_N^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_N^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(P)} & x_2^{(P)} & \cdots & x_N^{(P)} \end{bmatrix}.$$

In what follows, it will be useful to write the matrix X^T in terms of its column vectors. Therefore, we introduce the notation

$$X^T = [\mathbf{y}^{(1)}|\cdots|\mathbf{y}^{(N)}],$$

where $y_j^{(i)} = x_i^{(j)}$. One might view X^T as a new data matrix where the roles of time and space have been interchanged.
The motivation for collecting the data in matrix form comes from the observation that we may rewrite the ensemble-average covariance matrix in terms of XX^T. To see this, write out the jkth element of this matrix,

$$(XX^T)_{jk} = \sum_{\mu=1}^{P} x_j^{(\mu)} x_k^{(\mu)} = P\left\langle x_j x_k \right\rangle, \tag{32}$$

where $j, k = 1, \ldots, N$. In other words,

$$\frac{1}{P} XX^T = \left\langle \mathbf{x}\mathbf{x}^T \right\rangle.$$

From the above expression, we see that that patterns are being correlated over the spatial variable and average over time. Thus, we define the ensemble-average *spatial covariance matrix*[3] C_x of the observations as

$$C_x = \frac{1}{P} XX^T. \tag{33}$$

In an analogous manner we may form the ensemble-averaged *temporal covariance* matrix

$$C_t = \frac{1}{N} X^T X. \tag{34}$$

As before, let's write out the jkth element of this matrix:

$$(X^T X)_{jk} = \sum_{l=1}^{N} x_l^{(j)} x_l^{(k)} = N\left\langle y_j y_k \right\rangle, \tag{35}$$

[3]Strictly speaking, this matrix is not necessarily a covariance matrix, as it is not essential to apply a mean subtraction. We retain the term in analogy with the previous sections.

where $j,k = 1,\ldots,P$. In other words,

$$\frac{1}{N}X^TX = \left\langle \mathbf{y}\mathbf{y}^T \right\rangle,$$

so we see that C_t is in fact a temporal covariance matrix. Note that the definition of the ensemble, and hence ensemble average, has changed from that given above. In determining C_x the spatial covariances are averaged over time, while in determining C_t the temporal covariances are averaged over the spatial domain. Hence we will refer to the eigenvectors of C_x as the *spatial eigenvectors* and to the eigenvectors of C_t as the *temporal eigenvectors*.
We note that C_x is an $N \times N$ matrix and the spatial eigenvectors are solutions of

$$XX^TU = U\Lambda_x, \tag{36}$$

where the columns of U correspond to the eigenvectors of C_x, i.e., $U = [\mathbf{u}^{(1)}|\cdots|\mathbf{u}^{(N)}]$. Also, Λ_x is an $N \times N$ matrix

$$\Lambda_x = \begin{bmatrix} \lambda^{(1)} & & & \\ & \lambda^{(2)} & & \\ & & \ddots & \\ & & & \lambda^{(N)} \end{bmatrix}.$$

Similarly, C_t is a $P \times P$ matrix and the temporal eigenvectors are the solutions of

$$X^TXV = V\Lambda_t, \tag{37}$$

where the columns V correspond to the eigenvectors of C_t, i.e., $V = [\mathbf{v}^{(1)}|\cdots|\mathbf{v}^{(P)}]$. Also, Λ_t is an $P \times P$ matrix

$$\Lambda_t = \begin{bmatrix} \lambda^{(1)} & & & \\ & \lambda^{(2)} & & \\ & & \ddots & \\ & & & \lambda^{(P)} \end{bmatrix}.$$

We have used the same notation for the eigenvalues of XX^T and X^TX in view of the following proposition, which is a consequence of the fact $\det(XX^T) = \det(X^TX)$:

PROPOSITION 6.1. *The nonzero entries of Λ_x and Λ_t are equal.*

Hence, the KL spectrum can be determined from computing the eigenvalues of either C_x or C_t. Note that the eigenvalues of C_x are actually given by the matrix $(1/P)\Lambda_x$ and the eigenvalues of C_t are given by the matrix $(1/N)\Lambda_t$.
Now we recast these results in terms of the SVD. We recognize that the spatial eigenvectors U of the spatial covariance matrix XX^T are exactly the left-singular vectors of the data matrix X. Also, the temporal eigenvectors V of the temporal covariance matrix X^TX are exactly the right-singular vectors of the data matrix X. Thus, by the SVD we may decompose the data matrix

$$X = U\Sigma V^T,$$

where Σ is the $N \times P$ diagonal matrix given by

$$\Sigma = diag(\sigma^{(1)},\ldots,\sigma^{(r)}),$$

where $\sigma^{(i)} = \sqrt{\lambda^{(i)}}$, i.e., the singular values are the square roots of the eigenvalues of XX^T. Again, r is the rank of the matrix X, or, equivalently, the number of nonzero singular values.
Given the columns of U for a basis for the data, we have the *orthogonal expansion*

$$\mathbf{x}^{(\mu)} = \sum_{j=1}^{r} a_j^{(\mu)}\mathbf{u}^{(j)}.$$

Recall that this may be rewritten in terms of matrices as

$$X = UA,$$

where A is an $N \times P$ matrix of expansion coefficients $A = [\mathbf{a}^{(1)}|\cdots|\mathbf{a}^{(P)}]$. Furthermore, since $U^T = U^{-1}$, the expansion coefficients are given by

$$A = U^T X. \tag{38}$$

The following propositions provide very useful relationships between the expansion coefficients and the eigenvectors.

PROPOSITION 6.2. *If Σ is the matrix of singular values of X, and V the matrix of associated temporal eigenvectors (right-singular vectors), then the matrix of expansion coefficients A, i.e., the projections of the data onto the optimal spatial eigenvectors, is given by*

$$A = \Sigma V^T. \tag{39}$$

PROOF. By the SVD

$$X = U\Sigma V^T.$$

Multiplying both sides of the relationship by U^T and using the fact that $U^T U = I$ gives

$$U^T X = \Sigma V^T.$$

Recognizing $A = U^T X$ completes the result, which has an extremely useful interpretation: the expansion coefficients A are contained in the temporal eigenvectors, i.e., the right-singular vectors. □

Thus, the time-dependent coefficients given by the matrix A may be computed using two different methods. First, given the N-dimensional spatial eigenvectors, the projections may be found directly using Equation (38). Alternatively, given the P-dimensional temporal eigenvectors, the projection coefficients may be found indirectly by using Equation (39). Note that either P-dimensional or N-dimensional eigenvectors are required, but not both. Usually one approach is significantly more efficient than the other.

The next proposition states that the spatial eigenvectors may be written as the superposition of data where the appropriate expansion coefficients are provided by temporal eigenvectors, i.e., the right-singular vectors.

PROPOSITION 6.3.

$$\mathbf{u}^{(j)} = \frac{1}{\sigma_j} \sum_{k=1}^{P} v_k^{(j)} \mathbf{x}^{(k)},$$

where $j = 1, \ldots, rank X$.

PROOF. The result follows directly from the SVD

$$\begin{aligned} X &= U\Sigma V^T \\ XV &= U\Sigma, \end{aligned}$$

where the fact that V is an orthogonal matrix has been used. Defining Σ^+, a $P \times N$ matrix, as the pseudoinverse of Σ, we have

$$U = XV\Sigma^+,$$

from which the proposition follows. □

The next proposition presents a relation that is in a sense symmetrical to the previous one. It states that the temporal eigenvectors may be written as the superposition of data where the appropriate expansion coefficients are provided by spatial eigenvectors.

PROPOSITION 6.4.

$$\mathbf{v}^{(j)} = \frac{1}{\sigma_j} \sum_{k=1}^{N} u_k^{(j)} \mathbf{y}^{(k)},$$

where $j = 1, \ldots, rankX$ *and* $y_i^{(k)} = x_k^{(i)}$ *as defined above.*

PROOF. Again, this result follows directly from the SVD

$$\begin{aligned} X &= U\Sigma V^T \\ U^T X &= \Sigma V^T \\ V^T &= \Sigma^+ U^T X, \end{aligned}$$

where the fact that U is an orthogonal matrix has been used. Hence,

$$V = X^T U \Sigma^+,$$

from which the proposition follows. □

EXAMPLE 6.1. Recall Example 6.1, where the left- and right-singular vectors were computed for the data matrix

$$X = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad X^T = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

The columns of X^T are $\mathbf{y}^{(1)} = (1\,1)^T$, $\mathbf{y}^{(2)} = (0\,1)^T$, and $\mathbf{y}^{(3)} = (1\,0)^T$. We now confirm Proposition 6.3. To this end, we compute

$$\mathbf{u}^{(j)} = \frac{1}{\sigma_j} \sum_{k=1}^{P} v_k^{(j)} \mathbf{x}^{(k)}$$

for $\mathbf{u}^{(1)}$. Evaluating this formula gives

$$\mathbf{u}^{(1)} = \frac{1}{\sqrt{3}} \left(v_1^{(1)} \mathbf{x}^{(1)} + v_2^{(1)} \mathbf{x}^{(2)} \right).$$

Recalling $\mathbf{v}^{(1)} = \frac{1}{\sqrt{2}}(1 \quad 1)^T$, we obtain

$$\mathbf{u}^{(1)} = \frac{1}{\sqrt{3}} \left[\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \right].$$

Therefore,

$$\mathbf{u}^{(1)} = \frac{1}{\sqrt{6}}(2\,1\,1)^T,$$

which checks.
We now confirm Proposition 6.4 by computing

$$\mathbf{v}^{(j)} = \frac{1}{\sigma_j} \sum_{k=1}^{N} u_k^{(j)} \mathbf{y}^{(k)},$$

for $\mathbf{v}^{(1)}$. Employing the value for $\mathbf{u}^{(1)}$ found above, we obtain

$$\mathbf{v}^{(1)} = \frac{1}{\sqrt{3}} \left[\frac{2}{\sqrt{6}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \frac{1}{\sqrt{6}} \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \frac{1}{\sqrt{6}} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

which checks. It is also reassuring to confirm the formula $A = \Sigma V^T$ which provides the spatial expansion coefficients in terms of the temporal eigenvectors. By direct computation,

$$\begin{aligned} A &= U^T X \\ &= \begin{pmatrix} \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} \frac{3}{\sqrt{6}} & \frac{3}{\sqrt{6}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 \end{pmatrix}. \end{aligned}$$

According to the proposition, the $N \times P$ matrix $A = \Sigma V^T$ is also found as

$$\frac{1}{\sqrt{2}} \begin{pmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} \sqrt{\frac{3}{2}} & \sqrt{\frac{3}{2}} \\ -\sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \\ 0 & 0 \end{pmatrix},$$

which agrees with the previous result.

Translationally Invariant Data

We consider a data set consisting of the *row* vectors $\left\{\mathbf{x}^{(\mu)}\right\}$ to be translationally invariant if the spatial domain is periodic and a cyclic permutation of the components of a given data point generates another element of the data set. For example, if $\mathbf{x}^{(1)} = (123)$, then the translationally invariant data set generated by this point is

$$X = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{pmatrix}.$$

The columns of the $m \times n$ matrix X may be cyclically permuted by the $n \times n$ circulant matrix C, where $(C)_{ij} = c_{i-j}$ and $c_i = c_{i+ln}$. Since C is also a permutation matrix, its rows must be circular shifts of the first row of the identity matrix.
In this example, the 3×3 circulant permutation matrix is

$$C = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

The action of right-multiplying by C is to cyclically permute the columns, i.e.,

$$XC = \begin{pmatrix} 3 & 1 & 2 \\ 2 & 3 & 1 \\ 1 & 2 & 3 \end{pmatrix}.$$

Associated with the circulant matrix C is a permutation matrix P, obtained by interchanging rows or columns of the identity matrix, which rearranges the rows of X into their original order

$$PXC = X.$$

In our example it may be verified that

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Note that if P is a permutation , then $P^{-1} = P^T$. The right singular vectors of X may be determined by forming $X^T X$, i.e.,

$$X^T X = C^T X^T P^T P X C = C^T X^T X C;$$

hence

$$C X^T X = X^T X C.$$

So the circulant matrix C commutes with $X^T X$. It can be shown that both C and $X^T X$ are similar to diagonal matrices, i.e., they are *simple*. It follows that they share the same eigenvectors [**37**]. As it will be shown in Chapter 6, the eigenvectors of the circulant matrices are Fourier modes, so we may conclude that the right-singular vectors are also sinusoids. The given 3×3 circulant matrix C has the eigenvectors

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} 1 \\ e^{2\pi i/3} \\ e^{4\pi i/3} \end{pmatrix}, \quad \begin{pmatrix} 1 \\ e^{4\pi i/3} \\ e^{2\pi i/3} \end{pmatrix}.$$

The fact that the optimal basis for translationally invariant data consists of Fourier vectors is well known. The argument presented here follows [**8**].

CHAPTER 3

Additional Theory, Algorithms and Applications

In this chapter we continue our study of the KL procedure and apply it to a variety of problems. We begin with an extension of the KL procedure for *gappy* data in Section 1. This is followed by the application of the KL procedure in the presence of noise in Section 2.

1. Application with Missing Data

Now we turn to the problem of using the KL procedure on data sets that have *gaps*, or missing components. The algorithm presented here is due to Everson and Sirovich [**18**]. Our development follows [**18**], although here we simplify the setting of the presentation by using discrete vector spaces, rather than function spaces. We distinguish this extension of the KL procedure for *gappy data*, using the terminology of [**18**], from the case of *noisy data*, which is developed in the next section.

1.1. Estimating Missing Data. Suppose we can learn a set of best basis $\left\{\phi^{(1)}, \phi^{(2)}, \ldots, \phi^{(P)}\right\}$ from a training set. Let $\mathbf{x} \in \mathbb{R}^N$ be a vector that possesses a reduced expansion in terms of the KL basis as

$$\mathbf{x} \approx \mathbf{x}_D = \sum_{n=1}^{D} a_n \phi^{(n)}.$$

It follows that only D points of information are required to reproduce the original vector. Consider now an incomplete, or gappy, copy $\tilde{\mathbf{x}}$ of the original vector $\mathbf{x}$. This may be expressed as

$$\tilde{x}_i = \begin{cases} x_i, & m_i = 1, \\ 0, & m_i = 0, \end{cases}$$

where the vector $\mathbf{m} \in \mathbb{R}^N$ is an indicator vector, or *mask*, which identifies the indices of the missing data. We will also write this incomplete vector as

$$\tilde{\mathbf{x}} = \mathbf{m}.\mathbf{x},$$

where the product notation presents the point-wise multiplication: the ith component of the product is $(\mathbf{m}.\mathbf{x})_i = m_i x_i$.

Given a vector $\mathbf{x}$ that is an element of an ensemble of intrinsically low dimension, it may be possible to replace, or at least estimate, the missing entries. If the ambient dimension in which the vector resides is large, specifically if $D \ll N$, it is plausible that this repair may by possible even if a significant number of the entries of $\mathbf{x}$ are missing. The missing entries of the gappy vector $\tilde{\mathbf{x}}$ may be approximated by the corresponding elements of $\tilde{\mathbf{x}}_D$, yielding a D-term approximation to $\tilde{\mathbf{x}}$ as

$$\tilde{\mathbf{x}} \approx \tilde{\mathbf{x}}_D = \sum_{n=1}^{D} \tilde{a}_n \phi^{(n)}, \tag{40}$$

where the $\{\tilde{a}_n\}$ are to be found by requiring that

$$E = ||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_D||_{\mathbf{m}}^2 \tag{41}$$

be a minimum. The notation

$$||\mathbf{x}||_{\mathbf{m}}^2 = (\mathbf{x}, \mathbf{x})_{\mathbf{m}} = (\mathbf{m}.\mathbf{x}, \mathbf{m}.\mathbf{x})$$

indicates that the norm is defined only on the data that is not missing. With this norm it follows that the coefficients $\{\tilde{a}_n\}$ are estimated based on the available data only.

A few observations are in order. First, note that, by virtue of the eigenvectors $\left\{\phi^{(n)}\right\}$ being fully intact, the reconstruction $\tilde{\mathbf{x}}_D$ has no missing entries. In addition, since $\tilde{\mathbf{x}}_D$ approximates $\tilde{\mathbf{x}}$ on the mask $\mathbf{m}$, it also approximates $\mathbf{x}$ given

$$||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_D||_{\mathbf{m}} = ||\mathbf{x} - \tilde{\mathbf{x}}_D||_{\mathbf{m}}.$$

Now, using this definition of the (gappy) inner product, the error may be expanded as

$$\begin{aligned} E &= \left(\tilde{\mathbf{x}} - \sum_{n=1}^{D} \tilde{a}_n \phi^{(n)}, \tilde{\mathbf{x}} - \sum_{k=1}^{D} \tilde{a}_k \phi^{(k)}\right)_{\mathbf{m}} \\ &= (\tilde{\mathbf{x}}, \tilde{\mathbf{x}})_{\mathbf{m}} - 2\left(\tilde{\mathbf{x}}, \sum_{n=1}^{D} \tilde{a}_n \phi^{(n)}\right)_{\mathbf{m}} + \left(\sum_{n=1}^{D} \tilde{a}_n \phi^{(n)}, \sum_{k=1}^{D} \tilde{a}_k \phi^{(k)}\right)_{\mathbf{m}} \\ &= ||\tilde{\mathbf{x}}||^2_{\mathbf{m}} - 2\sum_{n=1}^{D} \tilde{a}_n (\tilde{\mathbf{x}}, \phi^{(n)})_{\mathbf{m}} + \sum_{k,n=1}^{D} \tilde{a}_k \tilde{a}_n (\phi^{(k)}, \phi^{(n)})_{\mathbf{m}}. \end{aligned}$$

Note that the eigenvectors $\left\{\phi^{(k)}\right\}$ are no longer orthogonal on the gappy inner product.

Differentiating the error term E with respect to the kth coefficient gives

$$\frac{\partial E}{\partial \tilde{a}_k} = 0 - 2(\tilde{\mathbf{x}}, \phi^{(k)})_{\mathbf{m}} + 2\sum_{i=1}^{D} \tilde{a}_i (\phi^{(i)}, \phi^{(k)})_{\mathbf{m}} = 0,$$

from which it follows that

$$\sum_{i=1}^{D} \tilde{a}_i (\phi^{(i)}, \phi^{(k)})_{\mathbf{m}} = (\tilde{\mathbf{x}}, \phi^{(k)})_{\mathbf{m}}.$$

This may be rewritten in the form of a linear system

$$M\tilde{\mathbf{a}} = \mathbf{f},$$

where

$$M_{ij} = (\phi^{(i)}, \phi^{(j)})_{\mathbf{m}}$$

and

$$f_i = (\tilde{\mathbf{x}}, \phi^{(i)})_{\mathbf{m}}.$$

The original vector $\mathbf{x}$ is then approximated by the repair of $\tilde{\mathbf{x}}$, which we denote by $\mathbf{r}$, i.e.,

$$(\mathbf{r}_D)_i = \begin{cases} \mathbf{x}_i, & m_i = 1, \\ (\tilde{\mathbf{x}}_D)_i, & m_i = 0. \end{cases}$$

1.2. Estimating a KL Basis with Missing Data. In the previous subsection we examined the question of estimating data missing from an observation. **The procedure required a KL basis derived from a training set with no gaps.** These ideas may now be applied to constructing a KL basis where only incomplete data sets are available. The procedure presented here was proposed in [**18**]. It is based on an iterative process that successively *repairs* the gappy data and improves the estimate for the associated KL basis.

A summary of the iterative procedure for repairing the gappy data set and computing the KL basis vectors is provided in the following box.

KL Procedure for Gappy Data

1. Initialize the missing data with the ensemble average.
2. Compute the first estimate of the KL basis.
3. Re-estimate the ensemble using the gappy approximation and the KL basis.
4. Re-compute the KL basis.
5. Repeat Steps 3–4 until stopping criterion is satisfied.

We now describe the steps in detail. The data set may be modeled by associating with each pattern a mask $\mathbf{m}^{(\mu)}$ of indices indicating which data is available and which components are missing. Each pattern with incomplete data may now be written

$$\tilde{\mathbf{x}}^{(\mu)} = \mathbf{m}^{(\mu)}.\mathbf{x}^{(\mu)}.$$

The ensemble average of the incomplete patterns is now

$$(\langle\tilde{\mathbf{x}}\rangle)_i = \frac{1}{P_i}\sum_{\mu=1}^{P}\tilde{x}_i^{\mu},$$

where

$$P_i = \sum_{\mu=1}^{P} m_i^{\mu}.$$

Once this ensemble average has been determined from the gappy data, the first stage of the ensemble repair procedure may be executed. **This repair is done by replacing the missing data with the point-wise mean of the existing data.** Specifically, the first stage of the repair process is then

$$(\mathbf{r}^{(\mu)}(0))_i = \begin{cases} x_i^{(\mu)}, & m_i^{(\mu)} = 1, \\ (\langle\tilde{\mathbf{x}}\rangle)_i, & m_i^{(\mu)} = 0. \end{cases} \tag{42}$$

The improved ensemble $\left\{\mathbf{x}^{(\mu)}(0)\right\}$ may be used to construct the first estimate, or initialize, the KL basis, which we denote

$$\left\{\phi^{(j)}(0)\right\}_{j=1}^{P}.$$

Now, given an initial estimate for the KL basis vectors, an improved approximation may be obtained using the procedure of the previous section.
Specifically, given the gappy pattern vector $\tilde{x}^{(\mu)} = \mathbf{m}^{\mu}.\mathbf{x}^{\mu}$, we may improve our estimate of $\mathbf{x}$ given by Equation 42 by using the first estimate of the KL basis. The improved estimate may be written

$$\tilde{\mathbf{x}}_D^{(\mu)}(1) = \sum_{k=1}^{D}\tilde{a}_k^{(\mu)}(1)\phi^{(k)}(0),$$

where the $\left\{\tilde{a}_k^{(\mu)}(1)\right\}$ are the solutions to

$$M^{\mu}(0)\tilde{\mathbf{a}}^{(\mu)}(1) = \mathbf{f}^{\mu}(0),$$

where

$$M_{ij}^{\mu}(0) = (\phi^{(i)}(0), \phi^{(j)}(0))_{\mathbf{m}^{(\mu)}}$$

and

$$f_k^{(\mu)}(0) = (\tilde{\mathbf{x}}^{\mu}, \phi^{(k)}(0))_{\mathbf{m}^{(\mu)}}.$$

The second iteration of the repair procedure uses the improved estimate

$$(\mathbf{r}^{(\mu)}(1))_i = \begin{cases} x_i^{(\mu)}, & m_i^{(\mu)} = 1, \\ \left(\left\langle \tilde{\mathbf{x}}_D^{(\mu)}(1) \right\rangle\right)_i, & m_i^{(\mu)} = 0, \end{cases} \tag{43}$$

for the gappy data ensemble.
The last two steps of this process are repeated until the KL basis is deemed to have converged in a satisfactory manner. In particular, we expect the sequence of repairs to approach the actual data

$$\mathbf{r}^{(\mu)}(n) \to \mathbf{x}^{(\mu)},$$

and consequently, the sequence of estimated eigenvectors to approach and actual eigenvectors:

$$\phi^{(i)}(n) \to \phi^{(i)}.$$

It is natural to end the iteration when the updates provide little change and it is concluded that no further progress is being made.

EXAMPLE 1.1. This example concerns the application of the KL procedure for incomplete data discussed above. Let the complete data set be translationally invariant:

$$f(x_m, t_\mu) = \frac{1}{N} \sum_{k=1}^{N} \frac{1}{k} \sin[k(x_m - t_\mu)],$$

where $m = 1, \ldots, M$, with M dimension of the ambient space (size of the spatial grid), and $\mu = 1, \ldots, P$, with P the number of points in the ensemble, as shown in Figure 1(a). Let $x_m = \dfrac{(m-1)2\pi}{M}$ and $t_\mu = \dfrac{(\mu-1)2\pi}{P}$. We select an ensemble of masks $\left\{\mathbf{m}^{(\mu)}\right\}$, $\mu = 1, \ldots, P$, where 10% of the indices are selected to be zero for each mask. Each pattern in the incomplete ensemble may be written as

$$\tilde{\mathbf{x}}^{(\mu)} = \mathbf{m}^{(\mu)} . \mathbf{f}^{(\mu)},$$

where $\left(\mathbf{f}^{(\mu)}\right)_m = \dfrac{1}{N} \sum_{k=1}^{N} \dfrac{1}{k} \sin[k(x_m - t_\mu)]$. Let $P = M = 64$ and $N = 3$. See Figure 1(b) for the resulted $f(x,t)$ with 10% mask. With the gappy algorithm [**18**] and the equation

$$\tilde{\mathbf{x}} \approx \tilde{\mathbf{x}}_D = \sum_{n=1}^{D} \tilde{a}_n \phi^{(n)},$$

we obtain the repaired data after a single pass of repair as shown in Figure 1(c) and the final repaired data after 3 iterations in Figure 1(d).

2. Application to Noisy Data

Now we turn to the case where the patterns have added noise, i.e.,

$$\mathbf{x}^{(\mu)} = \mathbf{s}^{(\mu)} + \mathbf{n}^{(\mu)},$$

or, in terms of data matrices,

$$X = S + N,$$

where X is assumed to be tall, i.e., if it is $n \times P$ then $n > P$. In addition, the columns are assumed to have zero mean.
In the general situation, neither the denoised signal $\mathbf{s}^{(\mu)}$ nor the noise component $\mathbf{n}^{(\mu)}$ is observable; only the noisy signal $\mathbf{x}^{(\mu)}$ is available. An optimal representation of the noisy data in terms of the eigenvectors of the ensemble-averaged covariance matrix $C_x = \left\langle \mathbf{x}\mathbf{x}^T \right\rangle$ does not, in general, provide a good separation of the signal and the noise.

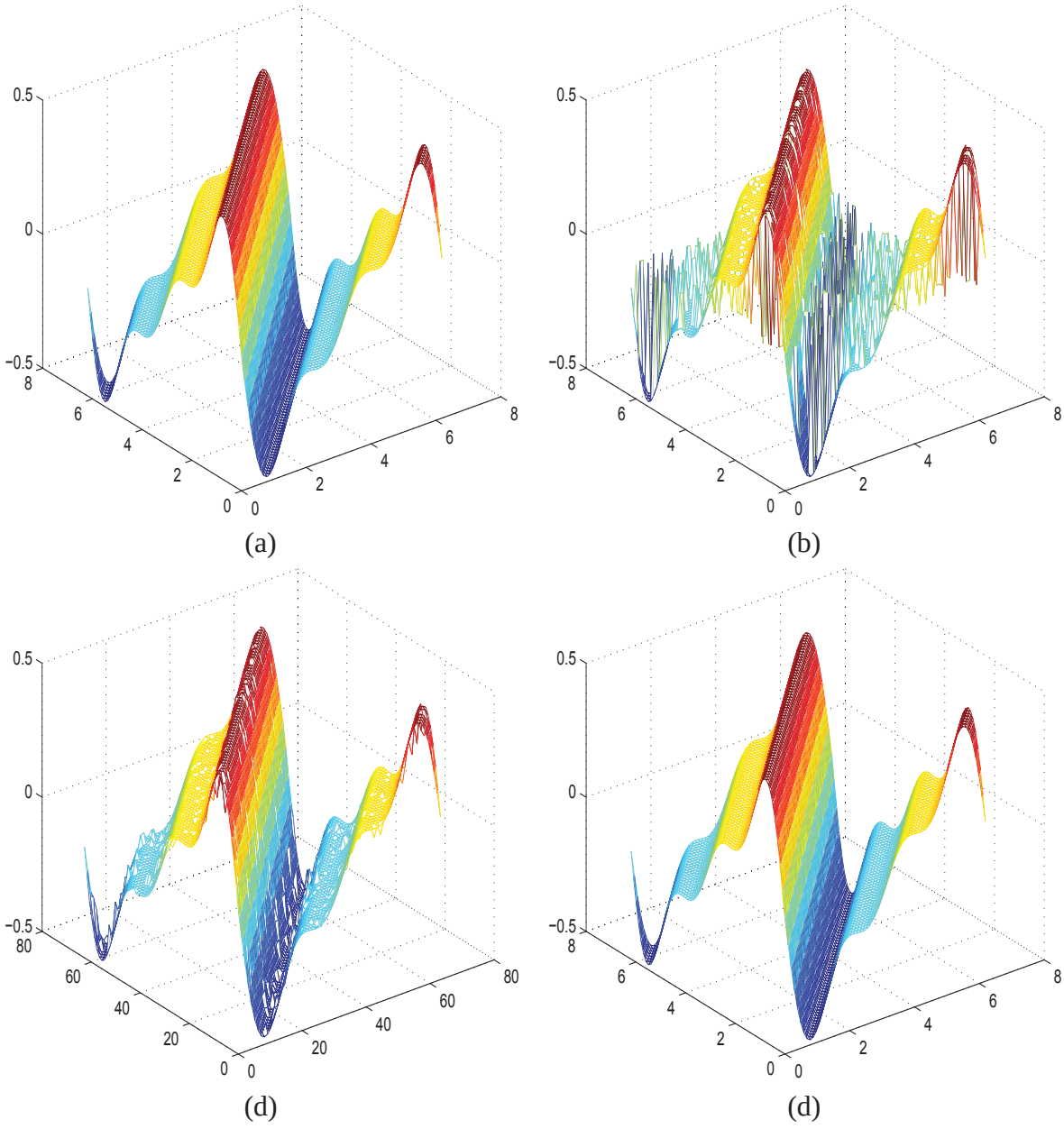


FIGURE 1. An illustration of the gappy algorithm [**18**] on a translationally invariant data. (a) The original data. (b) The gappy data. (c) Result after a single repair. (d) Final result after 3 iterations of repair.

The simple case is that of *white* noise, which is assumed to have zero mean, be uncorrelated with the signal and have a covariance matrix of the form αI where α is the variance of the noise and I is the identity matrix. In this instance, the covariance matrix of the signal may be decomposed as

$$\begin{aligned} C_{\mathbf{x}} &= \left\langle (\mathbf{s}+\mathbf{n})(\mathbf{s}+\mathbf{n})^T \right\rangle \\ &= \left\langle \mathbf{s}\mathbf{s}^T \right\rangle \left\langle \mathbf{n}\mathbf{n}^T \right\rangle \\ &= C_{\mathbf{s}} + \alpha I, \end{aligned}$$

where $C_{\mathbf{s}} = \left\langle \mathbf{s}\mathbf{s}^T \right\rangle$. The eigenvector of $C_{\mathbf{x}}$ are the same as those of $C_{\mathbf{s}}$, and the eigenvalues are all shifted upwards by the variance of the noise α, leaving the differences of the eigenvalues preserved.

The general situation is more complicated and the noise does indeed change the eigenvectors. Now the signal and noise may be separated in an optimal way by defining an appropriate variational principle. One approach for characterizing a basis in the presence of noise is to choose the direction with *maximum noise fraction* [**24**].
The optimal first basis vector, ϕ, is again taken as a superposition of the data, i.e.,

$$\phi = \psi_1 \mathbf{x}^{(1)} + \cdots + \psi_P \mathbf{x}^{(P)} = X_\psi.$$

It follows then that we may decompose ϕ into signal and noise components

$$\phi = \phi_{\mathbf{n}} + \phi_{\mathbf{s}},$$

where $\phi_{\mathbf{s}} = S\psi$ and $\phi_{\mathbf{n}} = N\psi$.
The basis vector ϕ is said to have maximum noise fraction if the ratio

$$D(\phi) = \frac{\phi_{\mathbf{n}}^T \phi_{\mathbf{n}}}{\phi^T \phi}$$

is a maximum. This may now be rewritten as

$$D(\phi(\psi)) = \frac{\psi^T N^T N \psi}{\psi^T X^T X \psi}.$$

Differentiating this with respect to ψ and setting the result to zero leads to the *symmetric definite generalized eigenproblem*

$$N^T N \psi = \mu^2 X^T X \psi. \tag{44}$$

This problem may be solved without actually forming the product matrices $N^T N$ and $X^T X$, using the generalized SVD, see [**22**].
The remaining maximum-nose-fraction basis vectors may be found using a similar approach. In fact, they are generated by the set of *generalized singular vectors* $\{\psi\}$ that come from solving Equation 44. These generalized singular vectors $\left\{\phi^{(i)}\right\}$ are ordered according to the generalized singular values $\{\mu\}$ with the largest μ corresponding to the basis vector with the **most noise**. Alternatively, the order of the generalized singular vectors may be reversed. This is convenient for data reconstruction purposes where the signal, not the noise, is the item of interest.
Notice that the signal matrix and the matrix of maximum noise fraction vectors are the same, i.e.,

$$S = \Phi.$$

The generalized singular vectors are normalized so that

$$\psi^{(j)} X^T X \psi^{(i)} \delta_{ij}.$$

Notice that they are not orthogonal with respect to the usual Euclidean inner product – the inner product is now weighted. The basis $\{\phi\}$ is, however, orthonormal.

EXAMPLE 2.1. *Noisy Time Series from a Physical Process.* As an example of the application of these ideas we consider the problem of filtering a set of $P = 7$ noisy time series shown in Figure 2(a), each of length $n = 250$. In this problem we take the data matrix X to be 250×7, where each column has had the mean removed. If we treat each of these time series as a point in $\mathbb{R}^{250}$, then the KL eigenvectors for the noisy data are shown in Figure 2(b). Note that these basis vectors are equivalent to the left-singular vectors of the data matrix X. The significant level of noise spread across all of these basis vectors is evident.
To apply the maximum-noise-fraction method for filtering the set of noisy time series we must estimate the covariance matrix of the noise, $N^T N$. **Assuming the noise is uncorrelated, the noise may be estimated simply by differencing signals shifted by one increment.** After computing N in this

way the generalized singular vectors $\{\psi\}$ were found and used to construct the orthonormal basis $\{\phi\}$ shown in Figure 2(c).

The result of projecting the first of the original seven time series onto the maximum-noise-fraction basis is shown in Figure 2(d) for one to seven modes. Observe how the noise-free portion of the signal is reconstructed last. For a closer look at the effect of truncating the noisy basis vectors, a full-mode reconstruction of the first time series is shown in Figure 3. Notice that significant detail of the signal is retained while a large amount of noise is removed.

Finally, we remark that although the method was introduced in the context of eliminating noise, it is potentially useful for separating signals in general when the covariance matrices for each component are available. Also, a striking feature of this approach is that non-differentiable functions with added noise may be recovered without smoothing, i.e., the filtered function may also be non-differentiable.

