6 Decomposition of Multichannel Data

6.1 Introduction

In the previous chapters we mainly focused on the analysis of single-input/single-output systems, single-channel data, or single images. Even when we worked with images, we worked with one row or column of pixels at a time. At most, we considered pairs of signals when we determined cross-correlation or coherence, or when we looked into input−output relationships. Although these techniques form a basis for analysis in neuroscience research, current studies usually collect multiple channels and/or movies of neural activity.

Examples of commonly encountered multichannel data sets are electroencephalograms (EEG), electrocorticograms (ECoG), recordings with multi-electrode arrays, a sequence of functional magnetic resonance images (fMRI), or movies made from neural tissue with voltage-sensitive or calcium indicator dyes. In these examples we deal not just with two or three simultaneously recorded signals, but with potentially overwhelming numbers of channels consisting of both spatial and temporal components. In the ECoG, each channel is at a certain location recording signals evolving over time; in both an fMRI sequence and a movie, the neural signals are represented by the intensity of each pixel as a function of time. Suppose we digitized an fMRI set with 128 samples in time where each image is 128×128 pixels, we would now have a huge data set consisting of $128^3 = 2{,}097{,}152$ points. Say we sample ECoG at a rate of 400 samples per channel per second and we record 128 channels for 60 s, resulting in a 1-min data set of $128 \times 400 \times 60 = 3{,}072{,}000$ data points. Examples of a small part of a 128-channel ECoG recording and a 21-channel EEG are shown in Fig. 6.1.

Typical goals for multichannel data processing are data reduction, decomposition, or investigation of the causal structure within the data. In the case of data reduction, we attempt to find the signal and noise components, and in the case of decomposition, our goal is to find the basic signal components that are mixed into the multichannel data set. Of course, both of these approaches are related. Suppose we have a measurement of brain activity during a task, and the activity associated with the task is signal while the remaining activity can be considered noise. If we can decompose our measured brain activity into these basic components, we have effectively used decomposition as a tool for data reduction.

Signal Processing for Neuroscientists, A Companion Volume. DOI: 10.1016/B978-0-12-384915-1.00006-1

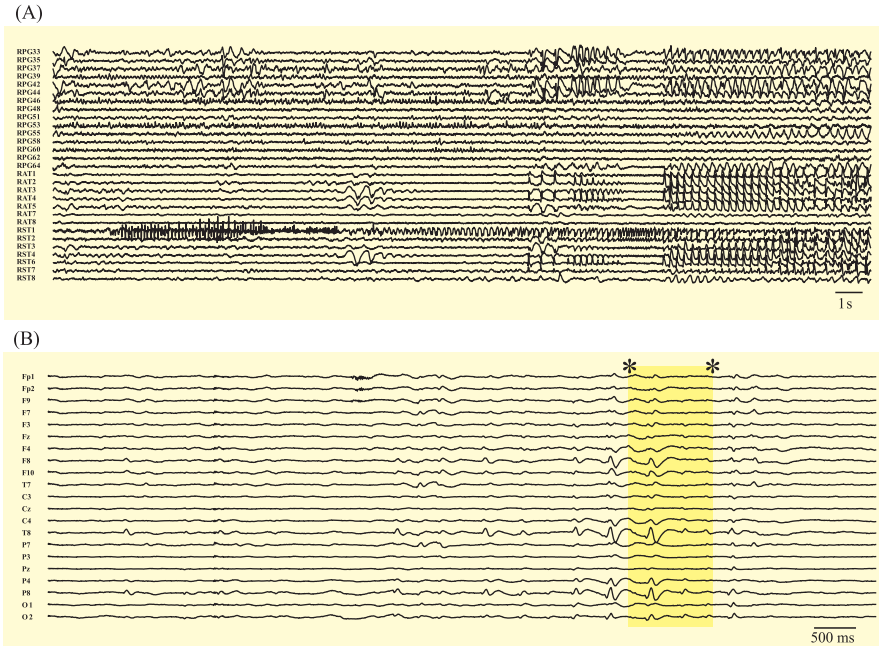

Figure 6.1 (A) Part of an intracranial 128-channel recording from a patient with epilepsy. Only a limited number of channels are shown: 15 channels of a right parietal grid (RPG), 7 channels from a right anterior temporal strip (RAT), and 7 from a right superior temporal strip (RST). The activity associated with a seizure onset starts with rapid oscillations in channel RST1 and eventually results in a spread of oscillatory activity in almost all channels. (B) A 21-channel scalp EEG recording from a different patient with epilepsy. The large-amplitude waveforms in this recording are epileptic spikes. The interval in between the asterisks (highlighted in yellow) is used for further analysis in Fig. 6.15.

Although we will discuss multichannel analysis of data sets with large numbers of signals, we will introduce examples with strongly reduced data sets of two or three channels. The idea is to illustrate principles that are valid in high-dimensional data space with 2D or 3D examples. Throughout this chapter, we will demonstrate principles rather than formally prove them. First, we will show the general principle of mixing and unmixing of signals (Section 6.2), and then we will go into the details of specific strategies: principal component analysis (PCA; Section 6.3) and independent component analysis (ICA; Section 6.4). If you are interested in proofs, see texts on this topic (e.g., Bell and Sejnowski, 1995; Stone, 2004), on linear algebra (e.g., Jordan and Smith, 1997; Lay, 1997), and on information theory (e.g., Cover and Thomas, 1991; Shannon and Weaver, 1949).

6.2 Mixing and Unmixing of Signals

The underlying model of decomposition is that the signals of interest are mixtures of sources. For instance, our ear detects sounds from different sources

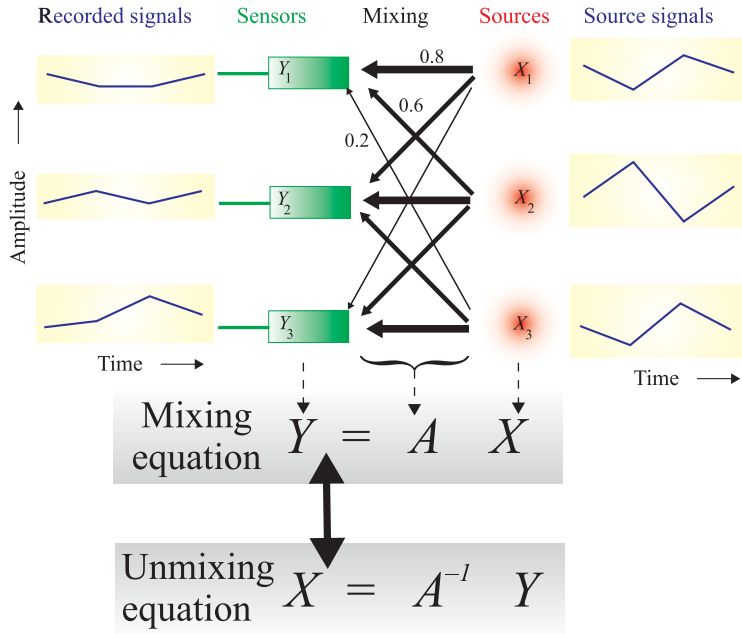

Figure 6.2 A mix of signals originating from three sources X (X_1–X_3) is recorded by three sensors Y (Y_1–Y_3). The signal from source to sensor is attenuated with distance. The amount of attenuation is symbolized by the width of the arrows (there are only three widths in this example: 0.8, 0.6, or 0.2). The mixing process can be represented by the matrix multiplication $Y = AX$, in which A is the mixing matrix. The unmixing process can be represented by $X = A^{-1}Y$, in which the unmixing matrix A^{-1} is the inverse of A.

simultaneously, such as someone talking to us, the noise of a fan, and music from a radio. Another example is an EEG electrode that picks up electrical activity from several parts of the brain. In other words, a set of measured signals Y_n consists of channels that are mixtures from a number of sources X_n. For now we assume that the number of sources is equal to or smaller than the number of signals we record. In this case the problem of mixing and unmixing can be defined mathematically. Let us consider a concrete example where we have three sources X_1–X_3 and three sensors Y_1–Y_3 (Fig. 6.2). The sensors pick up the signal from each source, but the signal is attenuated when traveling from source to sensor and the attenuation is proportional with distance. The level of attenuation in Fig. 6.2 is indicated by the width of the arrows: the signal from the source closest to the sensor attenuates least, only by a factor of 0.8. The signals from the other sources attenuate more, depending on distance, by a factor of 0.6 or a factor of 0.2 respectively. If we look at sensor Y_1 we can see that it will pick up $0.8\times$ the signal from source X_1 plus

0.6 × the signal from source X_2 plus 0.2 × the signal from source X_3. Similar rules can be found for the other sensors Y_2 and Y_3. The measurements at the three sensors in Fig. 6.2 can be represented by a linear system of three equations:

$$\begin{aligned} Y_1 &= 0.8X_1 + 0.6X_2 + 0.2X_3 \\ Y_2 &= 0.6X_1 + 0.8X_2 + 0.6X_3 \\ Y_3 &= 0.2X_1 + 0.6X_2 + 0.8X_3 \end{aligned}$$

Because this system has three equations with three unknowns X_1−X_3 (note that Y_1−Y_3 are known because they are measured), we can solve the system of equations above for the source signals X_1−X_3. We can put the equations in matrix form:

$$Y = AX \tag{6.1a}$$

in which,

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix}, \quad X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}, \text{ and the mixing matrix } A = \begin{bmatrix} 0.8 & 0.6 & 0.2 \\ 0.6 & 0.8 & 0.6 \\ 0.2 & 0.6 & 0.8 \end{bmatrix}$$

represents the attenuation coefficients for the setup in Fig. 6.2. The fact that $A(i,j) = A(j,i)$ (i.e., the mixing matrix A is symmetric) is due to the symmetric setup of the example in Fig. 6.2; this is not necessarily the case for a generic mixing matrix. Now suppose that we have recorded a time series of four samples from the three sensors and we want to know the signals from the individual sources. Because we know that $Y = AX$ we can compute X by $X = A^{-1}Y$, where A^{-1} is the inverse of A (here we assume that the inverse of A exists). If we take an example where the sources emit the following signals at times t_1−t_4 (indicated as source signals in the right panel in Fig. 6.2):

	t_1	t_2	t_3	t_4
X_1 :	24.1667	1.6667	33.3333	17.5000
X_2 :	−32.5000	0.0000	−55.0000	−22.5000
X_3 :	20.8333	3.3333	41.6667	17.5000

Our sensors will pick up:

$$Y = AX = \begin{bmatrix} 0.8 & 0.6 & 0.2 \\ 0.6 & 0.8 & 0.6 \\ 0.2 & 0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 24.1667 & 1.6667 & 33.3333 & 17.5000 \\ -32.5000 & 0.0000 & -55.0000 & -22.5000 \\ 20.8333 & 3.3333 & 41.6667 & 17.5000 \end{bmatrix}$$

$$= \begin{bmatrix} 4.0 & 2.0 & 2.0 & 4.0 \\ 1.0 & 3.0 & 1.0 & 3.0 \\ 2.0 & 3.0 & 7.0 & 4.0 \end{bmatrix}$$

So our measurement (indicated as recorded signals in the left panel in Fig. 6.2) will be:

$$\begin{array}{rcccc} & t_1 & t_2 & t_3 & t_4 \\ Y_1: & 4 & 2 & 2 & 4 \\ Y_2: & 1 & 3 & 1 & 3 \\ Y_3: & 2 & 3 & 7 & 4 \end{array}$$

Because we know mixing matrix A, we compute its inverse (if you want to check this example in MATLAB, the inverse of a matrix can be obtained with the inv command) so that we can estimate the source activity $\hat{X}$ from the measurements:

$$\hat{X} = A^{-1}Y \tag{6.1b}$$

That is:

$$\underbrace{\begin{bmatrix} 24.1667 & 1.6667 & 33.3333 & 17.5000 \\ -32.5000 & -0.0000 & -55.0000 & -22.5000 \\ 20.8333 & 3.3333 & 41.6667 & 17.5000 \end{bmatrix}}_{\hat{X}} =$$

$$\underbrace{\begin{bmatrix} 5.8333 & -7.5000 & 4.1667 \\ -7.5000 & 12.5000 & -7.5000 \\ 4.1667 & -7.5000 & 5.8333 \end{bmatrix}}_{A^{-1}} \underbrace{\begin{bmatrix} 4.0 & 2.0 & 2.0 & 4.0 \\ 1.0 & 3.0 & 1.0 & 3.0 \\ 2.0 & 3.0 & 7.0 & 4.0 \end{bmatrix}}_{Y}$$

As you can see, our estimate $\hat{X}$ for X is perfect (except for any precision errors due to computation). Although this example clarifies the mixing and unmixing process, it is not very helpful in practical applications (even if we ignore the effects of noise that would be present in any real recording) because the mixing matrix is unknown and/or the number of sources outnumbers the number of sensors. In the remainder of this chapter we will focus on what one can do if the mixing matrix is unknown. In this case, we want to separate the sources while we are "blind" with respect to the mixing process; therefore, these procedures are called blind source separation (BSS). We will specifically focus on two of these techniques: PCA and ICA.