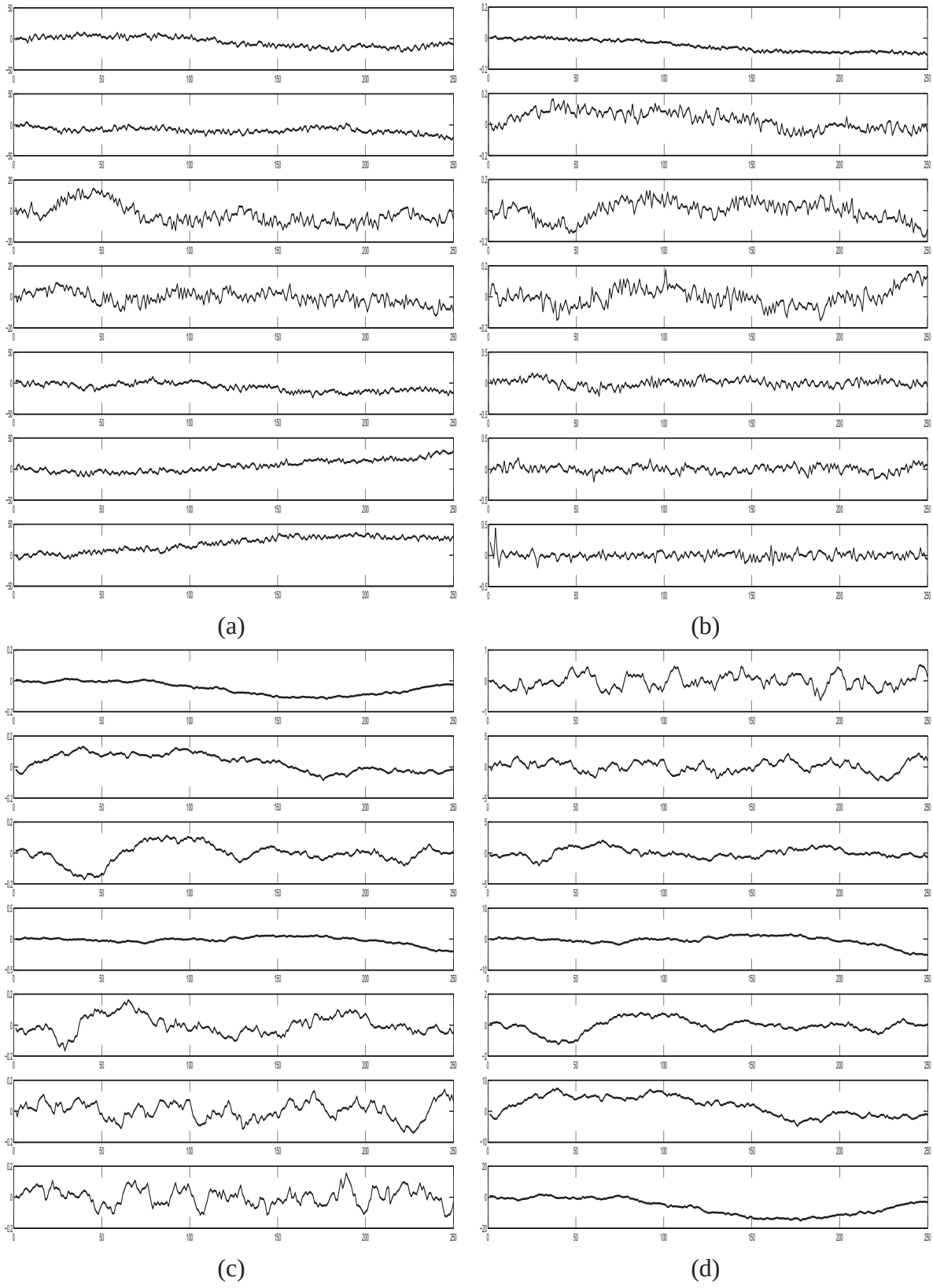


FIGURE 2. (a) A collection of seven noisy time series. (b) The orthonormal basis resulting from the noisy time series in (a). (c) The orthonormal basis resulting from implementing the maximum-noise-fraction method on seven noisy time series in (a). The basis vector with the most signal is at the top, and the one with the maximum-noise-fraction is at the bottom. (d) The result of projecting the first of the original seven time series onto the maximum-noise-fraction basis.

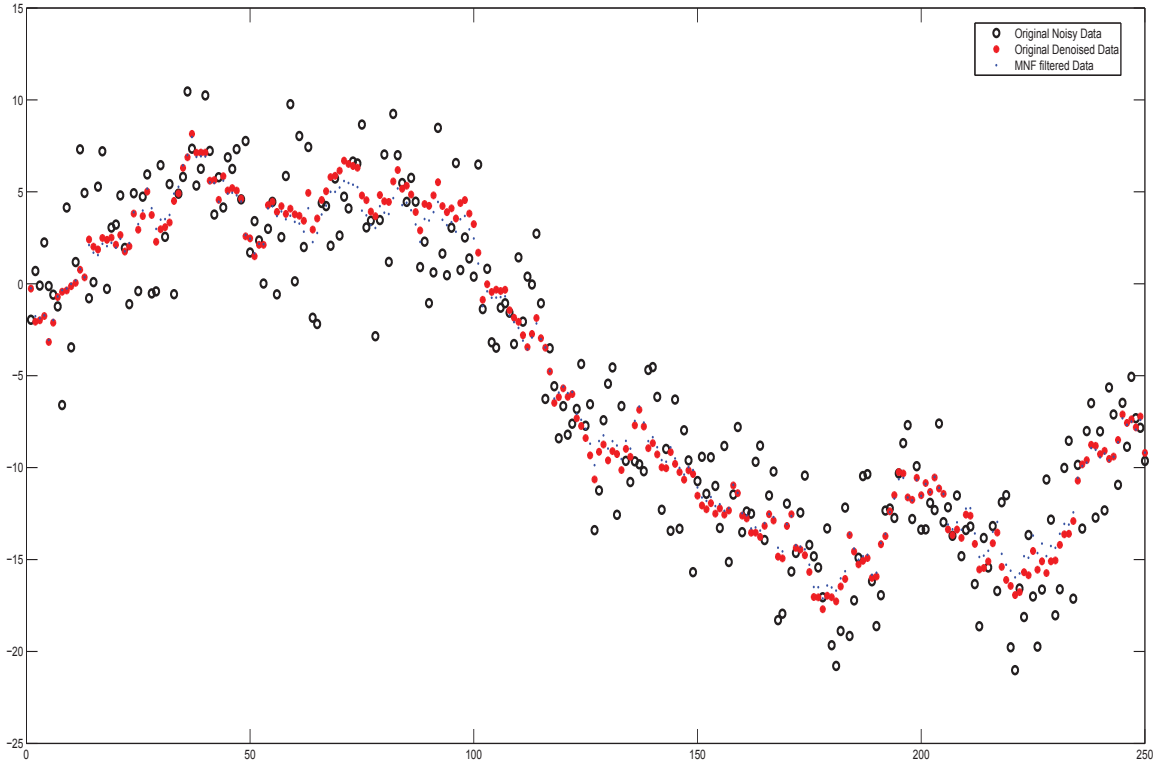


FIGURE 3. Full-mode reconstruction of the first noisy time series in Figure 2(a).

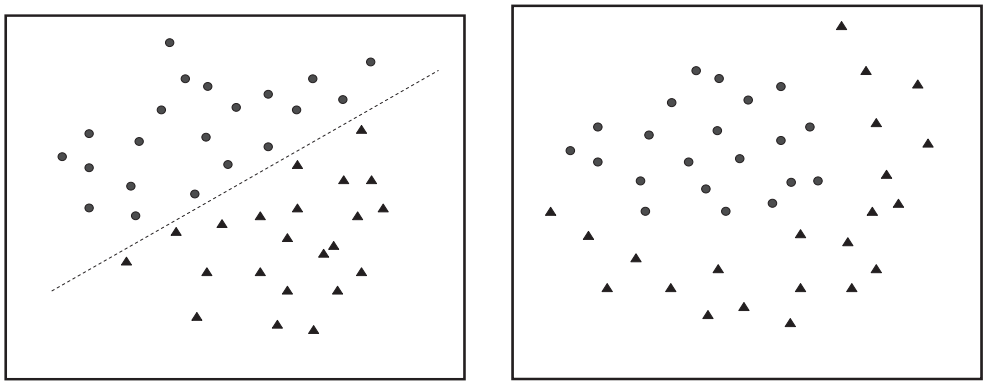
CHAPTER 4

Fisher's Linear Discriminant Analysis

Linear discriminant analysis (LDA) and the related Fisher's linear discriminant analysis (FDA) are methods used in statistics and machine learning to find a linear combination of features which characterize or separate two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.
The terms Fisher's linear discriminant and LDA are often used interchangeably, although Fisher's original article [**20**] actually describes a slightly different discriminant, which does not make some of the assumptions of LDA such as normally distributed classes or equal class covariances. In the following discussions, we will use FDA and LDA interchangeably without worries. We will motivate the method with a simple two-class classification problem first and generalize it to the multiclass case. The discussions here follow [**42**].

1. FDA for Two Classes

We will develop the idea of FDA in the context of a simple two-class problem. Namely, we will consider two classes, D_1 and D_2, of data points in $\mathbb{R}^2$. Applying FDA to this data in the plane will generate a line and a scalar which can then be used to classify novel points. The classification accuracy using Fisher's Discriminant Analysis depends on the linear separability of the classes of data. Two classes of points in $\mathbb{R}^2$ are linearly separable if it is possible to draw a line splitting the plane into two half-planes, with one class of points lying entirely on one side of the line and the other class of points lying on the opposite side of the line. Figure 1(a) illustrates two linearly separable classes of data with a separating line, and Figure 1(b) shows two non-linearly separable classes.
Of course, if data fails to be linearly separable it can be strongly or weakly non-linearly separable depending on how much the classes of data are mixed. The less the classes are mixed, the more likely that FDA will still perform fairly well. In the case where there is a severe non-linearity, it is difficult to directly compute the discriminating features between the two classes of patterns in the original input space. By defining a non-linear mapping from the input space to a high-dimensional feature



(a) (b)

FIGURE 1. (a) A linearly separable data cloud. (b) A non-linearly separable data cloud.

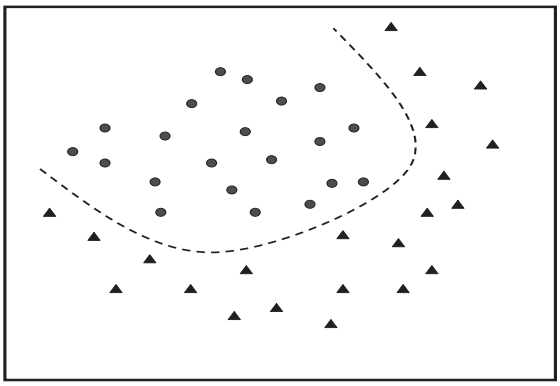


FIGURE 2. Kernel LDA. A non-linear map (i.e., kernel) can be found to separable two non-linearly separable classes.

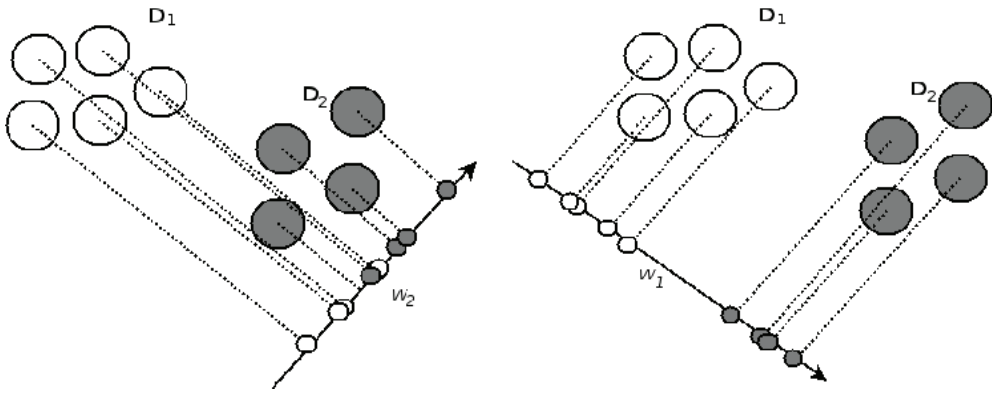


FIGURE 3. (a) A bad projection. (b) A good projection in the Linear Discriminant Analysis.

space, one can possibly obtain a linearly separable distribution in the feature space. Then LDA, the linear technique, can be performed in the feature space to extract the most significant discriminating features. This technique is generally called the Kernel LDA (KDA) [**3, 43**]. A graphical illustration is given in Figure 2.
Given a novel point $x \in \mathbb{R}^2$, the idea behind FDA is to

(1) Project x onto the line spanned by a certain unit vector w, thereby reducing the dimension of x from two to one, from a point in $\mathbb{R}^2$ to a scalar value in $\mathbb{R}$.
(2) Select the class of x based on whether $w^T x$ is above or below a certain scalar critical value α_c.

The question then becomes how to optimally select the direction of projection, w. Figure 3 compares the situations when a good and bad projection are used. When a good projection is found, two distinct classes can be separated without error; on the other hand, if a bad projection is chosen, it is impossible to find a real scalar α_c that separates the two classes perfectly. With that, optimally selecting w and α_c will be the focus of the following discussions.

1.1. Finding the Good Projection. First we discuss selecting the optimal projection, w. There are two aspects that effect the optimization of w. The goals are to separate data in distinct classes as far as possible while pull the data within the same class as close as possible.

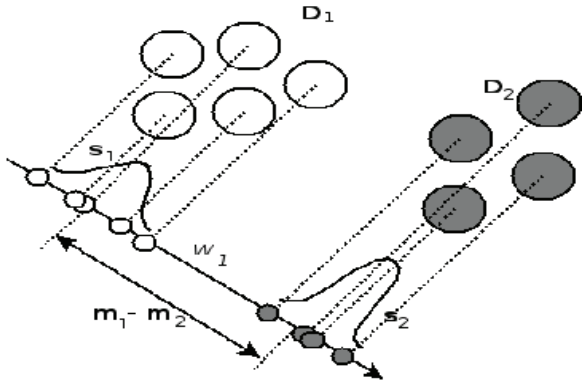


FIGURE 4. An illustration of a good LDA projection that maximizes the between-class scatter and minimizes the within-class scatter.

1.1.1. *Between-Class Separation and Within-Class Condensation.* Let n_1 and n_2 be the number of data points in D_1 and D_2, respectively. Define the class-wise means:

$$m_1 = \frac{1}{n_1}\sum_{x\in D_1} x \quad \text{and} \quad m_2 = \frac{1}{n_2}\sum_{y\in D_2} y.$$

We then define the means of the projected data accordingly as

$$\tilde{m}_1 = \frac{1}{n_1}\sum_{x\in D_1} w^T x \quad \text{and} \quad \tilde{m}_2 = \frac{1}{n_2}\sum_{y\in D_2} w^T y.$$

We desire that the projected means be far apart, in the effort to produce class-clusters that are far apart in the one-dimensional projected space. In other words, to achieve the between-class objective we find w such that

$$w = \arg\max_{w^*} \left(\tilde{m}_2 - \tilde{m}_1\right)^2. \tag{45}$$

Moreover, we want to minimize the scatter (variance) in each of the projected class clusters. Define the projected class-cluster scatter for D_1 and D_2, respectively, as

$$\tilde{S}_1^2 = \sum_{x\in D_1} (w^T x - \tilde{m}_1)^2 \quad \text{and} \quad \tilde{S}_2^2 = \sum_{y\in D_2} (w^T y - \tilde{m}_2)^2.$$

These two goals are summarized graphically in Figure 4.
Now the total within-class scatter, across all classes, is given by $\tilde{S}_1^2 + \tilde{S}_2^2$. The w that achieves the within-class objective is given by

$$\arg\min_{w} \left(\tilde{S}_1^2 + \tilde{S}_2^2\right). \tag{46}$$

In order to achieve Equations (45) and (46) at once, we seek to maximize the function

$$J(w) = \frac{(\tilde{m}_2 - \tilde{m}_1)^2}{\tilde{S}_1^2 + \tilde{S}_2^2} \tag{47}$$

over all choices of w with $\|w\|_2 = 1$, where unit length prevents the between-class scatter from being unbounded. Equation (47) is called the Fisher Criterion [**5**].

1.1.2. *Rewrite $J(w)$ and Maximize.* We can maximize J using expansions and matrix differentiation. First, we notice that we can write the projected mean in terms of the original mean

$$\begin{aligned}\tilde{m}_1 &= \frac{1}{n_1}\sum_{x\in D_1} w^T x \\ &= \frac{1}{n_1}\left(w^T x^{(i_1)} + w^T x^{(i_2)} + \cdots\right) \\ &= w^T\left(\frac{1}{n_1}\sum_{x\in D_1} x\right) \\ &= w^T m_1\end{aligned}$$

Similarly, $\tilde{m}_2 = w^T m_2$. Thus, the numerator $N(w)$ of $J(w)$ can be expanded as follows.

$$\begin{aligned}N(w) &= (\tilde{m}_2 - \tilde{m}_1)^2 = (w^T m_2 - w^T m_1)^2 = \left(w^T(m_2 - m_1)\right)^2 \\ &= w^T(m_2 - m_1)\, w^T(m_2 - m_1) = w^T(m_2 - m_1)(m_2 - m_1)^T w = w^T S_B w,\end{aligned}$$

where

$$S_B = (m_2 - m_1)(m_2 - m_1)^T \tag{48}$$

is the between-class scatter matrix.
For the denominator $D(w)$ of $J(w)$, we can write

$$\begin{aligned}D(w) &= \tilde{S}_1^2 + \tilde{S}_2^2 = \sum_{x\in D_1}\left(w^T x - \tilde{m}_1\right)^2 + \sum_{y\in D_2}\left(w^T y - \tilde{m}_2\right)^2 \\ &= \sum_{x\in D_1}\left(w^T x - w^T m_1\right)^2 + \sum_{y\in D_2}\left(w^T y - w^T m_2\right)^2 \\ &= \sum_{x\in D_1}\left(w^T(x - m_1)(x - m_1)^T w\right) + \sum_{y\in D_2}\left(w^T(y - m_2)(y - m_2)^T w\right) \\ &= w^T \sum_{i=1,2}\sum_{x\in D_i}(x - m_i)(x - m_i)^T w = w^T S_W w\end{aligned}$$

where

$$S_W = \sum_{i=1,2}\sum_{x\in D_i}(x - m_i)(x - m_i)^T \tag{49}$$

is the within-class scatter matrix.
We can finally write the Fisher Criterion as

$$J(w) = \frac{N(w)}{D(w)} = \frac{w^T S_B w}{w^T S_W w},$$

which is commonly known as the generalized Rayleigh quotient [**14**].
Setting $\nabla J(w) = 0$, a necessary condition for a global maximum, gives the generalized eigenvalue problem (see Appendix 5)

$$S_B w = \lambda S_W w. \tag{50}$$

But which of the generalized eigenvectors that satisfy Equation (50) is the optimal w? We notice that

$$\lambda = \frac{N(w)}{D(w)} = J(w)$$

is precisely the Fisher Criterion. Since we want to maximize $J(w) = \lambda$ we will find w as the generalized eigenvector corresponding to the largest generalized eigenvalue solving Equation (50).

1.2. Optimal α_c. Finally, the task of selecting an optimal α_c can be approached in a variety of ways. Several methods are available for heuristically selecting the threshold value, α_c. One method is to let α_c be the average of the projected means

$$\alpha_c = \frac{\tilde{m}_1 + \tilde{m}_2}{2}.$$

. In another method, one can assume, without loss of generality, that the projected class-cluster of D_1 lies above the projected class-cluster of D_2. Then it is reasonable to set α_c to be the average of the minimum projected element of D_1, call it $\hat{x}$, and the maximum projected element of D_2, call it $\hat{y}$, i.e.,

$$\alpha_c = \frac{\hat{x} + \hat{y}}{2}.$$

Or, one can select α_c by cross-validation. Namely, α_c is chosen empirically to maximize classification rates for validation data.

2. Multiclass FDA

In the case where there are more than two classes, the analysis used in the previous section can be extended to find a subspace which appears to contain all of the class variability.
We begin by supposing there are $M > 2$ classes, $D_1, D_2, ..., D_M$, partitioning n points $x_i \in \mathbb{R}^N, i = 1, ..., n$, where there are n_i points in each D_i. Let $X = \{x_i \in \mathbb{R}^N\}_{i=1}^n$. We take $l < N$ copies of the linear projection formula from the two class, two-dimensional case,

$$y_k = w_k^T x, \; k = 1, ..., l,$$

where each number y_k is the result of projection of x onto the line spanned by vector w_k. Combining these into a matrix equation yields

$$\mathbf{y} = W^T x$$

where $W = [w_1 \, w_2 \cdots w_l]$ is $N \times l$ and $\mathbf{y} = [y_1 \, y_2 \cdots y_l]^T$ is $l \times 1$. Thus, the projection of each x_i by W is an associated $\mathbf{y}_i$. This generates the set Y of n projected points, $Y = \{\mathbf{y}_i \in \mathbb{R}^l\}_{i=1}^n$. Each of the $\mathbf{y}_i$ maintains the same class label as its counterpart x_i. We want to find the best projection W, where the definition of quality has been extended from the two class problem in a natural way.
We are now ready to generalize the definitions of S_W and S_B for the multiclass, high-dimensional case, as shown in [**14**]. For the original data X, the within-class scatter is the sum of individual class variances

$$S_W = \sum_{i=1}^{M} \sum_{x \in D_i} (x - m_i)(x - m_i)^T.$$

The definition of the within-class scatter matrix is somewhat intuitive, and matches exactly the form in Equation (49). Each outer product used to create S_W communicates a measure of the variance of a data point from the mean, consisting of every pairwise comparison of entries. On the other hand, the definition of the between-class scatter matrix is not as intuitive. It arises from considering the total scatter

$$S_T = \sum_{i=1}^{n} (x_i - \hat{m})(x_i - \hat{m})^T,$$

where $\hat{m}$ is the mean across all data points in X. If we let the total scatter be the sum of the within-class and between-class scatter matrices,

$$S_T = S_W + S_B,$$

then we solve for the between-class scatter matrix by

$$S_B = S_T - S_W = \sum_{i=1}^{M} n_i (m_i - \hat{m})(m_i - \hat{m})^T.$$

Now we consider the projected data Y. If we let $\tilde{m}$ be the mean of Y and $\tilde{m}_i$ be the mean of the projected data in class D_i, we have the natural extension to the projected within-class scatter matrix

$$\tilde{S}_W = \sum_{i=1}^{M} \sum_{\mathbf{y} \in D_i} (\mathbf{y} - \tilde{m}_i)(\mathbf{y} - \tilde{m}_i)^T.$$

and the projected between-class scatter matrix

$$\tilde{S}_B = \sum_{i=1}^{M} n_i (\tilde{m}_i - \tilde{m})(\tilde{m}_i - \tilde{m})^T.$$

After simplification, we arrive at $\tilde{S}_W = W^T S_W W$ and $\tilde{S}_B = W^T S_B W$. We can produce a scalar measuring scatter using the determinant of the products, since the determinant is the product of the eigenvalues, and the magnitude of the product of eigenvalues corresponds to variance. In this light, the Fisher Criterion becomes

$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|},$$

a ratio of determinants. The columns of W are found as the l eigenvectors corresponding to the l largest eigenvalues, in descending order, of the generalized eigenvalue problem

$$S_B w = \lambda S_W w.$$

Higher-dimensional ambient spaces lead to new notions of α_c. The purpose of α_c is to separate projected data into distinct classes. This allows for numerous ways to define α_c, as with the two-class problem in $\mathbb{R}^2$, but what is important about the general FDA is that it gives us a transformation matrix W with the aforementioned favorable properties.

CHAPTER 5

Convolution in Digital Images

In this chapter we define the convolution operator $\star$ and become familiar with it. We then develop some basic properties of convolution and next, study the interplay between the convolution with filter and the Fourier series of the filter.

1. Convolution and Correlation

Convolution on two functions s and f can be viewed as a way to producing a modified version of s or f, whichever is appropriate. The formal definition is stated in the continuous form:

$$(f \star s)(t) = \int_{-\infty}^{\infty} f(\tau) \cdot s(t-\tau)\, d\tau = \int_{-\infty}^{\infty} s(\tau) \cdot f(t-\tau)\, d\tau.$$

In the discrete case,

DEFINITION 1.1. Let s and f be two bi-infinite sequences, where $s = (\ldots, s_{-2}, s_{-1}, s_0, s_1, s_2, \ldots)$ and $f = (\ldots, f_{-2}, f_{-1}, f_0, f_1, f_2, \ldots)$. Then the *convolution* product y of s and f, denoted by $f \star s$ is the bi-infinite sequence whose nth component is given by

$$y_n = \sum_{k=-\infty}^{\infty} s_k f_{n-k}. \tag{51}$$

In general, Equation 51 will diverge unless appropriate conditions are placed on s and f. In our applications, either s will be non-zero on a set of measure zero or terms in s and f will decay rapidly to ensure convergence.

A seemingly familiar concept that resembles a lot of the characteristics of convolution will be discussed next. Statistically speaking, the (Pearson) *correlation* $\rho_{X,Y}$ between two random variables X and Y is defined as

$$\rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y},$$

where $E(X)$ and $E(Y)$ refer to the expected value of X and Y, respectively. Since $\mu_X = E(X)$, the mean of X, and $\sigma_X^2 = E[(X-E(X))^2] = E(X^2) - E^2(X)$, the variance of X, the equation above becomes

$$\rho_{X,Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)}\sqrt{E(Y^2) - E^2(Y)}}.$$

In terms of finite signals, we have an expression that is similar to Equation 51

$$r_{sf} = \frac{1}{n-1} \cdot \frac{1}{\sigma_s \sigma_f} \sum_{i=1}^{n} (s_i - \mu_s)(f_i - \mu_f).$$

Notice that the signal has one fewer degree of freedom when the mean is taken out, explaining the division by $n-1$ instead of n.

These two concepts inspire the use of *correlation* and *convolution* in image processing applications. Intuitively, correlation is like the process of moving a filter mask (f over a signal (s) or image and computing the sum of products at each location while convolution works similarly except that the filter f is first rotated 180°. A simple 1D example is given in Figure 1 to illustrate the similarity and difference of these two concepts.

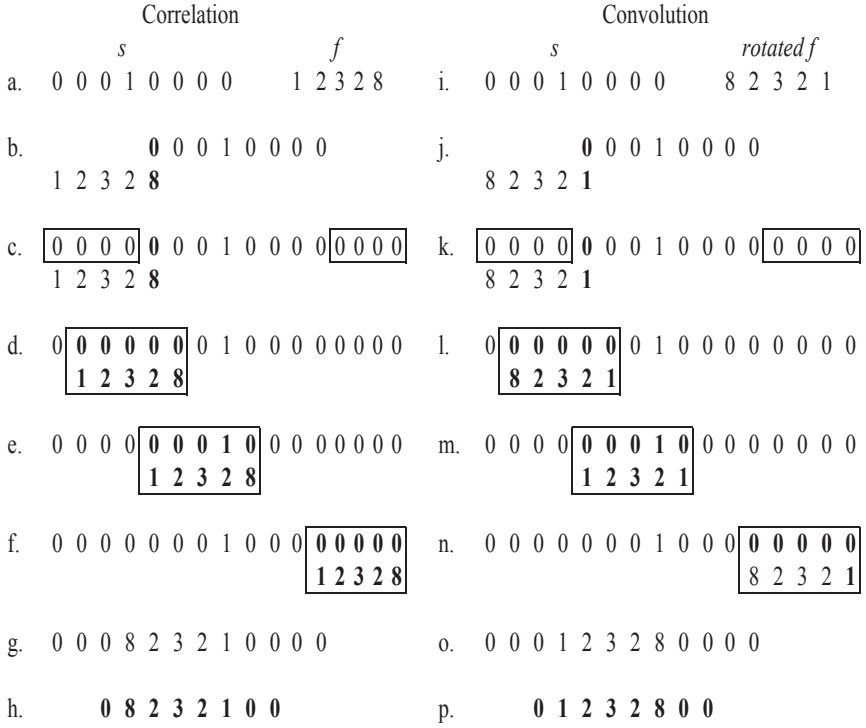


FIGURE 1. s is a discrete unit pulse of length 8 and f is a filter of length 5. The processes of correlation and convolution are illustrated on the left and right panels, respectively. In steps b. and j., the align the signal and filter. In steps c. and k., appropriate number of zeros are added to pad the empty cells in order to perform multiplications in the next step. In general, if the filter of of size m, then we need $m-1$ zeros on either side of s. Plots d. (resp. l.) and e. (resp. m.) show the positions of the filter after one and four shifts, respectively. For example, the result of convolution at four shifts is calculated from $0\cdot 8+0\cdot 2+0\cdot 3+1\cdot 2+0\cdot 1=2$, which corresponds to the first 2 from the left in the full convolution result shown in o. Finally, the full-length correlation and convolution are cropped to match the length of the original signal, s. Notice that the only major difference between the two methods is the action of rotating the filter mask in convolution before performing the product and sum.

We proceed with a similar fashion in the case of 2-dimensional signals, such as the case of images. In general, if the filter is of size $m\times n$, we first pad the image s with a minimum of $m-1$ rows of zero's on the top and bottom and $n-1$ columns of zero's on the left and right. The rest of the correlation and convolution process is similar to the 1D case and is illustrated in Figure 2.
Mathematically, we compute discrete 2D correlation with

$$f(x,y)*s(x,y)=\sum_{u=-a}^{a}\sum_{v=-b}^{b}f(u,v)s(x+u,y+v)=\sum_{u=-a}^{a}\sum_{v=-b}^{b}s(u,v)f(x+u,y+v)$$

and discrete 2D convolution with

$$f(x,y)\star s(x,y)=\sum_{u=-a}^{a}\sum_{v=-b}^{b}f(u,v)s(x-u,y-v)=\sum_{u=-a}^{a}\sum_{v=-b}^{b}s(u,v)f(x+u,y+v).$$

There are a few things to pay attention to here. First, both correlation and convolution are commutative, which means that it is irrelevant what we consider as a signal and what we consider as a filter.

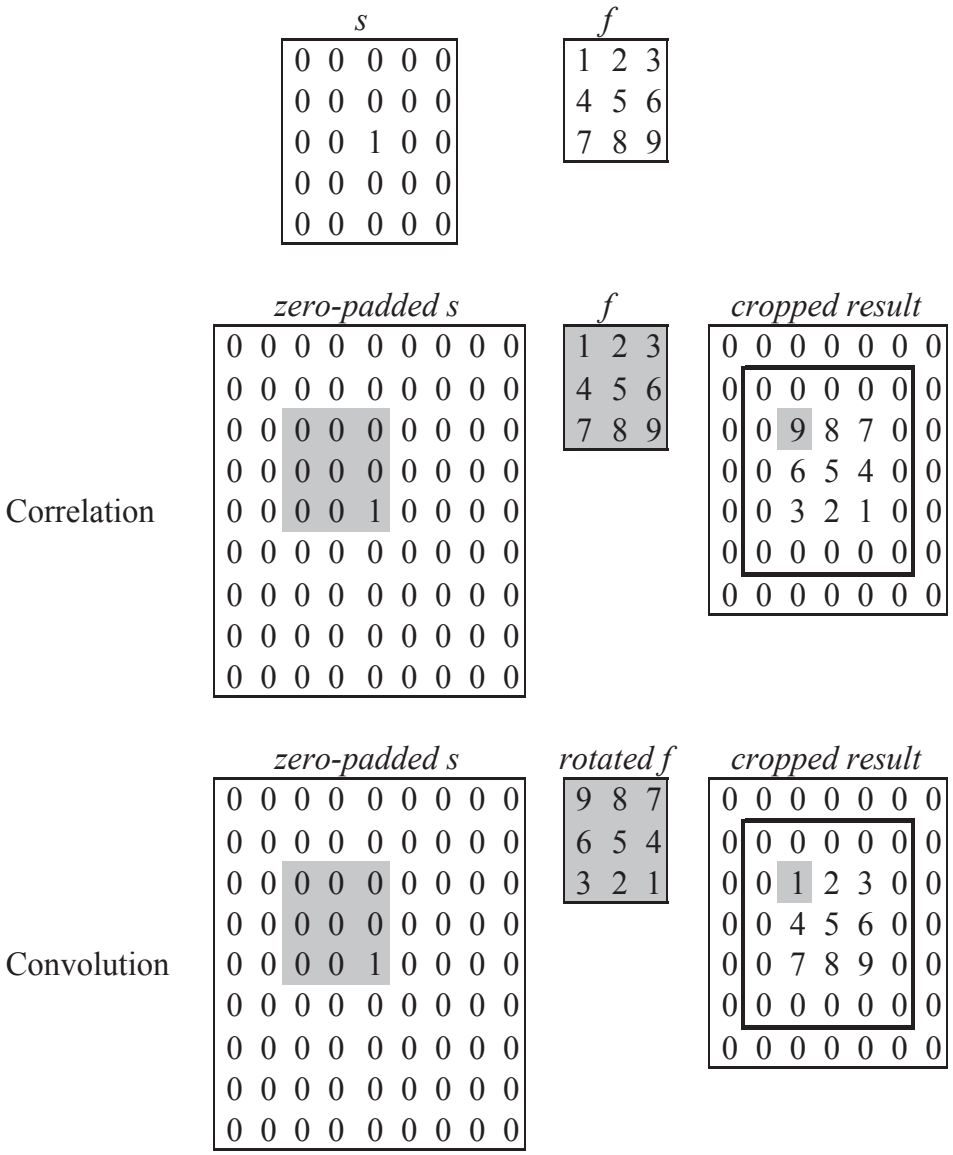


FIGURE 2. In this figure, s is the image that we are interested in studying while f is a filter window of size 3×3. The processes of 2D correlation and convolution are illustrated on the top and bottom panels, respectively. In both incidences, s is first padded with two rows of zeros on the top and bottom and two columns of zero on the left and right. The first non-zero result of correlation and convolution is highlighted as one slides f through the zero-padded s from the upper left corner to the lower-right corner. At the end of a series of matrix multiplications, we retain the boxed region as the final result of correlation and convolution.

Although in practice, filters are often smaller in size. Another important detail to notice is the subtraction in f in the second equation captures exactly the 180°rotation mentioned earlier. Now that we have acquainted ourselves with the notation and definitions of convolution, we will next briefly discuss how convolution is used in a lot of image processing applications.

2. Convolution as Filters

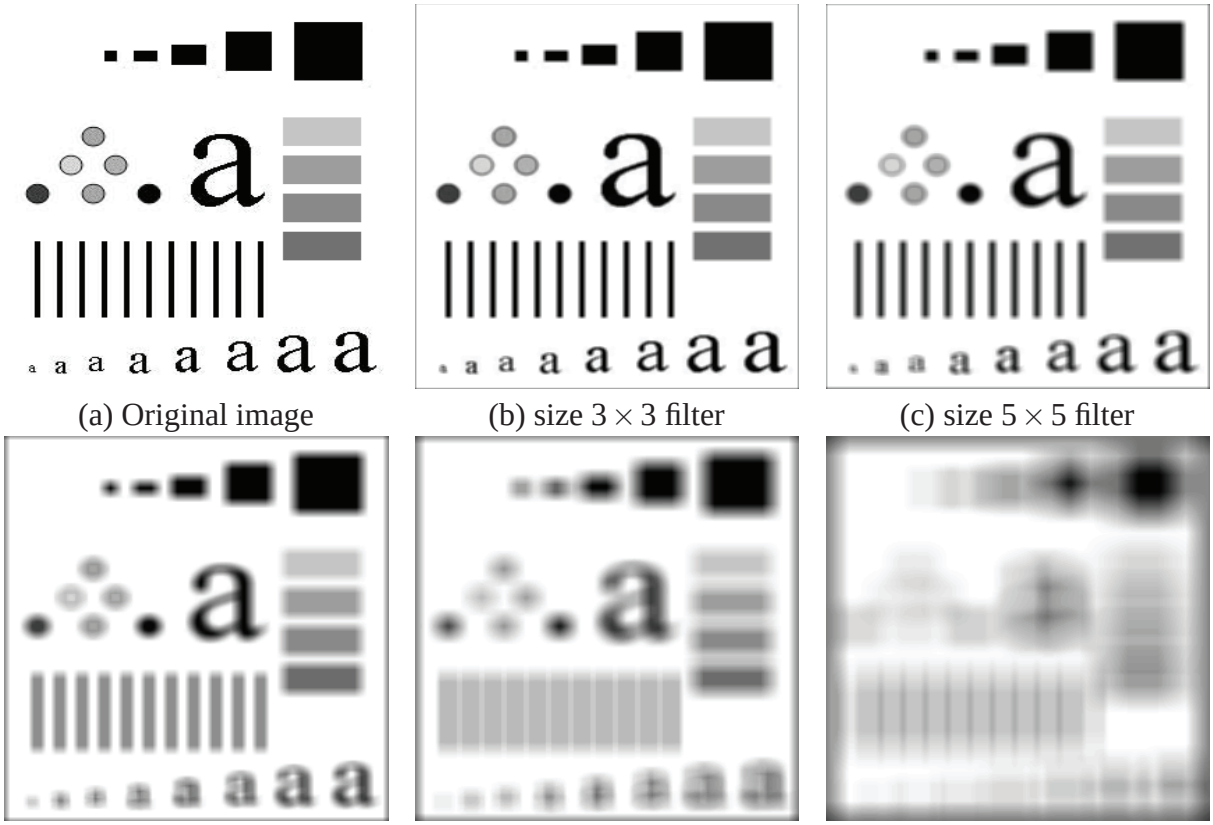
Convolution is considered as a form of *spatial filtering* in manipulating images. Common synonyms include convolution mask, convolution kernel, convolving a mask with an image, etc. The name filter is borrowed from frequency domain processing. Filtering refers to accepting/passing or rejecting certain frequency components. For example, a *low-pass* filter passes low-frequency information. In images, such filters correspond to the action of smoothing or blurring. On the other hand, a *high-pass* filter damps low frequency information while maintains high-frequency content. In images, such filters correspond to the action of sharpening. The theory on how to construct appropriate filters for various tasks has grown tremendously over the past few decades, for detailed information, please refer to [**23**]. We will only introduce a few simple yet fundamental filters from the literature here.

2.1. Smoothing Spatial Filters.

- Linear filters. The linearity in this context refers to the ability to represent the filter in a matrix form. A typical technique in constructing such filters is through the use of integration. Namely, summing and averaging. An averaging filter results in an image with reduced sharp transitions in intensities. Because random noise typically consists of sharp transitions in intensity levels, the most obvious application of smoothing is noise reduction. However, smoothing also has the undesirable side effect of edge blurring. A typical 3×3 averaging filter looks like
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$
The appropriate size of the filter depends on the application. Figure 3 depicts the various effects created by using filters of various sizes. In general, averaging filters of larger size have a more significant effect of blurring.

  Another type of averaging filter is a weighted average filter where neighbor pixels are inversely weighted as a function of distance from the center. In particular, the diagonal neighbors are considered as further away from the orthogonal neighbors of the center. A typical 3×3 weighted average filter looks like the following.
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$
The result of applying this filter to an image is shown in Figure 4 for convenience.

  An important application of spatial averaging is to blur an image for the purpose of getting a gross representation of objects of interest, such that the intensity of smaller objects blends with the background and larger objects become "bloblike" and easy to detect. The size of the filter/mask establishes the relative size of the objects that will be blended with the background. As an illustration, consider Figure 5(a), which is an image from the Hubble telescope. Figure 5(b) shows the result of applying a 15×15 averaging mask to this image We see that a number of objects have either blended with the background or their intensity has diminished considerably. It is typical to follow an operation like this with thresholding to eliminate objects based on their intensity. The result of using the thresholding function with a threshold value equal to 25% of the highest intensity in the blurred image is shown in Figure 5(c). Comparing this result with the original image, we see that is is a reasonable representation of what we would consider to be the largest, brightest objects in that image.
- Nonlinear filters. Filters of this sort can not be expressed nicely in a closed matrix form. This sort of filter is based on ordering (ranking) the pixels contained in the image area encompassed by the filter, and then replacing the value of the center pixel with the value determined



(a) Original image (b) size 3×3 filter (c) size 5×5 filter

(d) size 9×9 filter (e) size 15×15 filter (f) size 35×35 filter

FIGURE 3. (a) Original image, of size 906×712 pixels. (b)-(f) Results of smoothing with square averaging filter of sizes $m = 3, 5, 9, 15$, and 35, respectively.

(a) (b) (c)

FIGURE 4. (a) Original image, of size 906×712 pixels. (b) Result of applying the filter $\frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. (c) Result of applying the filter $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$.

by the ranking result. The best-known filter in this category is the *median* filter (among others such as max and min filters), which replaces the value of a pixel by the median of the intensity values in the neighborhood of that pixel. Median filters are quite popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with

(a) (b) (c)

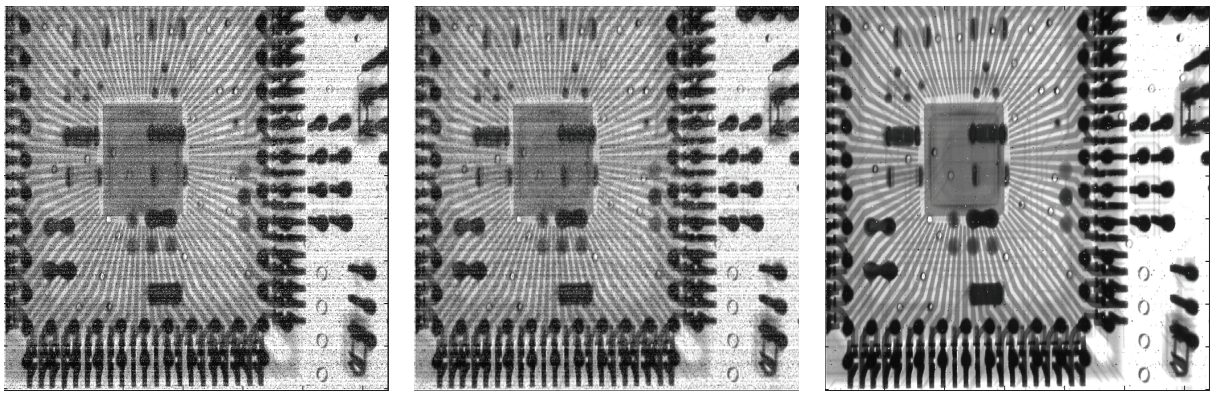
FIGURE 5. (a) Image of size 399×395 pixels from the Hubble Space Telescope. (b) Image Filtered with a 15×15 averaging filter. (c) Result of thresholding (b) with 25% of highest intensity.

considerably less blurring than linear smoothing filters of similar size. They are particularly useful in the presence of impulse noise (a.k.a. salt-and-pepper noise), intensity spikes caused by error in data transmission.

For example, suppose that a 3×3 neighborhood has values $(10, 20, 20, 20, 15, 20, 20, 25, 100)$. These values are sorted as $(10, 15, 20, 20, 20, 20, 20, 25, 100)$, which results in a median of 20. Thus, the principal function of median filters is to force points with distinct intensity levels to be more like their neighbors. In fact, isolated clusters of pixels that are light or dark with respect to their neighbors and whose area is less than $m^2/2$ (one-half of the filter area), are eliminated by an $m \times m$ median filter. In this case, "eliminated" means forced to the median intensity of the neighbors. Larger clusters are affected considerably less.

Figure 6(a) shows an X-ray image of a circuit board heavily corrupted by salt-and-pepper noise. To illustrate the point about the superiority of median filtering over average filtering in situations such as this, we show in Figure 6(b) the result of processing the noisy image with a 3×3 neighborhood averaging mask, and in Figure 6(c) the result of using a 3×3 median filter. The averaging filter blurred the image and its noise reduction performance was poor. The superiority in all respects of median over average filtering in this case is quite evident. In general, median filtering is much better suited than averaging for the removal of salt-and-pepper noise.

2.2. Sharpening Spatial Filters. The principal objective of sharpening is to highlight transitions in intensity. Uses of image sharpening vary and include applications ranging from electronic printing and medical imaging to industrial inspection and autonomous guidance in military systems. Contrary to averaging, sharpening can be accomplished by spatial differentiation. This boils down to numerically taking derivatives of sequence of numbers. The derivatives of a digital function are defined in terms of differences. There are various ways to define these differences. However, we require that any definition we use for a first derivative (1) must be zero in areas of constant intensity; (2) must be nonzero at the onset of an intensity step or ramp; and (3) must be nonzero along ramps. Similarly, any definition of a second derivative (1) must be zero in constant areas; (2) must be nonzero at the onset and end of an intensity step or ramp; and (3) must be zero along ramps of constant slope. Because we are dealing with digital quantities whose values are finite, the maximum possible intensity change also is finite, and the shortest distance over which that change can occur is between adjacent pixels.



(a) (b) (c)

FIGURE 6. (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging filter. (c) Noise reduction a 3×3 median filter.

A basic definition of the first-order derivative of a one-dimensional function $f(x)$ is the difference

$$\frac{\partial f}{\partial x} = f(x+1) - f(x).$$

We used a partial derivative here in order to keep the notation the same as when we consider an image function of two variables, $f(x,y)$, at which time we will be dealing with partial derivatives along the two spatial axes. We define the second-derivative of $f(x)$ as the difference

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x).$$

It is easy to verify that these two definitions satisfy the conditions stated above. To illustrate this, and to examine the similarities and differences between first- and second-derivatives of a digital function, consider the example in Figure 7. Center of the Figure 7 shows a section of a scan line (intensity profile). The values inside the small squares are the intensity values in the scan line, which are plotted as black dots above it. As the figure shows, the scan line contains an intensity ramp, three sections of constant intensity, and an intensity step. The circles indicate the onset or end of intensity transitions. The first- and second-order derivatives computed using the two preceding definitions are included below the scan line and plotted on the bottom of the figure.

Let us consider the properties of the first and second derivatives as we traverse the profile from left to right. First, we encounter an area of constant intensity, where both derivatives are zero there so condition (1) is satisfied for both. Next, we encounter an intensity ramp followed by a step, and we note that the first-order derivative is nonzero at the onset of the ramp and the step; similarly, the second derivative is nonzero at the onset and end of both the ramp and the step; therefore, property (2) is satisfied for both derivatives. Finally, we see that property (3) is satisfied also for both derivatives because the first derivative is nonzero and the second is zero along the ramp. Note that the sign of the second derivative changes at the onset and the end of a step or ramp. In fact, we see in the bottom of Figure 7 that in a step transition a line joining these two values crosses the horizontal axis midway between the two extremes. This *zero crossing* property is quite useful for locating edges.

Edges in digital images often are ramp-like transitions in intensity, in which case the first derivative of the image would result in thick edges because the derivative is nonzero along a ramp. On the other hand, the second derivative would produce a double edge one pixel thick, separated by zeros. From this, we conclude that the second derivative enhances fine detail much better than the first derivative, a property that is ideally suited for sharpening images. Also, second derivatives are much easier to implement than first derivatives, so we focus our attention initially on second derivatives.

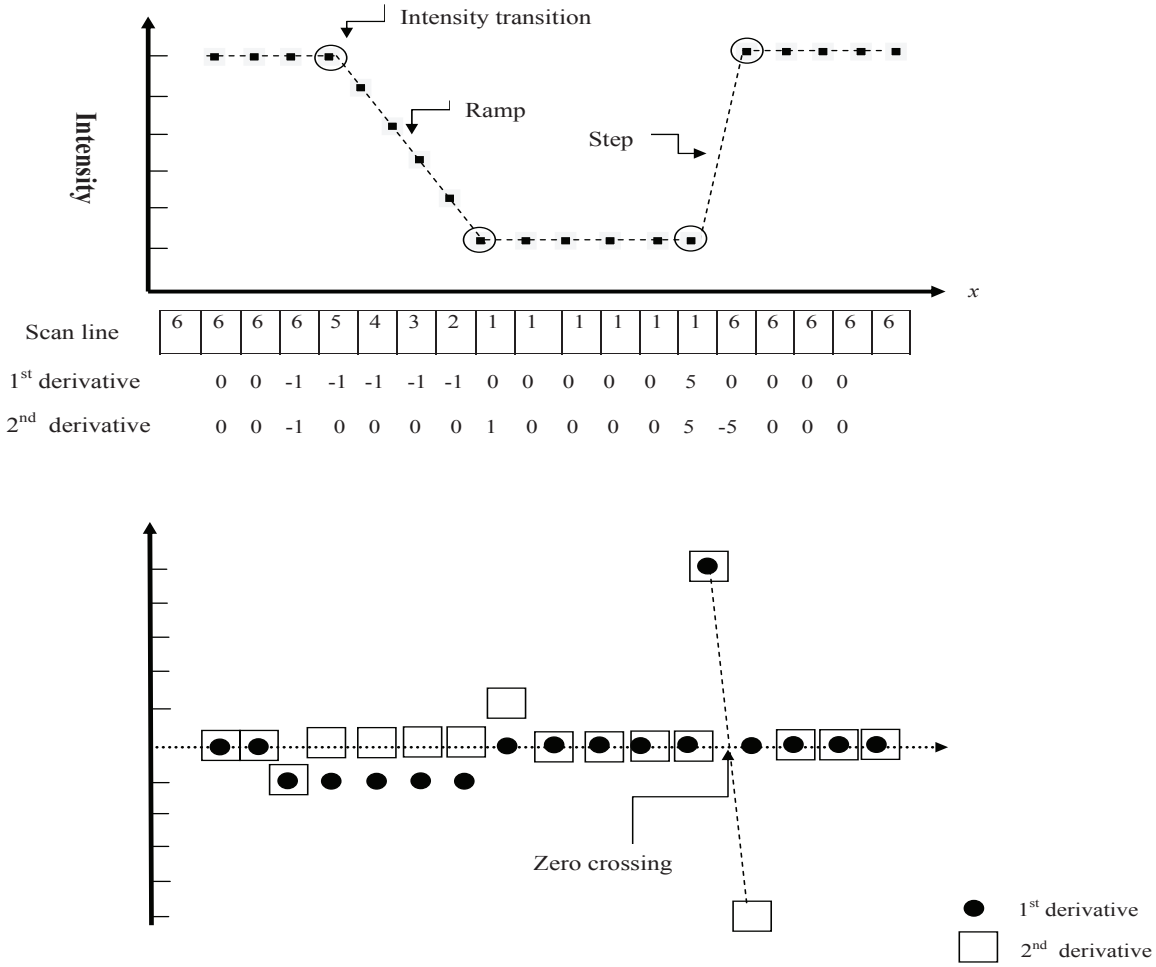


FIGURE 7. Illustration of the first and second derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image.

Here, we consider the implementation of 2-D, second-order derivatives and their use for image sharpening. We are interested in *isotropic* filters, whose response is independent of the direction of the discontinuities in the image to which the filter is applied. In other words, isotropic filters are *rotation invariant*, in the sense that rotating the image and then applying the filter gives the same result as applying the filter to the image first and then rotating the result.
It can be shown that the simplest isotropic derivative operator is the Laplacian, which, for a function (image) $f(x,y)$ of two variables, is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

Because derivatives of any order are linear operations, the Laplacian is a linear operator. To express this equation in discrete form, we use the definition introduced previously, keeping in mind that we have to carry a second variable. In the *x*-direction, we have

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \tag{52}$$

and, similarly, in the *y*-direction we have

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \tag{53}$$

Therefore, it follows that the discrete Laplacian of two variables is

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y). \tag{54}$$

This equation can be implemented using the filter mask

$$\begin{bmatrix} & (x,y-1) & \\ (x-1,y) & (x,y) & (x+1,y) \\ & (x,y+1) & \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{55}$$

which gives an isotropic result for rotations in increments of 90°.

The diagonal directions can be incorporated in the definition of the digital Laplacian by adding two more terms to Equation (54), one for each of the two diagonal directions. The form of each new term is the same as either Equation (52) or (53), but the coordinates are along the diagonals. Because each diagonal term also contains a $-2f(x,y)$ term, the total subtracted from the difference terms now would be $-8f(x,y)$. This new definition can be implemented with the mask

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{56}$$

This mask yields isotropic results in increments of 45°. You are likely to see in practice the Laplacian masks

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}.$$

They are obtained from definitions of the second derivatives that are the negatives of the ones we used in Equations (52) and (53). As such, they yield equivalent results, but the difference in sign must be kept in mind when combining (by addition or subtraction) a Laplacian-filtered image with another image.

Because the Laplacian is a derivative operator, its use highlights intensity discontinuities in an image and deemphasizes regions with slowly varying intensity levels. This will tend to produce images that have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background. Background features can be "recovered" while still preserving the sharpening effect of the Laplacian simply by adding the Laplacian image to the original. As noted earlier, it is important to keep in mind which definition of the Laplacian is used. If the definition used has a negative center coefficient, then we *subtract*, rather than add, the Laplacian image to obtain a sharpened result. Thus, the basic way in which we use the Laplacian for image sharpening is

$$g(x,y) = f(x,y) + c[\nabla^2 f(x,y)],$$

where $f(x,y)$ and $g(x,y)$ are the input and sharpened images, respectively. The constant is $c = -1$ if the Laplacian filters in Equation (55) or (56) are used, and $c = 1$ if either of the other two is used.

Figure 8(a) shows a slightly blurred image of a CT scan. Figure 8(b) shows the result of filtering this image with the Laplacian mask in Equation (55). Finally, Figure 8(c) shows the result of adding the original image to the Laplacian. By doing this, it restored the overall intensity variations in the image, with the Laplacian increasing the contrast at the locations of intensity discontinuities. The net result is an image in which small details were enhanced and the background tonality was reasonably preserved.

(a) (b) (c)

FIGURE 8. (a) Blurred image of a CT scan. (b) Laplacian mask in Equation (55). (c) Image sharpened using the mask in (b).

CHAPTER 6

Fourier Analysis

This chapter is concerned primarily with establishing a foundation for the Fourier transform and how it is used in basic image filtering. Most of the materials in this chapter follows [**23**].

The most significant contribution of the French mathematician Jean Baptiste Joseph Fourier in the field of image processing is perhaps the fact that any periodic function can be expressed as the sum of sines and/or cosines of different frequencies, each multiplied by a different coefficient (we now call this sum a *Fourier series*). It does not matter how complicated the function is; if it is periodic and satisfies some mild mathematical conditions, it can be represented by such a sum. This is now taken for granted but, at the time it first appeared, the concept that complicated functions could be represented as a sum of simple sines and cosines was not at all intuitive, see e.g., Figure 1, so it is not surprising that Fourier's ideas were met initially with skepticism.

Even functions that are not periodic (but whose area under the curve is finite) can be expressed as the integral of sines and/or cosines multiplied by a weighing function. The formulation in this case is the *Fourier transform*, and its utility is even greater than the Fourier series in many theoretical and

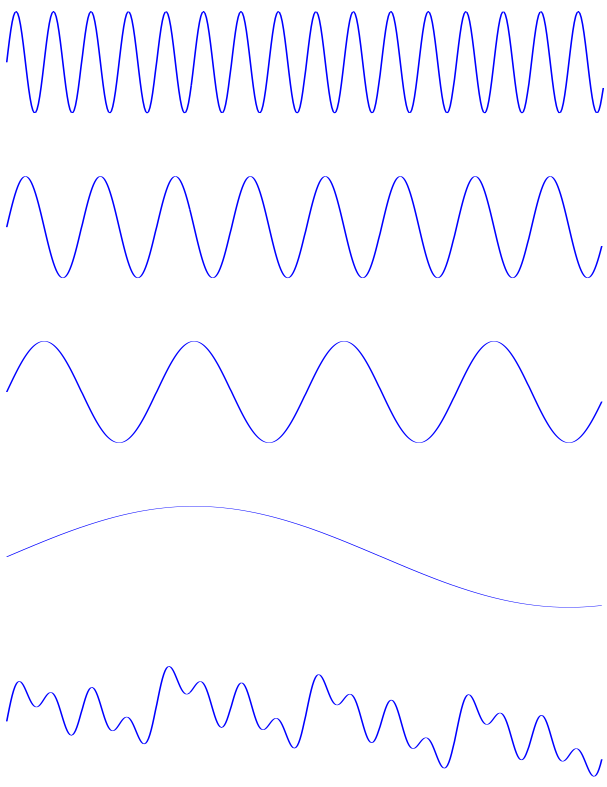


FIGURE 1. The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

applied disciplines. Both representations share the important characteristic that a function, expressed in either a Fourier series or transform, can be reconstructed (recovered) completely via an inverse process, with no loss of information. This is one of the most important characteristics of these representations because it allows us to work in the "Fourier domain" and then return to the original domain of the function without losing any information. Ultimately, it was the utility of the Fourier series and transform in solving practical problems that made them widely studied and used as fundamental tools. We will be dealing primarily with functions (images) of finite duration, so the Fourier transform is the tool in which we are interested. The material in the following sections introduces the Fourier transform and the frequency domain. It is shown that Fourier techniques provide a meaningful and practical way to study and implement a host of image processing approaches.

1. Review of Complex Exponential Functions

One of the most famous functions in all of mathematics is *Euler's formula*. The result expresses a complex exponential function int terms of cosine and sine. The result has numerous applications and we will see that the formula is important we we design tools to process digital signals and images. Recall Taylor's series for $e^t = 1 + t + \frac{t^2}{2!} + \frac{t^3}{3!} + \cdots$. Thus,

$$\begin{aligned} e^{i\theta} &= 1 + (i\theta) + \frac{(i\theta)^2}{2!} + \frac{(i\theta)^3}{3!} + \cdots \\ &= 1 + i\theta - \frac{\theta^2}{2!} - \frac{i\theta^3}{3!} + \frac{\theta^4}{4!} + \cdots \\ &= \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - + \cdots\right) + i\left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - + \cdots\right) \\ &= \cos\theta + i\sin\theta. \end{aligned}$$

Thus we have Euler's formula

$$e^{i\theta} = \cos\theta + i\sin\theta.$$

So $e^{i\theta}$ gives a graph of a circle on the complex plane. We can easily compute the conjugate as follows:

$$\overline{e^{i\theta}} = \overline{\cos\theta + i\sin\theta} = \cos\theta - i\sin\theta = e^{-i\theta}.$$

We will also be needing the notion of N^{th} roots of unity later in the discussion of Fourier transform. In particular, the N^{th} roots of unity are given by $\{e^{i2\pi k/N}\}_{k=0}^{N-1}$. For example, the 4^{th} roots of unity are

$$e^0 = 1, \quad e^{i\pi/2} = i, \quad e^{i\pi} = -1, \quad e^{i3\pi/2} = -i.$$

Now, we define a family of complex exponential functions

$$E_k(x) = e^{ikx}, \tag{57}$$

where $k \in \mathbb{Z}$ and $x \in \mathbb{C}$. These functions are $\frac{2\pi}{k}$-period. They give k copies of Sine and Cosine in each interval of length 2π.

THEOREM 1.1. *$\{E_k(x)\}_{k\in\mathbb{Z}}$ forms a basis for the space of all 2π-periodic square-integrable functions $f(x)$, i.e., $f(x)$ is such that $\int_{-\infty}^{\infty} |f(x)|^2\,dx < \infty$.*

PROOF. We will show that $\{E_k(x)\}$ forms an orthogonal set. It will then follow that $\{E_k(x)\}$ is a basis since it is easy to see that it spans the space of all 2π-periodic functions. Recall the definition of complex inner product:

$$\langle f(x), g(x)\rangle = \int_{-\pi}^{\pi} f(x)\cdot\overline{g(x)}\,dx$$

over any interval of 2π. (Alternatively, one can define $\langle f(x), g(x)\rangle = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)\cdot\overline{g(x)}\,dx$ so that the set forms an orthonormal basis.)
For $j,k \in \mathbb{Z}$,

$$\int_{-\pi}^{\pi} E_k(w)\overline{E_j(w)}\,dw = \int_{-\pi}^{\pi} e^{ikw}e^{-ijw}\,dw = \int_{-\pi}^{\pi} e^{i(k-j)w}\,dw. \tag{58}$$

If $j = k$, Equation (58) gives 2π. If $j \neq k$, let $u = (k-j)w$, then $dw = \frac{1}{k-j}du$. And Equation (58) becomes

$$\begin{aligned} & \frac{1}{k-j}\int_{-(k-j)\pi}^{(k-j)\pi} \cos u + i\sin u\,du \\ = & \frac{1}{k-j}\int_{-(k-j)\pi}^{(k-j)\pi} \cos u\,du + \frac{i}{k-j}\int_{-(k-j)\pi}^{(k-j)\pi} \sin u\,du \\ = & \frac{2}{k-j}\int_{0}^{(k-j)\pi} \cos u],du = \frac{2}{k-j}\sin u\Big|_0^{(k-j)\pi} = 0 \end{aligned}$$

Thus,

$$\left\langle E_k, E_j\right\rangle = \begin{cases} 2\pi & \text{if} \quad k = j \\ 0 & \text{if} \quad k \neq j, \end{cases}$$

which establishes the claim. □

2. Fourier Series

As we saw at the end of Section 1, if f is a 2π-periodic function and well-behaved (f is piecewise continuous and has a finite number of jump discontinuities), then we can represent it as a linear combination of the family of complex exponentials $\{E_k(x)\}_{k\in\mathbb{Z}}$. Such a linear combination is called a *Fourier series*.

DEFINITION 2.1. Suppose that f is a 2π-periodic, absolutely integrable function, then the Fourier series of f is given by

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx}. \tag{59}$$

As long as $f(x)$ is a continuous function and the derivative exists at x, then the right hand side of Equation (59) converges to $f(x)$. If f is piecewise continuous and x is a jump discontinuity, then the series converges to one-half the sum of the left and right limits of f at x. Notice that this value may or may not be the value of $f(x)$.
Now, a natural question to ask is how to find the c_k's in the Fourier series. It turns out that the procedure we follow to obtain the c_k's is a natural consequence of the E_k's being orthogonal, i.e., $c_k = \langle f, E_k\rangle$. In particular, consider the following

$$\begin{aligned} f(x) &= \sum_{k=-\infty}^{\infty} c_k e^{ikx} \\ f(x)\overline{E_k(x)} &= \sum_{k\in\mathbb{Z}} E_k(x)\overline{E_k(x)} \\ \int_{-\pi}^{\pi} f(x)\overline{E_k(x)}\,dx &= \int_{-\pi}^{\pi} \sum_{k\in\mathbb{Z}} E_k(x)\overline{E_k(x)}\,dx \\ &= \sum_{k\in\mathbb{Z}} \underbrace{\int_{-\pi}^{\pi} E_k(x)\overline{E_k(x)}\,dx}_{2\pi} \\ &= 2\pi c_k \end{aligned}$$

Thus, we obtain the Fourier coefficients

$$c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)\overline{E_k(x)}\,dx = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-ikx}\,dx. \tag{60}$$

EXAMPLE 2.1. In this example, we find the Fourier series for the sawtooth function $f(x) = x$ when $-\pi < x < \pi$ and $f(x) = f(x+2\pi)$, given in Figure 2.

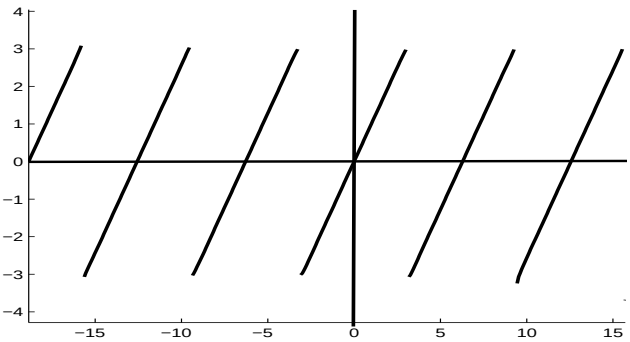


FIGURE 2. The sawtooth function.

We first compute the Fourier coefficients, c_k's:

$$\begin{aligned} c_k &= \int_{-\pi}^{\pi} \frac{1}{2\pi} x e^{ikx}\,dx \\ &= \frac{1}{2\pi}\left(\int_{-\pi}^{\pi} x\cos kx\,dx - i\int_{-\pi}^{\pi} x\sin kx\,dx\right) \\ &= 0 - \frac{i}{2\pi}\cdot 2\int_{0}^{\pi} x\sin kx\,dx \\ &= -\frac{i}{\pi}\int_{0}^{\pi} x\sin kx\,dx. \end{aligned}$$

We now integrate this by parts. Let $u = x$, $du = dx$, $dv = \sin kx\,dx$, and $v = -1/k\cos kx$:

$$\begin{aligned} -\frac{i}{\pi}\int_0^\pi x\sin kx\,dx &= -\frac{i}{\pi}\left[\frac{-1}{k}x\cos kx\Big|_0^\pi + \int_0^\pi \frac{1}{k}\cos kx\,dx\right], k\neq 0 \\ &= -\frac{i}{\pi}\left[\frac{-1}{k}x\cos kx\Big|_0^\pi + \frac{1}{k^2}\sin kx\Big|_0^\pi\right], k\neq 0 \\ &= \left(\frac{-i}{\pi}\right)\left(\frac{-\pi}{k}\right)(-1)^k = \frac{(-1)^k i}{k}, k\neq 0. \end{aligned}$$

Hence,

$$c_k = \begin{cases} \frac{(-1)^k i}{k} & \text{if} \quad k\neq 0 \\ 0 & \text{if} \quad k = 0, \end{cases}$$

which results the following Fourier series for f:

$$f(x) = \sum_{k\neq 0}\frac{(-1)^k i}{k}e^{ikx}. \tag{61}$$

It is not hard to see that if a function is odd (even), then its Fourier coefficients are imaginary (real) and the series can be reduced to one of sines (cosines). This should make sense to you — f is an odd function, so we should need sine functions to represent it. Indeed, one can show that Equation (61) can be reduced to

$$f(x) = -2\sum_{k=1}^{\infty}\frac{(-1)^k}{k}\sin(kx)$$

by grouping $\pm 1, \pm 2, \cdots$ terms and using the facts $\cos x = \dfrac{e^{ix}+e^{-ix}}{2}$ and $\sin x = \dfrac{e^{ix}-e^{-ix}}{2i}$.
Let's get an idea of what this series looks like. We define the sequence of partial sums by

$$f_n(x) = -2\sum_{k=1}^{n}\frac{(-1)^k}{k}\sin(kx)$$

and plot f_n for various values of n in Figure (3). As n gets larger, $\sin(nx)$ becomes more and more oscillatory. You will see a little undershoot and overshoot at the points of discontinuity of f. This is the famous Gibbs phenomenon.

As you can see, the computation of Fourier coefficients can be somewhat tedious. Fortunately, there exists an entire calculus for computing Fourier coefficients. The idea is to build Fourier series from a "library" of known Fourier series. In general, we can always build the Fourier series of a function that is expressed as a translation of another function.

PROPOSITION 2.1. *(Translation Rule). Suppose that $f(x)$ has the Fourier series representation*

$$f(x) = \sum_{k\in\mathbb{Z}} c_k e^{ikx}$$

and suppose that $g(x) = f(x-a)$. If the Fourier series representation for $g(x)$ is

$$g(x) = \sum_{k\in\mathbb{Z}} d_k e^{ikx}$$

then $d_k = e^{ika}c_k$.

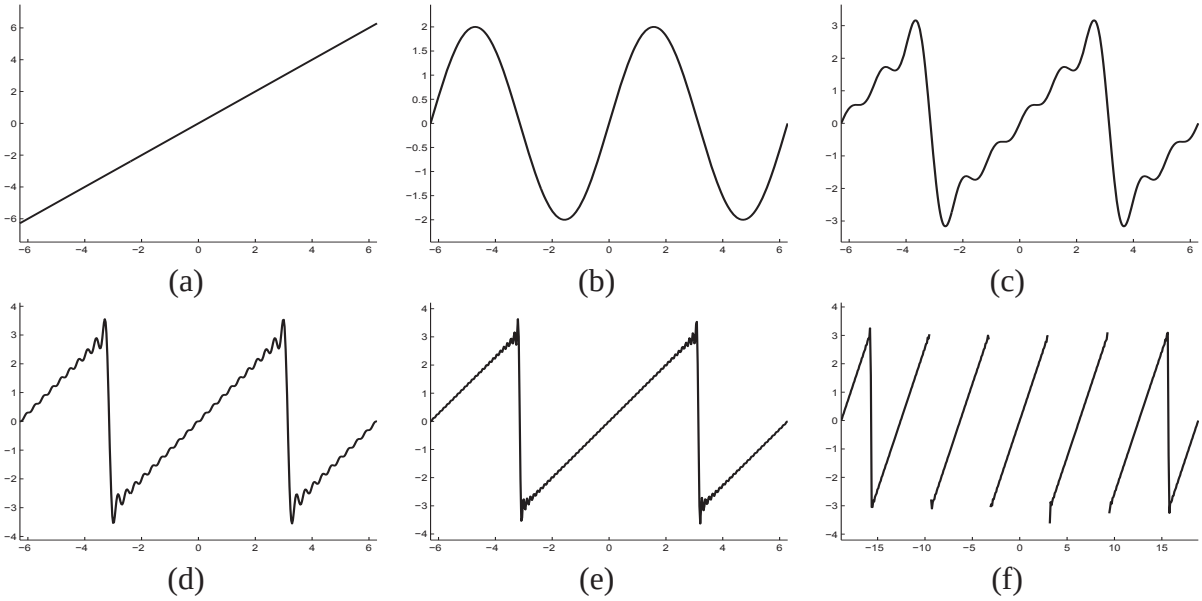


FIGURE 3. (a) The sawtooth function on $(-\pi,\pi)$. (b) 1-term Fourier partial sum. (c) 5-term Fourier partial sum. (d) 20-term Fourier partial sum. (e) 50-term Fourier partial sum. (f) 100-term Fourier partial sum.

PROOF.

$$d_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} g(x)e^{-ikx}\,dx = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x-a)e^{-ikx}\,dx.$$

Now we do a u-substitution. Let $u = x-a$. The $du = dx$ and $x = u+a$. Changing endpoints gives the formula

$$\begin{aligned} d_k &= \frac{1}{2\pi}\int_{-\pi-a}^{\pi-a} f(u)e^{-ik(u+a)}\,du \\ &= \frac{1}{2\pi}e^{-ika}\int_{-\pi-a}^{\pi-a} f(u)e^{-iku}\,du \\ &= \frac{1}{2\pi}e^{-ika}\int_{-\pi}^{\pi} f(u)e^{-iku}\,du \\ &= e^{-ika}c_k \end{aligned}$$

□

Although there are several such rules for obtaining Fourier series coefficients, we only state one more.

PROPOSITION 2.2. *(Modulation Rule). Suppose that $f(x)$ has the Fourier series representation*

$$f(x) = \sum_{k\in\mathbb{Z}} c_k e^{ikx}$$

and suppose that $g(x) = e^{imx}f(x)$ for some $m \in \mathbb{Z}$. If the Fourier series representation for $g(x)$ is

$$g(x) = \sum_{k\in\mathbb{Z}} d_k e^{ikx}$$

then $d_k = c_{k-m}$.

Finite-Length Fourier Series

You might have decided that it is a little unnatural to construct a series for a function that is already known. In many applications we typically do not know the function that generates a Fourier series — indeed, we usually have only the coefficients c_k or samples of some function such as an image. In those cases, what do we expect the Fourier coefficients to tell us?

EXAMPLE 2.2. Suppose we have a finite-length Fourier series

$$H(x) = \sum_{k=0}^{L} h_k e^{ikx}.$$

Let $h_0 = h_2 = \frac{1}{4}$ and $f_1 = \frac{1}{2}$. All other values for h_k are zero. Let us construct the Fourier series for these h_k, and plot of graph of $|H(x)|$, for $-\pi \leq x \leq \pi$.

$$\begin{aligned} H(x) &= \sum_{k=0}^{2} h_k e^{ikx} \\ &= \frac{1}{4} + \frac{1}{2}e^{ix} + \frac{1}{4}e^{i2x} \\ &= e^{ix}\left(\frac{1}{4}e^{-ix} + \frac{1}{2} + \frac{1}{4}e^{ix}\right) \\ &= e^{ix}\left(\frac{1}{2} + \frac{1}{2}\cdot\frac{e^{ix}+e^{-ix}}{2}\right) \\ &= e^{ix}\left(\frac{1}{2} + \frac{1}{2}\cos x\right) = \frac{1}{2}e^{ix}(1+\cos x). \end{aligned}$$

And the Fourier spectrum $|H(x)| = \left|\frac{1}{2}\right| \left|e^{ix}\right| |1+\cos x| = \frac{1}{2}(1+\cos x)$, since we know that e^{ix} describes a circle of radius 1 centered at the origin, so its modulus is 1. Also, $-1 \leq \cos x \leq 1$ so that $0 \leq 1+\cos x \leq 2$. We can then drop the absolute value signs from this factor. The graph of $|H(x)|$ for $-\pi \leq x \leq x$ is shown in Figure 4.

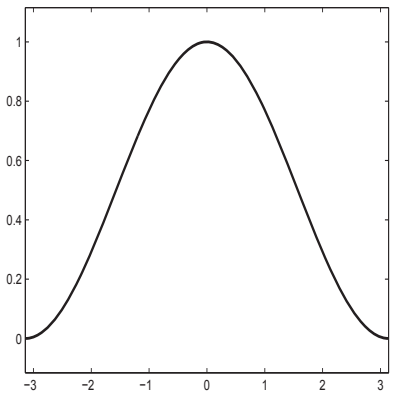


FIGURE 4. A graph of $\frac{1}{2}(1+\cos x)$.

Many scientists and engineers use the plot of spectrum to design the process they apply to signals and images. In this setting, x represents *frequency*, with $x = 0$ being the lowest frequency. We will see that functions $|H(x)|$ with large values at or near 0 will mean that the process constructed from the h_k's will leave data that are homogeneous (non-oscillatory or similar values) largely unchanged. If

values of $H(x)$ are near zero for values of x near π, then we will see that the process constructed from the h_k's will take oscillatory data ad replace them with very small values.
Conversely, when values of $|H(x)|$ are large at or near π, the process constructed from the h_k's will leave highly oscillatory data unchanged. If values of $|H(x)|$ area near 0 for x near zero, then the process constructed from the h_k's will take homogeneous data and replace them with very small values.
This important example describes in a nutshell exactly how we use Fourier series throughout the discussions. The numbers $h_0, h_1, \cdots, h_L$ are values that we use to process data that comprise signals and images. We have seen that the Fourier series can tell us how these processes will act on homogeneous or highly oscillatory data.