6.3 Principal Component Analysis

In this section we introduce the concept of decomposing multichannel data into its principal components. With PCA of a multidimensional measurement, one can find the directions of maximal and minimal variance in the multidimensional measurement space. We will see that these directions are orthogonal, indicating that the

components extracted with PCA are uncorrelated. We will introduce the technique by analyzing a concrete 3D example of four measurements S_1−S_4, each observation S_n having three values or signals s_1, s_2, and s_3 (one for each of the three dimensions):

$$S_1 = \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix} \quad S_2 = \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix} \quad S_3 = \begin{bmatrix} 2 \\ 1 \\ 7 \end{bmatrix} \quad S_4 = \begin{bmatrix} 4 \\ 3 \\ 4 \end{bmatrix} \tag{6.2a}$$

The mean vector of these four observations, M, contains the mean for each of the three signals m_1, m_2, and m_3:

$$M = \frac{1}{4}\{S_1 + S_2 + S_3 + S_4\} = \frac{1}{4}\left\{ \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \\ 7 \end{bmatrix} + \begin{bmatrix} 4 \\ 3 \\ 4 \end{bmatrix} \right\} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 4 \end{bmatrix} \tag{6.2b}$$

A 3D plot of the observations and their mean is shown in Fig. 6.3A. If we now demean our four observations—that is, we subtract M from S_1 to S_4 (as we generally do with our signals before processing them)—and we group the demeaned observation in matrix B, we have:

$$B = \begin{bmatrix} 4-3 & 2-3 & 2-3 & 4-3 \\ 1-2 & 3-2 & 1-2 & 3-2 \\ 2-4 & 3-4 & 7-4 & 4-4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -2 & -1 & 3 & 0 \end{bmatrix} \tag{6.3}$$

In statistics, a data set from multichannel observations such as the concatenated matrix $S = \begin{bmatrix} S_1 & S_2 & S_3 & S_4 \end{bmatrix}$ or matrix B is called multivariate data. A scatter plot of the demeaned observations is shown in Fig. 6.3B. Note that the new mean value is now at the origin, and so we have in effect translated the axes of our coordinate system. From B, we can compute the covariance matrix C. Since we have three variables (s_1, s_2, s_3) in each observation, the covariance matrix is 3 × 3. If we have N observations, each entry in the matrix can be computed by $C(i,j) = 1/(N-1)\sum_{n=1}^{N}(s_i - m_i)_n(s_j - m_j)_n$. In this example, C is a 3 × 3 matrix, i and j range from 1 to 3, and N is the number of observations, in this example $N = 4$ (since we have observations S_1−S_4). In matrix notation this notation can be simplified to:

$$\begin{aligned} C = \frac{1}{N-1}BB^{\mathrm{T}} &= \frac{1}{3}\begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -2 & -1 & 3 & 0 \end{bmatrix}\begin{bmatrix} 1 & -1 & -2 \\ -1 & 1 & -1 \\ -1 & -1 & 3 \\ 1 & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 1.33 & 0 & -1.33 \\ 0 & 1.33 & -0.67 \\ -1.33 & -0.67 & 4.67 \end{bmatrix} \end{aligned} \tag{6.4}$$

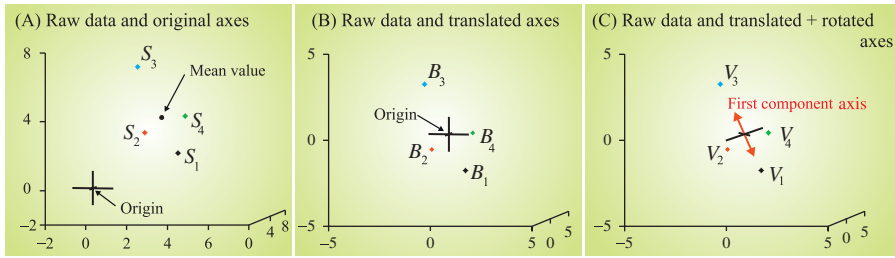

Figure 6.3 (A) A 3D plot of four observation vectors S_1-S_4 (Equation (6.2a)) and their mean value M (Equation (6.2b)). (B) The same points, now indicated as B_1-B_4 because they are plotted against axes that are translated so that the mean M becomes the new origin. (C) Finally we plot the same points (now indicated as V_1-V_4) against axes that are also rotated to reflect the directions of the three principal components. The first principal component is indicated by the double arrow (red). This illustration was made with MATLAB script Pr6_1.m (available on http://www.elsevierdirect.com/companions/9780123849151); the numerical values can be found in Table 6.1.

The superscript "T" indicates the transpose of matrix B (in the transpose rows and columns are interchanged such that $B(i,j) \rightarrow B(j,i)$). Each value in the diagonal of C represents the variance of the b_1, b_2, and b_3 values of observation vectors B_n. So the sum of the diagonal elements, the trace of C written as tr(C), is the total variance. Each off-diagonal element is a covariance value—for example, $C(2,3)$ is the covariance between the b_2 and b_3 coordinates. Of course, $C(3,2)$ is the same value because it is the covariance between the b_3 and b_2 coordinates. Therefore a covariance matrix is always a **symmetric** matrix (see the example in Equation (6.4)). A more formal way to establish symmetry for covariance matrices is to show that interchanging the rows and columns (transposition) of covariance matrix C results in the same matrix: that is, $C = C^{\mathrm{T}}$. From Equation (6.4) we can establish that C is proportional with BB^{T} (by a factor of $1/(N-1)$) and the transposing operation on C can be represented by $(BB^{\mathrm{T}})^{\mathrm{T}} = B^{\mathrm{TT}}B^{\mathrm{T}}$ (recall that the multiplication order of matrices switches when taking their transpose). Because the transpose of a transposed matrix is the original matrix again, we may simplify this outcome $B^{\mathrm{TT}}B^{\mathrm{T}} = BB^{\mathrm{T}}$, which shows that $(BB^{\mathrm{T}})^{\mathrm{T}} = BB^{\mathrm{T}}$—that is, the transpose of BB^{T} is BB^{T} again.

If $C(i,j)$ for $i \neq j$ is zero, there is no covariance or correlation between b_i and b_j. It may be clear that analysis of multivariate data is simpler when all signals are uncorrelated—that is, a covariance matrix that is **diagonal**, which means that all off-diagonal elements are zero. This is exactly the goal of the decomposition with PCA.

Note: Correlation (ρ_{xy}) between two variables x and y is a normalized version of the covariance (Cov(x,y)) between x and y—that is, $\rho_{xy} = \mathrm{Cov}(x,y)/\sigma_x\sigma_y$— with standard deviations σ_x and σ_y for x and y, respectively. The effect of this normalization is that the correlation coefficient ρ_{xy} is scaled between -1 and 1.

6.3.1 Finding Principal Components

To summarize the above, the strategy of PCA is to manipulate our demeaned observations B_n $(b_1, b_2, b_3)_n$ for which correlations between b_i and b_j may exist into transformed data V_n $(v_1, v_2, v_3)_n$ such that all correlations between v_i and v_j vanish. Again, mathematically this means that the covariance matrix C of B may contain nonzero off-diagonal elements (see, e.g., Equation (6.4)), but the covariance matrix Σ of V must be a diagonal matrix (all off-diagonal elements are zero). Let us continue with our example and use the PCA approach to find the components. We first introduce and apply the method; later we justify the procedure in the context of the above strategy.

Continuing the numerical example above, we will show that the 3×3 covariance matrix C in Equation (6.4) can be diagonalized by applying a linear transformation. To accomplish this, we first define a 3×3 matrix of orthogonal column vectors $U = \begin{bmatrix} U_1 & U_2 & U_3 \end{bmatrix}$ and a 3×3 diagonal matrix Σ with diagonal entries $\lambda_1 - \lambda_3$, and group our demeaned observations B_n in matrix B (Equation (6.3)). We can compute:

$$CU = \begin{bmatrix} CU_1 & CU_2 & CU_3 \end{bmatrix} \tag{6.5a}$$

and

$$U\Sigma = \begin{bmatrix} U_1 & U_2 & U_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} \lambda_1 U_1 & \lambda_2 U_2 & \lambda_3 U_3 \end{bmatrix} \tag{6.5b}$$

Note that Σ is a diagonal matrix. Now let us assume that our covariance matrix C is diagonalizable such that:

$$C = U\Sigma U^{-1} \quad \text{and} \quad \Sigma = U^{-1} CU \tag{6.5c}$$

(note that we also assumed that U is invertible). If we right-multiply the first expression in Equation (6.5c) by U we get:

$$CU = U\Sigma \tag{6.5d}$$

This result indicates that if C is diagonalizable, then the expressions in Equations (6.5a) and (6.5b) must be equal. If we equate the individual columns in the matrices in Equation (6.5d), we get:

$$CU_1 = \lambda_1 U_1, \quad CU_2 = \lambda_2 U_2, \quad \text{and} \quad CU_3 = \lambda_3 U_3 \tag{6.5e}$$

The result in Equation (6.5e) shows that $\lambda_1 - \lambda_3$ and $U_1 - U_3$ must be the eigenvalues and corresponding eigenvectors of the covariance matrix C. See Appendix 6.1 if you need to review the concept of eigenvalues and eigenvectors; if you need

more than a quick review, see a text on linear algebra such as the first part of Jordan and Smith (1997) or Lay (1997).

Because C is a symmetric matrix, its eigenvectors are orthogonal vectors. We can show this property of symmetric matrices by considering a simple 2D case where we have two distinct eigenvalues (λ_1 and λ_2) with two corresponding eigenvectors (U_1 and U_2). To show that these vectors are orthogonal, we show that their scalar product equals zero.

> *Note*: Recall that the inner product (also called scalar product or dot product) of two vectors $\vec{a}$ and $\vec{b}$ is given by $ab\ cos\ \phi$, where a and b are the lengths of the vectors and ϕ is the angle between them. If the vectors are orthogonal, ϕ equals 90° and the outcome of the dot product is zero.

We can show that the dot product $U_1 \cdot U_2 = 0$ by computing the following expression:

$$\lambda_1 U_1 \cdot U_2 = (\lambda_1 U_1)^{\mathrm{T}} U_2 = (CU_1)^{\mathrm{T}} U_2 = U_1^{\mathrm{T}} C^{\mathrm{T}} U_2 \tag{6.6a}$$

Here we changed the vector dot product notation into vector notation $U_1 \cdot U_2 = U_1{}^{\mathrm{T}} U_2$ (note the presence of the dot in the far-left expression), and we used the definition of the eigenvalue/eigenvector of C: $\lambda_1 U_1 = CU_1$ (Appendix 6.1). We know that C is a covariance matrix that must be symmetric; therefore, $C = C^{\mathrm{T}}$. Using this property for symmetric matrices, we get:

$$U_1^{\mathrm{T}} C^{\mathrm{T}} U_2 = U_1^{\mathrm{T}} (CU_2) = U_1^{\mathrm{T}} (\lambda_2 U_2) = \lambda_2 U_1^{\mathrm{T}} U_2 = \lambda_2 U_1 \cdot U_2 \tag{6.6b}$$

Note the dot in the last expression. Combining Equations (6.6a) and (6.6b), we may conclude that for the symmetric covariance matrix:

$$\lambda_1 U_1 \cdot U_2 = \lambda_2 U_1 \cdot U_2 \rightarrow (\lambda_1 - \lambda_2) U_1 \cdot U_2 = 0 \tag{6.6c}$$

Because we deal with two distinct eigenvalues, we know that $(\lambda_1 - \lambda_2) \neq 0$ and therefore the scalar product $U_1 \cdot U_2 = 0$, indicating that the angle between the two eigenvectors of a symmetric matrix must be 90°. Thus the two vectors are orthogonal (perpendicular):

$$U_1 \perp U_2 \tag{6.6d}$$

So if we need an orthogonal matrix, we can use the orthogonal eigenvectors of the covariance matrix to create the matrix U to transform the observed demeaned data.

Let us apply the results from the above paragraphs to our numerical example (Equations (6.2)–(6.4)). First we must find a 3×3 matrix of orthogonal eigenvectors vectors $U = \begin{bmatrix} U_1 & U_2 & U_3 \end{bmatrix}$ to transform the demeaned data—that is, $B = UV$.

Matrix V contains the transformed vectors $V_1 - V_4$. This means that for each demeaned observation B_n we want to identify an orthogonal change of variable V_n such that:

$$B_n = UV_n \rightarrow \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}_n = \begin{bmatrix} U_1 & U_2 & U_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_n \tag{6.7}$$

Recall that in the above $U_1 - U_3$ are column vectors so that U is a 3×3 matrix $u_{i,j}$, that is, $b_1 = u_{1,1} \times v_1 + u_{1,2} \times v_2 + u_{1,3} \times v_3$, etc. Assuming again that U is invertible, we can write the relationship in Equation (6.7) as $V_n = U^{-1}B_n$. Since U is an orthogonal matrix, its inverse is equal to its transpose (see a linear algebra text such as Lay, 1997, if you need to review this), so we may write $U^{-1}B_n = U^{\mathrm{T}}B_n$. Recalling how we computed the covariance matrix C from B and its transpose (Equation (6.4)), we can now find the covariance matrix Σ for V:

$$\begin{aligned} \Sigma &= \frac{1}{N-1} VV^{\mathrm{T}} = \frac{1}{N-1} (U^{\mathrm{T}}B)(U^{\mathrm{T}}B)^{\mathrm{T}} \quad \text{since} \quad V = U^{\mathrm{T}}B \\ &= \frac{1}{N-1} U^{\mathrm{T}}BB^{\mathrm{T}}U \qquad\qquad \text{since } (U^{\mathrm{T}}B)^{\mathrm{T}} = B^{\mathrm{T}}U \\ &= U^{\mathrm{T}} \underbrace{\frac{1}{N-1} BB^{\mathrm{T}}}_{C\text{: Equation (6.4)}} U \qquad U = U^{\mathrm{T}}CU \end{aligned} \tag{6.8a}$$

So the orthogonal matrix U can relate C to Σ:

$$\Sigma = U^{\mathrm{T}}CU = U^{-1}CU \tag{6.8b}$$

In the above we used again $U^{-1} = U^{\mathrm{T}}$ to obtain a result for Σ that is the same as the second expression in Equation (6.5c). Thus the covariance matrix for transformed observations V_n is the diagonal matrix Σ. Because the off-diagonal elements (the covariance values) of Σ are zero, v_1, v_2, and v_3 of the transformed observations are uncorrelated. The diagonal elements of Σ, eigenvalues $\lambda_1 - \lambda_3$, are the variance values for the transformed observations $v_1 - v_3$. Convention for PCA is that the eigenvalues and associated eigenvectors are sorted from high to low eigenvalues (variance).

6.3.2 A MATLAB Example

If we compute the eigenvalues and eigenvectors for covariance matrix C, we can transform our demeaned observations depicted in Fig. 6.3B. In MATLAB this can be easily accomplished with the `eig` command—that is, `[UU,SIGMA] = eig(C)`. In our example, we obtain three eigenvectors that form a rotated set of axes relative to the translated axes in Fig. 6.3B. This is because the eigenvectors are orthogonal (i.e., perpendicular) (Equation (6.6d)). If we arrange the eigenvectors according to

the magnitude of their associated eigenvalues (variance), we get the first, second, and third principal components (note then that the first principal component is along the direction of greatest variance). In Fig. 6.3C the first component is indicated by a double arrow (red) and the remaining two components by lines (black); in this example it is easy to see that the first component is in the direction of maximal variance. The covariance matrix C and its eigenvectors and eigenvalues (grouped in Σ and sorted for the eigenvalues in **descending** order) are:

$$C = \begin{bmatrix} 1.3333 & 0 & -1.3333 \\ 0 & 1.3333 & -0.6667 \\ -1.3333 & -0.6667 & 4.6667 \end{bmatrix}$$

(See also Equation (6.4))

$$U = \begin{bmatrix} -0.3192 & 0.4472 & 0.8355 \\ -0.1596 & -0.8944 & 0.4178 \\ 0.9342 & 0 & 0.3568 \end{bmatrix}$$

and

$$\Sigma = \begin{bmatrix} 5.2361 & 0 & 0 \\ 0 & 1.3333 & 0 \\ 0 & 0 & 0.7639 \end{bmatrix}$$

Note: If you do this example in MATLAB, the `eig` command sorts the eigenvalues from low to high. This is in **ascending** order, which is contrary to convention for PCA. Therefore, the order of the diagonal entries in SIGMA and Σ and the order of the associated eigenvectors (columns) in UU and U are reversed.

Suppose we want to find the coordinates of our observations S_1–S_4 on the translated-and-rotated set of axes (Fig. 6.3C)—in other words, the projections of the observations on the eigenvectors. Let us look into our numerical example how this can be accomplished by computing the projection of the first observation on the first principal component. First, we take point B_1 (corresponding to a demeaned version of the first observation S_1 in Equation (6.2a) and depicted in Fig. 6.3B), which is the first column of B in Equation (6.3):

$$B_1 = \begin{bmatrix} 1 \\ -1 \\ -2 \end{bmatrix}$$

Now let us take the first eigenvector, which is the first column of matrix U:

$$U_1 = \begin{bmatrix} -0.3192 \\ -0.1596 \\ 0.9342 \end{bmatrix}$$

The projection of the first point (black in Fig. 6.3) on this eigenvector can be determined by the scalar product of the two vectors:

$$B_1 \cdot U_1 = B_1^{\mathrm{T}} U_1 = [\,1 \quad -1 \quad -2\,] \begin{bmatrix} -0.3192 \\ -0.1596 \\ 0.9342 \end{bmatrix} = -2.0279$$

The above can easily be checked in MATLAB after running the example program Pr6_1.m (available on http://www.elsevierdirect.com/companions/9780123849151). Use B(:,1) and U(:,1) for B_1 and U_1 respectively; the scalar product can be computed with B(:,1)'*U(:,1) (note the ' for transposing B(:,1)). The outcome is −2.0279, the projection of the first point on the first eigenvector. The projection of the first point on the second and third eigenvectors will be scalar products $B_1 \cdot U_2$ and $B_1 \cdot U_3$ (note the dots). For the second point B_2 (red in Fig. 6.3) we can repeat the procedure: $B_2 \cdot U_1$, $B_2 \cdot U_2$, and $B_2 \cdot U_3$. The same, of course, is true for the third (blue, Fig. 6.3) and the fourth (green, Fig. 6.3) points. We can compute all the scalar products V at once with the matrix multiplication $B^{\mathrm{T}}U$. This will generate the coordinates of all four points on the three eigenvectors. The results for our numerical example are summarized in Table 6.1.

Table 6.1 Principal Component Analysis: Numerical Example

$S =$	$[S_1$	S_2	S_3	$S_4]$	**Original observations**
s_1	4.0000	2.0000	2.0000	4.0000	Fig. 6.3A
s_2	1.0000	3.0000	1.0000	3.0000	
s_3	2.0000	3.0000	7.0000	4.0000	
$B =$	$[B_1$	B_2	B_3	$B_4]$	**Demeaned observations**
b_1	1.0000	−1.0000	−1.0000	1.0000	Fig. 6.3B
b_2	−1.0000	1.0000	−1.0000	1.0000	
b_3	−2.0000	−1.0000	3.0000	0	
$V =$	$[V_1$	V_2	V_3	$V_4]$	**Projections on eigenvectors**
v_1	**−2.0279**	−0.7746	**3.2812**	−0.4787	Fig. 6.3C
v_2	1.3416	−1.3416	0.4472	−0.4472	
v_3	−0.2959	−0.7746	−0.1829	1.2533	

Summary of PCA on four observations S_1–S_4. These data points are plotted in Fig. 6.3A. First the data are demeaned in B_1–B_4 so that a new set of axes with its origin in the point of gravity of all points is obtained (Fig. 6.3B). Finally the axes are rotated using the PCA (Fig. 6.3C). Note that the first component axis (double arrow, red in Fig. 6.3C) indicates the direction of largest variance, easily appreciated when looking at the position of the first (V_1, black) and third (V_3, blue) points in Fig. 6.3C. For clarity, these extreme values for the first component v_1 are indicated in bold in the table (v_1 in vectors V_1 and V_3).

Note: In some texts the projection on the first eigenvector (row v_1 in Table 6.1) is indicated as the first principal component, the projections v_2 on the second eigenvector is then the second principal component, etc. To summarize, depending on the text, the principal components can be the eigenvectors U_1-U_3 or the projections of the observations on these vectors v_1-v_3, and in some texts the term "principal component" is used for both.

The variance in each direction (i.e., for each component v_1, v_2, and v_3) is easily calculated in MATLAB after running the program `Pr6_1.m` with the `std` command: `std(V').^2`. The outcome of this calculation is 5.2361, 1.3333, 0.7639; as expected, these values correspond to the eigenvalues in Σ. Because the origin of the axes in Fig. 6.3C is the same as in panel B, the mean of the components v_1-v_3 remains zero (`mean(V')`). Further we can test for zero covariance—that is, testing that the off-diagonal entries of the covariance matrix `(1/3)*V*V'` are indeed zero. The result is:

5.2361	0.0000	−0.0000
0.0000	1.3333	0.0000
−0.0000	0.0000	0.7639

The outcome is as expected: the diagonal elements are again the variances for v_1-v_3 and all covariance values are zero.

6.3.3 Singular Value Decomposition

A common technique to compute the eigenvalues and eigenvectors of the covariance matrix directly from the demeaned observations is singular value decomposition. This technique is based on the fact that any rectangular matrix, such as the demeaned observation matrix B, can be decomposed as:

$$B = U\Theta W^{\mathrm{T}} \tag{6.9}$$

Note that this expression looks similar to the first expression in Equation (6.5c). In Equation (6.9) U and W are orthogonal matrices, and Θ is a matrix that includes a matrix Σ for which the diagonal entries are the so-called singular values σ_1, σ_2, σ_3, …, σ_r. In our numerical example above where B is a 3×4 matrix (Equation (6.3)), U is a 3×3 matrix of eigenvectors, W is a 4×4 matrix of eigenvectors, and Θ is the same size as B, a 3×4 matrix in which the first 3×3 diagonal entries are the singular values $\sigma_1-\sigma_3$. In this example:

$$B = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -2 & -1 & 3 & 0 \end{bmatrix} \quad U = \begin{bmatrix} -0.3192 & 0.4472 & 0.8355 \\ -0.1596 & -0.8944 & 0.4178 \\ 0.9342 & -0.0000 & 0.3568 \end{bmatrix}$$

$$\Theta = \begin{bmatrix} 3.9634 & 0 & 0 & 0 \\ 0 & 2.0000 & 0 & 0 \\ 0 & 0 & 1.5139 & 0 \end{bmatrix}$$

$$W = \begin{bmatrix} -0.5117 & 0.6708 & -0.1954 & 0.5000 \\ -0.1954 & -0.6708 & -0.5117 & 0.5000 \\ 0.8279 & 0.2236 & -0.1208 & 0.5000 \\ -0.1208 & -0.2236 & 0.8279 & 0.5000 \end{bmatrix}$$

While the eigenvectors (columns of U) we find here correspond with those found for the covariance matrix above, you may be surprised that the singular values in Θ do not correspond with those in Σ above. This is because (unlike the eigenvalues $\lambda_1 - \lambda_3$ of the covariance matrix C) the singular values σ_i are the standard deviations and not the variance. Furthermore, the singular values are based on BB^{T} while the eigenvalues λ_i are based on the normalized version: BB^{T} divided by $(N-1)$. So if we compute $\Theta\Theta^{\mathrm{T}}$ and divide by $N-1=3$, we get the same values as the diagonal entries in Σ:

$$\Sigma = \frac{\Theta\Theta^{\mathrm{T}}}{N-1} = \frac{\Theta\Theta^{\mathrm{T}}}{3} = \begin{bmatrix} 5.2361 & 0 & 0 \\ 0 & 1.3333 & 0 \\ 0 & 0 & 0.7639 \end{bmatrix}$$

This result is identical to the values we obtained for covariance matrix Σ we obtained earlier. If we use Equation (6.9) to compute BB^{T}:

$$BB^{\mathrm{T}} = (U\Theta W^{\mathrm{T}})(U\Theta W^{\mathrm{T}})^{\mathrm{T}} = (U\Theta W^{\mathrm{T}})(W\Theta^{\mathrm{T}}U^{\mathrm{T}}) = U\Theta \underbrace{W^{\mathrm{T}}W}_{I} \Theta^{\mathrm{T}}U^{\mathrm{T}}$$

$$= U \underbrace{\Theta\Theta^{\mathrm{T}}}_{\Sigma''} U^{\mathrm{T}} = U\Sigma'' U^{\mathrm{T}} \tag{6.10}$$

In the above we used $W^{\mathrm{TT}} = W$. Since W is orthogonal, $W^{\mathrm{T}} = W^{-1}$, so we may state that $W^{\mathrm{T}}W = I$, where I is the identity matrix. Finally, because Θ's only nonzero entries are on the diagonal, we may state: $\Theta\Theta^{\mathrm{T}} = (N-1)\Sigma = \Sigma''$. Recalling that BB^{T} divided by $(N-1)$ is the covariance matrix C, the outcome of Equation (6.10) is (with the exception of the normalization $1/(N-1)$, reflected by the use of Σ'' instead of Σ) the same as the left expression in Equation (6.5c). Restating this here for convenience: $C = U\,\Sigma\,U^{-1}$ (recall that $U^{\mathrm{T}} = U^{-1}$ because U is an orthogonal matrix).