3. The Fourier Transform of Functions of One Variable

3.1. The Continuous Case. The *Fourier Transform* of a continuous function $f(t)$ of a continuous variable, t, denoted $\mathscr{F}\{f(t)\}$, is defined by the equation

$$\mathscr{F}\{f(t)\} = \int_{-\infty}^{\infty} f(t)e^{-i2\pi\mu t}\,dt \tag{62}$$

where μ is also a continuous variable. Because t is integrated out, $\mathscr{F}\{f(t)\}$ is a function of only μ . We denote this fact explicitly by writing the Fourier transform as $\mathscr{F}\{f(t)\} = F(\mu)$; that is, the Fourier transform of $f(t)$ may be written for convenience as

$$F(\mu) = \int_{-\infty}^{\infty} f(t)e^{-i2\pi\mu t}\,dt. \tag{63}$$

Conversely, given $F(\mu)$, we can obtain $f(t)$ back using the *inverse Fourier transform*, $f(t) = \mathscr{F}^{-1}\{F(\mu)\}$, written as

$$f(t) = \int_{-\infty}^{\infty} F(\mu)e^{i2\pi\mu t}\,d\mu \tag{64}$$

where we made use of the fact that variable μ is integrated out in the inverse transform and wrote simply $f(t)$ instead of the more cumbersome notation $f(t) = \mathscr{F}^{-1}\{F(\mu)\}$. Equations (63) and (64) comprise the so-called *Fourier transform pair*.
We need one more building block before proceeding. We introduced the idea of convolution in Section 5 . You learned in that section that convolution of two functions involves flipping (rotating by 180°) one function about its origin and sliding it past the other. At each displacement in the sliding process, we perform a computation, which in the case of Section 5, was a sum of products. In the present discussion, we are interested in the convolution of two continuous functions, $f(t)$ and $h(t)$, of one continuous variable, t, so we have to use integration instead of a summation. The convolution of these two functions, denoted as before by the operator $\star$, is defined as

$$f(t) \star h(t) = \int_{-\infty}^{\infty} f(\tau)h(t-\tau)\,d\tau \tag{65}$$

where the minus sign accounts for the flipping just mentioned, t is the displacement needed to slide one function past the other, and τ is a dummy variable that is integrated out. We assume for now that the functions extend from $-\infty$ to ∞. At the moment, we are interested in finding the Fourier transform of Equation (65). We start with Equation (62):

$$\begin{aligned}\mathscr{F}\{f(t)\star h(t)\} &= \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(\tau)h(t-\tau)\,d\tau\right]e^{-i2\pi\mu t}\,dt \\ &= \int_{-\infty}^{\infty} f(\tau)\left[\int_{-\infty}^{\infty} h(t-\tau)e^{-i2\pi\mu t}\,dt\right]d\tau.\end{aligned}$$

The term inside the brackets is the Fourier transform of $h(t-\tau)$. By the *circularity property*, which will be shown later in this Chapter, we have $\mathscr{F}\{h(t-\tau)\} = e^{-i2\pi\mu\tau}H(\mu)$, where $H(\mu)$ is the Fourier transform of $h(t)$. Using this fact in the preceding equation gives us

$$\begin{aligned}\mathscr{F}\{f(t)\star h(t)\} &= \int_{-\infty}^{\infty} f(\tau)\left[H(\mu)e^{-i2\pi\mu\tau}\right]d\tau \\ &= H(\mu)\int_{-\infty}^{\infty} f(\tau)e^{-2i\pi\mu\tau}d\tau \\ &= H(\mu)F(\mu).\end{aligned}$$

If we refer to the domain of t as the *spatial* domain, then the domain of μ is realized as the *frequency* domain. The preceding equation tells us that the Fourier transform of the convolution of two functions in the spatial domain is equal to the product in the frequency domain of the Fourier transforms of the two functions. Conversely, if we have the product of the two transforms, we can obtain the convolution in the spatial domain by computing the inverse Fourier transform. In other words, $f(t)\star h(t)$ and $H(\mu)F(\mu)$ are a Fourier transform pair. The result is one-half of the *convolution theorem* and is written as

$$f(t)\star h(t) \Leftrightarrow H(\mu)F(\mu). \tag{66}$$

The double arrow is used to indicate that the expression on the right is obtained by taking the Fourier transform of the expression on the left, while the expression on the left is obtained by taking the *inverse* Fourier transform of the expression on the right. Following a similar development would result in the other half of the convolution theorem:

$$f(t)h(t) \Leftrightarrow H(\mu)\star F(\mu), \tag{67}$$

which states that convolution in the frequency domain is analogous to multiplication in the spatial domain, the two being related by the forward and inverse transforms, respectively. As you will see later in this chapter, the convolution theorem is the foundation for filtering in the frequency domain.

4. The Discrete Case

In practice, continuous functions have to be converted into a sequence of discrete values before they can be processed in a computer. This is accomplished by sampling and quantization. In the following discussion, we examine sampling in more details.

With reference to Figure 5, consider a continuous function, $f(t)$, that we wish to sample at uniform intervals ΔT of the independent variable t. One way to model sampling is to multiply $f(t)$ by a sampling function equal to a train of impulses ΔT units apart. Then the value, f_k, of an arbitrary sample in the sequence is given by $f(k\Delta T)$, shown in Figure 5(d). A natural question to consider is whether or not we can *uniquely* recover f from f_k and if so, when?

Figure 6(a) is a sketch of the Fourier transform, $F(\mu)$, of a function $f(t)$, and Figure 6(b) shows the transform, $\tilde{F}(\mu)$, of the sampled function. The quantity $1/\Delta T$ is the sampling rate used to generate the sampled function. So, in Figure 6(b) the sampling rate was high enough to provide sufficient separation between the periods and thus preserve the integrity of $F(\mu)$. In Figure 6(c), the sampling rate was just enough to preserve $F(\mu)$, but in Figure 6(d), the sampling rate was below the minimum required to maintain distinct copies of $F(\mu)$ and thus failed to preserve the original transform. Figure 6(b) is the result of an *over-sampled* signal, while Figure 6(c) and (d) are the results of *critically-sampling* and *under-sampling* the signal, respectively.

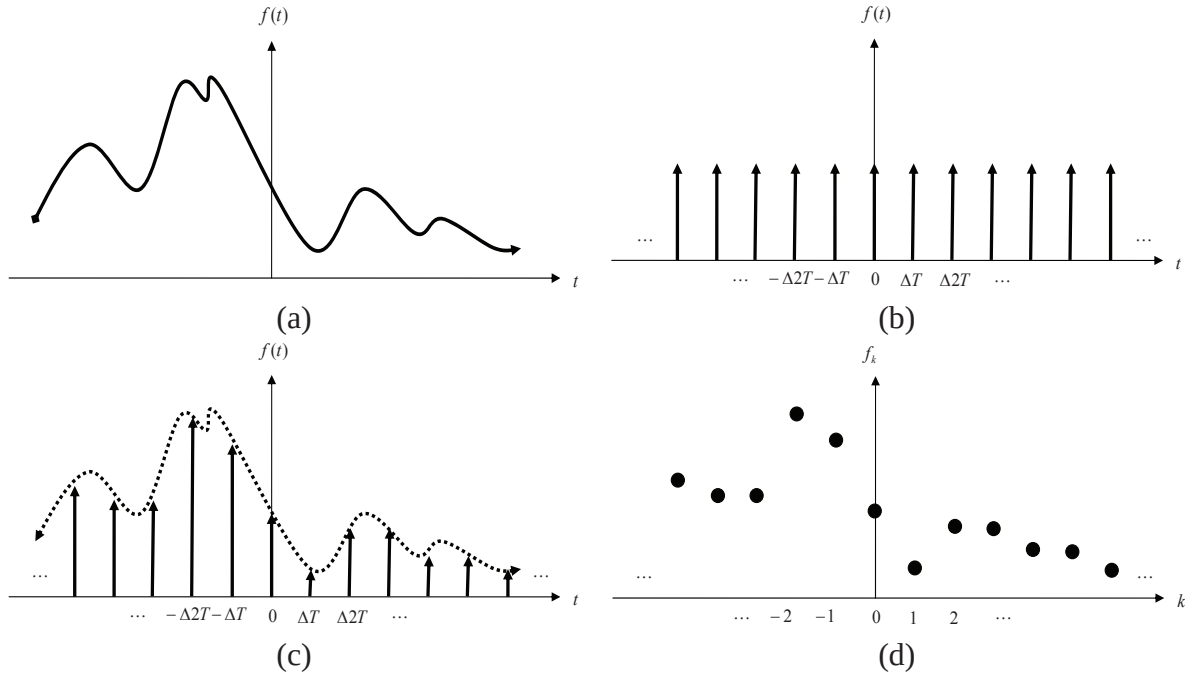


FIGURE 5. (a) A continuous function. (b) Train of impulses used to model the sampling process. (c) Sample function formed as the product of (a) and (b). (d) Sample values obtained.

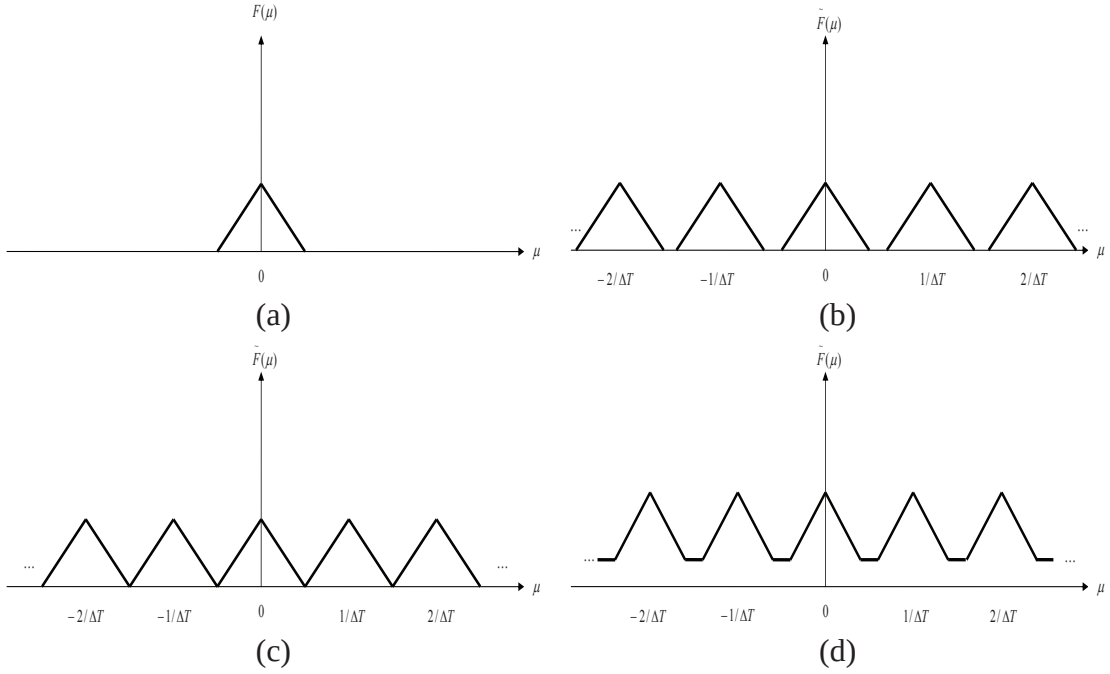


FIGURE 6. (a) Fourier transform of a band-limited function. (b)-(d) Transforms of the corresponding sampled function under the conditions of over-sampling, critically-sampling, and under-sampling, respectively.

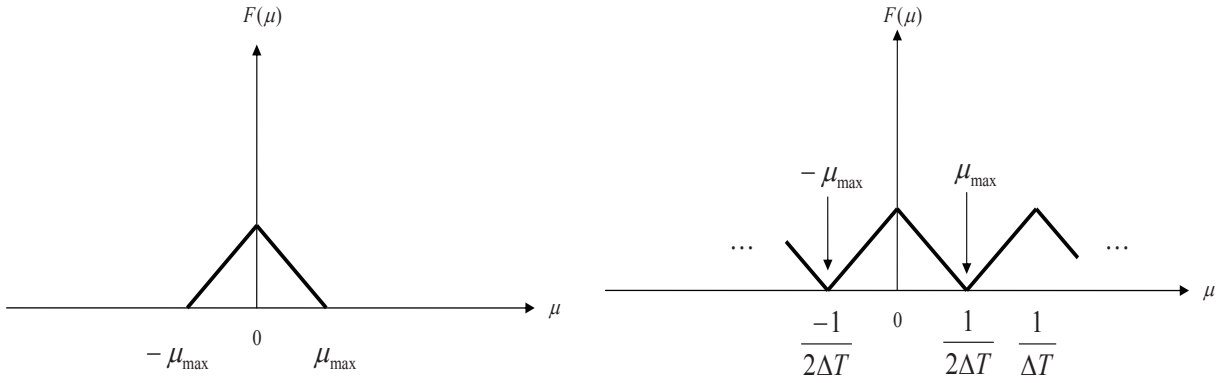


FIGURE 7. (a) Transform of a band-limited function. (b) Transform resulting from critically sampling the same function.

The Sampling Theorem

A function $f(t)$ whose Fourier transform is zero for values of frequencies outside a finite interval (band) $[-\mu_{\max}, \mu_{\max}]$ about the origin is called a *band-limited* function. Figure 7(a), which is a magnified section of Figure 6(a), is such a function. Similarly, Figure 7(b) is a more detailed view of the transform of a critically sampled function shown in Figure 6(c). A lower value of $1/\Delta T$ would cause the periods in $\tilde{F}(\mu)$ to merge; a higher value would provide a clean separation between the periods.

We can recover $f(t)$ from its sampled version if we can isolate a copy of $F(\mu)$ from the periodic sequence of copies of this function contained in $\tilde{F}(\mu)$, the transform of the sampled function $\tilde{f}(t)$. Since $\tilde{F}(\mu)$ is a continuous, periodic function with period $1/\Delta T$. Therefore, all we need is one complete period to characterize the entire transform. This implies that we can recover $f(t)$ from that single period by using the inverse Fourier transform.

Extracting from $\tilde{F}(\mu)$ a single period that is equal to $F(\mu)$ is possible if the separation between copies is sufficient, see Figure 6. In terms of Figure 7(b), sufficient separation is guaranteed if $1/2\Delta T > \mu_{\max}$ or

$$\frac{1}{\Delta T} > 2\mu_{\max}. \tag{68}$$

This equation indicates that a continuous, band-limited function can be recovered completely from a set of its samples if the samples are acquired at a rate exceeding twice the highest frequency content of the function. This result is known as the *sampling theorem*. We can say based on this result that no information is lost if a continuous, band-limited function is represented by samples acquired at a rate greater than twice the highest frequency content of the function. Conversely, we can say that the *maximum* frequency that can be captured by sampling a signal at a rate $1/\Delta T$ is $\mu_{\max} = \frac{1}{2\Delta T}$. Sampling at the Nyquist rate[1] sometimes is sufficient for perfect function recovery, but there are cases in which this leads to difficulties. Thus, the sampling theorem specifies that sampling must exceed the Nyquist rate.

To see how the recovery of $F(\mu)$ from $\tilde{F}(\mu)$ is possible in principle, consider Figure 8, which shows the Fourier transform of a function sampled at a rate slightly higher than the Nyquist rate. The function

[1] A sampling rate equal to exactly twice the highest frequency.

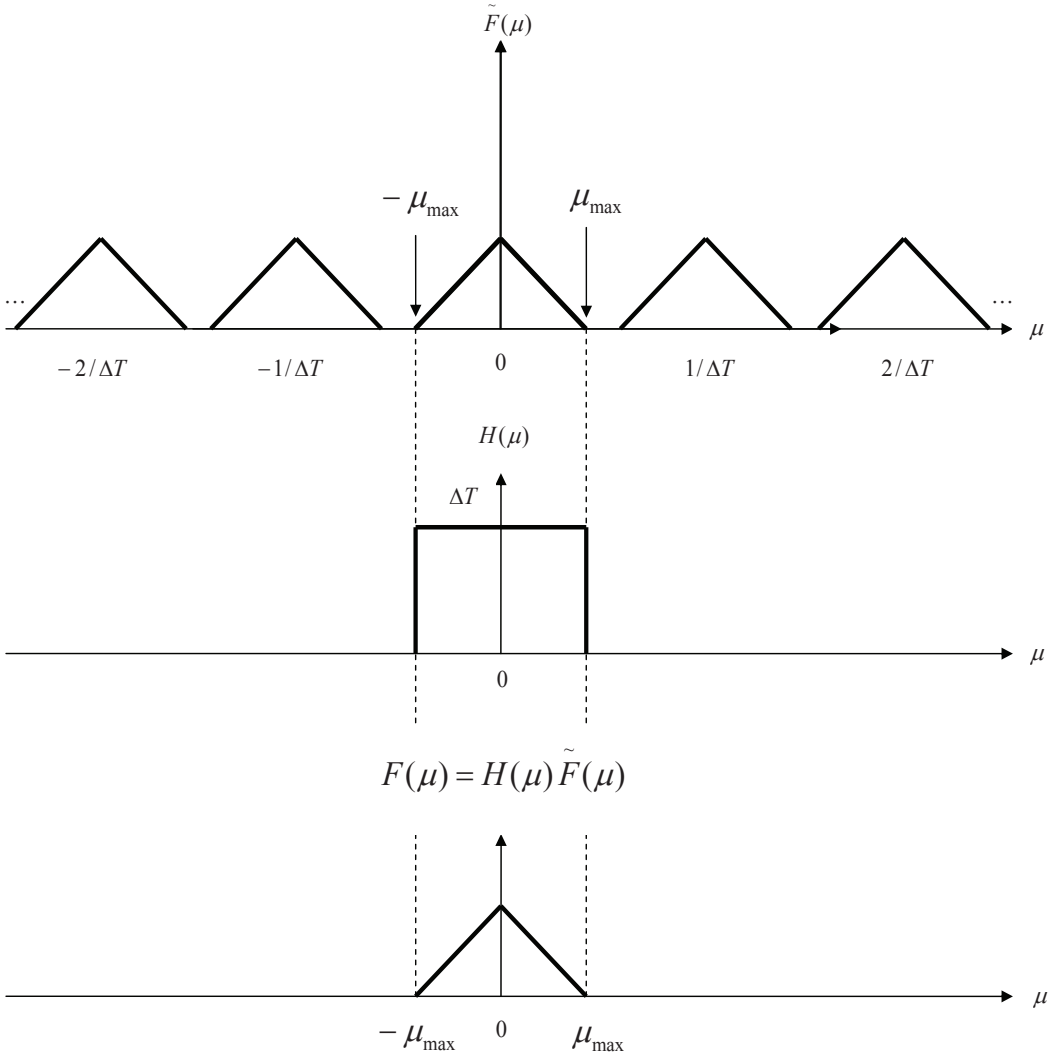


FIGURE 8. Extracting one period of the transform of a band-limited function using an ideal lowpass filter.

in the middle of Figure 8 is defined by the equation

$$H(\mu) = \begin{cases} \Delta T & -\mu_{\max} \le \mu \le \mu_{\max} \\ 0 & \text{otherwise} \end{cases}$$

When multiplied by the periodic sequence in the top of Figure 8, this function isolates the period centered on the origin. Then, as the bottom of Figure 8 shows, we obtain $F(\mu)$ by multiplying $\tilde{F}(\mu)$ by $H(\mu)$:

$$F(\mu) = H(\mu)\tilde{F}(\mu).$$

Once we have $F(\mu)$ we can recover $f(t)$ by using the inverse Fourier transform:

$$f(t) = \int_{-\infty}^{\infty} F(\mu) e^{i2\pi\mu t}\, d\mu.$$

The last three equations prove that, theoretically, it is possible to recover a band-limited function from samples of the function obtained at a rate exceeding twice the highest frequency content of the function. As we discuss in the following section, the requirement that $f(t)$ must be band-limited implies in general that $f(t)$ must extend from $-\infty$ to ∞, a condition that cannot be met in practice.

As you will see shortly, having to limit the duration of a function prevents perfect recovery of the function, except in some special cases.
Function $H(\mu)$ is called a *lowpass* filter because it passes frequencies at the low end of the frequency range but it eliminates (filters out) all higher frequencies. It is called an *ideal* lowpass filter because of its infinitely rapid transitions in amplitude (between 0 and ΔT at location $-\mu_{\max}$ and the reverse at $\mu_{\max}$), a characteristic that cannot be achieved with physical electronic components.

Aliasing

A logical question at this point is: what happens if a band-limited function is sampled at a rate that is less than twice its highest frequency? This corresponds to the under-sampled case discussed in the previous discussion. Top of Figure 9 is the same as Figure 6(d), which illustrates this condition. The net effect of lowering the sampling rate below the Nyquist rate is that the periods now overlap, and it becomes impossible to isolate a single period of the transform, regardless of the filter used. For instance, using the ideal lowpass filter in the middle of Figure 9 would result in a transform that is corrupted by frequencies from adjacent periods, as the bottom of Figure 9 shows. The inverse transform would then yield a corrupted function of t. This effect, caused by under-sampling a function, is known as *frequency aliasing* or simply *aliasing*. In words, aliasing is a process in which high frequency components of a continuous function "masquerade" as lower frequencies in the sampled function. This is consistent with the common use of the term *alias*, which means "a false identity".
Unfortunately, except for some special cases, aliasing is always present in sampled signals because, even if the original sampled function is band-limited, infinite frequency components are introduced the moment we limit the duration of the function, which we always have to do in practice. In practice, the effects of aliasing can be reduced by smoothing the input function to attenuate its higher frequencies (e.g., by defocusing in the case of an image). This process, called *anti-aliasing*, has to be done *before* the function is sampled because aliasing is a sampling issue that cannot be "undone after the fact" using computational techniques.

The Discrete Fourier Transform (DFT) of One Variable

The material up to this point may be viewed as the foundation of those basic principles, so now we are ready to derive the DFT. The Fourier transform of a sampled, band-limited function extending from $-\infty$ to ∞ is a *continuous, periodic* function that also extends from $-\infty$ to ∞. In practice, we work with a finite number of samples, and the objective of this section is to derive the DFT corresponding to such sample sets.
From the definition of the Fourier transform in Equation (63), one can obtain the *discrete Fourier transform pair*

$$F(u) = \sum_{x=0}^{M-1} f(x) e^{-i2\pi ux/M} \quad u = 0, 1, 2, \ldots, M-1 \tag{69}$$

and

$$f(x) = \frac{1}{M} \sum_{n=0}^{M-1} F(u) e^{i2\pi ux/M} \quad x = 0, 1, 2, \ldots, M-1 \tag{70}$$

where we used the functional notation instead of subscripts for simplicity and M is the total number of samples taken over one complete period. Clearly, $F(u) = F_u$ and $f(x) = f_x$. It can be shown that both the forward and inverse discrete transforms are infinitely periodic, with period M. That is,

$$F(u) = F(u + kM)$$

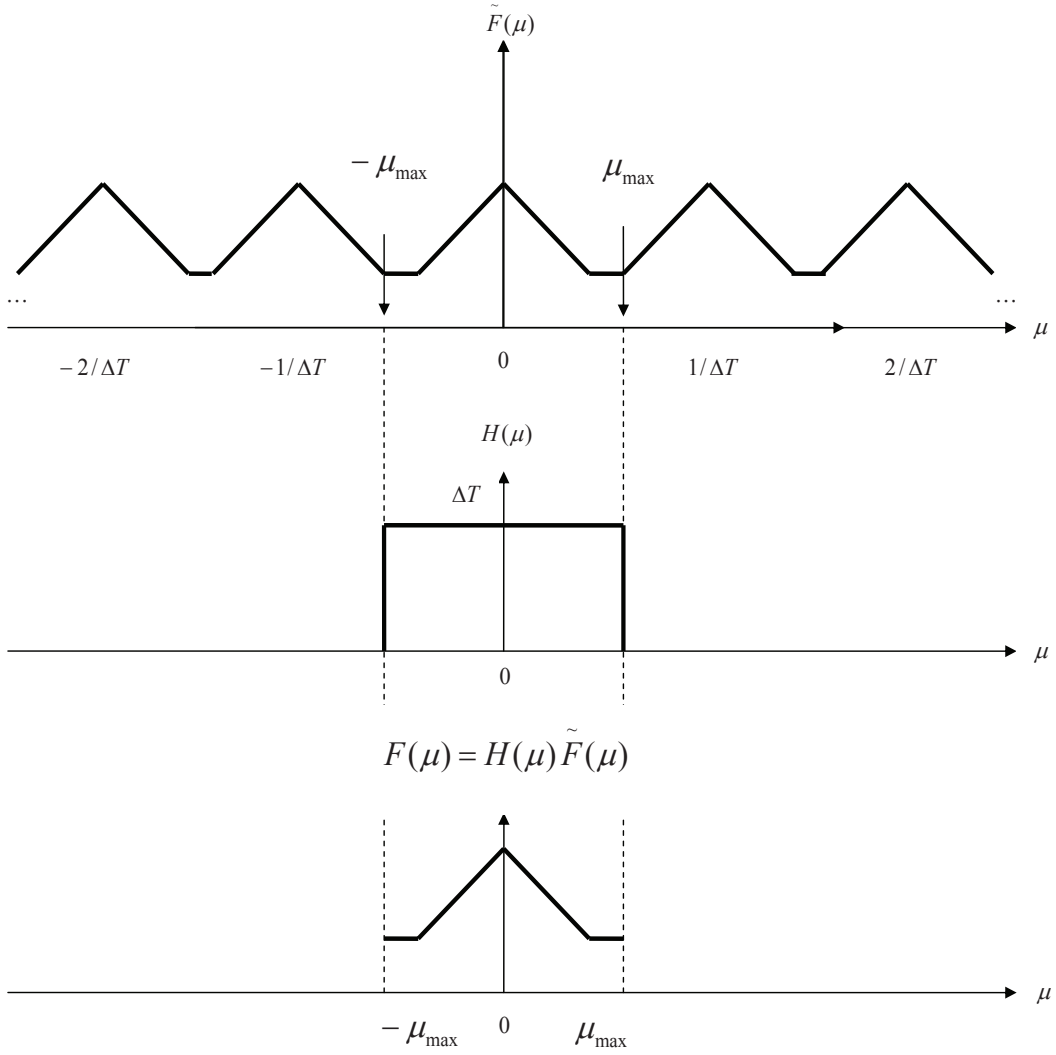


FIGURE 9. Top: Fourier transform of an under-sampled, band-limited function. Middle: The same ideal lowpass filter used in Figure 8. Bottom: The product of top and middle. The interference from adjacent periods results in aliasing that prevents perfect recovery of $F(\mu)$ and, therefore, of the original, band-limited continuous function.

and

$$f(x) = f(x+kM)$$

where k is an integer. This is called the *periodicity*. The discrete equivalent of the convolution is

$$f(x) \star h(x) = \sum_{m=0}^{M-1} f(m)h(x-m) \tag{71}$$

for $x = 0, 1, 2, \ldots, M-1$. Because in the preceding formulations the functions are periodic, their convolution also is periodic. Equation (71) gives one period of the periodic convolution. For this reason, the process inherent in this equation often is referred to as *circular convolution*, and is a direct result of the periodicity of the DFT and its inverse. This is in contrast with the convolution you studied earlier, in which values of the displacement, x, were determined by the requirement of sliding one function completely past the other, and were not fixed to the range $[0, M-1]$ as in circular convolution.

5. The Fourier Transform of Functions of Two Variables

The 2-D Continuous Fourier Transform Pair

Let $f(t,z)$ be a continuous function of two continuous variables, t and z. The two-dimensional, continuous Fourier transform pair is given by the expressions

$$F(\mu,\nu) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t,z)e^{-i2\pi(\mu t+\nu z)}\,dt\,dz \tag{72}$$

and

$$f(t,z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\mu,\nu)e^{i2\pi(\mu t+\nu z)}\,d\mu\,d\nu \tag{73}$$

where *mu* and ν are the frequency variables. When referring to images, t and z are interpreted to be continuous *spatial* variables. As in the 1-D case, the domain of the variables μ and ν defines the *continuous frequency domain*.

The 2-D Discrete Fourier Transform and its Inverse

The 2-D *discrete Fourier transform (DFT)* is given by

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{74}$$

where $f(x,y)$ is a digital image of size $M \times N$. This is computed through repeated applications of 1-D version of the transform, i.e., M 1-D DFT of length N and N 1-D DFT of length M.
Given the transform $F(u,v)$, we can obtain $f(x,y)$ by using the *inverse discrete Fourier transform (IDFT)*

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{i2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{75}$$

for $x = 0,1,2,\ldots,M-1$ and $y = 0,1,2,\ldots,N-1$. The rest of this section is based on properties of the 2-D discrete Fourier transform pair and their use for image filtering in the frequency domain.

Properties

1. Relationships Between Spatial and Frequency Intervals

Suppose that a continuous function $f(t,z)$ is sampled to form a digital image, $f(x,y)$, consisting of $M \times N$ samples taken in the t- and z-directions, respectively. Let ΔT and ΔZ denote the separation between samples. Then the separations between the corresponding discrete, frequency domain variables are given by

$$\Delta u = \frac{1}{M\Delta T}$$

and

$$\Delta v = \frac{1}{N\Delta Z}$$

respectively. Note that the separations between samples in the frequency domain are inversely proportional both the spacing between spatial samples and the number of samples.

2. Translation and Rotation

It can be shown by direct substitution into Equations (74) and (75) that the Fourier transform pair

satisfies the following translation properties

$$f(x,y)e^{i2\pi(u_0x/M+v_0y/N)} \Leftrightarrow F(u-u_0,v-v_0) \tag{76}$$

and

$$f(x-x_0,y-y_0) \Leftrightarrow F(u,v)e^{-i2\pi(x_0u/M+y_0v/N)}. \tag{77}$$

That is, multiplying $f(x,y)$ by the exponential shown shifts the origin of the DFT to (u_0,v_0) and, conversely, multiplying $F(u,v)$ by the negative of that exponential shifts the origin of $f(x,y)$ to (x_0,y_0). Notice that translation has no effect on the magnitude (spectrum) of $F(u,v)$.
Using the polar coordinates

$$x=\cos\theta \quad y=r\sin\theta \quad u=\omega\cos\phi \quad v=\omega\sin\phi$$

results in the following transform pair:

$$f(r,\theta+\theta_0) \Leftrightarrow F(\omega,\phi+\theta_0) \tag{78}$$

which indicates that rotating $f(x,y)$ by an angle θ_0 rotates $F(u,v)$ by the same angle. Conversely, rotating $F(u,v)$ rotates $f(x,y)$ by the same angle.

3. Periodicity

As in the 1-D case, the 2-D Fourier transform and its inverse are infinitely periodic in the u and v directions; that is

$$F(u,v)=F(u+k_1M,v)=F(u,v+k_2N)=F(u+k_1M,v+k_2N)$$

and

$$f(x,y)=F(x+k_1M,y)=F(x,y+k_2N)=F(x+k_1M,y+k_2N)$$

where k_1 and k_2 are integers.
The periodicities of the transform and its inverse are important issues in the implementation of DFT-based algorithms. Consider the 1-D spectrum in Figure 10(a), the transform data in the interval from 0 to $M-1$ consists of two back-to-back half periods meeting at point $M/2$. For display and filtering purposes, it is more convenient to have in this interval a complete period of the transform in which the data are contiguous, as in Figure 10(b). It follows from Equation (76) that

$$f(x)e^{i2\pi(u_0x/M)} \Leftrightarrow F(u-u_0).$$

In other words, multiplying $f(x)$ by the exponential term shown shifts the data so that the origin, $F(0)$, is located at u_0. If we let $u_0=M/2$, the exponential term becomes $e^{i\pi x}$ which is equal to $(-1)^x$ because x is an integer. In this case,

$$f(x)(-1)^x \Leftrightarrow F(u-M/2).$$

That is, multiplying $f(x)$ by $(-1)^x$ shifts the data so that $F(0)$ is at the *center* of the interval $[0,M-1]$, which corresponds to Figure 10(b), as desired.
In 2-D the situation is more difficult to graph, but the principle is the same, as Figure 10(c) shows. Instead of two half periods, there are now four quarter periods meeting at the point $(M/2,N/2)$. The dashed rectangles correspond to the infinite number of periods of the 2-D DFT. As in the 1-D case, visualization is simplified if we shit the data so that $F(0,0)$ is at $(M/2,N/2)$. Letting $(u_0,v_0)=(M/2,N/2)$ in Equation (76) results in the expression

$$f(x,y)(-1)^{x+y} \Leftrightarrow F(u-M/2,v-N/2).$$

Using this equation shifts the data so that $F(0,0)$ is at the center of the *frequency rectangle* defined by the intervals $[0,M-1]$ and $[0,N-1]$, as desired. Figure 10(d) shows the result.

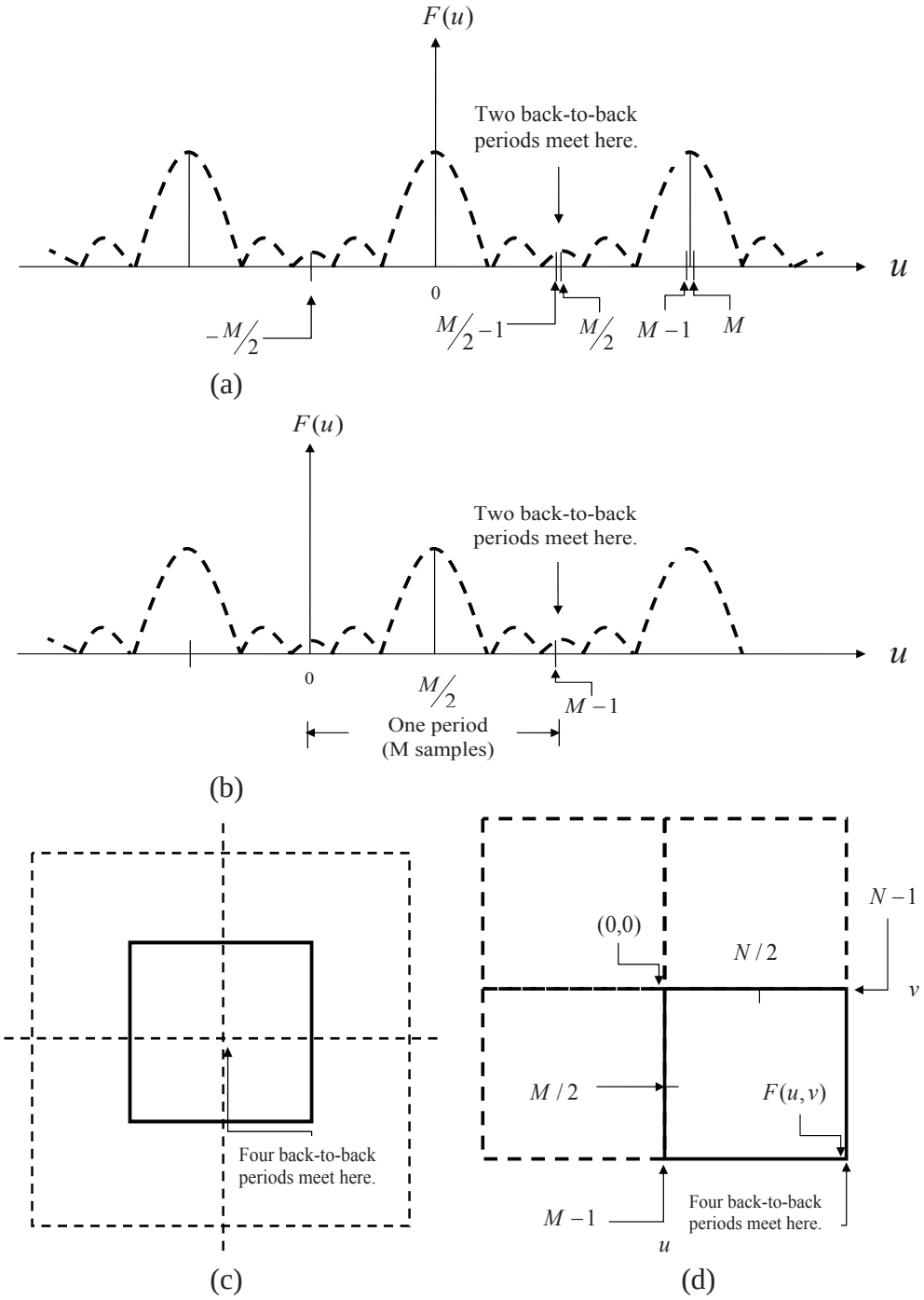


FIGURE 10. Centering the Fourier transform. (a) A 1-D DFT showing an infinite number of periods. (b) Shifted DFT obtained by multiplying $f(x)$ by $(-1)^x$ before computing $F(u)$. (c) A 2-D DFT showing an infinite number of periods. The solid area is the $M \times N$ data array, $F(u,v)$, obtained with Equation (74). This array consists of four quarter periods. (d) A shifted DFT obtained by multiplying $f(x,y)$ by $(-1)^{x+y}$ before computing $F(u,v)$. The data now contains one complete, centered period, as in (b).