We can use standard MATLAB functions to compute the eigenvalues and eigenvectors from the covariance matrix using the **eig** ***command, or directly from the demeaned***

observations using singular value decomposition with the **svd** ***command. A part of Pr6_1.m (available on http://www.elsevierdirect.com/companions/9780123849151) shows the use of these commands.***

```
% Two Methods to Perform PCA using MATLAB standard functions eig and svd
% 1. Eigenvalues and Eigenvectors (eig) of Covariance Matrix C
% [=(1/(N-1)*B*B′]
[ei_vectors1,ei_values1]=eig(C)
['NOTE that the eigenvalues above are sorted in ASCENDING order']

% 2. Singular Value Decomposition (svd) of DEMEANED Observation Matrix B
[ei_vectors2,singular_values,vv]=svd(B)
['NOTE that the eigenvalues above are sorted in DESCENDING order']
% IMPORTANT NOTE
% singular_values is the sqrt of the eigenvalues of the non-normalized
% covariance B*B′ [i.e. sqrt(eig(B*B′))]
```

As a final note, you can see that the PCA would do a bad job distinguishing source signals from a mixture, since PCA separates components based purely on variance. In our computations above, our S matrix was the same as the measured signals Y in the example of Fig. 6.2 in Section 6.2. However, the temporal sequences in the decomposed results in V (Table 6.1) do not even come close to unmixing the source signals X in that example. In the following section we will see how PCA can be used with more success as a tool to separate signal(s) from noise.

6.3.4 Using PCA as a Filter

The PCA technique detects uncorrelated components with decreasing variance. One application that uses this property is to remove noisy components from mixtures of signals. The reasoning for this application is that signal components should display high variance, while the added noise components have smaller variance. Of course, the truth of this assumption depends on the type of signal and may not always be valid.

In the MATLAB script Pr6_2.m (available on http://www.elsevierdirect.com/companions/9780123849151) we explore this technique by purposely corrupting an image (Lena) with random noise to examine how well we can clean up the mess using PCA. Subsequently we use singular value decomposition to define the principal components. The program displays a series of 30 figures each with four panels: the original image, its noisy contaminated version, the image reflecting the nth principal component, and the image reflecting the sum of components 1 to n. It can be seen that the PCA cannot retrieve the original image, but it certainly can improve the noisy contaminated version. At some point (around component 10−15 in this example) the higher components do not seem to further improve image quality in the sum of the components 1 to n. This is due to the fact that the higher components indeed contain more of the noisy aspect of the corrupted image, thus decreasing corrupted image quality when added to the sum of components.

6.4 Independent Component Analysis

In the previous section we introduced PCA, a technique to decompose multichannel data into uncorrelated components. If we use PCA to decompose x and y into variables a and b, we have shown that the covariance between decomposed variables a and b is zero and the covariance matrix of these transformed observations is entirely determined by variance. ICA moves beyond the constraint of decorrelation and looks for components that are statistically independent. When two signals a and b are statistically independent, they are each drawn from an independent probability density function (pdf), and the joint pdf of $[a\ b]$ is simply the product of the individual pdfs:

$$p_{ab}([ab]) = p_a(a)p_b(b) \tag{6.11}$$

The joint and individual pdfs are symbolized by p_{ab}, p_a, and p_b, respectively. Suppose we have two processes a and b, with probability distributions $p_a = \begin{bmatrix} 0.1 & 0.2 & 0.2 & 0.4 & 0.1 \end{bmatrix}$ and $p_b = [0.1 \quad 0.3 \quad 0.3 \quad 0.2 \quad 0.1]$; then, if they are independent, we may use Equation (6.11) to compute the joint probability p_{ab} (Fig. 6.4). Note that the probability functions in Fig. 6.4 all add up to 1 ($\sum_{i=1}^{5} p_{a_i} = 1$, $\sum_{i=1}^{5} p_{b_i} = 1$, and $\sum_{i=1}^{5}\sum_{j=1}^{5} p_{a_i b_j} = 1$). Statistical independence between a and b means that all the moments and central moments (moments about the mean) of the distributions for a and b must also be independent:

$$E\{a^p b^q\} = E\{a^p\}E\{b^q\} \tag{6.12a}$$

Here $E\{\ldots\}$ denotes the Expectation (see section 3.2 in van Drongelen, 2007, if you need to refresh your knowledge about Expectation). If a and b are demeaned, the first central moment (exponents $p = 1$ and $q = 1$ in Equation (6.12a)) of the

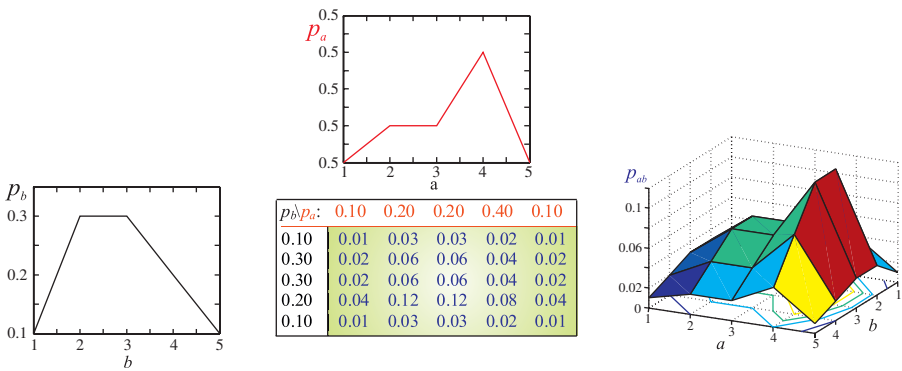

Figure 6.4 An example of a 2D joint probability density function p_{ab} (in the green panel) and its marginal distributions p_a and p_b. The graphs show the individual, marginal distributions of a (top graph) and b (left graph). In the 3D graph on the right, the joint probability is plotted on the vertical axis against the variables a and b.

joint pdf $E\{a\ b\}$ is the covariance. If a and b are uncorrelated (as in the decomposed result from PCA), we have:

$$E\{ab\} = \underbrace{E\{a\}}_{0}\underbrace{E\{b\}}_{0} = 0 \tag{6.12b}$$

You can see that demanding that a and b are uncorrelated (Equation (6.12b)) is not as strong a condition as asking for statistical independence of a and b (Equation (6.12a)). There is an exception when PCA does generate statistically independent components: this is the case when the extracted signals are normally distributed. Normally distributed signals are determined by their first two moments; once these are known, all higher-order moments are determined. **To summarize: two signals that are statistically independent are also uncorrelated, but uncorrelated signals are not statistically independent except when the signals are normally distributed.**

In real cases, signal mixtures tend to be normally distributed due to the central limit theorem. To put it informally, the central limit theorem states that the sum or a mixture (=weighted sum) of multiple variables tends to be normally distributed even when individual components are not drawn from a normal distribution. An example of this theorem at work is shown in Fig. 6.5. In this example we study a mixture of variables that are each uniformly distributed (Fig. 6.5A). Interestingly, the mixture of only five such variables already shows a tendency toward a normal distribution (Fig. 6.5B). This shows that the practical application of PCA for extracting statistically independent components will be fairly limited because (unlike signal mixtures) it is less likely that individual signal components are normally distributed. Because ICA demands statistical independence of the individual components, ICA is much better at extracting components that are not normally distributed. To visualize the difference in component distributions, we can look at the distribution of observations from a uniform distribution (Fig. 6.5A), where we observe that the points are scattered more or less evenly over a line in the 1D case (Fig. 6.5C) or a plane in the 2D case (Fig. 6.5E). On the other hand, normally distributed (Gaussian) mixtures are concentrated around the mean value of the distribution (Fig. 6.5B, D, and F).

6.4.1 Entropy of Sources and Mixtures

Recall that the entropy $S(X)$ of a random variable X (see van Drongelen, 2007, chapter 14, section 14.3) depends on the probability distribution of X. It can be defined as the sum (in the case of a discrete variable) or the integral (in case of a continuous variable) of the product $p(x)\log_2 1/(p(x)) = -p(x)\log_2 p(x)$ over all x:

$$S(X) = -\sum_{\text{All } x} p(x)\log_2\ p(x) \tag{6.13}$$

In this case we defined S for a discrete variable and we use $\log_2$, a base 2 logarithm, so that S is in units of bits.

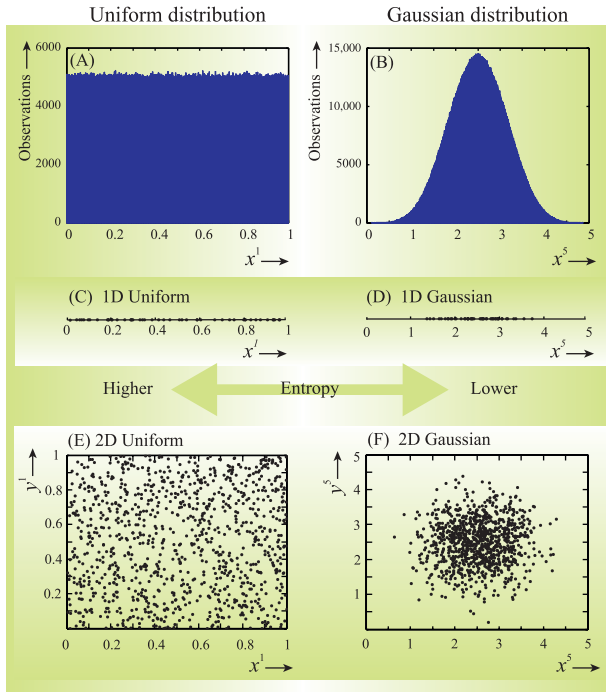

Figure 6.5 (A) Histogram of a variable x^1 that is uniformly distributed between 0 and 1. (B) The sum of only five of these uniformly distributed variables x^5 tends to be almost normally distributed. (A) and (B) were made with script Pr6_3.m (available on http://www.elsevierdirect.com/companions/9780123849151). A series of one-dimensional observations from uniform (x^1) and (almost) Gaussian (x^5) distributions are shown in (C) and (D), respectively. The scatter plots in (E) and (F) are examples of a series of 2D observations: two variables x^1 y^1 for the uniform case, and two variables x^5 y^5 for the Gaussian one. As expected, the uniform distribution results in a scatter of points throughout the plane, whereas the Gaussian case shows a concentration of points around a center (the mean). Consequently, the entropy of the uniformly distributed points is higher than the entropy for the Gaussian distributed observations.

Let us consider a very simple case, a coin toss. If we have the usual situation, we have probability $p = \frac{1}{2}$ for both outcomes heads and tails (scenario III, Fig. 6.6A), and the entropy according to Equation (6.13) is:

$$S(X) = -\left[\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right] = 1 \text{ bit}$$

This is a reasonable result, because we have an outcome that fits in a single bit: either heads (1) or tails (0). Now suppose we have a "faulty" (deterministic) coin that always lands on one side, either heads or tails. In these scenarios (I and V, Fig. 6.6A) we have $p = 0$ for one outcome and $p = 1$ for the other; now the entropy is:

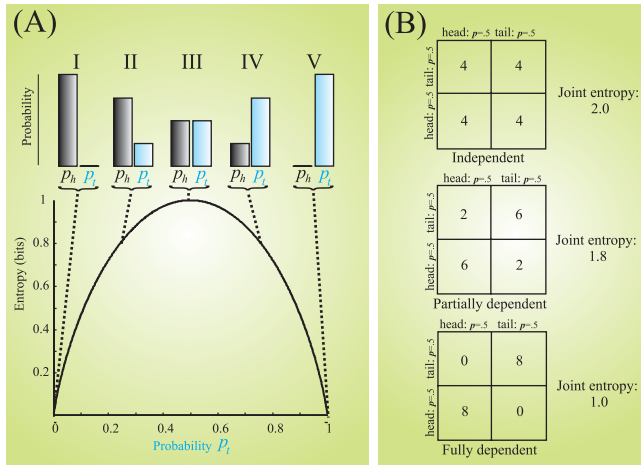

Figure 6.6 Statistics of a coin toss and entropy. (A) Five scenarios of probability distributions of heads (p_h) and tails (p_t). The graph depicts that each scenario is associated with a specific entropy value. (B) Statistics of coin tosses. The numbers in the tables show the idealized outcomes for 16 tosses. Two coins are used for each observation and in the upper diagram the outcomes of each toss is completely independent. In the two lower diagrams, there (magically) is some dependence between the two tosses in each observation—there is either a full dependence (bottom diagram; the pair of outcomes in each observation are identical) or a partial dependence (middle diagram).

$$S(X) = -\left[0 \log_2 0 + 1 \log_2 1\right] = 0 \text{ bit}$$

Note that we use $0 \log_2 0 = 0$. This outcome of zero bit also seems reasonable since there is no surprise (information) with each outcome: it will always be heads in one scenario and always tails in the other. If our coin is biased and we get heads or tails in 75% of the cases (scenarios II and IV in Fig. 6.6A), we have probabilities $p = 0.75$ and $p = 0.25$ and the associated entropy is:

$$S(X) = -\left[\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right] = 0.81 \text{ bit}$$

Thus, for every probability distribution, we find a specific entropy (see the graph in Fig. 6.6A). We find that the maximum entropy for tossing a coin is when probability is equal ($\frac{1}{2}$) for heads and tails, which occurs when the probability distribution is uniform (scenario III, Fig. 6.6A).

Without further proof, we state that the above result may be generalized to any probability distribution: variables show maximum entropy when they are uniformly distributed. Thus, in the example in Fig. 6.5, the variable in the left panel with a uniform distribution has higher entropy than the variable with a Gaussian distribution.

In panels E and F in Fig. 6.5, we consider a 2D distribution where the observations are represented by dots in a plane. In this example we have a joint distribution

similar to the one for variables a and b shown in Fig. 6.4 (where each random variable has five possible outcomes). To compute the entropy associated with such a joint probability distribution, we follow the same approach as for the 1D case: we summate $-p(x)\log_2 p(x)$ over the domain of x, which leads us to actually have two sums:

$$S(a,b) = -\sum_{i=1}^{5}\sum_{j=1}^{5} p_{a_i b_j}\log_2 p_{a_i b_j}$$

The entropy $S(a,b)$ of the joint distribution (the table in Fig. 6.4) is:

$$-[0.01\log_2 0.01 + 0.03\log_2 0.03 + 0.03\log_2 0.03 + \cdots + 0.03\log_2 0.03 + 0.02\log_2 0.02 + 0.01\log_2 0.01] = 4.29 \text{ bits}$$

The entropy for the individual variables a and b can be obtained from the marginal distributions (i.e., the five probabilities corresponding to each outcome, given one random variable). Using the distribution for a (see marginal distribution [red] in Fig. 6.4), we find that entropy $S(a)$ is:

$$-[0.1\log_2 0.1 + 0.2\log_2 0.2 + 0.2\log_2 0.2 + 0.4\log_2 0.4 + 0.1\log_2 0.1] = 2.12 \text{ bits}$$

and for b (marginal distribution [black] in Fig. 6.4) we find that $S(b)$ is:

$$-[0.1\log_2 0.1 + 0.3\log_2 0.3 + 0.3\log_2 0.3 + 0.2\log_2 0.2 + 0.1\log_2 0.1] = 2.17 \text{ bits}$$

Now we see that $S(a) + S(b) = 2.12 + 2.17 = 4.29$, which is equal to $S(a,b)$. This is not so surprising because the probability distributions for a and b were independent such that $p_{ab}([ab]) = p_a(a)p_b(b)$. If our distribution in Fig. 6.4 had been uniform, we would have found different values for the entropies. If this were the case, the five probabilities of p_a and p_b would be $\begin{bmatrix}0.2 & 0.2 & 0.2 & 0.2 & 0.2\end{bmatrix}$ and the joint distribution would also be uniform, with all 25 probabilities equal to 0.04. The associated entropies would now be:

$$S(a,b) = -25 \times 0.04 \times \log_2(0.04) = 4.64 \text{ bits}$$

and

$S(a)$ and $S(b)$ are both $-5 \times 0.2 \times \log_2(0.2) = 2.32$ bits

In all cases the entropies are higher (because of the uniform distribution), but,

$$S(a) + S(b) = S(a,b) \tag{6.14a}$$

still holds.

If there were dependence between the two distributions of a and b, we would have found a different result. Let us explore the effect of dependence with an even simpler example by getting back to our coin toss. Assuming that we toss two coins 16 times, and in one case we have the usual situation where the tosses are independent (the idealized outcome is shown in Fig. 6.6B, top diagram). However, in the other case there is a "magical" full dependence between the two coins: if one coin lands on heads or tails, the other coin will too (the idealized outcome is shown in Fig. 6.6B, bottom diagram).

When tossing two coins, we have four alternative outcomes: heads/heads, heads/tails, tails/heads, and tails/tails. Assuming we have equal probability for heads and tails, we get in the independent case that the probability for each outcome is $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$. In the fully dependent case, however, the probabilities for heads/tails and tails/heads are zero because one coin will magically copy the outcome of the other (we imagine this just for the sake of this example; do not worry about how you would actually do such a thing). The probabilities for heads/heads and tails/tails are therefore each $\frac{1}{2}$. Regardless of independence or dependence, we can first compute entropies S_1 and S_2 for each individual coin toss from the marginal distributions, finding that S_1 and S_2 are both:

$$-2 \times 0.5 \times \log_2(0.5) = 1 \text{ bit}$$

As was the case for $S(X)$ computed earlier in this chapter, it makes sense that the entropies S_1 and S_2 should be equal to 1 bit, since the outcome of each individual coin toss can fit in 1 bit (0 for heads, 1 for tails).

In the independent case (top diagram in Fig. 6.6B), we find for the joint entropy, $S_{1,2}$,

$$-4 \times 0.25 \times \log_2(0.25) = 2 \text{ bits}$$

Here we see that just as in the case for the independent distribution in Fig. 6.4, the sum of the individual entropies equals the joint entropy:

$$S_1 + S_2 = S_{1,2}$$

We can also see that the result we get for $S_{1,2}$ is reasonable since the possible joint outcomes fill four possible states, or 2 bits.

Now we compute the joint entropies for the two other scenarios in Fig. 6.6B. In the fully dependent case (the bottom diagram in Fig. 6.6B), the joint entropy $S_{1,2}$ is:

$$-\left[2 \times 0.5 \times \log_2(0.5) + 2 \times 0 \times \log_2(0)\right] = 1 \text{ bit}$$

It should be unsurprising that the joint entropy ($S_{1,2}$) in this case is identical to the entropy of an individual coin toss (S_1 or S_2); since there is total dependence, no additional information is provided by the flipping of a second coin.

The (idealized) outcomes for a case with partial dependence between the two coins are shown in the middle diagram in Fig. 6.6B. Note that due to the partial dependence, in most but not all cases, the outcomes of the first and second coin toss are identical. This results in a joint entropy of:

$$-\left[2 \times 0.125 \times \log_2(0.125) + 2 \times 0.375 \times \log_2(0.375)\right] = 1.8 \text{ bits}$$

In both cases where there is (full or partial) dependence between the tosses of the coins, we find that $S_1 + S_2 > S_{1,2}$, and the more dependence that exists between the tosses, the larger the difference between $S_1 + S_2$ and $S_{1,2}$. Accordingly, we need to adapt Equation (6.14a) to account for the possibility of dependence between the two variables by including a term that reflects this dependence. This term is commonly indicated by mutual information (MI), which is an indication of the level of dependence between variables. In other words, MI quantifies the amount of information that variable 1 provides about variable 2. In case of the dependence between coin tosses, the outcome of one coin toss determines the outcome of the other. In the normal, fair tosses the outcome of one toss does not provide any information about the other since they are independent.

To summarize, we find:

	Independent (bit)	**Slightly dependent (bit)**	**Fully dependent (bit)**
Entropy coin 1, S_1	1.0	1.0	1.0
Entropy coin 2, S_2	1.0	1.0	1.0
Sum, $S_1 + S_2$	2.0	2.0	2.0
Joint entropy, $S_{1,2}$	2.0	1.8	1.0
Difference $(S_1 + S_2) - S_{1,2}$ (=MI)	0.0	0.2	1.0

It can be seen in this example that the MI variable is indeed proportional to the level of dependence. Without further proof, we assume that we may generalize our findings and state that for any two random processes X and Y, we can compute the entropy for each of the individual processes $S(X)$ and $S(Y)$ such that their joint entropy $S(X,Y)$ is the sum of the individual entropy values when X and Y are independent (Equation (6.14a)), and otherwise:

$$S(X, Y) = S(X) + S(Y) - \text{MI}(X, Y) \tag{6.14b}$$

in which MI(X,Y) is the mutual information between X and Y. Note that Equation (6.14b) holds even when X and Y are independent, since then MI(X,Y) merely becomes zero. We could define joint entropy $S(X,Y)$ as the total information of the joint process X,Y. To summarize, it can be concluded from the above that **for any given pair of processes X and Y, maximal independence occurs at maximal joint entropy (or joint information) $S(X,Y)$ and minimal mutual information**

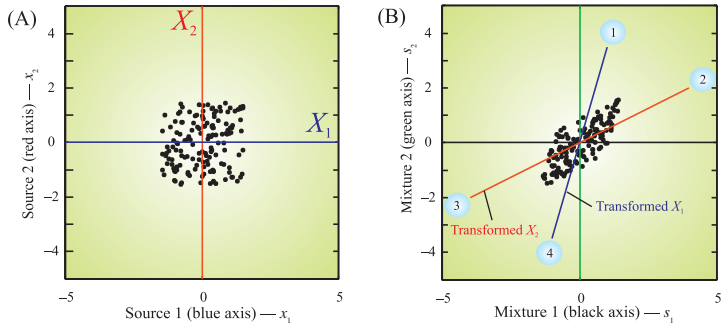

Figure 6.7 (A) Scatter plot of two source signals x_1 and x_2. (B) Scatter plot of two mixtures s_1 and s_2 that were created from the source signals. The transformed source axes (X_1, indicated by 1−4 [dark blue], and X_2, indicated by 2−3 [red]) are indicated in this mixture plot.

MI(X,Y) of the joint process. This is a basis for the ICA technique: independence of separated sources is evaluated by joint entropy (joint information) and MI. For the separation of independent sources, their joint information $S(X,Y)$ must be maximized (therefore, this ICA technique is also called infomax), which is the same as minimizing their mutual information MI(X,Y).

6.4.2 Using the Scalar Product to Find Independent Components

After we obtain the criteria for unmixing a mixture of signals (e.g., decorrelation, statistical independence, maximizing joint entropy), the procedure for separating components from mixtures in ICA is essentially the same as was outlined for PCA in Section 6.2: source signals are found from the product of the unmixing matrix and the recorded signals (Fig. 6.2). The unmixing matrix contains the vectors along which the components are extracted. **The difference between ICA and PCA is the strategy for finding the directions of the vectors in the unmixing matrix. In PCA we found directions of maximal variance (Fig. 6.3C) while the components were decorrelated. For ICA we demand statistical independence.**

To illustrate an ICA-type extraction procedure, let us consider a 2D case: two sources x_1 and x_2 creating two mixtures s_1 and s_2. Scatter plot representations of the sources and the mixtures are shown in Fig. 6.7; the sources are plotted in panel A and the resulting mixtures in panel B. Assuming that we know the mixing matrix A in this example,

$$A = \begin{bmatrix} 0.2 & 0.8 \\ 0.7 & 0.4 \end{bmatrix}$$

we can determine the orientation of the original axes X_1 and X_2 from the source scatter plot (depicted in Fig. 6.7A) in the mixture scatter plot (shown in Fig. 6.7B).

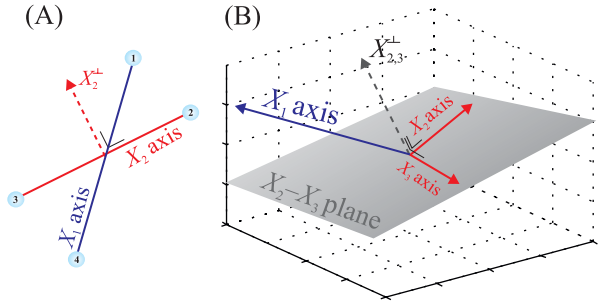

Figure 6.8 (A) and (B) show the strategy for unmixing. In (A) we have a 2D case: if we cancel all components in the direction of axis X_2 (by using the inner product with vector $X_2^{\perp}$ perpendicular to X_2), the remainder must be a component of the X_1 axis. This approach can be extended to higher-dimensional cases (B): by canceling components for X_2 and X_3 (by using the inner product with vector $X_{2,3}^{\perp}$ perpendicular to X_2 and X_3), we keep the ones for X_1.

The first axis $X_1 = [1\ 0]$, so the transformed version of source axis X_1 in the scatter plot of the mixtures is:

$$\underbrace{A}_{\text{Mixing matrix}} \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{X_1} = \begin{bmatrix} 0.2 & 0.8 \\ 0.7 & 0.4 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.2 \times 1 & + & 0.8 \times 0 \\ 0.7 \times 1 & + & 0.4 \times 0 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.7 \end{bmatrix}$$

This is the first column of mixing matrix A. Similarly, the transformed second source axis X_2 in the mixture plot is the second column of A. The axes X_1 and X_2 from the scatter plot in Fig. 6.7A are also depicted, after transformation with mixing matrix A, in Fig. 6.7B. After this transformation, X_1 becomes the axis 1−4 (dark blue) and X_2 becomes axis 2−3 (red).