4. Fourier Spectrum and Phase Angle

Because the 2-D DFT is complex in general, it can be expressed in polar form:

$$F(u,v) = |F(u,v)|e^{i\phi(u,v)} \tag{79}$$

where the magnitude

$$|F(u,v)| = [R^2(u,v) + I^2(u,v)]^{1/2}$$

is called the *Fourier* (or *frequency*) *spectrum*, and

$$\phi(u,v) = \arctan\left[\frac{I(u,v)}{R(u,v)}\right]$$

is the *phase angle*. Finally, the *power spectrum* is defined as

$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v).$$

R and I are the real and imaginary parts of $F(u,v)$ and all computations are carried out for the discrete variables $u = 0,1,2,\ldots,M-1$ and $v = 0,1,2,\ldots,N-1$. Therefore, $|F(u,v)|, \phi(u,v)$, and $P(u,v)$ are arrays of size $M \times N$.

The Fourier transform of a real function is conjugate symmetric which implies that the spectrum has *even* symmetry about the origin:

$$|F(u,v)| = |F(-u,-v)|.$$

The phase angle exhibits the following *odd* symmetry about the origin:

$$\phi(u,v) = -phi(-u,-v).$$

It follows from Equation (74) that

$$F(0,0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)$$

which indicates that the zero-frequency term is proportional to the average value of $f(x,y)$. That is,

$$\begin{aligned} F(0,0) &= MN\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y) \\ &= MN\bar{f}(x,y) \end{aligned}$$

where $\bar{f}$ denotes the average value of f. Then,

$$|F(0,0)| = MN|\bar{f}(x,y)|.$$

Because the proportionality constant MN usually is large, $|F(0,0)|$ typically is the largest component of the spectrum by a factor that can be several orders of magnitude larger than other terms. Because frequency components u and v are zero at the origin, $F(0,0)$ sometimes is called the *dc component* of the transform. This terminology is from electrical engineering, where "dc" signifies direct current (i.e., current of zero frequency).

EXAMPLE 5.1. Figure 11(a) shows a simple image of 30×30 and Figure 11(b) shows its spectrum and displays it in image form. The origins of both the spatial and frequency domains are at the top left. As expected, the area around the origin of the transform contains the highest values (thus appears red in the image). However, the four corners of the spectrum contain similarly high values. The reason is the periodicity property. The reason is the periodicity property discussed in the previous section. To center the spectrum, we simply multiply the image in (a) by $(-1)^{x+y}$ before computing the DFT. Figure 11(c) shows the result, which clearly is much easier to visualize (note the symmetry about the center point).

Because the dc term dominates the values of the spectrum, the dynamic range of other intensities in the displayed image are compressed. To bring out those details, we perform a log transformation, $1+\log|F(u,v)|^2$. Figure 11(d) shows the display of that. The increased rendition of detail is evident.

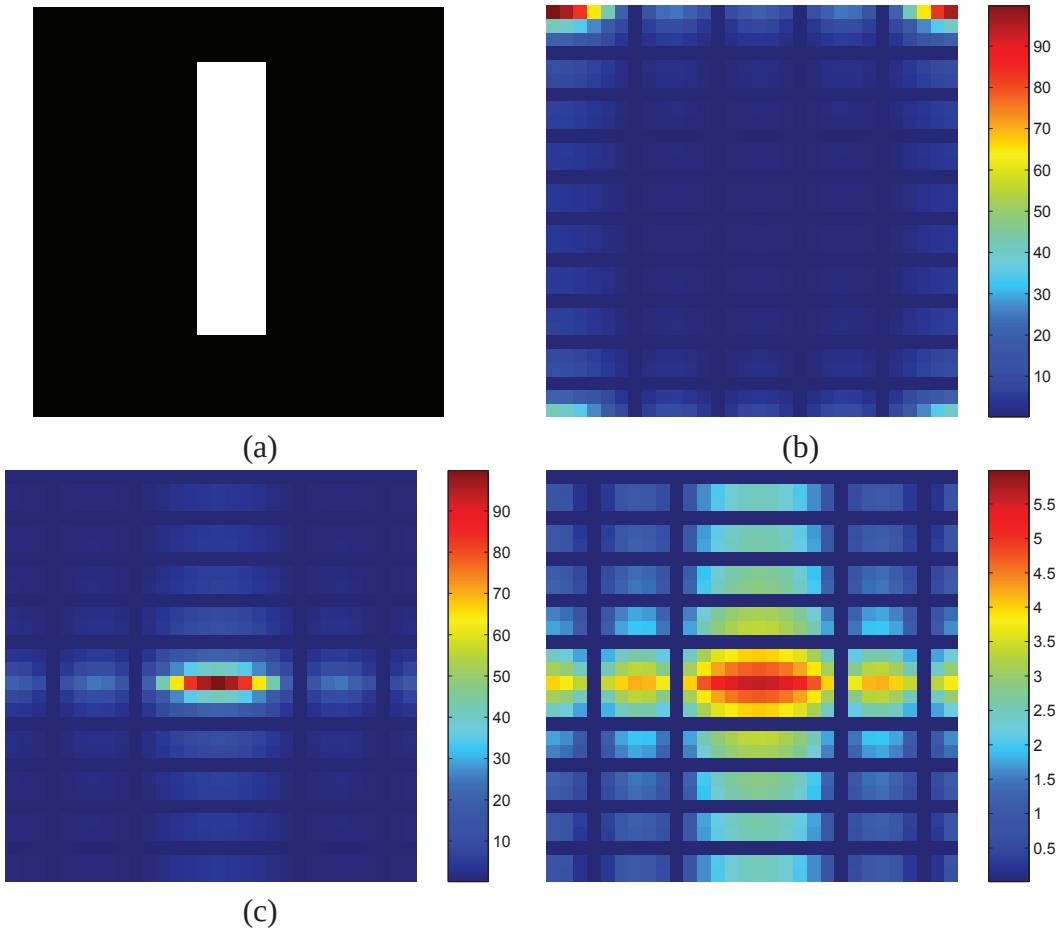


FIGURE 11. (a) Image. (b) Spectrum showing red spots in four corners. (c) Centered spectrum. (d) Result showing increase detail after a log transformation. The zero crossings of the spectrum are closer in the vertical direction because the rectangle in (a) is longer in that direction. The coordinate convention used here places the origin of the spatial and frequency domains at the top left.

It follows from Equations (77) and (78) that the spectrum is insensitive to image translation, but it rotates the same angle of a rotated image.

The components of the spectrum of the DFT determine the amplitudes of the sinusoids that combine to form the resulting image. At any given frequency in the DFT of an image, a large amplitude implies a greater prominence of a sinusoid of that frequency in the image. Conversely, a small amplitude implies that less of that sinusoid is present in the image. Thus, while *the magnitude of the 2-D DFT is an array whose components determine the intensities in the image*, the *corresponding phase is an array of angles that carry much of the information about where discernable objects are located in the image.*

[2]We commonly use log map to map a narrow range of low intensity values into a wider range of output levels, see, e.g., Figure 12

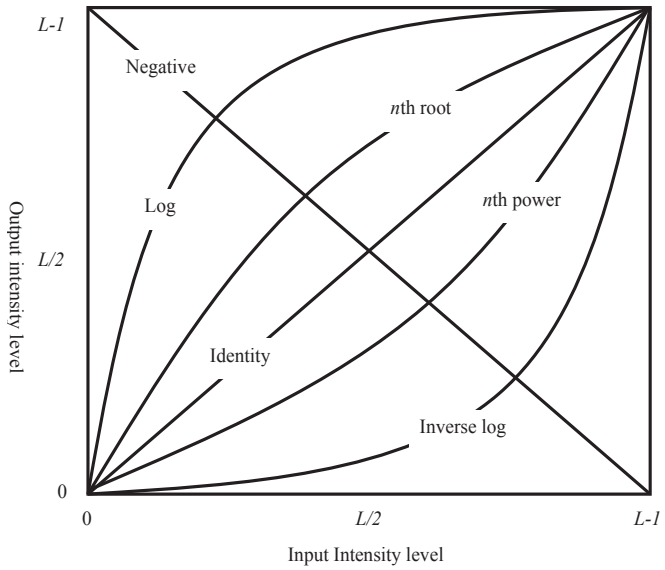


FIGURE 12. Some basic intensity transformation functions. All curves were scaled to fit in the range shown.

5. The 2-D Convolution Theorem

Extending Equation (71) to two variables results in the following expression for 2-D *circular convolution*:

$$f(x,y) \star h(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)h(x-m,y-n) \tag{80}$$

for $x = 0,1,2,\ldots,M-1$ and $y = 0,1,2,\ldots,N-1$. The 2-D convolution theorem is given by the expressions

$$f(x,y) \star h(x,y) \Leftrightarrow F(u,v)H(u,v) \tag{81}$$

and, conversely,

$$f(x,y)h(x,y) \Leftrightarrow F(u,v) \star H(u,v). \tag{82}$$

6. The Basics of Filtering in the Frequency Domain

In this section, we lay the the groundwork for all the filtering techniques discussed in the remainder of the chapter.

6.1. Additional Characteristics of the Frequency Domain. We begin by observing in Equation (74) that *each* term of $F(u,v)$ contains *all* values of $f(x,y)$, modified by the values of the exponential terms. Thus, with the exception of trivial cases, it usually is impossible to make direct associations between specific components of an image and its transform. However, some general statements can be made about the relationship between the frequency components of the Fourier transform and spatial features of an image. For instance, because frequency is directly related to spatial rates of change, it is not difficult intuitively to associate frequencies in the Fourier transform with patterns of intensity variations in an image. For example, the slowest varying frequency component ($u = v = 0$) is proportional to the average intensity of an image. As we move away from the origin of the transform, the low frequencies correspond to the slowly varying intensity component of an image. As we move further away from the origin, the higher frequencies begin to correspond to faster and faster intensity

changes in the image. These are the edges of objects and other components of an image characterized by abrupt changes in intensity.

Filtering techniques in the frequency domain are based on modifying the Fourier transform to achieve a specific objective and then computing the inverse DFT to get us back to the image domain. It follows from Equation (79) that the two components of the transform to which we have access are the transform magnitude (spectrum) and the phase angle. We learned from before that visual analysis of the phase component generally is not very useful; the spectrum, however, provides some useful guidelines as to gross characteristics of the image from which the spectrum was generated.

6.2. Frequency Domain Filtering Fundamentals. Filtering in the frequency domain consists of modifying the Fourier transform of an image and then computing the inverse transform to obtain the processed result. Thus, given a digital image, $f(x,y)$, of size $M \times N$, then basic filtering equation in which we are interested has the form

$$g(x,y) = \mathscr{F}^{-1}[H(u,v)F(u,v)] \tag{83}$$

where $\mathscr{F}^{-1}$ is the IDFT, $F(u,v)$ is the DFT of the input image, $f(x,y)$, $H(u,v)$ is a *filter function* (also called simply the *filter*, or the *filter transfer function*), and $g(x,y)$ is the filtered (output) image. Functions F, H, and g are arrays of size $M \times N$, the same as the input image. The product $H(u,v)F(u,v)$ is formed using array multiplication. The filter function modifies the transform of the input image to yield a processed output, $g(x,y)$. Specification of $H(u,v)$ is simplified considerably by using functions that are symmetric about their center, which requires that $F(u,v)$ be centered also. As explained in Section 5, this is accomplished by multiplying the input image by $(-1)^{x+y}$ prior to computing its transform.

We are now in the position to consider the filtering process in some details. One of the simplest filters we can construct is a filter $H(u,v)$ that is 0 at the center of the transform and 1 elsewhere. This filter would reject the dc term and pass all other terms of $F(u,v)$ when we form the product $H(u,v)F(u,v)$. We know that the dc term is responsible for the average intensity of an image, so setting it to zero will reduce the average intensity of the output image to zero.

As noted earlier, low frequencies in the transform are related to slowly varying intensity components in an image, such as the walls of a room or a cloudless sky in an outdoor scene. On the other hand, high frequencies are caused by sharp transitions in intensity, such as edges and noise. Therefore, we would expect that a filter $H(u,v)$ that attenuates high frequencies while passing low frequencies (appropriately called a *lowpass filter*) would blur an image while a filter with the opposite property (called a *highpass filter*) would enhance sharp detail, but cause a reduction in contrast in the image.

6.3. Summary of Steps for Filtering in the Frequency Domain.

(1) Given an input image $f(x,y)$ of size $M \times N$, obtain the padding parameters P and Q. P and Q are taken so that

$$P \geq M + m - 1, Q \geq N + n - 1,$$

where m and n are sizes for the filter H. Typically, $m = M$ and $n = N$. Thus, select $P = 2M$ and $Q = 2N$. (Please refer to [**23**] for the discussion on wraparound error and zero padding. Rule of thumb: zero-pad images and then create filters in the frequency domain to be of the same size as the padded images. Images and filters must of the same size when using the DFT.)

(2) Form a padded image $f_p(x,y)$, of size $P \times Q$ by appending the necessary number of zeros to $f(x,y)$.

(3) Multiply $f_p(x,y)$ by $(-1)^{x+y}$ to center its transform.

(4) Compute the DFT, $F(u,v)$, of the image from step (3).

(5) Generate a real, symmetric filter function, $H(u,v)$, of size $P \times Q$ with center at coordinates $(P/2, Q/2)$[3]. Form the product $G(u,v) = H(u,v)F(u,v)$ using array multiplication; that is, $G(i,j) = H(i,j)F(i,j)$.
(6) Obtain the processed image:

$$g_p(x,y) = \left\{\text{real}\left[\mathscr{F}^{-1}[G(u,v)]\right]\right\}(-1)^{x+y}$$

where the real part is selected in order to ignore parasitic complex components resulting from computational inaccuracies, and the subscript p indicates that we are dealing with padded arrays.
(7) Obtain the final processed result, $g(x,y)$, by extracting the $M \times N$ region from the top, left quadrant of $g_p(x,y)$.

6.4. Correspondence Between Filtering in the Spatial and Frequency Domains. The link between filtering in the spatial and frequency domains in the convolution theorem. In practice, we prefer to implement convolution with small filter masks because of computational coasts and speed. However, DFT and IDFT give us a more intuitive idea. One way to take advantage of the properties of both domains is to specify a filter in the frequency domain, compute its IDFT, and then use the resulting, full-sized spatial filter as a *guide* for constructing smaller spatial filter masks.
In the following discussion, we use Gaussian filters to illustrate how frequency domain filters can be used as guides for specifying the coefficients of some of the small masks. Filters based on Gaussian functions are of particular interest because both the forward and inverse Fourier transforms of a Gaussian function area real Gaussian functions.
We limit the discussion to 1-D to illustrate the underlying principles. Two-dimensional Gaussian filters are discussed in the next section.
Let $H(u)$ denote the 1-D frequency domain Gaussian filter:

$$H(u) = Ae^{-u^2/2\sigma^2} \tag{84}$$

where σ is the standard deviation of the Gaussian curve. The corresponding filter in the spatial domain is obtained by taking the inverse Fourier transform of $H(u)$:

$$h(x) = \sqrt{2\pi}\sigma A e^{-2\pi^2\sigma^2 x^2}. \tag{85}$$

These equations are important for two reasons: (1) They are a Fourier transform pair, both components of which are Gaussian and *real*. This facilitates analysis because we do not have to be concerned with complex numbers. In addition, Gaussian curves are intuitive and easy to manipulate. (2) The functions behave reciprocally. When $H(u)$ has a broad profile (large value of σ), $h(x)$ has a narrow profile, and vice versa. In fact, as σ approaches infinity, $H(u)$ tends toward a constant function and $h(x)$ tends toward an impulse, which implies no filtering in the frequency and spatial domains, respectively.
Figure 13(a) and (b) show plots of a Gaussian lowpass filter in the frequency domain and the corresponding lowpass filter in the spatial domain. Suppose that we want to use the shape of $h(x)$ in Figure 13(b) as a *guide* for specifying the coefficients of a small spatial mask.The key similarity between the two filters is that all their values are positive. Thus, we conclude that we can implement lowpass filtering in the spatial domain by using a mask with all positive coefficients. For reference, Figure 13(b) shows two of the masks discussed in Section 5. Note the reciprocal relationship between the width of the filters, as discussed in the previous paragraph. The narrower the frequency domain filter, the more it will attenuate the low frequencies, resulting in increased blurring. In the spatial domain, this means that a larger mask must be used to increase blurring.

[3]If $H(u,v)$ is to be generated from a given spatial filter, $h(x,y)$, then we form $h_p(x,y)$ by padding the spatial filter to size $P \times Q$, multiply the expanded array by $(-1)^{x+y}$, and compute the DFT of the result to obtain a centered $H(u,v)$.

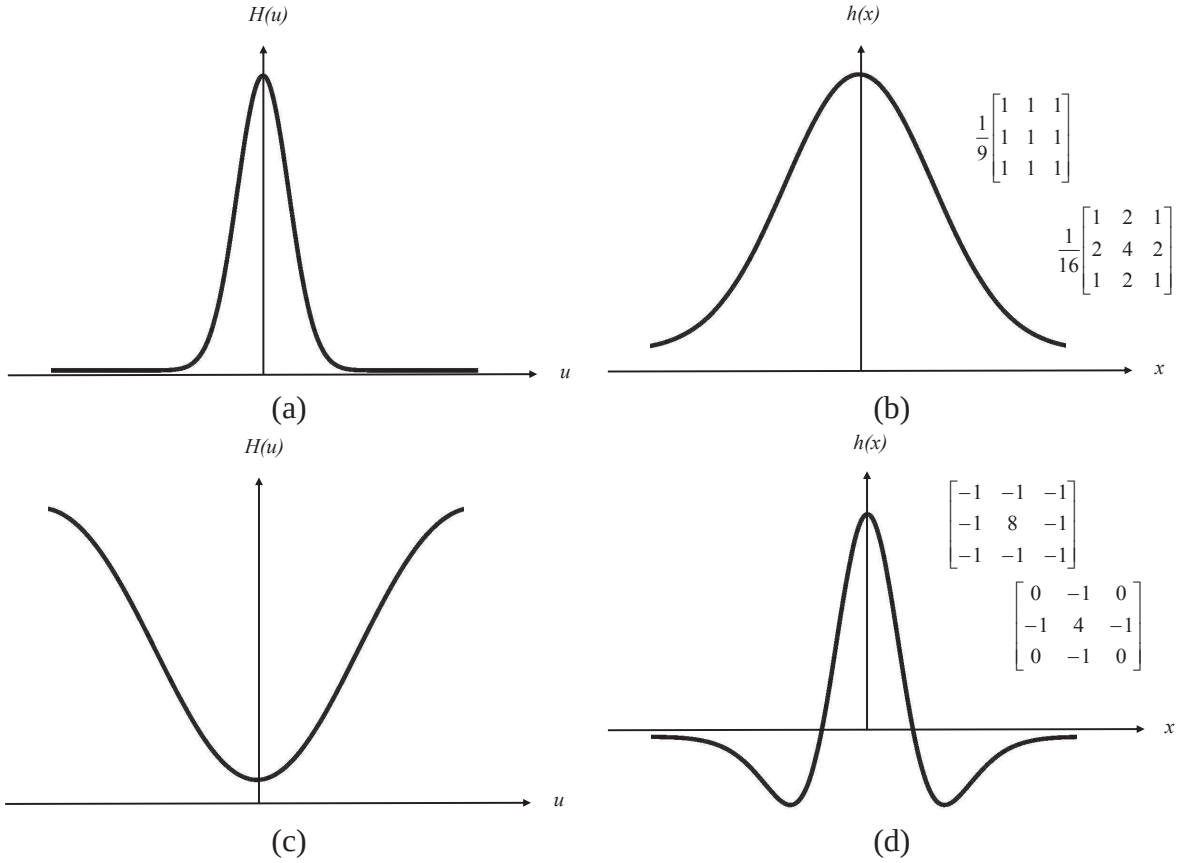


FIGURE 13. (a) A 1-D Gaussian lowpass filter in the frequency domain. (b) Spatial lowpass filter corresponding to (a). (c) Gaussian highpass filter in the frequency domain. (d) Spatial highpass filter corresponding to (c). The small 2-D masks shown are spatial filters discussed in Section 5.

More complex filters can be constructed using the basic Gaussian function of Equation (84). For example, we can construct a highpass filter as the *difference* of Gaussians:

$$H(u) = Ae^{-u^2/2\sigma_1^2} - Be^{-u^2/2\sigma_2^2}$$

with $A \geq B$ and $\sigma_1 > \sigma_2$. The corresponding filter in the spatial domain is

$$h(x) = \sqrt{2\pi}\sigma_1 Ae^{-2\pi^2\sigma_1^2x^2} - \sqrt{2\pi}\sigma_2 Be^{-2\pi^2\sigma_2^2x^2}.$$

Figure 13(c) and (d) show plots of these two equations. We note again the reciprocity in width, but the most important feature here is that $h(x)$ has a positive center term with negative terms on either side. The small masks shown in Figure 13(d) capture this property.

6.5. Image Smoothing Using Frequency Domain Filters. The remainder of this chapter deals with various filtering techniques in the frequency domain. We begin with lowpass filters. Edges and other sharp intensity transitions (such as noise) in an image contribute significantly to the high-frequency content of its Fourier transform. Hence, smoothing (blurring) is achieved in the frequency domain by high-frequency attenuation; that is, by *lowpass* filtering. In this section, we consider three types of lowpass filters: ideal, Butterworth, and Gaussian. These three categories cover the range from very sharp (ideal) to very smooth (Gaussian) filtering. The Butterworth filter has a parameter called the *filter order*. For high order values, the Butterworth approaches the ideal filter. For lower

Ideal	Butterworth	Gaussian
$H(u,v)=\begin{cases}1 & \text{if } D(u,v)\le D_0\\ 0 & \text{if } D(u,v)>D_0\end{cases}$	$H(u,v)=\dfrac{1}{1+[D(u,v)/D_0]^{2n}}$	$H(u,v)=e^{-D^2(u,v)/2D_0^2}$

TABLE 1. Lowpass filters. D_0 is the cutoff frequency and n is the order of the Butterworth filter.

order values, it behaves more like a Gaussian filter. Thus, the Butterworth filter may be viewed as providing a transition between two "extremes".

6.5.1. *Ideal Lowpass Filters.* A 2-D lowpass filter that passes without attenuation all frequencies within a circle of radius D_0 from the origin and "cuts off" all frequencies outside this circle is called an *ideal lowpass filter* (ILPF); it is specified by the function

$$H(u,v)=\begin{cases}1 & \text{if } D(u,v)\le D_0\\ 0 & \text{if } D(u,v)>D_0\end{cases}$$

where D_0 is a positive constant and $D(u,v)$ is the distance between a point (u,v) in the frequency domain and the center of the frequency rectangle; that is

$$D(u,v)=\left[(u-P/2)^2+(v-Q/2)^2\right]^{1/2}$$

where, as before, P and Q are the padded sizes. The ideal lowpass filter is radially symmetric about the origin, which means that the filter is completely defined by a radial cross section.

6.5.2. *Butterworth Lowpass Filters.* The transfer function of a Butterworth lowpass filter (BLPF) of order n, and with cutoff frequency at a distance D_0 from the origin, is defined as

$$H(u,v)=\frac{1}{1+[D(u,v)/D_0]^{2n}}$$

where $D(u,v)$ is defined similarly as before. Unlike the ILPF, the BLPF transfer function does not have a sharp discontinuity that gives a clear cutoff between passed and filtered frequencies. For filters with smooth transfer functions, defining a cutoff frequency locus at points for which $H(u,v)$ is down to a certain fraction of its maximum value is customary.

6.5.3. *Gaussian Lowpass Filters.* Gaussian lowpass filters (BLPFs) of two dimension is given by

$$H(u,v)=e^{-D^2(u,v)/2\sigma^2}.$$

Here we do not use a multiplying constant as in Section 5 in order to be consistent with the filters discussed in the present section, whose highest value is 1. As before, σ is a measure of spread about the center. By letting $\sigma=D_0$, we can express the filter using the notation of the other filters in this section:

$$H(u,v)=e^{-D^2(u,v)/2D_0^2}$$

where D_0 is the cutoff frequency. When $D(u,v)=D_0$, the GLPF is down to 0.607 of its maximum value. All three filters are summarized in Table 1.

6.6. Image Sharpening Using Frequency Domain Filters. In the previous section, we showed that an image can be smoothed by attenuating the high-frequency components of its Fourier transform. Because edges and other abrupt changes in intensities are associated with high-frequency components, image sharpening can be achieved in the frequency domain by highpass filtering, which attenuates the low-frequency components without disturbing high-frequency information in the Fourier transform. We consider only zero-phase-shift filters that are radially symmetric. A highpass filter is obtained from a given lowpass filter using the equation

$$H_{HP}(u,v)=1-H_{LP}(u,v) \tag{86}$$

Ideal	Butterworth	Gaussian
$H(u,v) = \begin{cases} 0 & \text{if} D(u,v) \le D_0 \\ 1 & \text{if} D(u,v) > D_0 \end{cases}$	$H(u,v) = \dfrac{1}{1+[D_0/D(u,v)]^{2n}}$	$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$

TABLE 2. Highpass filters. D_0 is the cutoff frequency and n is the order of the Butterworth filter.

where H_{LP} is the transfer function of the lowpass filter. That is, when the lowpass filter attenuates frequencies, the highpass filter passes them, and vice versa. Guided by this principle, we obtain three highpass filters given in Table 2. A variety of applications using these and other filters cab be found in [**23**].

7. Implementation

We have focused attention thus far on the theoretical concepts and on examples of filtering in the frequency domain. Ont thing that should be clear by now is that computational requirements in this area of image processing are not trivial. Thus, it is important to develop a basic understanding of methods by which Fourier transform computations can be simplified and speeded up. This section deals with these issues.

7.1. Separability of the 2-D DFT. The 2-D DFT is separable into 1-D transform. We can write Equation (74) as

$$F(u,v) = \sum_{x=0}^{M-1} e^{-i2\pi ux/M} \sum_{y=0}^{N-1} f(x,y) e^{-i2\pi vy/N} \tag{87}$$

$$= \sum_{x=0}^{M-1} F(x,v) e^{-i2\pi ux/M} \tag{88}$$

where

$$F(x,v) = \sum_{y=0}^{N-1} f(x,v) e^{-i2\pi vy/N}. \tag{89}$$

For each value of x and for $v = 0,1,2,\ldots,N-1$, we see that $F(x,v)$ is simply the 1-D DFT of a row of $f(x,y)$. By varying x from 0 to $M-1$ in Equation (89), we compute a set of 1-D DFTs for all rows of $f(x,y)$. The computations in Equation (87) similarly are 1-D transforms of the columns of $F(x,v)$. Thus, we conclude that the 2-D DFT of $f(x,y)$ can be obtained by computing the 1-D transform of each row of $f(x,y)$ and then computing the 1-D transform along each column of the result. This is an important simplification because we have to deal only with one variable at a time. A similar development applies to computing the 2-D IDFT using the 1-D IDFT. However, as we show in the following section, we can compute the IDFT using an algorithm designed to compute the DFT.

7.2. Computing the IDFT Using a DFT Algorithm. Taking the complex conjugate of both sizes of Equation (75) and multiplying the results by MN yields

$$MNf^*(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u,v) e^{-i2\pi(ux/M+vy/N)}. \tag{90}$$

But, we recognize the form of the right side of this result as the DFT of $F^*(u,v)$. Therefore, Equation (90) indicates that if we substitute $F^*(u,v)$ into an algorithm designed to compute the 2-D forward Fourier transform, the result will be MNf^*. Taking the complex conjugate and multiplying this result by MN yields $f(x,y)$, which is the inverse of $F(u,v)$.

Computing the 2-D inverse from a 2-D forward DFT algorithm that is based on successive passes of 1-D transforms (as in the previous section) is a frequent source or confusion involving the complex conjugates and multiplication by a constant, neither of which is done in the 1-D algorithms. The key concept to keep in mind is that we simply input $F^*(u,v)$ into whatever forward algorithm we have. The result will be $MNf^*(x,y)$. All we have to do with this result to obtain $f(x,y)$ is to take its complex conjugate and multiply it by the constant MN. Of course, when $f(x,y)$ is real, as typically is the case, $f^*(x,y) = f(x,y)$.

7.3. The Fast Fourier Transform (FFT). Work in the frequency domain would not be practical if we had to implement Equations (74) and (75) directly. Brute-force implementation of these equations requires on the order of $(MN)^2$ summations and additions. For images of moderate size (say, 1024×1024 pixels), this means on the order of a trillion multiplications and additions for just one DFT, excluding the exponentials, which could be computed once and stored in a look-up table. This would be a challenge even for super computers. Without the discovery of the *fast Fourier transform (FFT)*, which reduces computations to the order of $MN \log_2 MN$ multiplications and additions, it is safe to say that the material presented in this chapter would be of little practical value. The computational reductions afforded by the FFT are impressive indeed. For example, computing the 2-D FFT of a 1024×1024 image would require on the order of 20 million multiplication and additions, which is a significant reduction from the one trillion computations mentioned above.

Although the FFT is a topic covered extensively in the literature on signal processing, this subject matter is of such significance in our work that this chapter be incomplete if we did not provide at least an introduction explaining why the FFT works as it does. The algorithm we selected to accomplish this objective is the so-called *successively-doubling* method, which was the original algorithm that led to the birth of an entire industry [**7**].

This algorithm assumes that the number of samples, M, is an integer power of 2, i.e., $M = 2^n$ for some n. But this is not a general requirement of other approaches. Suppose $M = 2K$, then

$$\begin{aligned}
F(u) &= \sum_{x=0}^{M_1} f(x)e^{-i2\pi ux/M} \\
&= \sum_{x=0}^{2K-1} f(x)e^{-i2\pi ux/2K} \\
&= \sum_{x=0}^{K-1} f(2x)e^{-i2\pi u(2x)/2K} + \sum_{x=0}^{K-1} f(2x+1)e^{-i2\pi u(2x+1)/2K} \quad \text{(splitting into even and odd parts)} \\
&= \sum_{x=0}^{K-1} f(2x)e^{-i2\pi ux/K} + \sum_{x=0}^{K-1} f(2x+1)e^{-i2\pi ux/K} \cdot e^{-i2\pi u/2K} \\
&= F_e(u) + e^{-i2\pi u/2K} \cdot F_o(u).
\end{aligned}$$

With simple calculations, we can obtain $F(u+K) = F_e(u) - F_o(u) \cdot e^{-i2\pi u/2K}$. These calculations reveal some interesting properties of these expressions. An M-point transform can be computed by dividing the original expression into two parts. Computing the first half of $F(u)$ requires evaluation of the two $(M/2)$-point transforms in F_e and F_o. The other half then following directly from the equation $F(u+K) = F_e(u) - F_o(u) \cdot e^{-i2\pi u/2K}$ without additional transform evaluations.

CHAPTER 7

Wavelet and Multiresolution Analysis

1. Introduction

Although the Fourier Transform has been the mainstay of transform-based image processing since the late 1950s, a more recent transformation, called the *wavelet transform*, is now making it even easier to compress, transmit, and analyze many images. Unlike the Fourier transform, whose basis functions are sinusoids, wavelet transform are based on small waves, called *wavelets*, of varying frequency and *limited duration*. This allows them to provide the equivalent of a musical score for an image, revealing not only what notes (or frequencies) to play but also when to play them. Fourier transforms, on the other hand, provide only the notes or frequency information; temporal information is lost in the transform process.

In 1987, wavelets were first shown to be the foundation of a powerful new approach to signal processing and analysis called *multiresolution* theory [**39**]. Multiresolution theory incorporates and unifies techniques from a variety of disciplines, including subband coding from signal process, quadrature minor filtering from digital speech recognition, and pyramidal image processing. As its name implies, multiresolution theory is concerned with the representation and analysis of signals (or images) at more than one resolution. The appeal of such an approach is obvious — features that might go undetected at one resolution may be easy to detect at another. Although the imaging community's interest in multiresolution analysis was limited until the late 1980s, it is now difficult to keep up with the number of papers, theses, and books devoted to the subject.

In this chapter, we examine wavelet-based transformations from a multiresolution point of view. Although such transforms can be presented in other ways, this approach simplifies both their mathematical and physical interpretations. We begin with an overview of imaging techniques that influenced the formulation of multiresolution theory. Our objective is to introduce the theory's fundamental concepts within the context of image processing the simultaneously provide a brief historical perspective of the method and its application. The bulk of this chapter is focused on the development and use of the discrete wavelet transform.

A powerful, yet conceptually simple structure for representing images at more than one resolution is the *image pyramid* [**9**]. Originally devised for machine vision and image compression applications, an image pyramid is a collection of decreasing resolution images arranged in the shape of a pyramid. As can be see in Figure 1(a), the the base of the pyramid contains a high-resolution representation of the image being processed; the apex contains a low-resolution approximation. As you move up the pyramid, both size and resolution decrease. Base level 0 is of size $M \times N$, apex level is of size 1×1. The number of maximum levels that can be computed depends on the image resolution, M and N. Without loss of generality, let $N = \min(M,N)$, then the maximum number of levels one can take is $J = \log_2 N$. The image resolution at a general level j is $M/2^j \times N/2^j$, where $0 \leq j \leq J$. Figure 1(b) gives an idea how image size changes as we move up the image pyramid. We are now ready for the formal discussion of the Multiresolution analysis.

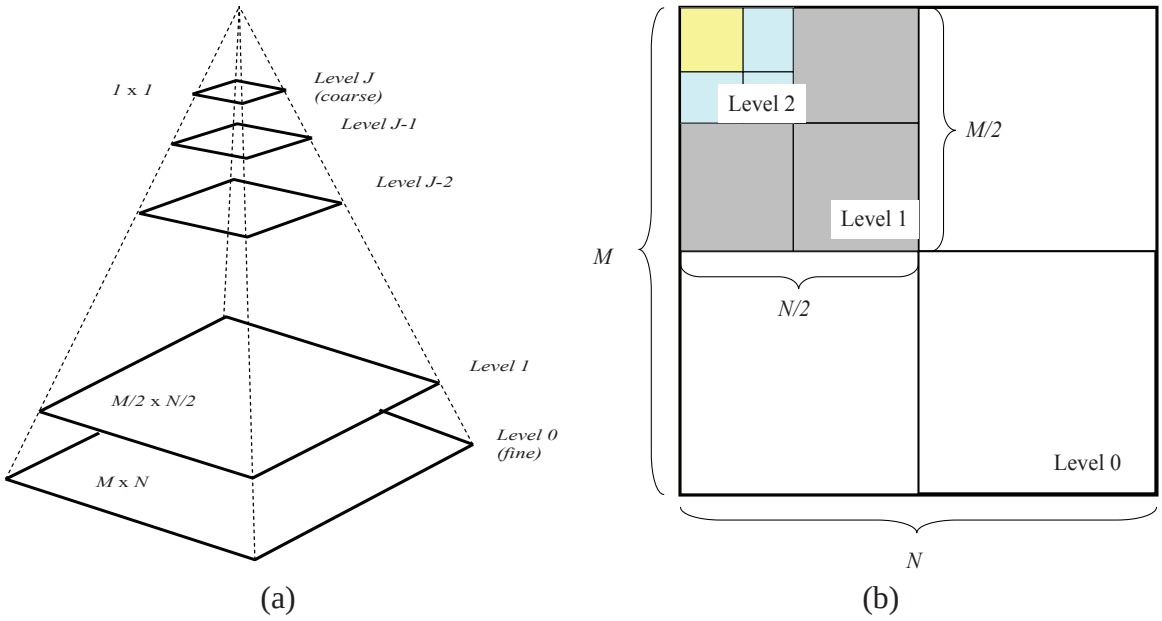


FIGURE 1. (a) An image pyramid. (b) An illustration of how image resolution changes as one moves up the pyramid.

2. The Continuous Wavelet Transform

The continuous wavelet transform (CWT) is defined by

$$X(b,a) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t)\overline{\psi\left(\frac{t-b}{a}\right)}\,dt, \tag{91}$$

where $a, b \in \mathbb{R}$ (typically a represents the *scale* parameter and b is the *shift* parameter) and $\bar{\psi}$ is the complex conjugate of the possibly complex function ψ. One interpretation of the transform $X(b,a)$ is that it provides a measure of similarity between the signal $x(t)$ and the continuously translated and dilated *mother wavelet* $\psi(t)$. The inverse wavelet transform is then provided by

$$x(t) = \frac{1}{\sqrt{|a|C_\psi}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} X(b,a)\psi\left(\frac{t-b}{a}\right)\frac{da\,db}{a^2}, \tag{92}$$

where

$$C_\psi = 2\pi \int_{-\infty}^{\infty} \frac{|\tilde{\psi}(\omega)|^2}{|\omega|}\,d\omega.$$

Here $\mathscr{F}(\psi(t)) = \tilde{\psi(\omega)}$, i.e., the Fourier integral transform of $\psi(t)$. Together, Equations (91) and (92) form the CWT pair.

2.1. Discretization of the CWT. The number of instances for which the CWT of a signal may be computed analytically is very small. In general, the evaluation of the CWT $X(b,a)$ is actually done numerically at a discrete set of points. So, consider the discretization of the (b,a)-plane. We consider the restriction of $X(b,a)$ to a collection of fixed values, a_j:

$$a_j = \alpha^j \quad \text{where} \quad \alpha > 1 \quad \text{and} \quad j \in \mathbb{Z}.$$

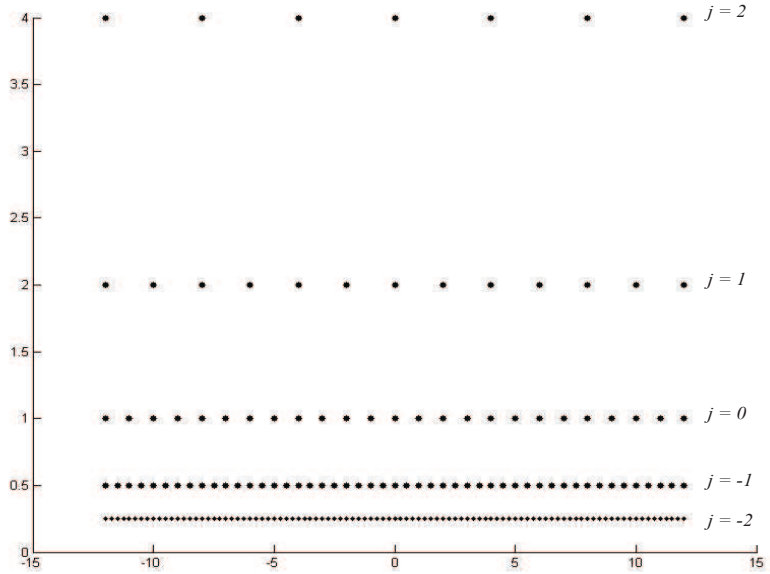


FIGURE 2. The dyadic sampling grid with $\alpha = 2$, $\beta = 1$. The horizontal axis represents the shift b and the vertical axis represents scale a.

Observe that the width of the function $\psi((t-b)/\alpha^j)$ is dependent on the magnitude of α^j. In fact is it α^j times wider, in terms of its support[1], than the mother wavelet, i.e.,

$$\text{supp}\psi\left(\frac{t-b}{\alpha^j}\right) = \alpha^j \text{supp}\psi(t).$$

This fact must be taken into account when discretizing this shift variable b. The discretized shift size should be proportional to the width of the dilated wavelet. This may be accomplished by taking

$$b_k^j = k\beta\alpha^j, \quad \beta > 0\,(\text{typically between}\,0.1\,\text{and}\,1.0).$$

Putting these all together, we write the wavelet on the discretized grid as

$$\begin{aligned} g_{ab} &= \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \\ &= \alpha^{-1/2j}\psi\left(\frac{t-k\beta\alpha^j}{\alpha^j}\right) \\ &= \alpha^{-j/2}\psi(\alpha^{-j}t - k\beta) \\ &= \psi_k^j(t) \end{aligned}$$

and the discretized wavelet coefficient from Equation (91) would be

$$X(b_k, a_j) = 2^{-j/2}\int_{-\infty}^{\infty} x(t)\overline{\psi(2^{-j}t-k)}\,dt.$$

EXAMPLE 2.1. Let $\alpha = 2$ and $\beta = 1$. This coarse sampling of the (b,a)-plane produces a *dyadic* grid, that is used a lot in practice. See Figure 2 for an illustration.
In general, calculations of these wavelet coefficients can be expensive, one needs to be smart about choosing appropriate α and β values.

[1]The support of a function $f(x)$, denoted supp $f(x)$, is the closure of the subset of its domain (i.e., that subset of f and its boundary) for which $f(x) \neq 0$. A function is said to have compact support if its support is closed and bounded.

3. Multiresolution Analysis (MRA)

In MRA, a *scaling function* (father wavelet) is used to create a series of approximations of a function or image, each differing by a factor of 2 in resolution from its nearest neighboring approximations. Additional functions, called *wavelets* (dilated and shifted versions of mother wavelet ψ), are then used to encode the difference in information between adjacent approximations.

3.1. Series Expansions. A signal or function $f(x)$ can often be better analyzed as a linear combination of expansion functions

$$f(x) = \sum_k \alpha_k \phi_k(x), \tag{93}$$

where k is an integer index of a finite or infinite sum, the α_k are real-valued *expansion coefficients*, and the $\phi_k(x)$ are real-valued *expansion functions*. If the expansion is unique — that is, there is only one set of α_k for any given $f(x)$ — the $\phi_k(x)$ are called *basis* functions, and the expansion set $\{\phi_k(x)\}$, is called a *basis* for the class of functions that can be so expressed. The expressible functions form a *function space* that is referred to as the *closed span* of the expansion set, denoted

$$V = \overline{\text{Span}_k\{\phi_k(x)\}}.$$

To say that $f(x) \in V$ means that $f(x)$ is in the closed span of $\{\phi_k(x)\}$ and can be written in the form of Equation (93).

For any function space V and corresponding expansion set $\{\phi_k(x)\}$, there is a set of *dual* functions denoted $\{\tilde{\phi}_k(x)\}$ that can be used to compute the α_k coefficients of Equation (93) for any $f(x) \in V$. These coefficients are computed by taking the *integral inner products*[2] of the dual $\tilde{\phi}_k(x)$ and function $f(x)$. That is

$$\alpha_k = \langle \tilde{\phi}_k(x), f(x) \rangle = \int \overline{\tilde{\phi}_k(x)}\, f(x)\, dx. \tag{94}$$

Depending on the orthogonality of the expansion set, this computation assumes one of three possible forms.

Case 1: If the expansion functions form an orthonormal basis for V, meaning that

$$\langle \phi_j(x), \phi_k(x) \rangle = \delta_{jk} = \begin{cases} 0 & j \neq k \\ 1 & j = k \end{cases}$$

the basis and its dual are equivalent, i.e., $\phi_k(x) = \tilde{\phi}_k(x)$. So,

$$\alpha_k = \langle \phi_k(x), f(x) \rangle. \tag{95}$$

Case 2: If the expansion functions are not orthonormal, but are an orthogonal basis for V, then

$$\langle \phi_j(x), \phi_k(x) \rangle = 0 \quad j \neq k$$

and the basis functions and their duals are called *biorthogonal*. The α_k are computed using Equation (94), and their biorthogonal basis and its dual are such that

$$\langle \phi_j(x), \tilde{\phi}_k(x) \rangle = \delta_{jk} = \begin{cases} 0 & j \neq k \\ 1 & j = k \end{cases}$$

[2]The inner product of two real or complex-valued function $f(x)$ and $g(x)$ is $\langle f(x), g(x) \rangle = \int \overline{f(x)} g(x)\, dx$, where $\overline{f(x)}$ is the complex conjugate of $f(x)$.

Case 3: If the expansion set is not a basis for V, but supports the expansion defined in Equation (93), it is a spanning set in which that is more than one set of α_k for any $f(x) \in V$. The expansion functions and their duals are said to be *overcomplete* or redundant. They form a *frame* in which[3]

$$A\|f(x)\|^2 \leq \sum_k |\langle \phi_k(x), f(x)\rangle|^2 \leq B\|f(x)\|^2$$

for some $A > 0$, $B < \infty$, and all $f(x) \in V$. Dividing this equation by the norm squared of $f(x)$, we see that A and B "frame" the normalized inner products of the expansion coefficients and the function. Equations similar to Equation (94) and (95) can be used to find the expansion coefficients for frames. If $A = B$, the expansion set is called a *tight frame* and it can be shown that [**12**]

$$f(x) = \frac{1}{A}\sum_k \langle \phi_k(x), f(x)\rangle \phi_k(x).$$

3.2. Scaling Functions. Consider the set of expansion functions composed of integer translations and binary scalings of the real, square-integrable function $\phi(x)$; that is the set $\{\phi_k^j(x)\}$, where

$$\phi_k^j(x) = 2^{-j/2}\phi\left(2^{-j}x - k\right) \tag{96}$$

for all $j,k \in \mathbb{Z}$ and $\phi(x) \in L^2(\mathbb{R})$. Here, k determines the position of $\phi_k^j(x)$ along the x-axis, and j determines the width of $\phi_k^j(x)$ — that is, how broad or narrow it is along the x-axis. The term $2^{-j/2}$ controls the amplitude of the function. Because the shape of of $\phi_k^j(x)$ changes with j, $\phi(x)$ is called a *scaling function*. By choosing $\phi(x)$ properly, $\{\phi_k^j(x)\}$ can be made to $L^2(\mathbb{R})$, which is the set of all measurable, square-integrable functions. Further define

$$V_j = \overline{\text{span}_k\{\phi_k^j(x)\}}$$

to be the subspace spanned by the set of $\phi_k^j(x)$ ranging all values of k. As will be seen in the following example, decreasing j decreases the size of V_j, allowing functions with smaller variations or finer details to be included in the subspace. This is a consequence of the fact that, as j decreases, the $\phi_k^j(x)$ that are used to represent the subspace functions become narrower and separated by smaller changes in x.

EXAMPLE 3.1. Consider the unit-height, unit-width scaling function [**26**]

$$\phi(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{97}$$

Figure 3 show six of the many expansion functions that can be generated by substituting this pulse-shaped scaling function into Equation (96). Notice that Figure 3(a) and (b) are members of V_0 and do not belong to V_1 since members of V_1 (e.g., Figure 3(c) and (d)) are too coarse to represent them. On the other hand, Figure 3(e) and (f), which are elements in V_{-1}, could be used to represent the functions in Figure 3(a) and (b) since they are of higher resolution.

The simple scaling function in the preceding example obeys the four fundamental requirements of multiresolution analysis [**41**]:

[3]The norm of $f(x)$, denoted $\|f(x)\|$, is defined as the square root of the absolute value of the inner product of $f(x)$ with itself.

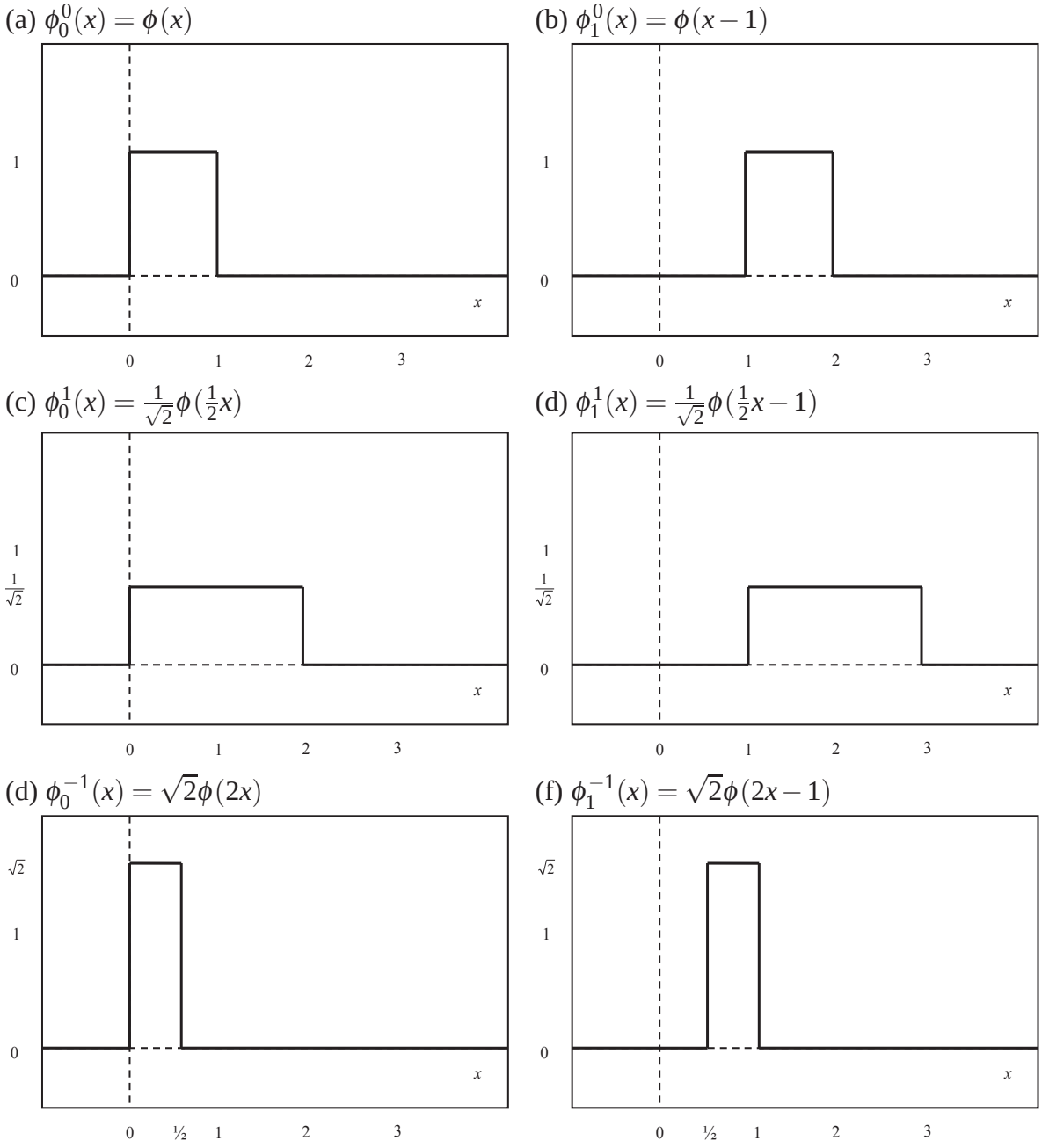


FIGURE 3. Some Haar scaling functions.

MRA Requirement 1: The scaling function is orthogonal to its integer translates.

This is easy to see in the case of Haar function, because whenever it has a value of 1, its integer translates are 0, so that the product of the two is 0. The Haar scaling function is said to have *compact support*, which means that it is 0 everywhere outside a finite interval called the *support*. In fact, the width of the support is 1; it is 0 outside the half open interval $[0,1)$. It should be noted that the requirement for orthogonal integer translates becomes harder to satisfy as the width of support of the scaling function becomes larger than 1.

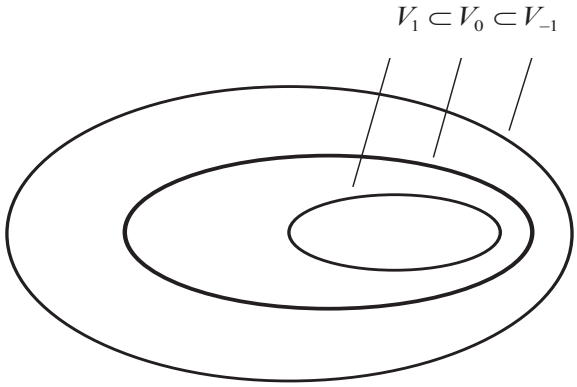


FIGURE 4. The nested function spaces spanned by a scaling function.

MRA Requirement 2: The subspaces spanned by the scaling function at low scales are nested within those spanned at higher scales.

As can be seen in Figure 4, subspaces containing high-resolution functions must also contain all lower resolution functions. That is

$$V_\infty \subset \cdots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \cdots \subset V_{-\infty}$$

Moreover, the subspaces satisfy the intuitive condition that if $f(x) \in V_j$, then $f(2x) \in V_{j-1}$.

MRA Requirement 3: The only function that is common to all V_j is $f(x) = 0$.

If we consider the coarsest possible expansion functions (i.e., $j = \infty$), the only representable function is the function of no information. That is,

$$V_\infty = \{0\}.$$

MRA Requirement 4: Any function can be represented with arbitrary precision.

Though it may not be possible to expand a particular $f(x)$ at an arbitrarily coarse resolution, all measurable, square-integrable functions can be represented by the scaling functions in the limit as $j \to -\infty$. That is,

$$V_{-\infty} = \{L^2(\mathbb{R})\}.$$

Under these assumptions, the coarser scale can be represented by the finer scale as a weighted sum:

$$\phi_k^{j+1} = \sum_n h_n \phi_n^j(x).$$

But

$$\phi_n^j(x) = 2^{-j/2}\phi(2^{-j}x - n),$$

thus

$$\phi_k^{j+1}(x) = \sum_n h_n \cdot 2^{-j/2}\phi(2^{-j}x - n).$$

When $j+1 = 0 = k$ ($j = -1$), we get the generic non-subscripted expression

$$\phi(x) = \sum_n h_n \sqrt{2}\phi(2x - n) = \sqrt{2}\sum_n h_n \phi(2x - n). \tag{98}$$

The h_n coefficients in this recursive equation are called *scaling coefficients*. This equation is fundamental to multiresolution analysis and is called the *refinement equation*, the *MRA equation*, or the

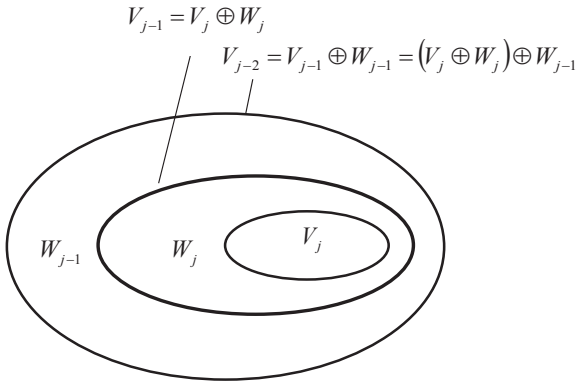


FIGURE 5. The relationship between scaling and wavelet function spaces.

dilation equation. It says that the expansion functions of any subspace can be built from double-resolution copies of themselves — that is, from expansion functions of the next higher resolution space. Tho choice of a reference subspace is arbitrary.

EXAMPLE 3.2. The scaling coefficients for the Haar function are $h_0 = h_1 = 1/\sqrt{2}$, thus

$$\phi(x) = \sqrt{2}\left(\frac{1}{\sqrt{2}}\phi(2x) + \frac{1}{\sqrt{2}}\phi(2x-1)\right) = \phi(2x) + \phi(2x-1).$$

3.3. Wavelet Functions. Given a scaling function that meets the MRA requirements of the previous section, we can define a *wavelet function* $\psi(x)$ that, together with its integer translates and binary scalings, spans the difference between any two adjacent scaling subspaces, V_j and V_{j-1}. The situation is illustrated graphically in Figure 5. We define the set $\{\psi_k^j(x)\}$ of wavelets and its subspace closure as

$$\psi_k^j(x) = 2^{-j/2}\psi(2^{-j}x - k) \tag{99}$$

and

$$W_j = \overline{\text{span}_k\{\psi_k^j(x)\}}.$$

The scaling and wavelet function subspaces in Figure 5 are related by

$$V_{j-1} = V_j \oplus W_j \tag{100}$$

where $\oplus$ denotes the union of spaces. The orthogonal complement of V_j in V_{j-1} is W_j, and all members of V_j are orthogonal to the members of W_j. Thus,

$$\langle \phi_k^j(x), \psi_l^j(x) \rangle = 0$$

for all appropriate $j,k,l \in \mathbb{Z}$.
Since wavelet spaces reside within the spaces spanned by the next higher resolution scaling functions, any wavelet function — like its scaling function counterpart — can be expressed as a weighted sum of shifted, double-resolution scaling functions. That is, we can write

$$\psi(x) = \sqrt{2}\sum_n g_n \phi(2x-n) \tag{101}$$

where the g_n are called the *wavelet coefficients*. Using the condition that wavelets span the orthogonal complement spaces and that integer wavelet translates are orthogonal, it can be shown that g_n is related to h_n by

$$g_n = (-1)^n h_{1-n}.$$

EXAMPLE 3.3. In the previous example, $h_0 = h_1 = \frac{1}{2}$. Using $g_n = (-1)h_{1-n}$, we get $g_0 = (-1)^0 h_{1-0} = \frac{1}{\sqrt{2}}$ and $g_1 = (-1)^1 h_{1-1} = -\frac{1}{\sqrt{2}}$. Substituting these values into Equation (101), we get

$$\psi(x) = \phi(2x) - \phi(2x-1),$$

which is plotted in Figure 6(a). Thus, the Haar wavelet function is

$$\psi(x) = \begin{cases} 1 & 0 \le x < 1/2 \\ -1 & 1/2 \le x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{102}$$

Using Equation (99), we can now generate the universe of scaled and translated Haar wavelets. Two such wavelets, $\psi_2^0(x)$ and $\psi_0^{-1}(x)$ are plotted in Figure 6(b) and (c), respectively. Note that wavelet $\psi_0^{-1}(x)$ for space W_{-1} is narrower than $\psi_2^0(x)$ for W_0; it can be used to represent finer detail.
Figure 6(d) shows a function of subspace V_{-1} that is not in subspace V_0. Although this function cannot be represented accurately in V_0, Equation (100) indicates that it can be expanded using V_0 and W_0 expansion functions. The resulting expansion is

$$f(x) = f_a(x) + f_d(x)$$

where

$$f_a(x) = \frac{3\sqrt{2}}{4}\phi_0^0(x) - \frac{\sqrt{2}}{8}\phi_2^0(x)$$

and

$$f_d(x) = -\frac{\sqrt{2}}{4}\psi_0^0(x) - \frac{\sqrt{2}}{8}\psi_2^0(x).$$

Here, $f_a(x)$ is an approximation of $f(x)$ using V_0 scaling functions, while $f_d(x)$ is the difference $f(x) - f_a(x)$ as a sum of W_0 wavelets. The two expansions, which are shown in Figure 6(e) and (f), divide $f(x)$ in a manner similar to a lowpass and highpass filter. The low frequencies of $f(x)$ are captured in $f_a(x)$ — it assumes the average value of $f(x)$ in each integer interval — while the high-frequency details are encoded in $f_d(x)$.

4. Wavelet Transforms in One Dimension

4.1. The Wavelet Series Expansions. We begin by defining the *wavelet series expansion* of function $f(x) \in L^2(\mathbb{R})$ relative to wavelet $\psi(x)$ and scaling function $\phi(x)$. Recall that $V_j = V_{j+1} \oplus W_{j+1}$, which can be factored further

$$\begin{aligned} V_j &= V_{j+1} \oplus W_{j+1} \\ &= V_{j+2} \oplus W_{j+2} \oplus W_{j+1} \\ &= V_{j+3} \oplus W_{j+3} \oplus W_{j+2} \oplus W_{j+1} \\ &= \cdots \\ &= V_J \oplus W_J \oplus \cdots \oplus W_{j+1}. \end{aligned}$$

Thus,

$$\begin{aligned} f(x) &= f_a^j(x) + f_d^j(x) \\ &= \sum_k c_k^j \phi_k^j(x) + \sum_l d_l^j \psi_l^j(x) \\ &= \sum_k c_k^J \phi_k^J + \sum_{r=j+1}^{J} \sum_l d_l^r \psi_l^r(x). \end{aligned}$$

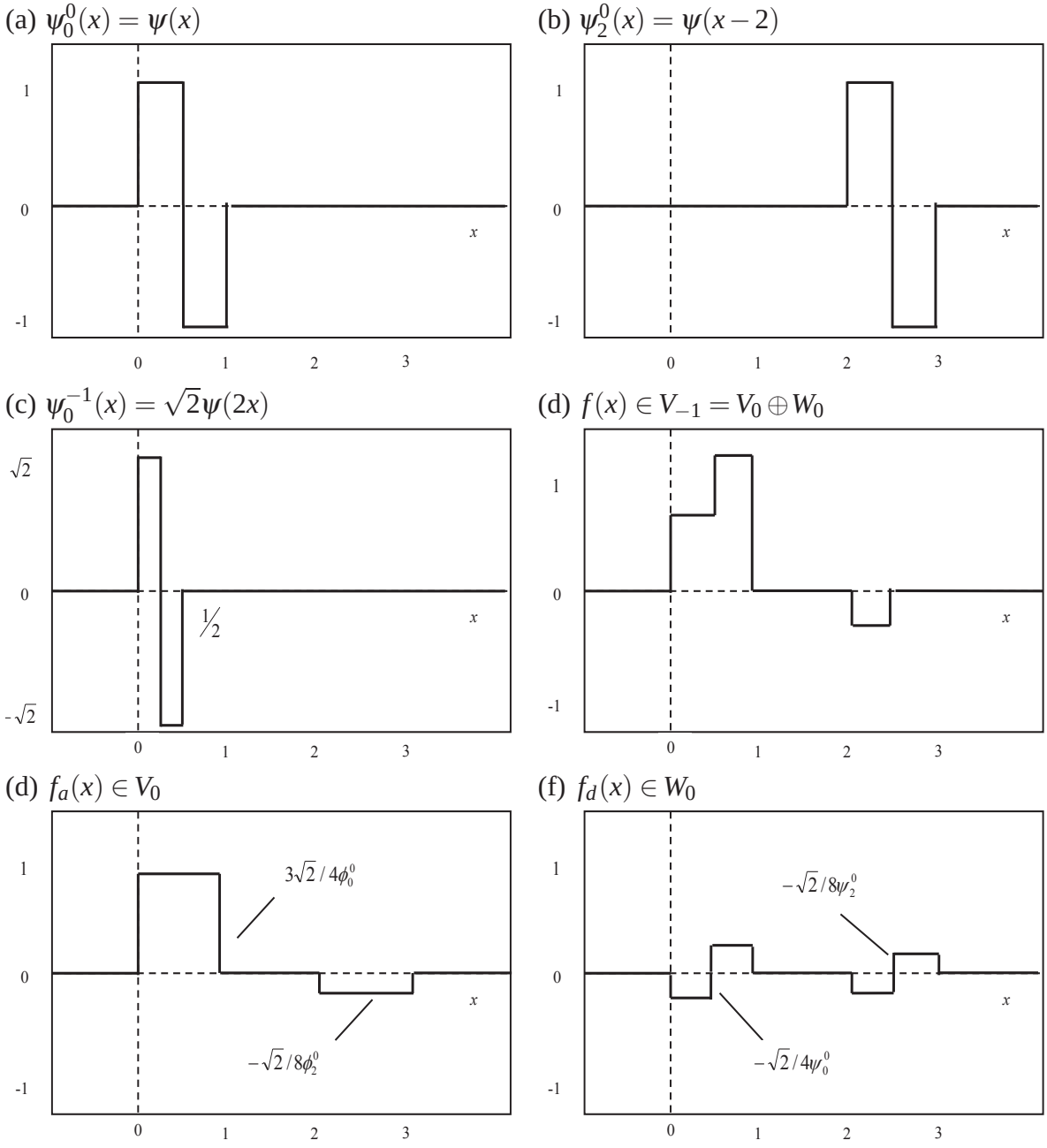


FIGURE 6. Haar wavelet functions in W_0 and W_{-1}.

The c_k's are normally called *approximation* and/or *scaling coefficients*; the d_k's are referred to as the *detail* and/or *wavelet coefficients*.

If the expansion functions form an orthonormal basis or tight frame, which is often the case, the expansion coefficients are calculated as

$$c_k^j = \langle f(x), \phi_k^j \rangle = \int_{-\infty}^{\infty} f(x)\phi_k^j(x)\,dx = \int_{-\infty}^{\infty} f(x)2^{-j/2}\phi\left(2^{-j}x-k\right)\,dx \tag{103}$$

and

$$d_k^j = \langle f(x), \psi_k^j \rangle = \int_{-\infty}^{\infty} f(x)\psi_k^j(x)\,dx = \int_{-\infty}^{\infty} f(x)2^{-j/2}\psi\left(2^{-j}x-k\right)\,dx. \tag{104}$$

EXAMPLE 4.1. Consider the function

$$y = \begin{cases} x^2 & 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$$

shown in Figure 7(a). Using Haar wavelets — see Equations (97) and (102) — and a starting scale $j = -2$. Along with the scaling and wavelet functions

$$\phi_0^0 = \phi(x), \quad \psi_0^0 = \psi(x), \quad \psi_0^{-1} = \sqrt{2}\psi(2x), \quad \psi_1^{-1} = \sqrt{2}\psi(2x-1),$$

Equations (103) and (104) can be used to compute the following expansion coefficients:

$$\begin{aligned}
c_0^0 &= \int_0^1 x^2 \phi_0^0(x)\,dx = \int_0^1 x^2\,dx = \frac{x^3}{3}\Big|_0^1 = \frac{1}{3} \\
d_0^0 &= \int_0^1 x^2 \psi_0^0(x)\,dx = \int_0^{1/2} x^2\,dx - \int_{1/2}^1 x^2\,dx = -\frac{1}{4} \\
d_0^{-1} &= \int_0^1 x^2 \psi_0^{-1}(x)\,dx = \int_0^{1/4} \sqrt{2}x^2\,dx - \int_{1/4}^{1/2} \sqrt{2}x^2\,dx = -\frac{\sqrt{2}}{32} \\
d_1^{-1} &= \int_0^1 x^2 \psi_1^{-1}(x)\,dx = \int_{1/2}^{3/4} \sqrt{2}x^2\,dx - \int_{3/4}^1 \sqrt{2}x^2\,dx = -\frac{3\sqrt{2}}{32}
\end{aligned}$$

Thus, the wavelet series expansion for y is

$$y = \underbrace{\underbrace{\underbrace{\frac{1}{3}\phi_0^0(x)}_{V_0} + \underbrace{\left[-\frac{1}{4}\psi_0^0(x)\right]}_{W_0}}_{V_{-1}=V_0\oplus W_0} + \underbrace{\left[-\frac{\sqrt{2}}{32}\psi_0^{-1}(x) - \frac{3\sqrt{2}}{32}\psi_1^{-1}(x)\right]}_{W_{-1}}}_{V_{-2}=V_{-1}\oplus W_{-1}=V_0\oplus W_0\oplus W_{-1}} + \cdots$$

The first term in this expansion uses c_0^0 to generate a subspace V_0 approximation of the function being expanded. This approximation is shown in Figure 7(b) and is the average value of the original function. The second term uses d_0^0 to refine the approximation by adding a level of detail from subspace W_0. The added detail and resulting V_{-1} approximation are shown in Figure 7(c) and (d), respectively. Another level of detail is added by the subspace W_{-1} coefficients d_0^{-1} and d_1^{-1}. This additional detail is shown in Figure 7(e), and the resulting V_{-2} approximation is depicted in Figure 7(f). Note that the expansion is now beginning to resemble the original function. As finer scales (greater levels of detail) are added, the approximation becomes a more precise representation of the function, realizing it in the limit as $j \to -\infty$.

4.2. The Pyramidal Algorithm. Wavelet analysis on a dyadic grid is a form of *multiresolution analysis (MRA)*. This framework is a powerful tool for representing scale information in data by decomposing it in terms of scaling and wavelet functions as described above. Furthermore, the MRA framework is the natural setting for deriving the pyramidal algorithm referred to next, which is used for efficient computation of the multiresolution decomposition and reconstruction of a function (or Mallat's algorithm [**40**]).

The decomposition of the function $f(x)$ into spaces requires a means to compute the content of $f(x)$ associated with each of the subspaces. This is accomplished by the appropriate projection operators. The entire discussion will be tremendously simplified by the fact that there is an especially simple relationship between these operators as revealed by the relationships between their expansion coefficients.

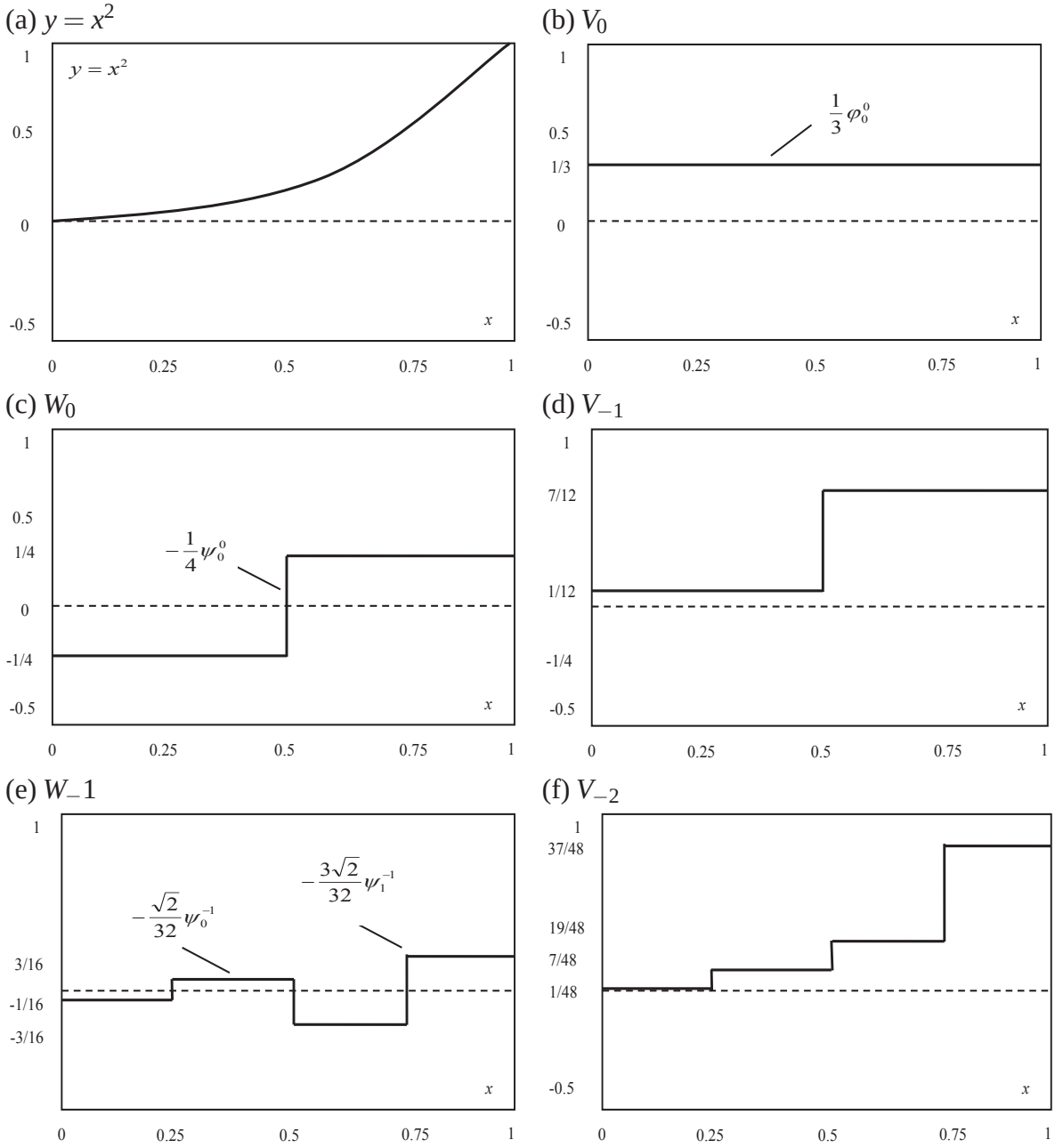


FIGURE 7. A wavelet series expansion of $y = x^2$ using Haar wavelets.

With this in mind we define the projection operators P_j and Q_j such that for any $f(x) \in L^2(\mathbb{R})$

$$P_j f \in V_j \quad \text{and} \quad Q_j f \in W_j.$$

As a consequence of the results established in the previous section, we have

$$P_{j-1} f = P_j f + Q_j f.$$

The Pyramidal Decomposition (finer to coarser)

Given the scaling coefficients $\{c_k^{j-1} : k \in \mathbb{Z}\}$ for some fixed resolution j, we seek *simple* expression for $\{c_k^j\}$ and $\{d_k^j\}$.