For the following explanation it helps to look at the plot of the mixtures in Fig. 6.7B and the orientation of axes and vectors in Fig. 6.8A. First we establish that we know there are two sources, and that we have two mixtures. If we know the orientation of axes X_1 and X_2, we can find the contribution of x_1 to the mixtures by excluding all contributions of x_2. Because the contributions of x_2 are in the direction of axis X_2, we can use the scalar product of (1) the observation vectors of the mixture (all points in Fig. 6.7B) and (2) a vector $X_2^{\perp}$ perpendicular to axis X_2 (Fig. 6.8A). All components parallel to X_2, which we will indicate by $X_2^{=}$, will cancel since the inner product $X_2^{\perp} \cdot X_2^{=} = 0$. Therefore, the only component remaining in the scalar product of each observation in the mixture plot $[s_1\ s_2]$ with $X_2^{\perp}$ will be independent of x_2, and thus must be from x_1.

Note: Summarizing the above approach in a few words, **all components independent of X_2, the axis for source x_2, can only be a component of x_1.** Furthermore, we found that **we can use the inner product to remove x_2 components and only keep the ones independent from X_2.**

We can use a similar reasoning for mixtures from three or more sources. Let us consider a three-source and three-mixture case, creating a 3D space (Fig. 6.8B). If we want to find the components for x_1, we need to remove the components for x_2 and x_3. So if we construct a plane through the axes for x_2 and x_3, we can come up with $X_{2,3}^{\perp}$ (Fig. 6.8B). The inner product of observation $[s_1\ s_2\ s_3]$ with $X_{2,3}^{\perp}$ (perpendicular to the $X_2{-}X_3$ plane) removes all components associated with x_2 and x_3, and must therefore be the contribution of x_1. With a higher number of dimensions (that is, with more sources and mixtures), we can always construct a hyperplane through all-but-one selected axis (i.e., the axis of one selected source) and find a vector perpendicular to this hyperplane. This vector (analogous to $X_{2,3}^{\perp}$ in Fig. 6.8B) can then be used to remove the contributions from all directions embedded in the hyperplane (analogous to the $X_2{-}X_3$ plane in Fig. 6.8B) so that the remainder must be the contribution from the selected source.

6.4.3 A MATLAB Example

We have now set the stage for an example with two sources and two mixtures. To extract sources from the mixtures, we will follow the strategy below:

(1) We find that our mixtures are Gaussian-like distributed, but we assume that our source signals have a uniform distribution (Fig. 6.5).
(2) We use entropy to evaluate independence of the sources (Fig. 6.6).
(3) If we know the axes associated with the sources, we know how to extract a component from a mixture by using the inner product (Fig. 6.8).

Now we must deal with the fact that we (pretend that we) do not know the transformed axes for the sources in the scatter plot of the mixture (Fig. 6.7B). We will solve this problem by applying a brute force iterative approach—that is, we systematically evaluate a series of angles for source axes X_1 and X_2 and for each pair of angles we use the inner product to compute the associated sources x_1 and x_2. For every solution of x_1 and x_2 (i.e., for each direction associated with axes X_1 and X_2), we determine the level of independence of x_1 and x_2 (while we assume each satisfies a uniform distribution). We can do this in multiple ways, but for now we will evaluate how independent x_1 and x_2 are by looking at the level of mutual information $\mathrm{MI}(x_1, x_2)$ between x_1 and x_2 (Equation (6.14b)). The more independent x_1 and x_2 are estimated to be, the lower their MI. So by following this brute force procedure, we get a series of angles for the axes X_1 and X_2 each with an associated value for $\mathrm{MI}(x_1, x_2)$. Finally we complete our procedure by selecting the angles for axes X_1 and X_2 that correspond to the minimal value of $\mathrm{MI}(x_1, x_2)$.

The brute force iterative procedure is followed in MATLAB script Pr6_4 (available on http://www.elsevierdirect.com/companions/9780123849151). The following is a snippet from this script showing the iteration loops. Each iteration loop goes through a range of angles: $0{-}2\pi$ rad. For each loop (i.e., for each angle in the brute force search), the entropy (H) and mutual information (MI) are determined using function entropy_2D.m (which must be in the same directory). Due to the large number of loops in the brute force search, running this script may take ∼30 min.

```
% ------------------------------------------------------------------
% rotate the unmixing vector and determine the mutual information of result
% ------------------------------------------------------------------
MI_min=100000000000000;              % set minimum of the
                                     % mutual-information to
                                     % large number
phi_min1=0;phi_min2=0;               % set the angles for axes X1 and X2
                                     % to zero
ct_phi1=0;                           % initialize counter 1
for phi1=0:2*pi/precision:2*pi;      % LOOP for rotating axis X1
   ct_phi1=ct_phi1+1;                % update counter 1
   ct_phi2=0;                        % initialize counter 2
   for phi2=0:2*pi/precision:2*pi;   % LOOP for rotating axis X2
      ct_phi2=ct_phi2+1;             % update counter 2
      v1=[cos(phi1) sin(phi1)];      % unit vector along X1 with angle
                                     % phi1
      ic1=v1*S;                      % unmix mixture S
      ic1=ic1-mean(ic1);sigma=std(ic1); % demean and determine standard
                                     % deviation
      v2=[cos(phi2) sin(phi2)];      % unit vector along X2 with angle
                                     % phi2
      ic2=v2*S;                      % unmix mixture S
      ic2=ic2-mean(ic2);sigma=std(ic2); % demean and determine standard
                                     % deviation

      % Use 2D entropy estimate function entropy_2D to compute mutual
      % information (MI) and entropy (H) as a function of the
      % position of axes X1 (counter for phi1) and X2 (counter for phi2)
      [H(ct_phi1,ct_phi2), MI(ct_phi1,ct_phi2)]=entropy_2D(ic1,ic2);

      if MI(ct_phi1,ct_phi2) < MI_min; % TEST: current MI < current
                                     % minimum of MI ?
         MI_min=MI(ct_phi1,ct_phi2); % if so a new minimum (minimum
                                     % for MI) is found
         imin=ct_phi1;jmin=ct_phi2;  % the indices for the new minimum
                                     % are saved
         phi_min1=phi1;              % and so are the other relevant data
         v_min1=v1;                  % the angles, the vectors &
                                     % components
         ic1_min=ic1;
         phi_min2=phi2;
         v_min2=v2;
         ic2_min=ic2;
```

```
      end;
    end;
end;
```

Running the script Pr6_4.m (available on http://www.elsevierdirect.com/companions/9780123849151; also available in a low-resolution version if you are in a hurry) will show you the source and mixed signals plus their scatter plots. The scatter plot of the mixture generated by the script resembles the detail in Fig. 6.9B. The rhomboid shape of the scattered dots indicates that we are dealing with a mix of uniformly distributed source signals. For a range of combinations of hypothetical directions for axes X_1 (indicated by 1−4 in Fig. 6.9) and X_2 (indicated by 2−3 in Fig. 6.9), the MI is computed.

Note: This is a rather computationally intensive procedure associated with our brute force approach, because we (pretend that) we do not know the directions for X_1 and X_2 and just compute the MI for all directions with a precision of 1° for each axis, resulting in 360^2 combinations. In the low-resolution version of script Pr6_4 we compute MI every 10° for each axis (36^2 combinations), which greatly reduces the run time for the program.

For illustration purposes in Fig. 6.9, we kept the angle for MI associated with axis X_1 at its minimum and plotted the MI associated with axis X_2 for each angle (from 0° to 360°, incremented in steps of 1°) as a single dot and connected the dots with a line. The minima of the MI and the line connecting them are indicated in the detailed plot in Fig. 6.9B. It is obvious that the line between these minima (the double arrow indicated by $X_2^{\perp}$ in Fig. 6.9B) is a very good estimate of the vector perpendicular to axis X_2 (indicated by 2−3). The script Pr6_4 also has the option of computing the principal components; the first principal component is also indicated in Fig. 6.9B (indicated as "PCA: axis-1"; light-blue line). It is obvious that this line is in the direction of maximal variance, but it would not do a good job separating the sources in this example.

6.4.4 What If Sources Are Not Uniformly Distributed?

For the ICA examples so far, we assumed that the sources were characterized by a uniform distribution (e.g., Fig. 6.5A, C, and E) and we used entropy estimates to determine the level of independence (e.g., Fig. 6.6) of the separated candidate sources. So what should we do if we know that our sources are not uniformly distributed—take for example a human voice or a chirp—in a recording of a sound mixture? Such sources usually show a distribution with many values around zero

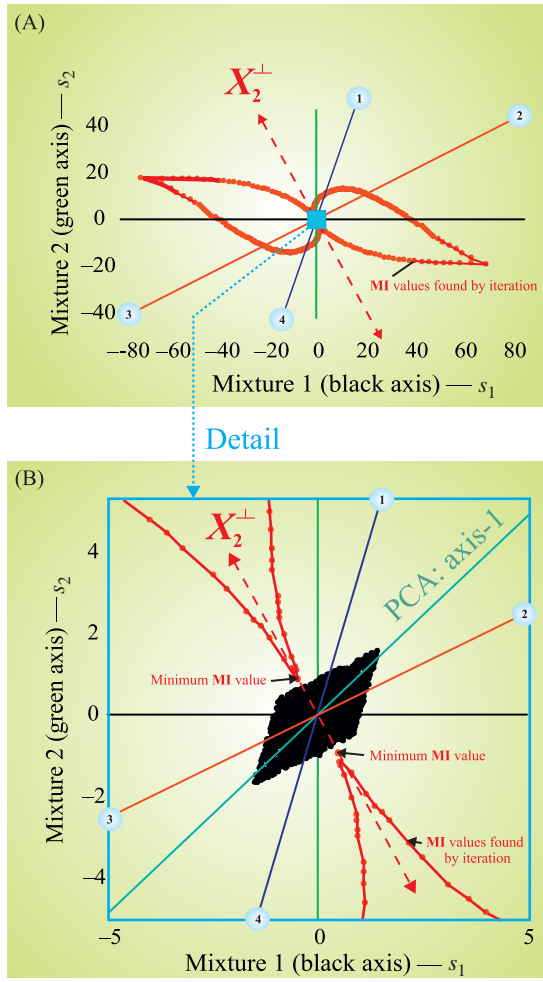

Figure 6.9 An example of an application of the ICA unmixing procedure; (B) is a detail of (A). This example shows a mixture where the sources are uniformly distributed. By iteration, we determine the MI of the candidate sources for each unmixing angle and plot this result (dots [red] in A and B). The best choice of angle is obtained when MI is minimal ("Minimum MI value" in B). Since we know, for this example, the orientation of source axis X_2 (the axis labeled at its ends by 2−3, red), we can see that our estimate for $X_2^{\perp}$ (dashed, double arrow, red) running through the minima we found for MI is indeed perpendicular to X_2. This estimate for $X_2^{\perp}$ will therefore perform a good unmixing operation. The axis labeled "PCA: axis-1" (light blue) in (B) indicates the direction of the eigenvector associated with the largest eigenvalue (the first principal component). As you can see, the principal component is not perpendicular to any of our source axes (1−4 or 2−3) and consequently would not achieve a good unmixing result for our two mixtures. The graph in (B) can be obtained with MATLAB script `Pr6_4.m` (available on http://www.elsevierdirect.com/companions/9780123849151).

(Fig. 6.10). In such a case one could look for another function to maximize or minimize (instead of using entropy or MI). In this example of a peaky distribution (Fig. 6.10B), one could maximize for peakyness of the distribution (kurtosis, a measure for how peaky a pdf is, might do the job in such a case). Alternatively, one could transform the non-uniform distribution into a uniform one and subsequently apply the same procedures that are available for separating sources with a uniform distribution. This approach is depicted in Fig. 6.11. The example in Fig. 6.11 shows the transformation of a Gaussian distribution into a uniform one. The function used for the transformation is the cumulative probability density function (cdf). For a normally distributed variable x with zero mean, the cdf is $\frac{1}{2}\left[1 + \mathrm{erf}(x/\sigma\sqrt{2})\right]$, in which `erf` is the error function (available in MATLAB) and σ is the standard deviation of x. This shows that it is plausible that for any pdf, the cdf is the optimal transformation to obtain a uniform distribution. The cdf will have the steepest slope where the probability is highest (and where you will therefore collect most observations), and this steeper slope will distribute the more densely packed observations over a wider area (see the area in between the [red] dotted lines in Fig. 6.11A–C). In contrast, at low probabilities (where fewer observations occur), the slope of the cdf will be less steep, and consequently the observations will be distributed over a smaller area. The overall effect of the transformation is thus to spread out observations more uniformly, exactly what we want for our purpose. After we transform our unmixed result into a uniform distribution, we can apply exactly the same procedure for separating mixtures (from uniformly distributed sources) we followed earlier.

The approach of transforming the source data is demonstrated in `Pr6_5.m` (available on http://www.elsevierdirect.com/companions/9780123849151). The following part, with the iteration loops (similar to the ones shown above for script `Pr6_4`), shows the transformation from the Gaussian distributed signals `ic1` and `ic2` to the uniformly distributed `T_ic1` and `T_ic2`.

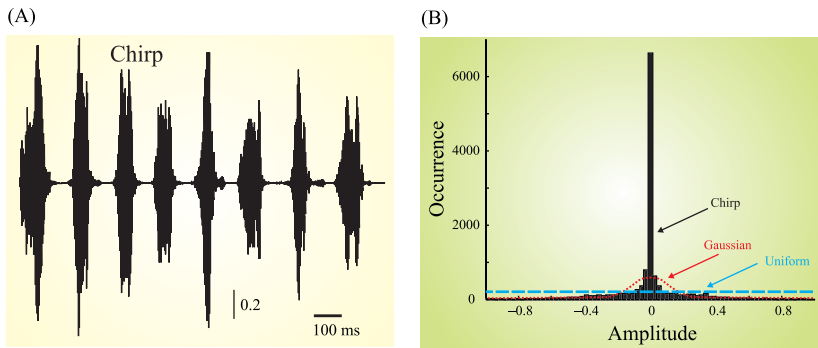

Figure 6.10 (A) Plot of a chirp (available in MATLAB by `load chirp`). (B) The amplitude distribution of the chirp signal compared to Gaussian (red) and uniform (blue) distributions. Note that the histogram of the chirp signal shows a peaky distribution, which is typical for many audio signals.

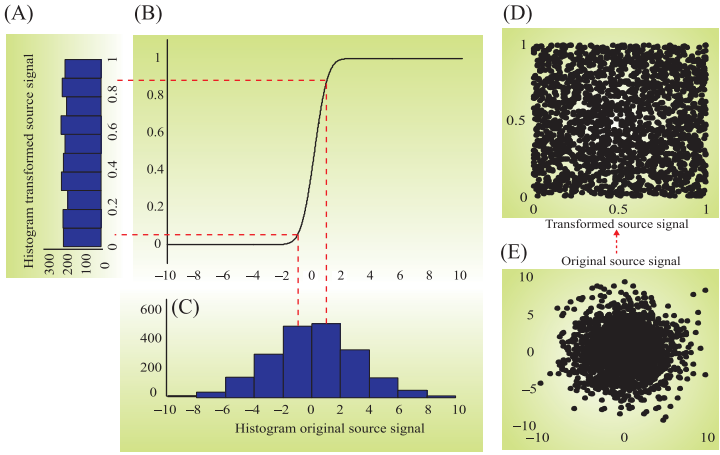

Figure 6.11 A uniform distribution (A) can be obtained from a non-uniform distribution by a transformation. This example shows a transformation of a histogram of observations drawn from a Gaussian distribution (C) using the cumulative probability density of the Gaussian distribution shown in (B). It can be seen that the majority of observations of the Gaussian distribution are located around zero in between the (red) dotted lines. Following these dotted lines from (C) to (A), it can be seen that the transformation of the Gaussian data (C) with the function in (B) distributes these points more evenly (A). Accordingly, if such a transformation is applied to a 2D scatter plot of a Gaussian variable (E), we get a scatter plot of uniformly distributed points (D).

```
% ------------------------------------------------------------------
% rotate the unmixing vector and determine the mutual information of result
% ------------------------------------------------------------------
MI_min=100000000000000;          % set minimum of the
                                 % mutual-information
                                 % to large number
phi_min1=0;phi_min2=0;           % set the angles for X1 and X2 to
                                 % zero
ct_phi1=0;                       % initialize counter 1
for phi1=0:2*pi/precision:2*pi;  % LOOP for X1
   ct_phi1=ct_phi1+1;            % update counter 1
   ct_phi2=0;                    % initialize counter 2
   for phi2=0:2*pi/precision:2*pi; % LOOP for X2
      ct_phi2=ct_phi2+1;         % update counter 2
      v1=[cos(phi1) sin(phi1)];  % unit vector along X1 with
                                 % angle phi1
      ic1=v1*S;                  % unmix mixture S
```

```
    ic1=ic1-mean(ic1);sigma1=std(ic1); % demean and determine standard
                                       % deviation
    v2=[cos(phi2) sin(phi2)];          % unit vector along X2 with angle
                                       % phi2
    ic2=v2*S;                          % unmix mixture S
    ic2=ic2-mean(ic2);sigma2=std(ic2); % demean and determine standard
                                       % deviation
    % TRANSFORMATION to make the estimates uniformly distributed
    % Transform ic1 and ic2 to a uniform distribution using erf to
    % transform the demeaned ic1 and ic2
    T_ic1=0.5*(1+erf((ic1)/(sigma1*sqrt(2))));
    T_ic2=0.5*(1+erf((ic2)/(sigma2*sqrt(2))));

  % Use custom estimate function entropy_2D to compute mutual
  % information (MI) and entropy (H)
  [H(ct_phi1,ct_phi2), MI(ct_phi1,ct_phi2)]=entropy_2D(T_ic1,T_ic2);

  % Sum of the variances should max at independence
  sum_var(ct_phi1,ct_phi2)=std(ic1)^2+std(ic2)^2;

    if MI(ct_phi1,ct_phi2) < MI_min;   % TEST: current MI < current
                                       % minimum of MI ?
      MI_min=MI(ct_phi1,ct_phi2);      % if so a new minimum (maximum
                                       % for MI) is found
      imin=ct_phi1;jmin=ct_phi2;       % the indices for the new minimum
                                       % are saved
      phi_min1=phi1;                   % and so are the other relevant data
      v_min1=v1;                       % the angles, the vectors &
                                       % components
      ic1_min=ic1;
      phi_min2=phi2;
      v_min2=v2;
      ic2_min=ic2;
    end;
  end;
end;
```

Because script Pr6_5.m is also computationally demanding, there is a low-resolution version available as well (both available on http://www.elsevierdirect.com/companions/9780123849151). Running the script will show you the source signals and the mixtures with their associated scatter plots. The scatter plot of the mixtures will resemble Fig. 6.12 and will also show the transformed axes X_1 (dark blue, indicated by 1−4 in Fig. 6.12) and X_2 (red, indicated by 2−3 in Fig. 6.12) plus the estimated MI values associated with each axis (red and dark-blue dots interconnected with lines in Fig. 6.12). The

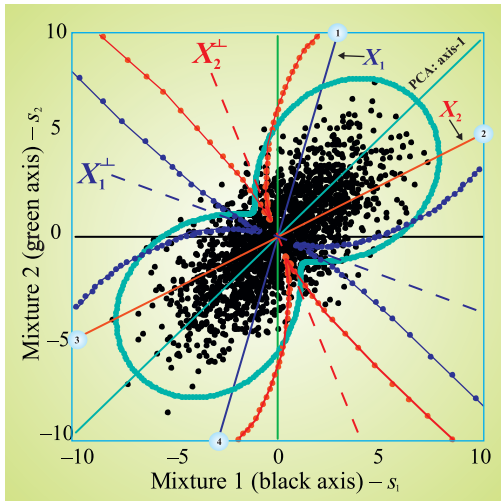

Figure 6.12 The example in this figure is the same as in Fig. 6.9, except now the source signals are normally distributed. As in the previous examples in Figs. 6.7−6.9, the two source axes are X_1 (labeled at its ends by 1−4, dark blue) and X_2 (labeled 2−3, red). The dashed, double arrows $X_1^{\perp}$ (dark blue) and $X_2^{\perp}$ (red) indicate the directions for minimal MI found by iteration. The MI values we computed by iteration are indicated by the dots interconnected by lines: dark blue for $X_1^{\perp}$ and red for $X_2^{\perp}$. The axis labeled "PCA: axis-1" (light blue) is the direction of the eigenvector associated with the largest eigenvalue (the first principal component). The 8-shaped contour (light-blue dots) denotes the variance for each direction. Further details can be found in the text.

program Pr6_5.m also allows us to compute the principal components (the first component, "PCA: axis-1," is indicated in Fig. 6.12) and the variance associated with each angle (the 8-shaped contour, light blue in Fig. 6.12). Note again how the first PCA component is not orthogonal to any of the source axes X_1 and X_2.

The implicit underlying thought of the algorithm in Pr6_5.m is that by transforming the unmixed sources into a uniform distribution, we can follow the same procedure as earlier without affecting the information content. This assumption may seem like a bit of a stretch, but if we transform the data using an invertible function (such as our cumulative probability density function in Fig. 6.11B), we do not affect the mutual independence of the signals (see Stone, 2004).

An invertible function is defined as a function that creates a unique new data point for each original data point and (because the function is invertible) this transformation can also be reversed. This means that if we have several independent data sets, they will remain independent after transformation with the invertible function into the other domain and vice versa.

6.4.5 Can We Apply Smarter Approaches Than the Brute Force Technique?

In the above brute force approach we looked into a 2D case. We determined the source axes X_1 and X_2 with a precision of 1°; this corresponds to 360 computations for each axis. For the 2D case this evaluates to $360^2 = 129{,}600$ iterations. In other words, for each iteration, we guess candidate sources and we compute their MI. For more dimensions and/or higher precisions, the number of iterations grows rapidly—for example, if we wanted a $\frac{1}{2}^\circ$ precision in a six-source case we have $720^6 \approx 1.4 \times 10^{17}$ iterations. As you can see, we need a more efficient procedure to find the best angles for the source axes; otherwise, source extraction very rapidly becomes a computational nightmare. Let us look at the surface of the inverse of the MI. This is the approach we take in MATLAB script Pr6_6.m, which is almost identical to Pr6_4.m, but now we use the inverse of the MI instead of the MI itself for visualization reasons; minima in MI are maxima in 1/MI, and maxima are easier to show in a 3D plot (both scripts are available on http://www.elsevierdirect.com/companions/9780123849151). Such a plot generated by Pr6_6.m for our two-mixtures/two-sources scenario is depicted in Fig. 6.13. The horizontal axes are the angles for axes X_1 and X_2, and the vertical axis is 1/MI. There are clearly eight maxima in the 1/MI landscape (again, corresponding to minima of MI) present in the plot. These eight maxima are not surprising if we consider the example of the source axes in the mixture scatter plot in Fig. 6.7B. Each source axis is labeled at each end (1 and 4 for axis X_1 and 2 and 3 for axis X_2), since each axis can be characterized twice: by its optimal angle ϕ or by the same angle plus

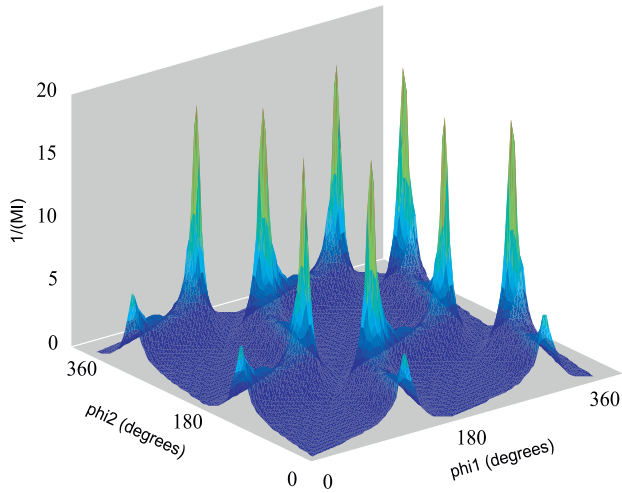

Figure 6.13 The inverse of the MI (1/MI) of two components extracted from two mixtures as a function of the angles (phi1, phi2) of the two source axes. Minima for the MI show up as maxima in 1/MI surface. The eight maxima correspond to a pair of angles of the axes that extract sources with minimum MI from the mixtures. To find these two axes, we need to identify only one of the eight maxima. Therefore, we can use the gradient in the landscape to locate one of the maxima. This procedure is more efficient than a brute force computation of the whole 1/MI surface. This graph was obtained with Pr6_6.m (available on http://www.elsevierdirect.com/companions/9780123849151).

180° ($\phi + 180^\circ$). An optimal combination (associated with a maximum 1/MI) is any combination of the angles for which we can successfully unmix our mixture into its source components. Using the labels in Fig. 6.9, we find that we have eight of these combinations: 1,2 1,3 2,1 2,4 3,1 3,4 4,2 4,3. Note that the combinations 1,4 and 2,3 are not valid because they denote the same axis and not a pair of axes. The eight combinations correspond to the eight maxima in the 1/MI landscape in Fig. 6.13. Because each of the eight combinations describes the optimal angles for the two axes, we only need to find one pair (i.e., one maximum in the landscape in Fig. 6.13) for our unmixing procedure. Since we find that the landscape of 1/MI has a clear-cut structure, we can use this to our benefit and avoid a lengthy brute force computation. Instead of computing the values for all angles from $0-360^\circ$, we apply the so-called simplex method, which uses the gradient in the landscape to locate one of the maxima. Globally, this procedure works as follows:

(1) We randomly pick an initial point in the 1/MI landscape (a pair of angles ϕ_1 and ϕ_2 associated with our pair of source axes X_1 and X_2) and determine its 1/MI value.
(2) Then we pick two neighboring points in the landscape, and determine 1/MI for these points as well (now we have defined a triangle in the landscape).
(3) We determine which of the three points has the lowest 1/MI value and move it in the direction where 1/MI is largest.
(4) We repeat Step 3 above until we cannot find a neighboring point with a low value of 1/MI, at which point we conclude that we have reached a peak in the landscape.
(5) Finally we will make our triangle of points smaller and repeat Steps 3 and 4 to locate the maximum with optimal precision.