PROPOSITION 4.1. *In the MRA framework,*

$$\phi_k^j(x) = \sum_m h_{m-2k}\phi_m^{j-1}(x) \tag{105}$$

and

$$\psi_k^j(x) = \sum_m g_{m-2k}\phi_m^{j-1}(x). \tag{106}$$

PROOF. Recall that $\phi(x) = \sqrt{2}\sum_n h_n\phi(2x-n)$ and $\phi_k^j(x) = 2^{-j/2}\phi(2^{-j}x-k)$. Thus,

$$\phi(2^{-j}y-k) = \sqrt{2}\sum_n h_n\phi(2^{-j+1}y-2k-n).$$

Let $m = 2k+n$, we have

$$\phi(2^{-j}y-k) = \sqrt{2}\sum_m h_{m-2k}\phi(2^{-(j-1)}y-m),$$

which gives the claimed expression

$$\phi_k^j(y) = 2^{-j/2}\phi(2^{-j}y-k) = \sum_m h_{m-2k}2^{-(j-1)/2}\phi(2^{-(j-1)}y-m) = \sum_m h_{m-2k}\phi_m^{j-1}(y).$$

Equation (106) can be shown similarly. □

To determine a recursion relation for the scaling coefficients, first write

$$c_k^j = \langle f, \phi_k^j\rangle.$$

By Equation (105) in Proposition 4.1, this becomes

$$\begin{aligned} c_k^j &= \langle f, \sum_m h_{m-2k}\phi_m^{j-1}\rangle \\ &= \sum_m h_{m-2k}\langle f, \phi_m^{j-1}(x)\rangle \\ &= \sum_m h_{m-2k}c_m^{j-1}. \end{aligned}$$

Similarly, we can get a recursion relation for the wavelet coefficients, d_k^j with Equation (106):

$$\begin{aligned} d_k^j &= \langle f, \psi_k^j\rangle \\ &= \langle f, \sum_m g_{m-2k}\phi_m^{j-1}\rangle \\ &= \sum_m g_{m-2k}\langle f, \phi_m^{j-1}(x)\rangle \\ &= \sum_m g_{m-2k}c_m^{j-1}. \end{aligned}$$

EXAMPLE 4.2. We can now use these recursion formulas to get a simplified formulas for the Haar scaling and wavelet coefficients. With $h_0 = h_1 = 1/\sqrt{2}$, $g_0 = 1/\sqrt{2}$, and $g_1 = -1/\sqrt{2}$, the recursion formula for the Haar scaling and wavelet functions become

$$c_k^j = \frac{c_{2k}^{j-1} + c_{2k+1}^{j-1}}{\sqrt{2}} \tag{107}$$

and

$$d_k^j = \frac{c_{2k}^{j-1} - c_{2k+1}^{j-1}}{\sqrt{2}}. \tag{108}$$

Notice that the coefficients of c_k^j at a given level j are seen to be *smoothed* versions of the coefficients at the higher resolution level $j-1$; while the coefficients of d_k^j at a given level j are produced by differencing the scaling coefficients which creates a *sharpened* versions of the coefficients at the higher resolution level $j-1$.

The Pyramidal Reconstruction (coarser to finer)

Now we derive the recursion relations going the other way. Oppose to the recursion relation for the decomposition, illustrated in Figure 8(a), we start with the function at its coarsest level and add on the detail from each of the wavelet subspaces, illustrated in Figure 8(b). Let $f^j(x) = P_j f(x)$ and $s^j(x) = Q_j f(x)$. At each level we have

$$P_{j-1} f(x) = P_j f(x) + Q_j f(x),$$

or as functions,

$$f^{j-1}(x) = f^j(x) + s^j(x) = \sum_k c_k^j \phi_k^j(x) + \sum_k d_k^j \psi_k^j(x),$$

where $f^j \in V_j$ and $s^j \in W_j$. This decomposition is represented in terms of the subspaces in Figure 8(a). The scaling coefficients at the $(j-1)^{\text{th}}$ level is

$$\begin{aligned} c_n^{j-1} &= \langle f^{j-1}, \phi_n^{j-1} \rangle \\ &= \langle \sum_k c_k^j \phi_k^j(x) + \sum_k d_k^j \psi_k^j(x), \phi_n^{j-1} \rangle \\ &= \sum_k c_k^j \langle \phi_k^j, \phi_n^{j-1} \rangle + \sum_k d_k^j \langle \psi_k^j, \phi_n^{j-1} \rangle, \end{aligned}$$

where

$$\begin{aligned} \langle \phi_k^j, \phi_n^{j-1} \rangle &= \langle \sum_m h_{m-2k} \phi_m^{j-1}, \phi_n^{j-1} \rangle \\ &= \sum_m h_{m-2k} \langle \phi_m^{j-1}, \phi_n^{j-1} \\ &= h_{n-2k} \end{aligned}$$

and

$$\begin{aligned} \langle \psi_k^j, \phi_n^{j-1} \rangle &= \langle \sum_m g_{m-2k} \phi_m^{j-1}, \phi_n^{j-1} \rangle \\ &= \sum_m g_{m-2k} \langle \phi_m^{j-1}, \phi_n^{j-1} \\ &= g_{n-2k}. \end{aligned}$$

Thus, we have the general reconstruction formula

$$c_n^{j-1} = \sum_k h_{n-2k} c_k^j + \sum_k g_{n-2k} d_k^j. \tag{109}$$

In the case of the Haar wavelet, the reconstruction formulas are

$$c_{2k}^{j-1} = \frac{\sqrt{2}}{2} \left(c_k^j + d_k^j \right) \tag{110}$$

and

$$c_{2k+1}^{j-1} = \frac{\sqrt{2}}{2} \left(c_k^j - d_k^j \right). \tag{111}$$

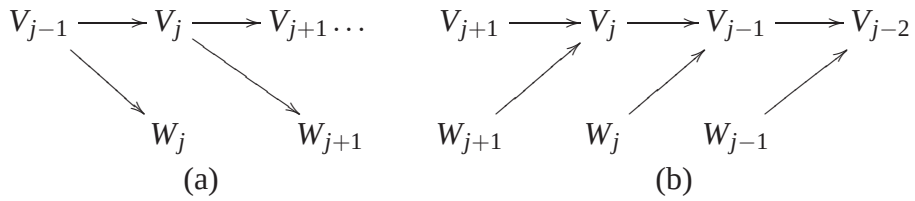


FIGURE 8. (a) The relationship of the wavelet and scaling subspaces in the pyramidal decomposition. Only the data represented in the finest-resolution scaling subspace is required. (b) The relationship of the wavelet and scaling subspaces in the pyramidal reconstruction. Only the data represented in the coarsest-resolution scaling subspace is required, in addition to all the wavelet-subspace projections, for perfect reconstruction.

EXAMPLE 4.3. *Haar Multiresolution Analysis*. We are now in a position to compute a wavelet decomposition of a function (or vector). We take as an example the decomposition of the vector $f = [9\,1\,2\,0]$. Viewing f as a function, we can write

$$f(x) = \begin{cases} 9 & \text{if } x \in \left[0, \frac{1}{4}\right), \\ 1 & \text{if } x \in \left[\frac{1}{4}, \frac{1}{2}\right), \\ 2 & \text{if } x \in \left[\frac{1}{2}, \frac{3}{4}\right), \\ 0 & \text{if } x \in \left[\frac{3}{4}, 1\right). \end{cases}$$

We need to arbitrarily specify the size of the smallest scale to start the pyramidal decomposition algorithm. We will choose $\dim f = 4 = 2^{-j}$. Thus the finest resolution required is at the level V_{-2}. We shall see that the decomposition in this case will involve the subspaces

$$\begin{aligned} V_{-2} &= \text{span}\{2\phi(4x-k) : k \in \mathbb{Z}\}, \\ V_{-1} &= \text{span}\{\sqrt{2}\phi(2x-k) : k \in \mathbb{Z}\}, \\ V_0 &= \text{span}\{\phi(x-k) : k \in \mathbb{Z}\}, \end{aligned}$$

where the coefficients normalize the functions so that they are orthonormal.
At the finest resolution, $j = -2$, the scaling coefficients $\{c_k^j\}$ are found by projection f onto the basis for V_{-2}. Thus, since $P_{-2}f \in V_{-2}$, we write

$$P_{-2}f(x) = \sum_{k\in\mathbb{Z}} c_k^{-2}\phi_k^{-2}(x),$$

where

$$\begin{aligned} c_k^{-2} &= \langle f, \phi_k^{-2}\rangle \\ &= \int_{-\infty}^{\infty} f \cdot 2^{2/2}\phi(2^2x-k)\,dx \\ &= 2\int_{\infty}^{\infty} f\phi(4x-k)\,dx \\ &= 2\int_{k/4}^{(k+1)/4} f(x)\,dx. \end{aligned}$$

Evaluating the integrals for appropriate values of k, we have

$$c_0^{-2} = \frac{9}{2}, \quad c_1^{-2} = \frac{1}{2}, \quad c_2^{-2} = 1, \quad c_3^{-2} = 0, \quad c_k^{-2} = 0 \quad \forall k > 3 \quad \text{and} \quad k < 0.$$

Now project f onto V_{-1} via $P_{-1}f(x)$ and onto W_{-1} via $Q_{-1}f(x)$. We have

$$P_{-1}f(x)=\sum_k c_k^{-1}\phi_k^{-1}(x) \quad \text{and} \quad Q_{-1}f(x)=\sum_k d_k^{-1}\psi_k^{-1}(x)$$

where, using the recursion formulas,

$$c_k^{-1}=\frac{c_k^{-2}+c_{2k+1}^{-2}}{\sqrt{2}} \quad \text{and} \quad d_k^{-1}=\frac{c_k^{-2}-c_{2k+1}^{-2}}{\sqrt{2}},$$

we get

$$\begin{aligned}
c_0^{-1} &= \frac{1}{\sqrt{2}}\left(c_0^{-2}+c_1^{-2}\right)=\frac{9/2+1/2}{\sqrt{2}}=\frac{5}{\sqrt{2}},\\
c_1^{-1} &= \frac{1}{\sqrt{2}}\left(c_2^{-2}+c_3^{-2}\right)=\frac{1+0}{\sqrt{2}}=\frac{1}{\sqrt{2}},\\
d_0^{-1} &= \frac{1}{\sqrt{2}}\left(c_0^{-2}-c_1^{-2}\right)=\frac{9/2-1/2}{\sqrt{2}}=\frac{4}{\sqrt{2}},\\
d_1^{-1} &= \frac{1}{\sqrt{2}}\left(c_2^{-2}-c_3^{-2}\right)=\frac{1-0}{\sqrt{2}}=\frac{1}{\sqrt{2}}.
\end{aligned}$$

Therefore,

$$P_{-1}f(x)=c_0^{-1}\phi_0^{-1}(x)+c_1^{-1}\phi_1^{-1}(x)=5\phi(2x)+\phi(2x-1)$$

and

$$Q_{-1}f(x)=d_0^{-1}\psi_0^{-1}(x)+d_1^{-1}\psi_1^{-1}(x)=\frac{4}{\sqrt{2}}\psi_0^{-1}+\frac{1}{\sqrt{2}}\psi_1^{-1},$$

which can then be rewritten (using the Haar relations) as

$$\begin{aligned}
Q_{-1}f(x) &= \frac{4}{\sqrt{2}}\left[\frac{\phi_0^{-2}-\phi_1^{-2}}{\sqrt{2}}\right]+\frac{1}{\sqrt{2}}\left[\frac{\phi_2^{-2}-\phi_3^{-2}}{\sqrt{2}}\right]\\
&= \frac{4}{\sqrt{2}}\left[\frac{2}{\sqrt{2}}\left(\phi(4x)-\phi(4x-1)\right)\right]+\frac{1}{\sqrt{2}}\cdot\frac{2}{\sqrt{2}}\left(\phi(4x-2)-\phi(4x-3)\right)\\
&= 4\phi(4x)-4\phi(4x-1)+\phi(4x-2)-\phi(4x-3)\in W_{-1}.
\end{aligned}$$

The projection onto V_0 proceeds similarly via the computation of $P_0f(x)$. The scaling coefficients at the level $j=0$ are found using

$$c_k^0=\frac{c_{2k}^{-1}+c_{2k+1}^{-1}}{\sqrt{2}}.$$

The single nonzero scaling coefficient is given by

$$c_0^0=\frac{\frac{5}{\sqrt{2}}+\frac{1}{\sqrt{2}}}{\sqrt{2}}=3$$

and the associated wavelet coefficient is

$$d_0^0=\frac{\frac{5}{\sqrt{2}}-\frac{1}{\sqrt{2}}}{\sqrt{2}}=2.$$

So, we have

$$P_0f(x)=c_0^0\phi_0^0(x)=3\phi(x)\in V_0$$

and

$$\begin{aligned} Q_0 f(x) &= d_0^0 \psi_0^0(x) \\ &= 2\left[\frac{1}{\sqrt{2}}\left(\phi_0^{-1} - \phi_1^{-1}\right)\right] \\ &= \frac{2}{\sqrt{2}}\left(\sqrt{2}\phi(2x) - \sqrt{2}\phi(2x-1)\right) \\ &= 2\phi(2x) - 2\phi(2x-1) \in W_0. \end{aligned}$$

In summary,

$$\begin{aligned} f(x) &= P_{-2} f(x) \in V_{-2} \\ &= P_{-1} f(x) + Q_{-1} f(x) \in V_{-1} \oplus W_{-1} \\ &= P_0 f(x) + Q_0 f(x) + Q_{-1} f(x) \in V_0 \oplus W_0 \oplus W_{-1}. \end{aligned}$$

4.3. Discrete Wavelet Transform (DWT). Like the Fourier series expansion, the wavelet series expansion of the previous sections maps a function of a continuous variable into a sequence of coefficients. If the function being expanded is discrete (i.e., a sequence of numbers), the resulting coefficients are called *discrete wavelet transform (DWT)*. For example, if $f(n) = f(x_0 + n\Delta x)$ for some x_0, Δx, and $n = 0, 1, 2, \ldots, M-1$, then the wavelet series expansion coefficients for $f(x)$ become the *forward* DWT coefficients for sequence $f(n)$:

$$c_k^j = \frac{1}{\sqrt{M}} \sum_n f(n) \phi_k^i(n), \tag{112}$$

$$d_k^j = \frac{1}{\sqrt{M}} \sum_n f(n) \psi_k^j(n) \tag{113}$$

and

$$f^j(n) = \frac{1}{\sqrt{M}} \left(\sum_k c_k^{j+1} \phi_k^{j+1}(n) + \sum_k d_k^{j+1} \right) \tag{114}$$

EXAMPLE 4.4. To illustrate the use of Equations (112) through (114), consider the discrete function of eight points: $f(0) = 448$, $f(1) = 768$, $f(2) = 704$, $f(3) = 640$, $f(4) = 1280$, $f(5) = 1408$, $f(6) = 1600$, $f(7) = 1600$. Because $M = 8 = 2^{-j}$, there will be 3 ($J = -3$) steps (levels) of MRA with $j = -3, -2, -1$, and 0. We will use the Haar scaling and wavelet functions and assume that the eight samples of $f(x)$ are distributed over the support of the basis functions, which is 1 in width. Notice that at the finest level, $c_k^{-3} = \frac{1}{\sqrt{8}} \sum_n f(n) \sqrt{8} \phi(8n-k) = f(n)$. At the next level, we use the recursion formulas:

$$c_k^{-2} = \frac{c_{2k}^{-3} + c_{2k+1}^{-3}}{\sqrt{2}} \quad \text{and} \quad d_k^{-2} = \frac{c_{2k}^{-3} - c_{2k+1}^{-3}}{\sqrt{2}}.$$

Thus, in matrix notations, we get the system

$$\overbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ & & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ & & & & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ & & & & & & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ & & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ & & & & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ & & & & & & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}}^{\mathbf{H_1}} \overbrace{\begin{bmatrix} 448 \\ 768 \\ 704 \\ 640 \\ 1280 \\ 1408 \\ 1600 \\ 1600 \end{bmatrix}}^{f} = \overbrace{\frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{608} \\ \mathbf{672} \\ \mathbf{1344} \\ \mathbf{1600} \\ -160 \\ 32 \\ -64 \\ 0 \end{bmatrix}}^{y_1} = \begin{bmatrix} \mathbf{y_1}|_{\mathbf{V_{-2}}} \\ y_1|_{W_{-2}} \end{bmatrix}$$

The action of left multiplying the signal f by the Haar matrix $\mathbf{H}_1$ produces the vector y_1 that consists of the *approximation* coefficients in V_{-2} in its upper half and the *detail* coefficients in W_{-2} in its lower half. To get the approximation and detail at the next level ($j=-1$), we consider only the components of y_1 in V_{-2}, i.e.,

$$\overbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}}^{\mathbf{H_2}} \overbrace{\begin{bmatrix} 608 \\ 672 \\ 1344 \\ 1600 \end{bmatrix}}^{y_1|_{V_{-2}}} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{640} \\ \mathbf{1472} \\ -32 \\ -128 \end{bmatrix} = \begin{bmatrix} \mathbf{y_2}|_{\mathbf{V_{-1}}} \\ y_2|_{W_{-1}} \end{bmatrix}$$

with

$$y_2 = [y_2|_{V_{-1}} \quad y_2|_{W_{-1}} \quad y_1|_{W_{-2}}]^T = [\mathbf{640} \quad \mathbf{1472} \quad -32 \quad -128 \quad -160 \quad 32 \quad -64 \quad 0]^T.$$

Lastly, to obtain the approximation and detail components at the bottom level $j=0$, we consider only the components of y_2 in V_{-1}, i.e.,

$$\overbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}}^{\mathbf{H_3}} \begin{bmatrix} 640 \\ 1472 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{1056} \\ -416 \end{bmatrix} = \begin{bmatrix} \mathbf{y_3}|_{\mathbf{V_0}} \\ y_3|_{W_0} \end{bmatrix}$$

with

$$y_3 = [y_3|_{V_0} \quad y_3|_{W_0} \quad y_2|_{W_{-1}} \quad y_1|_{W_{-2}}]^T = \frac{1}{\sqrt{2}}[\mathbf{1056} \quad -416 \quad -32 \quad -128 \quad -160 \quad 32 \quad -64 \quad 0]^T.$$

Notice that this process is entirely reversible since the **H**'s are invertible. A simply compression method takes advantage of this fact. For example, a threshold compression method eliminates all entries below a threshold value and retains all entries above it. In this example, set $-64/\sqrt{2}$ and $-160/\sqrt{2}$ in y_1 to 0 to obtain $\tilde{y}_1 = \frac{1}{\sqrt{2}}[608, 672, 1344, 1600, 0, 32, 0, 0]^T$, then a compressed version of f is given by $\tilde{f} = \mathbf{H}_1^{-1}\tilde{y}_1$.

5. Wavelet Transforms in Two Dimension

The one-dimensional transforms of the previous sections are easily extended to two-dimensional functions like images. In two dimensions, a two-dimensional scaling function, $\phi(x,y)$, and three two-dimensional wavelets, $\psi^H(x,y)$, $\psi^V(x,y)$, and $\psi^D(x,y)$ are required. Each is the product of two

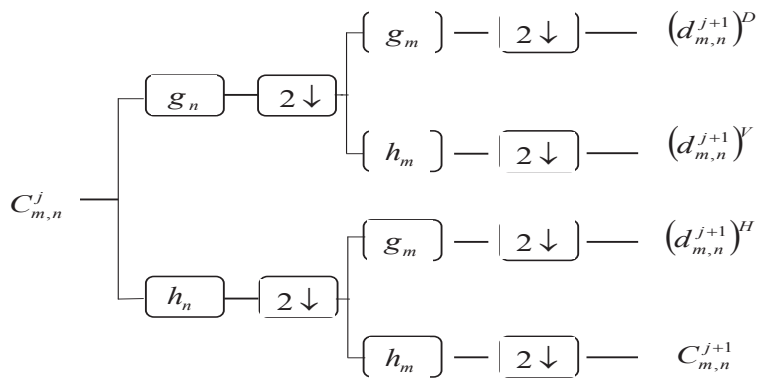


FIGURE 9. The analysis filter bank.

one-dimensional functions. Excluding products that produce one-dimensional results, like $\phi(x)\psi(x)$, the four remaining products produce the *separable* scaling function

$$\phi(x,y) = \phi(x)\phi(y) \tag{115}$$

and separable, "directionally sensitive" wavelets

$$\psi^H(x,y) = \psi(x)\phi(y), \tag{116}$$

$$\psi^V(x,y) = \phi(x)\psi(y), \tag{117}$$

and

$$\psi^D(x,y) = \psi(x)\psi(y). \tag{118}$$

These wavelets measure functional variations — intensity variations for images — along different directions: ψ^H measures variations along columns (for example, horizontal edges), ψ^V responds to variations along rows (like vertical edges), and ψ^D corresponds to variations along diagonals. The directional sensitivity is a natural consequence of the separability in Equations (116) to (118); it does not increase the computational complexity of the 2-D transform discussed in this section.

Given separable two-dimensional scaling and wavelet functions, extension of the 1-D DWT to two dimensions is straightforward. We first define the scaled and translated basis functions:

$$\begin{aligned} \phi^j_{m,n}(x,y) &= 2^{-j/2}\phi(2^{-j}x-m, 2^{-j}y-n) \\ \left(\psi^j_{m,n}(x,y)\right)^i &= 2^{-j/2}\psi^i(2^{-j}x-m, 2^{-j}y-n), \quad i=\{H,V,D\}. \end{aligned}$$

Rather than an exponent, i is a superscript that assumes the values H, V, and D. The discrete wavelet transform of image $f(x,y)$ of size $M \times N$ is then

$$\begin{aligned} c^j_{m,n} &= \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\phi^j_{m,n}(x,y) \\ \left(d^j_{m,n}\right)^i &= \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\left(\psi^j_{m,n}(x,y)\right)^i, \quad i=\{H,V,D\}. \end{aligned}$$

Like the 1-D discrete wavelet transform, the 2-D DWT can be implemented using digital filters and downsamplers. With separable two-dimensional scaling and wavelet functions, we simply take the 1-D FWT of the rows of $f(x,y)$, followed by the 1-D FWT of the resulting columns. Figure 9 shows the process in block diagram form.

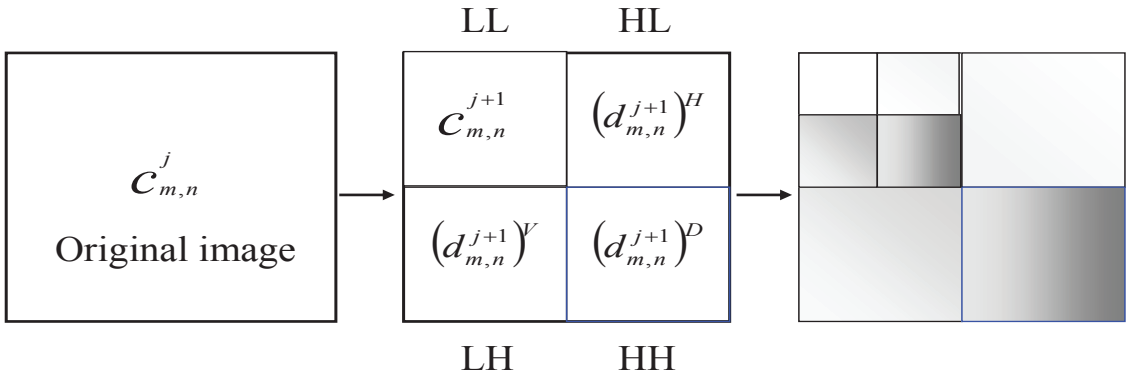


FIGURE 10. The 2-dimensional wavelet transform takes an image on the left, and applies four combinations of smoothing and differencing to obtain the transform in the middle. At the next level, we perform the same operations to only the upper left sub-image as shown in the right.

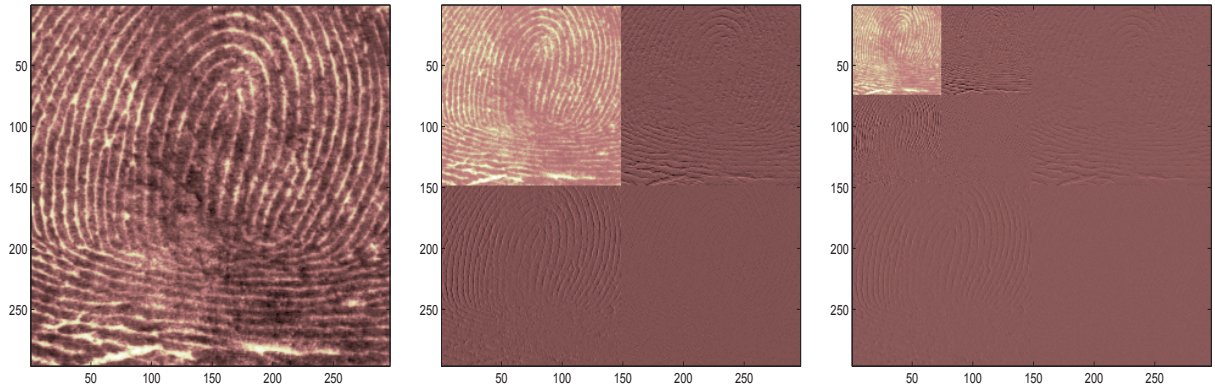


FIGURE 11. (a) Original fingerprint. (b) The resulted 4 sub-images after a single level of discrete wavelet transform. (c) Result after 2 levels of DWT, corresponding to the right panel in Figure 10.

In general, the 2-dimensional wavelet transform proceeds by creating four sub-images of the original image. The image in quadrant II is the result of reducing the resolution of the columns and rows of the original image. This is achieved by applying the 1-dimensional scaling transform (thus, lowpass filter) to both the columns and rows; for short we refer to these operations as L column, L row. The image in quadrant I is obtained by applying the scaling transform to the rows and wavelet transform to the columns of the original image, i.e., H column and L row. Quadrant III is obtained via L column and H row while quadrant IV is obtained via H column and H row. These operations are summarized in Figure 10. An example is shown in Figure 11.

The image in quadrant II may be used to calculate the next level of the decomposition. The result is to divide this quadrant into four new quadrants. This basic idea may be applied to the upper left corner image as many times as deemed useful, illustrated in the right panel of Figure 10.

The two-dimensional wavelet transform has found a number of important applications, including an FBI fingerprint identification system as well as JPEG 2000, a standard for image storage and retrieval.

CHAPTER 8

Suggested Exercises

1. Set One

1.1. Theory.

(1) Let the basis $\mathscr{B}_1$ be the standard basis, i.e., $\mathbf{e}^{(1)} = (1\,0)^T$, $\mathbf{e}^{(2)} = (0\,1)^T$, and the basis $\mathscr{B}_2$ be given by the two vectors $\mathbf{v}^{(1)} = (1\,1)^T$, $\mathbf{v}^{(2)} = (-1\,1)^T$. Given $\mathbf{u}_{\mathscr{B}_1} = (1\,1)^T$, find $\mathbf{u}_{\mathscr{B}_2}$.

1.2. Computing.

(1) Write a code to generate 1000 random numbers contained on the unit circle. Apply several random matrices to this data and describe your results in the terminology of bases and change of bases. How do your results differ if the multiplying matrix is constrained to be orthogonal?

(2) Given an algorithm [**35**] for computing small principal angles between two subspaces given by the real matrices X and Y, where X is in $\mathbb{R}^{n\times p}$ and Y is in $\mathbb{R}^{n\times q}$ (Principal angles are defined to be between 0 and $\pi/2$ and listed in ascending order):

Input: matrices X (n-by-p) and Y (n-by-q).
Output: principal angles θ between subspaces $\mathscr{R}(X) = \mathscr{X}$ and $\mathscr{R}(Y) = \mathscr{Y}$.

(a) Find orthonormal bases Q_x and Q_y for $\mathscr{X}$ and $\mathscr{Y}$ such that

$$Q_x^T Q_x = Q_y^T Q_y = I \quad \text{and} \quad \mathscr{R}(Q_x) = \mathscr{X}, \mathscr{R}(Q_y) = \mathscr{Y}.$$

(b) Compute SVD for cosine: $Q_x^T Q_y = H\Sigma Z^T$, where $\Sigma = diag(\sigma_1, \ldots, \sigma_q)$.

(c) Compute matrix

$$Y = \begin{cases} Q_y - Q_x(Q_x^T Q_y) & \text{if } rank(Q_x) \geq rank(Q_y); \\ Q_x - Q_y(Q_y^T Q_x) & \text{otherwise.} \end{cases}$$

(d) SVD for sine: $[H, diag(\mu_1, \ldots, \mu_q), Z] = \text{svd}(Y)$.

(e) Compute the principal angles, for $k = 1, \ldots, q$

$$\theta_k = \begin{cases} \arccos(\sigma_k) & \text{if} \quad \sigma_k^2 < \frac{1}{2}; \\ \arcsin(\mu_k) & \text{if} \quad \mu_k^2 \leq \frac{1}{2}. \end{cases}$$

(a) Implement this algorithm in MATLAB under the function name prinAngles. Your function should have the input and output arguments:

[theta] = prinAngles(X1,X2)

where theta is the principal angles listed from the smallest to the largest, X1 is the first set of images listed by the columns, and X2 is the second set of images listed the columns as well.

(b) To verify that your implementation is correct, download face1.mat and face2.mat from the course website where face1.mat contains 21 distinct images of person 1 in its columns and face2.mat contains 21 distinct images of person 2 in its columns and test your implementation with this data. Note: in case you want to see what the images look like, the

images are of resolution 160×138. The following MATLAB commands will display the first image of person 1:

```
>> load face1
>> imagesc(reshape(face1(:,1),160,138)), colormap(
       gray), axis off
```

(3) Write a function in MATLAB using the homogeneous coordinates that scales (enlarge and shrink) a 2D image about a point P = [tx, ty, 1]'. Specifically, the first line of your function will be (other than the comments)

```
function [newImg] = scale(Img, alpha, P)
```

where alpha = [sx, sy]' is the scale parameter that controls how much scaling is applied in x-direction and in y-direction, respectively. Make sure your routine works by applying it to an image of your choice. The MATLAB commands that are useful here:

```
>> meshgrid
>> interp2
>> imread
>> imagesc
>> reshape
```

For example, to find out how to use meshgrid, type

```
>> Help meshgrid
```

in the MATLAB command prompt.

(4) Writ a routine in MATLAB using the homogeneous coordinates that translates a 2D image horizontally and a routine that translates the image vertically. Specifically, one of your functions should have the input and output arguments

[newImg] = translateH(Img, tx)

where tx is the amount of horizontal translation applied (make sure it works for both positive and negative values). Test your routine by applying it to an image of your choice.

(5) Write a routine in MATLAB using the homogeneous coordinates that rotates a 2D image about a point P = [tx, ty, 1]'. Specifically, your function should have the input and output arguments

[newImg] = rotate[Img, theta, P]

where theta is the amount of rotation applied in counterclockwise orientation. Make sure your routine works by applying it to an image of your choice.

2. Set Two

2.1. Theory.

(1) Let W_1 and W_2 be vector subspaces and $W = W_1 + W_2$. Show, by giving an example, that the decomposition of a vector $\mathbf{x} \in W$ is not unique, i.e.,

$$\mathbf{x} = \mathbf{w}_1 + \mathbf{w}_2 = \mathbf{w}_1^{'} + \mathbf{w}_2^{'},$$

where $\mathbf{w}_1 \neq \mathbf{w}_1^{'}$, $\mathbf{w}_2 \neq \mathbf{w}_2{}'$, $\mathbf{w}_1, \mathbf{w}_1^{'} \in W_1$, $\mathbf{w}_2, \mathbf{w}_2^{'} \in W_2$.

(2) Consider the matrix

$$A = \begin{pmatrix} 1 & -1 \\ 2 & -2 \\ 3 & -3 \end{pmatrix}.$$

Determine bases for the column space, row space, null space, and left null space of A.

(3) Let $V = \mathbb{R}^3$, let

$$\mathbf{u}^{(1)} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \quad \mathbf{u}^{(2)} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \quad , \mathbf{x} = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix},$$

and define $W_1 = \text{span}\,(\mathbf{u}^{(1)}, \mathbf{u}^{(2)})$. Find the orthogonal projection of $\mathbf{x}$ onto W_1. Also find the projection matrix $\mathbb{P}$ associated with this mapping.

(4) Reconsider Problem 3. Find vectors such that $x = UU^T x$ and $x \neq UU^T x$ where the matrix U consists of the basis vectors from Problem 3. Draw a picture to show the set of vectors for which UU^T acts as the identity.

(5) Determine the SVD of the data matrix

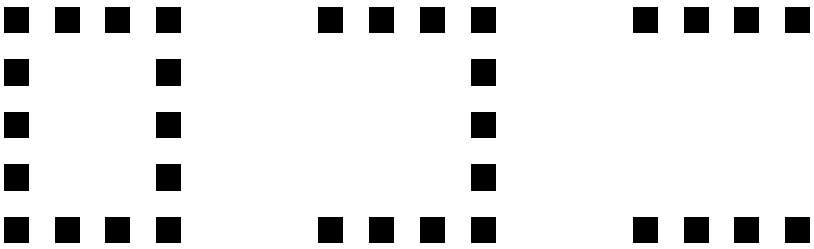
$$\begin{pmatrix} -2 & -1 & 1 \\ 0 & -1 & 0 \\ -1 & 1 & 2 \\ 1 & -1 & 1 \end{pmatrix},$$

and compute the rank-one, -two, and -three approximations to A.

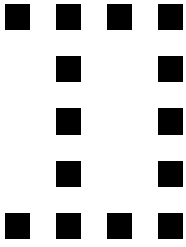
(6) Propose a method to compute a random orthogonal transformation.

2.2. Computing.

(1) Consider the training set consisting of the following three patterns consisting of 5×4 arrays of black squares

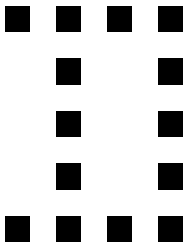


Using Kohonen's novelty filter, find the novelty in the pattern



Proceed by assuming that the black square entries have numerical value one and the blank entries have numerical value zero. Concatenate the columns of each pattern to make vectors in $\mathbb{R}^{20}$.

(2) Compute the SVD of the matrix A whose entries come from the pattern



and display the reconstructions A_1, A_2, A_3, A_4. Again, treat the squares as ones and the blanks as zeros. Your reconstructions should be matrices with numerical values. Interpret your results.

(3) This assignment requires the use of a MATLAB image. Choose your favorite image for this exercise (be sure to choose an image whose resolution is at least 300-by-300.). All the necessary MATLAB syntax is described as follows. To begin, load the MATLAB image into the matrix A using

```
>> A = imread('myImage.tif');
% adjust accordingly with the image extension
```

If your image is in color, A will have three dimensions where the first two give the resolution of the image and the last one contains a layer of red, a layer of green, and a layer of blue. To turn a color image into a monochrome one, use the MATLAB command

```
>> B = rgb2gray(A);
```

You don't necessarily have to work with a black and white picture, but it is definitely easier and computationally cheaper to start with one. The full and reduced SVD may then be executed simply by

```
>> A = double(A); % data matrix has to be in double
        precision in order to perform mathematics on
         it
>> [U,S,V] = svd(A);
>> [U_thin,S_thin,V_thin] = svd(A,0);
%% the zero (not the letter "o") indicates economy
        size
%% 'double' converts any 8-bit single (uint8) into
        16-bit double precision
```

where U (resp. U_thin), S (resp. S_thin), and V (resp. V_thin) are the left-singular vectors, the singular values, and the right-singular vectors (resp. reduced). A rank-k approximation of the image may be found via

```
>> A_k = U(:,1:k)*S(1:k,1:k)*V(:,1:k)';
```

The resulting image may be displayed using

```
>> imagesc(A_k);
% may use the option: axis off, axis square for
        better display
```

or

```
>> image(A_k);
```

For better viewing, one can reset the colormap to gray scales with the command

```
>> imagesc(A_k), colormap(gray)
```

Now, do the following:

(a) Plot the singular value distribution of your image, where the *x*-axis is the counting index while the *y*-axis is the magnitude of the singular values. If we define the **cumulative energy** of $A \in \mathbb{R}^{m\times n}$ with rank r to be

$$E_k = \frac{\sum_{i=1}^{k}\sigma_i^2}{\sum_{i=1}^{r}\sigma_i^2}, \text{ where } k \le r,$$

identify the number of singular values (σ_is) needed to retain at least 95% of the energy. This number is often called *numerical rank* of the matrix A. Is the numerical rank for your image large or small? Explain why.

(b) Compute the rank-10, rank-50, rank-100, and rank-200 approximations to your chosen image along with the *relative errors* of approximation (use the title, xlabel, and ylabel commands to specify appropriate information), display them on the same figure (using subplot), and interpret your results. (Recall that the *absolute* error of a rank-k approximation is measured by the $k+1^{\text{th}}$ singular value, so the *relative* error is given by σ_{k+1}/σ_1.)

3. Set Three

3.1. Theory.

(1) Show that

$$\nabla_{\mathbf{v}}(\mathbf{v},\mathbf{v}) = 2\mathbf{v}$$

and that if C is a symmetric matrix, then

$$\nabla_{\mathbf{v}}(\mathbf{v},C\mathbf{v}) = 2C\mathbf{v}.$$

(2) Show that

$$(\phi^{(1)},C\phi^{(2)}) = (C\phi^{(1)},\phi^{(2)}).$$

Assume C is symmetric.

(3) Does periodic data imply that the ensemble average covariance matrix C will have eigenvalues with multiplicity greater than 1? If C has eigenvalues of multiplicity greater than 1, is the data necessarily periodic?

(4) Given the data matrix

$$X = \begin{pmatrix} -2 & -1 & 1 \\ 0 & -1 & 0 \\ -1 & 1 & 2 \\ 1 & -1 & 1 \end{pmatrix},$$

compute the eigenvalues and eigenvectors of XX^T and X^TX. For $\mathbf{u}^{(1)}$, confirm the statement

$$\mathbf{u}^{(j)} = \frac{1}{\sigma_j}\sum_{k=1}^{P} v_k^{(j)}\mathbf{x}^{(k)},$$

where $j = 1,\ldots,rankX$.

(5) It was shown that the expansion coefficients may be computed using formula

$$A = \Sigma V^T,$$

providing an alternative to the direct computation via

$$A = U^TX.$$

Compute the number of add/multiplies required to compute A via both formulas, using the data matrix and eigenvectors given in the previous Problem. Which way is computationally cheaper in general? Why?

3.2. Computing.

(1) The object of this programming assignment is to write a code to apply the *snapshot* method to a collection of P high-resolution image files. Your program should compute (and order) the eigenpictures. It should also have a subroutine to determine the projection of a given picture onto the best D-element subspace (D is typically chosen empirically). Your program report should include the following information:
   (a) A display of the ensemble-average image.
   (b) A picture of a mean-subtracted image, for one of the images chosen at random from the ensemble.
   (c) A collection of eigenpictures (based on mean-subtracted data) for a broad range of eigenvalues. The eigenpictures must be mapped to integers on the interval $[0,255]$.
   (d) Partial reconstructions of a selected image for various value of D. Include the reconstruction error $||\mathbf{x}-\mathbf{x}_D||$ in each case and confirm that you obtain perfect reconstruction when D is equal to the rank of the data matrix.
   (e) A graph of $\lambda_i/\lambda_{\max}$ vs i, where λ_i is the i^{th} eigenvalue of the mean-subtracted, ensemble-averaged covariance matrix. How does this plot help you determining the best D value to use?
   (f) Now, devise (describe) a classification algorithm that uses this idea of best basis to classify a probe (testing) data against a given gallery. (For your reference: this process is called *Principal Component Analysis*.) Why is this more efficient than classifying data points in their resolution dimension?

   The data for this problem may be downloaded from the course website. The data file faces1.mat contains 109 images whose dimensions are 120×160. It is a single matrix, where each column has length $19,200$, which is 120×160. The format of the data is "uint8", which stands for unsigned integer, 8 bits. Before you use the data for KL, change it to "double" format.

(2) Test your theory from 1(f) on the following data set: Digits.mat can be downloaded from the course website. It contains three variables: Gallery, Probe, and photo_size, where Gallery is a 1024×500 matrix with 50 digits of 0 in its first 50 columns, 50 digits of 9 in its last 50 columns, etc. The row dimension comes from the resolution of the images stored in photo_size. The variable Probe stores a set of novel digits from 0 to 9 that do not appear in the Gallery. Use *Principal Component Analysis* to classify the probe images against the gallery images. How well did the algorithm perform? Report and analyze your result.

4. Set Four

4.1. Theory.

(1) Consider the two eigenvector problems

$$C_{\mathbf{x}}\mathbf{u}=\lambda_x\mathbf{u}$$

and

$$C_{\mathbf{s}}\mathbf{v}=\lambda_s\mathbf{v}$$

where the matrices are related by $C_{\mathbf{x}}=C_{\mathbf{s}}+\alpha I$, where α is a real number and I is the usual identity matrix. Show that if $\mathbf{u}$ is an eigenvector of $C_{\mathbf{x}}$, then it is also an eigenvector of $C_{\mathbf{s}}$ associated with eigenvalue $\lambda_s=\lambda_x-\alpha$.

(2) Let A be a real $m \times n$. Show that the matrix M defined as

$$M = \alpha^2 I + AA^T$$

is nonsingular, where $I = I_m$ and α is a nonzero real number.

(3) Show that the between-class scatter matrix, S_B, in the multi-class *Fisher Discriminant Analysis* is given by

$$S_B = \sum_{i=1}^{M} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T,$$

where M is the total number of distinct classes, n_i is the number of data points in class i, $\mathbf{m}_i$ is the class mean of the i^{th} class, and $\mathbf{m}$ is the mean across all n data points. You may use the facts that

$$S_T = S_B + S_W, \quad S_W = \sum_{i=1}^{M} \sum_{x \in D_i} (x - \mathbf{m}_i)(x - \mathbf{m}_i)^T, \quad \text{and} \quad S_T = \sum_{i=1}^{n} (x_i - \mathbf{m})(x_i - \mathbf{m})^T.$$

4.2. Computing.

(1) This project concerns the application of the KL procedure for incomplete data [**18**]. Let the complete data set be translation- invariant:

$$f(x_m, t_\mu) = \frac{1}{N} \sum_{k=1}^{N} \frac{1}{k} \sin[k(x_m - t_\mu)],$$

where $m = 1, \ldots, M$, with M dimension of the ambient space (size of the spatial grid), and $\mu = 1, \ldots, P$, with P the number of points in the ensemble. Let $x_m = \dfrac{(m-1)2\pi}{M}$ and $t_\mu = \dfrac{(\mu - 1)2\pi}{P}$. Select an ensemble of masks $\left\{\mathbf{m}^{(\mu)}\right\}$, $\mu = 1, \ldots, P$, where 10% of the indices are selected to be zero for each mask. Each pattern in the incomplete ensemble may be written as

$$\tilde{\mathbf{x}}^{(\mu)} = \mathbf{m}^{(\mu)} . \mathbf{f}^{(\mu)},$$

where $\left(\mathbf{f}^{(\mu)}\right)_m = \dfrac{1}{N} \sum_{k=1}^{N} \dfrac{1}{k} \sin[k(x_m - t_\mu)]$. Let $P = M = 64$ and $N = 3$.

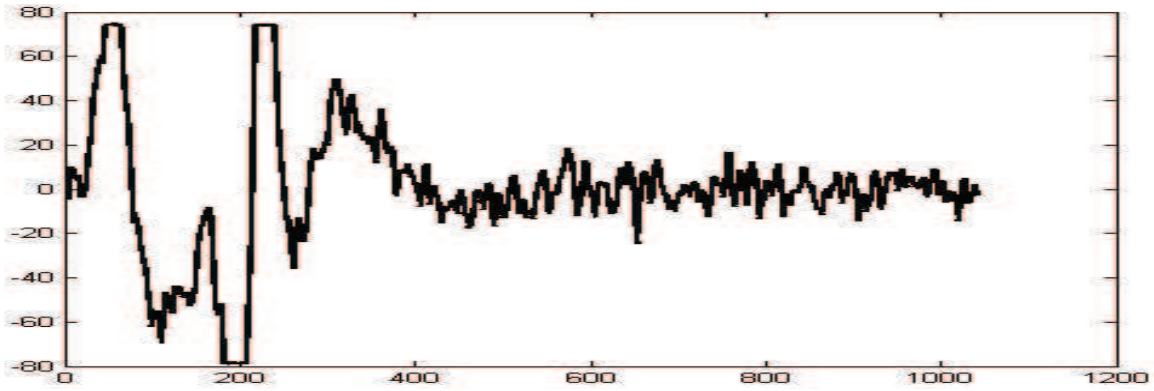
(a) Compute the eigenvectors of this ensemble using the gappy algorithm [**18**].
(b) Plot the eigenvalues as a function of the iteration, and continue until they converge.
(c) Plot your final eigenfunctions corresponding to the 10 largest eigenvalues.
(d) Plot the element $\tilde{\mathbf{x}}^{(1)}$ and the vector $\tilde{\mathbf{x}}_D$ repaired according to Equation

$$\tilde{\mathbf{x}} \approx \tilde{\mathbf{x}}_D = \sum_{n=1}^{D} \tilde{a}_n \phi^{(n)}. \tag{119}$$

Determine the value of D that provides the best approximation to the original non-gappy pattern vector.

(2) This project allows you to apply the two-class *Linear Discriminant Analysis (LDA)* on a simple EEG data. You will download the zipped file EEG_for_LDA from the course website. Once you unzip the archive, you will find 20 files whose file names follow the format "class-C_seq-T", where C stands for the task number (C = 2 and C = 3) and T stands for the trial number which ranges from 0 to 9. The participants were asked to count in task 2 and to perform visual rotation in task 3. The EEG data were collected in 19 channels with sampling at 256 Hz over 10 trials for each task. Upon loading the files, the variable "class_C_seq_T" is a 19-by-1040 matrix, where each row represents a reading from one of the 19 channels

(electrodes on the skull) and each column represents a reading at a single time stamp. A sample reading for task 2, trial 0 is shown below.



(a) Write a MATLAB routine to produce an optimal projection direction, w, using the two-class LDA criterion

$$w = \arg\max_w J(w) = \arg\max_w \frac{w^T S_B w}{w^T S_W w},$$

where

$$S_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \quad \text{and} \quad S_W = \sum_{i=1}^{2} \sum_{x \in D_i} (x - \mathbf{m}_i)(x - \mathbf{m}_i)^T$$

are the between-class scatter matrix and the within-class scatter matrix, respectively. That is, your code should take in a set of data points with a clear indication which points belong to class one and which points belong to class 2, and output a single vector w that is the solution of the generalized eigenvalue problem $S_B w = \lambda S_W w$. (If you are interested in the implementation of multi-class LDA, see [**4**] for more details on how to deal with the singularity of S_W.)

(b) Now, use your subroutine in part (a) to project the EEG data onto a real line. Particularly, we can form a data point in $\mathbb{R}^{1040\times 19}$ by concatenating the columns for each trial, therefore having 10 data points for task 0 and 10 data points for task 2. You would then project these 20 points onto the real line with the w found with part (a). Plot the projected data on the real line and distinguish the classes with different symbols. Do you see a clear separation? Analyze your results.

(3) Construct a 250×10 data set of your choice with correlated noise in the columns. (You may construct the noise by first constructing a data set of 250 points in $\mathbb{R}^3$ and map it to $\mathbb{R}^{10}$ via right multiplication by a random 3×10 matrix. This accomplishes the correlation aspect.) That is, the data matrix will contain $P = 10$ noisy signals, each of length $n = 250$ where each column has had mean removed. Apply the MNF method to filter the data. In particular, examine the effect of a D-mode reconstruction on a singe noisy signal for various values of D. Plot the the result of filtered data, noisy data, as well as the original den-noised data in the same graph to compare.

(4) Implement a 3×3 *median filter* and apply the filtering process on a corrupted image of "app-ndt-Chip-5.JPG" located via the course website. Specifically, corrupt "app-ndt-Chip-5.JPG" with *salt-and-pepper* noise, where the corrupted pixels are either set to the maximum value (which looks like snow in the image) or have single bits flipped over. In some cases, single pixels can be set alternatively to zero or to the maximum value (i.e., 255 on a 8-bit machine). Then apply the median filter to de-noise the corrupted image. Compare your result with the original.

(5) Given an image "CTimage.JPG" on the course website. Perform the following operations:
   (a) Construct a 3×3 average filter to smooth the image.
   (b) Then use a $2D$ Laplacian filter mask to extract the edges of the smoothed image.
   (c) Finally, enhance the smoothed image with the result from part (b). How does this image compare to the original?

5. Set Five

5.1. Theory.

(1) Find the Fourier series for the 2π-periodic square wave function

$$f(\omega) = \begin{cases} -k & \text{if} \quad -\pi < \omega < 0 \\ k & \text{if} \quad 0 < \omega < \pi \end{cases} \quad \text{and} \quad f(\omega + 2\pi) = f(\omega)$$

(2) Compute by hand the Haar wavelet decomposition (Pyramidal decomposition) of the vector $\mathbf{x}^T = [1, 7, -3, 2]$ by viewing it as

$$f(x) = \begin{cases} 1 & \text{if} x \in \left[0, \frac{1}{4}\right), \\ 7 & \text{if} x \in \left[\frac{1}{4}, \frac{1}{2}\right), \\ -3 & \text{if} x \in \left[\frac{1}{2}, \frac{3}{4}\right), \\ 2 & \text{if} x \in \left[\frac{3}{4}, 1\right). \end{cases}$$

Graphically show the projections onto the scaling and wavelet subspaces.

(3) Let $\mathbf{x} \in \mathbb{R}^M$, and let $\mathbf{y}$ be the 1D DWT of $\mathbf{x}$. If we write the Haar wavelet transform and its inverse as matrix operations, i.e.,

$$\mathbf{y} = W\mathbf{x}$$

and

$$\mathbf{x} = \tilde{W}\mathbf{y},$$

what are W and $\tilde{W}$? This should be done in terms of the Haar Pyramidal Decomposition algorithm, i.e., the expressions of W and $\tilde{W}$ depend on the level of the decomposition/reconstruction. If $\mathbf{x} = [576, 704, 1152, 1280, 1344, 1472, 1536, 1536]$, what is its Haar wavelet transform after 3 $(2^3 = 8)$ levels of decompositions?

5.2. Computing.

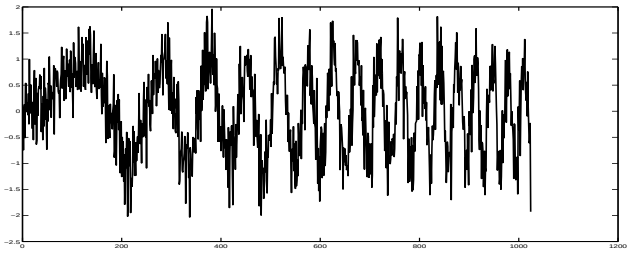
(1) Write a MATLAB code to implement the 1D Discrete Haar Wavelet Transform (1D HWT) including the algorithms for
   - Haar pyramidal decomposition, and
   - Haar pyramidal reconstruction.

Compute the six-level decomposition of the data

$$f_n = \sin\left(\frac{n^2}{10000}\right) + \eta_n,$$

where $n = 1, \ldots, 1024$, and η_n is selected from a normal distribution with mean zero and variance 0.2. (See figure below) Initialize the transform by assuming that $\mathbf{f} \in V_0$. Include the following plots in your report:
   (a) $P_i\mathbf{f} = \mathbf{f}_a^i \in V_i$ for all $i = 1, \ldots, 6$,
   (b) $Q_i\mathbf{f} = \mathbf{f}_d^i \in W_i$ for all $i = 1, \ldots, 6$.

Be sure that each plot has domain $[1, 1024]$.



(2) Test your codes with your favorite image for this problem. Make sure your codes are as general as possible and be sure to plot the results.
   (a) Write a function in MATLAB to compute the *approximation* and each of the three *detail* components of an image. (i.e., you will produce 4 *extremely* short codes here) Notice that the resolution of LL, HL, LH, and HH will be $M/2 \times N/2$, where (M,N) is the resolution of the original image.
   (b) Write a subroutine to reconstruct an image from **only** the *approximation* component as a function of level. Notice that the resolution of the reconstructed image will be of size $M \times N$.
   (c) Compress the image up to level 3. Compute the compression ratio as a function of each compression level. Plot the compressed images for each level along with their compression ratio. Note that the compression ratio is defined as

$$CR = \frac{\text{\# of nonzero entries in the transformed}}{\text{\# of nonzero entries in the compressed}}.$$

Some useful MATLAB commands for this problem:

```
>> dwt2
>> wavedec2
>> wrcoef2
```

6. Group Final Project

6.1. Data: Imges of Cats and Dogs. The following data are available on the course website.

(1) PatternRecData.mat which contains two variables: the 198-by-198 matrix KLDATA.mat and a row vector sub-labels of length 160. The data matrix KLDATA contains distinct images of cats and dogs (courtesy of Dave Bolme and Dr. Ross J. Beveridge, Department of Computer Science, Colorado State University) in its columns. There are 99 of each animal and they are randomly placed in the columns of KLDATA. The vector sub-labels gives you the identity (with cat = 1 and dog = 0) of the first 160 patterns.
(2) TIFFtraining.zip which contains TIFF images for the first 160 patterns in KLDATA. There is a little glitch to the file Dog96.tif, which is a $64 \times 64 \times 2$ matrix instead of simply 64×64. The first layer is what you would need.

6.2. Project Assignment. Use the data given above to build different pattern recognition architectures from the methods that you learned in class over the semester or methods you acquired elsewhere that are relevant to the problem. *Include as many methods as the number of members in the group. Note there is no limit to group size but that the entire group will receive the same grade.*
Submit a write-up that that are coherent to the format described in syllabus. Once you are satisfied with your pattern recognition routine on the known data, classify the last 38 unknown columns in KLDATA as either cats or dogs. Save the result as a row vector of zeros (= dogs) and ones (= cats) and email it to me by the end of the semester. Alternatively, if you wish to classify the raw data

(instead of the KL data), you would design your classifier to take in either a 4096-by-1 column vector or a 4096-by-38 matrix and output their class labels as either cats or dogs.
Your final write-up will be sure to include the following items.

(1) Classification errors as a 2-by-2 confusion matrix (dogs classified as dogs, dogs classified as cats, cats classified as cats, cats classified as dogs). You can accomplish this by splitting the data into testing and training and provide classification errors on the testing set.
(2) Description of the classification method and details about how the classifier is constructed.
(3) Predicted class membership for the 38 unlabeled data.
(4) Codes used in the exploration process. These codes can be inserted wherever they fit or shuffled all the way at the end of the report depending on your writing style.

6.3. Suggestions for Possible Approaches. The following list is by no means complete. The purpose is to provide you with some initial directions.

(1) Determine the covariance matrix of the cats and the covariance matrix of the dogs and construct optimal bases for each using maximum noise fraction. Project new samples onto the cat basis and dog basis and see which gives a better representation.
(2) Use vector quantization, e..g, Kohonen's self-organizing map on a 2D lattice.
(3) Use eigen-cats and eigen-dogs and the Principal Component Analysis (PCA) on the raw data.
(4) 2D Discrete Wavelet Analysis (DWT) or Fourier Analysis on the raw data for frequency content information.
(5) Radial Basis Function (map cats to ones and dogs to zeros).
(6) Fisher's Linear Discriminant Analysis (LDA).
(7) Labeled Voronoi cell classification.
(8) One-sided or two-sided tangent distances.
(9) Set-to-set comparison with principal angles and Grassmannian distances.
(10) A combination of any of the methods above with weights.

A prize will be given to the team who has the highest classification rate (or lowest misclassification rate).

Bibliography

[1] PhD thesis.

[2] A. Barg and D. Nogin. Bounds on packings of spheres in the Grassmann manifold. *IEEE Trans. Information Theory*, 48(9):2450–2454, 2002.

[3] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12:2385–2404, 2000.

[4] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[5] C. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, New Jersey, 2006.

[6] A. Björck and G. Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of Computation*, 27(123):579–594, 1973.

[7] E. Brigham. *The Fast Fourier Transform and its Applications*. Prentice Hall, Upper Saddle River, New Jersey, 1988.

[8] D. Broomhead and M. Kirby. A new approach for dimensionality reduction: theory and algorithms. *SIAM J. of Applied Mathematics*, 60(6):2114–2142, 2000.

[9] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE. Trans. Commun.*, 31(4):532–540, 1983.

[10] J.-M. Chang, M. Kirby, and C. Peterson. Set-to-set face recognition under variations in pose and illumination. In *Proc. Biometrics Symposium*, Baltimore, MD, U.S.A., September 2007.

[11] J. Conway, R. Hardin, and N. Sloane. Packing lines, planes, etc.: Packings in Grassmannian spaces. *Experimental Mathematics*, 5:139–159, 1996.

[12] I. Daubechies. Ten lectures on wavelets. In *CBMS-NSF Regional Conference Series in Applied Mathematics*, Philadelphia, PA, 1992. SIAM.

[13] P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, 1982.

[14] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, New York, second edition, 2001.

[15] H. Eastment and W. Krzawnowski. Cross-validatory choice of the number of components from a principal component analysis. *Technometrics*, 24:73–77, 1982.

[16] A. Edelman, T. Arias, and S. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20(2):303–353, 1999.

[17] L. Eldén. *Matrix Methods in Data Mining and Pattern Recognition*. SIAM, 2007.

[18] R. Everson and L. Sirovich. The karhunen-loéve transform for incomplete data. *J. Opt. Soc. Am., A*, 12(8):1657–1664, 1995.

[19] G. Farin and D. Hansford. *Practical Linear Algebra: A Geometry Toolbox*. A K Peters, Ltd., Massachusetts, 2004.

[20] R. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[21] K. Fukunaga and D.R. Olsen. An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, C-20(2):176–183, 1971.

[22] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, third edition, 1996.

[23] R. Gonzalez and R. Woods. *Digital Image Processing*. Prentice Hall, New Jersey, third edition, 2008.

[24] A. Green, M. Berman, P. Switzer, and M. Craig. A transformation for ordering multispectral data in terms of image quality with implications for noise removal. *IEEE Trans. on Geoscience and Remote Sensing*, 26(1):65–74, 1988.

[25] P. Griffiths and J. Harris. *Principles of Algebraic Geometry*. Wiley & Sons, 1978.

[26] A. Haar. Zur theorie der orthogonalen funktionensysteme. *Math. Annal.*, 69:331–371, 1910.

[27] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, U.K., 1985.

[28] H. Hotelling. Analysis of a complex statisfical variables into principal components. *Journal of Educational Psychology*, September 1933.

[29] I.T. Jolliffe. *Principal Component Analysis*. Springer, New York, 1986.

[30] W. Kendall. Probability, convexity and harmonic maps with small image I: Uniqueness and fine existence. In *Proc. of the London Mathematical Society*, volume 61, pages 371–406, 1990.

[31] M. Kirby. *Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns*. Wiley & Sons, 2001.

[32] M. Kirby and L. Sirovich. Application of the Karhunen-Loève procedure for the characterization of human faces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(1):103–108, 1990.

[33] M. Kirby, F. Weisser, and G. Dangelmayr. A problem in facial animation: analysis and synthesis of lip motion. In *Proc. 7th Scandinavian Conf. on Image Analysis*, pages 529–536, Aalborg, Denmark, 1991.

[34] M. Kirby, F. Weisser, and G. Dangelmayr. Speaking with images: A model problem in the representation of still and moving images. *Pattern Recognition*, 26(1):63–73, 1993.

[35] A. Knyazev and M. Argentati. Principal angles between subspaces in an *a*-based scalar product: Algorithms and perturbation estimates. *SIAM J. Sci. Comput.*, 23(6), 2002.

[36] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 1984.

[37] P. Lancaster and M. Tismenetsky. *The Theory of Matrices*. Academic Press, New York, 1985.

[38] E. Lorenz. Empirical orthogonal eigenfunctions and statistical weather prediction. Technical Report Science Report No. 1, Statistical Forecasting Project 1, M.I.T., 1956.

[39] S. Mallat. A compact multiresolution representation: The wavelet model. In *Proc. IEEE Computer Society Workshop on Computer Vision*, pages 2–7, Washington, D.C., 1987. IEEE Computer Society Press.

[40] S. Mallat. Multiresolution approximations and wavelet orthonormal bases of $L^2(\mathbb{R})$. *Trans. Amer.Math. Soc.*, 315(1):69–87, 1989.

[41] S. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Analysis & Machcine Intelligence*, pages 674–693, 1989.

[42] J. Marks. Discriminative canonical correlations: An offspring of fisher's discriminant analysis. Master Paper, May 2009. Department of Mathematics.

[43] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Muller. Fisher discriminant analysis with kernels. In *Proc. IEEE Neural Networks for Signal Processing Workshop*, pages 41–48. IEEE Computer Society Press, 1999.

[44] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Stanford Digital Library Working Papers, 1998.

[45] L. Sirovich and M. Kirby. A low-dimensional procedure for the characterization of human faces. *J. of the Optical Society of America A*, 4(3):529–524, 1987.

[46] S. Watanabe. Karhunen-Loéve expansion and factor analysis. In *Trans. 4th. Prague Conf. on Inf. Theory. Decision Functions, and Random Proc.*, pages 635–660, Prague, 1965.

APPENDIX A

Supplementary Materials

1. Linear Independence and Bases

Given a set of vectors $\{v_1, v_2, \ldots, v_n\}$ in $\mathbb{R}^m$, $m \geq n$, the set of linear combinations

$$\text{span}\,(v_1, v_2, \ldots, v_n) = \{y \quad | \quad y = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n\}$$

is called the *span* of the set of vectors $v_1, v_2, \ldots, v_n$. If we concatenate the vectors into a single matrix X, i.e.,

$$X = [v_1 \,|\, v_2 \,|\, \cdots \,|\, v_n],$$

then the span$(v_1, v_2, \ldots, v_n)$ is equal to the *range* of the (transformation) matrix X or the *column space* of the transformation X, denoted by $\mathscr{R}(X)$. The vectors $v_1, v_2, \ldots, v_n$ are *linearly independent* if the only way to write $\sum_{j=1}^{n} \alpha_j v_j = 0$ is when $\alpha_i = 0$ for all $i = 1, \ldots, n$. A set of m linearly independent vectors in $\mathbb{R}^m$ is called a *basis* in $\mathbb{R}^m$. This is equivalent to say that any vector in $\mathbb{R}^m$ can be written as a linear combination of the basis vectors.

If we have a set of linearly dependent vectors, then we can keep a linearly independent subset and express the rest in terms of the linearly independent ones. Thus we can consider the number of linearly independent vectors as a measure of the information contents of the set and compress the set accordingly: take the linearly independent vectors as representatives (basis vectors) for the set, and compute the coordinates of the rest in terms of the basis. However, in real applications we seldom have *exactly linearly dependent vectors* but rather *almost linearly dependent vectors*. It turns out that for such a *data reduction procedure* to be practical and numerically stable, we need the basis vectors to be not only linearly independent but *orthogonal*. So, how do we find out whether or not a set of vectors is linearly independent or not in large data sets? The answer lies within the concept of *rank*.

2. The Rank of a Matrix

The *rank* of a matrix is defined as the maximum number of linearly independent columns. It is a standard result in linear algebra that the number of linearly independent column vectors is equal to the number of linearly independent row vectors. In real data set, we seldom calculate the exact rank of the data matrix. Instead, we consider the *numerical rank* of the data matrix. We will visit this concept in the main text. If a matrix is not full rank, then it is said to be *rank-deficient*.

A square matrix $A \in \mathbb{R}^{n \times n}$ with rank n (full rank) is called *nonsingular* and has an inverse A^{-1} satisfying

$$AA^{-1} = A^{-1}A = I.$$

If we multiply the linearly independent vectors by a nonsingular matrix, then the resulting vectors remain linearly independent.

PROPOSITION 2.1. *Assume that the vectors $v_1, v_2, \ldots, v_p$ are linearly independent. Then for any nonsingular matrix T, the vectors $Tv_1, Tv_2, \ldots, Tv_p$ are linearly independent.*

EXAMPLE 2.1. The matrix $A = \mathbf{u}\mathbf{v}^T$ has rank 1.

3. The Similarity Transformation in Registration Problem

Let

$$S=\begin{bmatrix} s_x & 0 & 0\\ 0 & s_y & 0\\ 0 & 0 & 1\end{bmatrix}, T=\begin{bmatrix} 1 & 0 & t_x\\ 0 & 1 & t_y\\ 0 & 0 & 1\end{bmatrix}, \text{and} \quad R=\begin{bmatrix} \cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix} a & -b & 0\\ b & a & 0\\ 0 & 0 & 1\end{bmatrix}$$

represent the scaling, translation, and rotation matrix, respectively. Composition of the three gives a similarity transform (order doesn't matter)

$$M=RST=\begin{bmatrix} as_x & -bs_y & at_xs_x-bt_ys_y\\ bs_x & as_y & bt_xs_x+at_ys_y\\ 0 & 0 & 1\end{bmatrix}.$$

If the horizontal and vertical scaling are not the same, this transformation matrix has 6 degrees of freedom. To simplify the problem, we will assume $s_x=s_y$. Therefore

$$M=\begin{bmatrix} A & -B & C\\ B & A & D\\ 0 & 0 & 1\end{bmatrix},$$

which has 4 degrees of freedom. To solve the parameters A,B,C, and D, we will need 2 points. Say, for example, we have two known eye coordinates (left and right pupil positions) given in $[x_0,y_0,1]^T$ and $[x_1,y_1,1]^T$, which are called *source points*. Given the corresponding *target points* (where the two eye coordinates are mapped to at the end of the transformation) $[u_0,v_0,1]^T$ and $[u_1,v_1,1]^T$, we have the following relations

$$\begin{bmatrix} u_0\\ v_0\\ 1\end{bmatrix}=\begin{bmatrix} A & -B & C\\ B & A & D\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix} x_0\\ y_0\\ 1\end{bmatrix}, \text{and} \quad \begin{bmatrix} u_1\\ v_1\\ 1\end{bmatrix}=\begin{bmatrix} A & -B & C\\ B & A & D\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix} x_1\\ y_1\\ 1\end{bmatrix}.$$

Multiplication gives the system of 4 equations

$$\begin{aligned} x_0A-y_0B+C &= u_0\\ y_0A+x_0B+D &= v_0\\ x_1A-y_1B+C &= u_1\\ y_1A+x_1B+D &= v_1\end{aligned}$$

In matrix notation, we have

$$\underbrace{\begin{bmatrix} x_0 & -y_0 & 1 & 0\\ y_0 & x_0 & 0 & 1\\ x_1 & -y_1 & 1 & 0\\ y_1 & x_1 & 0 & 1\end{bmatrix}}_{m}\begin{bmatrix} A\\ B\\ C\\ D\end{bmatrix}=\begin{bmatrix} u_0\\ v_0\\ u_1\\ v_1\end{bmatrix}.$$

Therefore

$$\begin{bmatrix} A\\ B\\ C\\ D\end{bmatrix}=m^{-1}\begin{bmatrix} u_0\\ v_0\\ u_1\\ v_1\end{bmatrix}.$$

With the parameters A,B,C and D are now known, the final similarity transformation M can be used to transform any 2D image to a prescribed image plane.

4. Generalized Singular Value Decomposition

THEOREM 4.1. [**22**] *(Generalized Singular Value Decomposition) If we have $A \in \mathbb{R}^{m\times p}$ with $m \geq p$ and $B \in \mathbb{R}^{n\times p}$, there exist orthogonal $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ and an invertible $X \in \mathbb{R}^{p\times p}$ such that*

$$U^T AX = C = diag(c_1,\ldots,c_p) \quad c_i \geq 0 \tag{120}$$

and

$$V^T BX = S = diag(s_1,\ldots,s_q) \quad s_i \geq 0 \tag{121}$$

where $q = \min(p,n)$.

To see that these assignments do give rise to the relation

$$S^2 A^T AX = C^2 B^T BX,$$

consider the following. From Equation 120 and 121, we have $C^T = X^T A^T U$ and $S^T = X^T B^T V$. Since C and S are diagonal, so $C^T C = C^2$, $S^T S = S^2$. Thus,

$$\begin{aligned} C^T C &= (X^T A^T U)(U^T AX) = X^T A^T AX = C^2 \Rightarrow X^T = C^2 X^{-1} A^{-1} (A^T)^{-1} \\ S^T S &= (X^T B^T V)(V^T BX) = X^T B^T BX = S^2 \Rightarrow X^T = S^2 X^{-1} B^{-1} (B^T)^{-1} \end{aligned}$$

Equating $(X^T)^{-1}$'s to obtain

$$A^T AX(C^2)^{-1} = B^T BX(S^2)^{-1}.$$

Multiply through by C^2 and S^2 appropriately to get

$$S^2 A^T AX = C^2 B^T BX.$$

To see how this is related to the symmetric definite generalized eigenproblem

$$N^T N\psi = \mu^2 X^X \psi$$

in the maximum-noise-fraction problem, first suppose that if $\psi = [\psi^{(1)},\ldots,\psi^{(p)}]$ satisfy

$$s_i^2 A^T Ax_i = c_i^2 B^T Bx_i, \quad i = 1:p$$

and $s_i \neq 0$, then $A^T Ax_i = \mu^2 B^T Bx_i$, where $\mu_i = c_i/s_i$. Thus, x_i are termed the generalized singular vectors of the pair (A,B). Replace N with A, X with B, and ψ with x, we see that the maximum-noise-fraction problem can be solved by using the GSVD method.

4.1. MATLAB Syntax. To compute GSVD of two data matrices $A \in \mathbb{R}^{m\times p}$ and $B \in \mathbb{R}^{n\times p}$, use the command

```
%% full version
[U,V,X,C,S] = gsvd(A,B);
[U,V,X,C,S] = gsvd(A,B,0); % thin version
```

so that

$$A = UCX^T \tag{122}$$

and

$$B = VSX^T, \tag{123}$$

$C^T C + S^T S = I$ with unitary matrices U and V and non-negative diagonal matrices C and S, also satisfying

$$S^2 A^T A(X^T)^{-1} = C^2 B^T B(X^T)^{-1}.$$

Note that $\#col(A) = \#col(B)$, but $\#row(A)$ does not necessarily have to be equal to $\#row(B)$. The command

```
S = gsvd(A,B);
```

returns the vector of generalized singular values, i.e.,

```
S = sqrt(diag(C'*C)./diag(S'*S))
```

In the economic version, the resulting U and V have at most p columns, and C and S have at most p rows. The generalized singular values are $diag(C)./diag(S)$.
There is a minor bug in the gsvd command. From Equations 122 and 123, we get $C = UA(X^T)^{-1} = U^TA(X^{-1})^T$ and $S = V^TB(X^T)^{-1} = V^TB(X^{-1})^T$, which are equivalent to $C^T = X^{-1}A^TU$ and $S^T = X^{-1}B^TV$. Thus,

$$\begin{aligned} C^TC &= C^2 = X^{-1}A^TA(X^{-1})^T = X^{-1}A^TA(X^T)^{-1} \Rightarrow X^{-1} = C^2X^TA^{-1}(A^T)^{-1} \\ S^TS &= S^2 = X^{-1}B^TB(X^{-1})^T = X^{-1}B^TB(X^T)^{-1} \Rightarrow X^{-1} = S^2X^TB^{-1}(B^T)^{-1} \end{aligned}$$

Equating X's to obtain

$$\begin{aligned} & \left(C^2X^TA^{-1}(A^T)^{-1}\right)^{-1} = \left(S^2X^TB^{-1}(B^T)^{-1}\right)^{-1} \\ \Rightarrow\ & (C^2)^{-1}A^TA(X^T)^{-1} = (S^2)^{-1}B^TB(X^T)^{-1} \\ \Rightarrow\ & S^2A^TA(X^T)^{-1} = C^2B^TB(X^T)^{-1} \end{aligned}$$

Thus, let $\psi = (X^T)^{-1}$ in the maximum-noise-fraction algorithm to achieve the correct solutions.

```
function [Phi] = mnf(N,X)
%% this algorithm requires the knowledge of N (noise)
%% treat N as noise and X as data in the gsvd.

[U,V,A,C,S] = gsvd(N,X,0);
psi = inv(A.');
Phi = X*psi;  % optimal basis vectors
```

5. Derivation of Generalized Eigenvalue Problem

We establish the generalized eigenvalue problem found in Section 1.1. Recall $J(w) = \dfrac{N(w)}{D(w)} = \dfrac{w^TS_Bw}{w^TS_Ww}$. Since S is symmetric, $\dfrac{d}{dw}\left(w^TSw\right) = 2Sw$. Thus,

$$\nabla J(w) = \frac{w^TS_Ww\,(2S_Bw) - w^TS_Bw\,(2S_Ww)}{\left(w^TS_Ww\right)^2}.$$

Setting $\nabla J(w) = 0$ gives

$$w^TS_Ww\,(2S_Bw) - w^TS_Bw\,(2S_Ww) = 0.$$

That is,

$$D(w)S_Bw = N(w)S_Ww \Rightarrow S_Bw = \lambda S_Ww,$$

where $\lambda = \dfrac{N(w)}{D(w)}$.
This is a generalized eigenvalue problem.