By using this method, we use the slope in the landscape to climb toward a maximum, a procedure that is much faster than iteration and that scales much better when we increase the number of sources in each mixture and the number of measurements of mixtures. See Press et al. (2007) for the details of this and other parameter search techniques.

6.4.6 An Example of ICA Applied to EEG Signals

The signals of brain electrical activity in Fig. 6.1 are recordings directly from the cortex (ECoG) and from the scalp (EEG). In the context of this chapter, it is fairly reasonable to assume that the signals generated at different locations separated by several millimeters in the brain will be statistically independent. Another way of saying the same thing is that since brain signals carry a lot of information, recording sites that are relatively remote must have low levels of MI. When recording directly from the cortex, we can indeed observe this principle. When we use ICA to decompose an ECoG (Fig. 6.1A), our statistically independent components are almost identical to the recorded channels, indicating that the different sites on the cortex generate statistically independent signals. For the EEG (Fig. 6.1B), this reasoning does not hold because the skull and scalp have a tendency to smear (mix) the contributions of the underlying sources, so ICA may be a good tool to find individual sources in the signals.

The procedure that is commonly applied to EEG analysis is to find source signals that are temporally independent (because the EEG matrix has a temporal and spatial component, we could also look for components that are spatially independent, but this is usually not done). The underlying thought here is that source signals contribute to the signal at each EEG electrode. An example of two sources contributing to

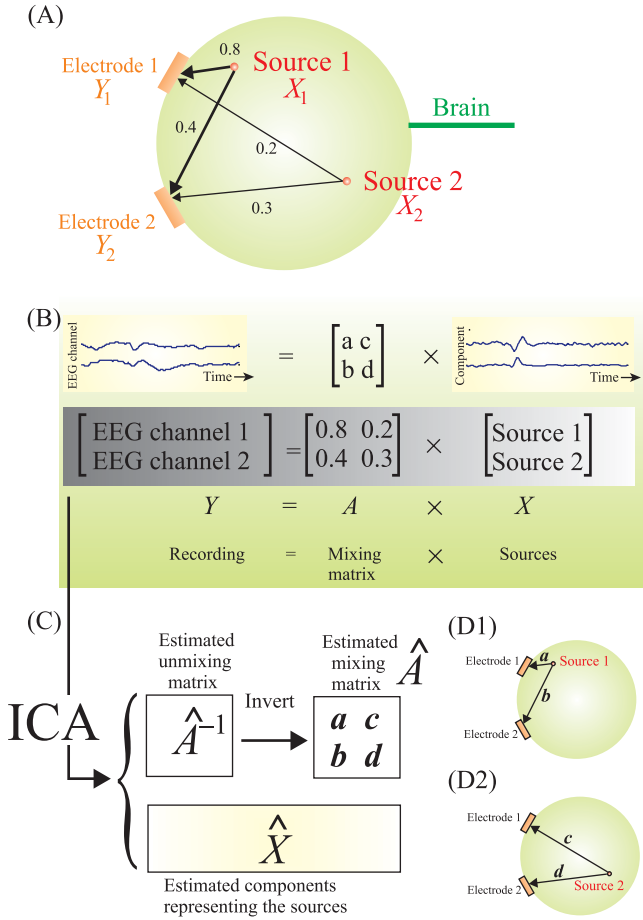

Figure 6.14 ICA analysis in EEG, a two-source and two-channel example. (A) Electrodes Y_1 and Y_2 record channels 1 and 2, each containing a mixture of sources X_1 and X_2. In each mixture, the attenuation of the source signal is proportional with distance between source and electrode (symbolized by the arrows). (B) shows the mathematics underlying the mixing process that can be represented by matrix multiplication $Y = AX$, with A being the mixing matrix (similar to the example in Fig. 6.2). The next step, (C), is to estimate the mixing matrix and source components with the ICA procedure. The estimated source activity can give an impression of the distribution of activities across the brain and the estimate of the mixing matrix can be used to determine the effect for each source on the electrodes (D).

two EEG electrodes/channels is depicted in Fig. 6.14A. The EEG data can therefore be considered a linear mixture of the sources. Because the electrodes register the fields of the locally generated activity traveling at the speed of light, the delays for propagation between source and electrode are negligible. Our finding with the ECoG (that the ICA components resemble the original time series) shows that if sources are not too close, they will be independent. The simplified scenario in Fig. 6.14A shows how two electrodes Y_1 and Y_2 each record a different mixture from sources X_1 and X_2. Similar to the example in Fig. 6.2, we have that $Y = AX$, with A being the mixing matrix. The first column in mixing matrix A (a and b, in the example 0.8 and 0.4) indicate the coupling strength (proximity) of source X_1 to electrodes Y_1 and Y_2 (Fig. 6.14B). The second column in A (c and d, in the example 0.2 and 0.3) reflect the same coupling strength (proximity) of source X_2 to electrodes Y_1 and Y_2 (Fig. 6.14B). In this sense, mixing matrix A contains spatial information because the values of its elements reflect the positions of sources and electrodes.

Now we can use ICA to estimate our source signals $\hat{X}$ and the unmixing matrix $\hat{A}^{-1}$ (Fig. 6.14C). Assuming that $\hat{A}^{-1}$ is invertible, we can determine an estimate $\hat{A}$ of the mixing matrix. Matrix $\hat{A}$ contains the estimates for coupling strengths between each of the sources and the electrodes. These estimates $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{d}$ in Fig. 6.14C and D can now be used to depict the coupling between sources and electrodes. For Source 1 we find coupling strengths $\boldsymbol{a}$ and $\boldsymbol{b}$ for Electrode 1 and Electrode 2 (Fig. 6.14D1); for Source 2 we have strengths $\boldsymbol{c}$ and $\boldsymbol{d}$ for Electrode 1 and

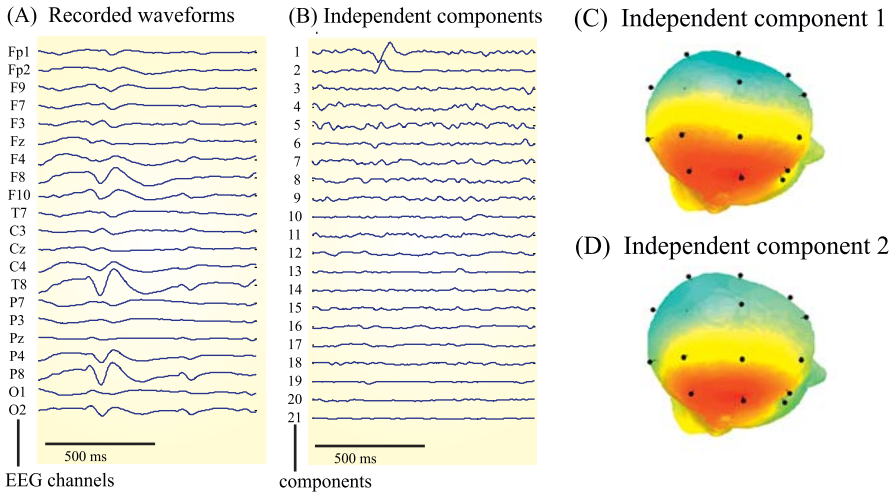

Figure 6.15 Part of the EEG recording shown in Fig. 6.1B is shown in (A); the 21 independent components are shown in (B). Here it can be seen that the epileptic spike waveforms are only represented in the first two independent components. Topographic maps of the scalp potential associated with these two components are shown in (C) and (D). These distributions are indicative for a source that is located right temporally. This figure was prepared with eeglab software. This MATLAB-based package can be downloaded from the Web site http://sccn.ucsd.edu/~scott/ica.html.

Electrode 2 (Fig. 6.14D2). As we will demonstrate in the following example (because usually the EEG recording includes multiple channels), it is common practice to show the coupling strength for each component (source) at each electrode in a color-coded fashion.

A detail of the EEG recording in Fig. 6.1 (the epoch in between the asterisks in Fig. 6.1B) is shown in Fig. 6.15A. This EEG recording contains a high-amplitude epileptic spike. The ICA of this 21-channel recording shows two components that seem associated with this spike signal (Components 1 and 2 in Fig. 6.15B). Since each electrode corresponds to a location, we can use the multichannel EEG to map our independent components topographically (as we did in Fig. 6.14D). The topographic maps of both these independent sources show a right temporal location (Fig. 6.15C and D). It was confirmed clinically that the epileptic focus was indeed located in the right temporal lobe in this patient. Although this confirmation is reassuring, it should be noted here that the brain area where epileptic spikes are generated and the focus where the epileptic seizures originate are not always the same location.

Appendix 6.1

Eigenvalues and Eigenvectors

The eigenvalues and eigenvectors of a matrix play a role in multiple applications, including the determination of principal components described in this chapter. "Eigen" is a German word that, in this context, may be translated into "characteristic." Recall that a matrix can be used to efficiently represent a set of expressions—for example,

$$x + 6y$$

$$5x + 2y$$

can be written as the product of a matrix A and vector v:

$$\underbrace{\begin{bmatrix} 1 & 6 \\ 5 & 2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{v} = Av$$

Vector v is an eigenvector of A if multiplication with matrix A scales it by a constant λ without changing the direction of v (Fig. A6.1):

$$Av = \lambda v \tag{A6.1.1}$$

The constant λ is the so-called eigenvalue of A. We can rewrite this expression as:

$$Av - \lambda v = 0 \rightarrow$$

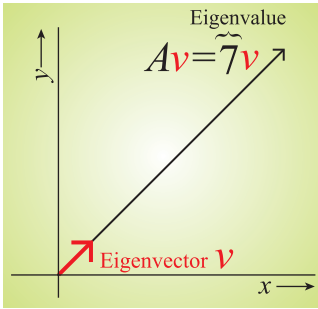

Figure A6.1 Eigenvector $v = (1,1)$ of matrix A. Note that the product Av does not change the direction of v; it only scales it by the eigenvalue λ (7 in this example). This is essentially the property associated with eigenvectors and eigenvalues of a matrix A, here v and λ, respectively.

$$(A - \lambda I)v = 0$$

where I is the identity matrix.

This expression always has the trivial solution $v = 0$, while according to Cramer's rule (see, e.g., Jordan and Smith, 1997), non-trivial solutions (solutions for which $v \neq 0$) can only exist if:

$$|A - \lambda I| = 0 \tag{A6.1.2}$$

The $|\ldots|$ indicates the determinant of the matrix. If we go back to our numerical example above, and apply the condition in Equation (A6.1.2), we have:

$$\left| \underbrace{\begin{bmatrix} 1 & 6 \\ 5 & 2 \end{bmatrix}}_{A} - \lambda \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{I} \right| = 0 \rightarrow$$

$$\left| \underbrace{\begin{bmatrix} 1-\lambda & 6 \\ 5 & 2-\lambda \end{bmatrix}}_{A - \lambda I} \right| = 0 \tag{A6.1.3}$$

This leads to the characteristic equation:

$$(1 - \lambda) \times (2 - \lambda) - 5 \times 6 = 0 \rightarrow \lambda^2 - 3\lambda - 28 = 0 \rightarrow (\lambda - 7)(\lambda + 4) = 0$$

Thus, for our numerical example we find $\lambda_1 = 7$ and $\lambda_2 = -4$.

If we generalize the matrix, let us say,

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

the characteristic equation becomes:

$$(a-\lambda)(d-\lambda)-bc=0\rightarrow\lambda^2-(a+d)\lambda+(ad-bc)=0$$

The solutions for the eigenvalues are now:

$$\lambda_{1,2}=\frac{(a+d)\pm\sqrt{(a+d)^2-4(ad-bc)}}{2}$$

Once the eigenvalues are known, we can compute an eigenvector for each eigenvalue. The only thing we need to determine is the direction of the eigenvector, since the length is unimportant (the scalability, as shown in Fig. A6.1, holds for any length of vector as long as the direction is correct). So we can set the value of x in the eigenvector v arbitrarily to 1 and we substitute for the eigenvalue $\lambda_1=7$ in Equation (A6.1.1):

$$Av=\lambda v\rightarrow\underbrace{\begin{bmatrix}1&6\\5&2\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}1\\y\end{bmatrix}}_{v}=\underbrace{7}_{\lambda}\underbrace{\begin{bmatrix}1\\y\end{bmatrix}}_{v}$$

This results in two equations:

$$1+6y=7$$
$$5+2y=7y$$

Both have the same solution, $y=1$. Therefore,

$$v=\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}1\\1\end{bmatrix}$$

is an eigenvector for eigenvalue 7; this is the eigenvector shown in Fig. A6.1. The same approach can be followed for the other eigenvalue, -4.