Chapter 15 - BLIND SOURCE SEPARATION:

Principal & Independent Component Analysis

Introduction

In this chapter we will examine how we can generalize the idea of transforming a time series into an alternative representation, such as the Fourier (frequency) domain, to facilitate systematic methods of either removing (filtering) or adding (interpolating) data. In particular, we will examine the techniques of **Principal Component Analysis** (PCA) using *Singular Value Decomposition* (SVD), and **Independent Component Analysis** (ICA). Both of these techniques utilize a representation of the data in a statistical domain rather than a time or frequency domain. That is, the data are projected onto a new set of axes that fulfill some statistical criterion, which implies independence, rather than a set of axes that represent discrete frequencies such as with the Fourier transform, where the independence is *assumed*.

Another important difference between these statistical techniques and Fourier-based techniques is that the Fourier components onto which a data segment is projected are fixed, whereas PCA- or ICA-based transformations *depend* on the structure of the data being analyzed. The axes onto which the data are projected are therefore *discovered*. If the structure of the data (or rather the statistics of the underlying sources) changes over time, then the axes onto which the data are projected will change too[1].

Any projection onto another set of axes (or into another space) is essentially a method for separating the data out into separate components or **sources** which will hopefully allow us to see important structure more clearly in a particular projection. That is, the direction of projection increases the signal-to-noise ratio (SNR) for a particular signal source. For example, by calculating the power spectrum of a segment of data, we hope to see peaks at certain frequencies. The power (amplitude squared) along certain frequency vectors is therefore high, meaning we have a strong component in the signal at that frequency. By discarding the projections that correspond to the unwanted sources (such as the noise or artifact sources) and inverting the transformation, we effectively perform a filtering of the recorded observation. This is true for both ICA and PCA as well as Fourier-based techniques. However, one important difference between these techniques is that Fourier techniques *assume* that the projections onto each frequency component are independent of the other frequency components. In PCA and ICA we attempt to *find* a set of axes which are independent of one another in some sense. We assume there are a set of independent

[1] (The structure of the data can change because existing sources are non-stationary, new signal sources manifest, or the manner in which the sources interact at the sensor changes.

sources in the data, but do not assume their exact properties. (Therefore, they may overlap in the frequency domain in contrast to Fourier techniques.) We then define some measure of independence and attempt to *decorrelate* the data by maximising this measure for (or between) projections onto each axis of the new space which we have transformed the data into. The **sources** are the data projected onto each of the new axes. Since we *discover*, rather than define the the new axes, this process is known as **blind source separation**. That is, we do not look for specific pre-defined components, such as the energy at a specific frequency, but rather, we allow the data to determine the components.

For PCA the measure we use to discover the axes is **variance** and leads to a set of orthogonal axes (because the data are decorrelated in a second order sense and the dot product of any pair of the newly discovered axes is zero). For ICA this measure is based on non-Gaussianity, such as **kurtosis**, and the axes are not necessarily orthogonal. Kurtosis is the fourth moment (mean, variance, and skewness are the first three) and is a measure of how non-Gaussian is a probability distribution function (PDF). Large positive values of kurtosis indicate a highly peaked PDF that is much narrower than a Gaussian. A negative kurtosis indicates a broad PDF that is much wider than a Gaussian (see §15.4). Our assumption is that if we maximize the non-Gaussianity of a set of signals, then they are maximally independent. This comes from the central limit theorem; if we keep adding independent signals together (which have highly non-Gaussian PDFs), we will eventually arrive at a Gaussian distribution. Conversely, if we break a Gaussian-like observation down into a set of non-Gaussian mixtures, each with distributions that are as non-Gaussian as possible, the individual signals will be independent. Therefore, kurtosis allows us to separate non-Gaussian independent sources, whereas variance allows us to separate independent Gaussian noise sources.

This simple idea, if formulated in the correct manner, can lead to some surprising results, as you will discover in the applications section later in these notes and in the accompanying laboratory. However, we shall first map out the mathematical structure required to understand how these independent sources are discovered and what this means about our data (or at least, our beliefs about the underlying sources). We shall also examine the assumptions we must make and what happens when these assumptions break down.

15.1 Signal & noise separation

In general, an observed (recorded) time series comprises of both the *signal* we wish to analyze and a *noise* component that we would like to remove. Noise or artifact removal often comprises of a data reduction step (filtering) followed by a data reconstruction technique (such as interpolation). However, the success of the data reduction and reconstruction steps is highly dependent upon the nature of the noise and the signal.

By definition, noise is the part of the observation that masks the underlying signal we wish to analyze[2], and in itself adds no information to the analysis. However, for a noise signal to carry no information, it must be *white* with a flat spectrum and an autocorrelation function

[2] It lowers the SNR!

(ACF) equal to an impulse[3]. Most *real* noise is not really white, but colored in some respect. In fact, the term *noise* is often used rather loosely and is frequently used to describe signal contamination. For example, muscular activity recorded on the electrocardiogram (ECG) is usually thought of as noise or artifact. (See Fig. 1.) However, increased muscle artifact on the ECG actually tells us that the subject is more active than when little or no muscle noise is present. Muscle noise is therefore a source of information about activity, although it reduces the amount of information we can extract from the signal concerning the cardiac cycle. Signal and noise definitions are therefore task-related and change depending on the nature of the information you wish to extract from your observations. In this sense, muscle noise is just another independent information 'source' mixed into the observation.

Table 1 illustrates the range of signal contaminants for the ECG[4]. We shall also examine the statistical qualities of these contaminants in terms of estimates of their PDFs since the power spectrum is not always sufficient to characterize a signal. The shape of a PDF can be described in terms of its **Gaussianity**, or rather, departures from this idealized form (which are therefore called super- or sub-Gaussian). The fact that these signals are not Gaussian turns out to be an extremely important quality, which is closely connected to the concept of independence, which we shall exploit to separate contaminants form the signal.

Although noise is often modeled as *Gaussian white noise*[5], this is often not the case. Noise is often correlated (with itself or sometimes the source of interest), or concentrated at certain values. For example, 50Hz or 60Hz mains noise contamination is sinusoidal, a waveform that spends most of its time at the extreme values (near its turning points), rather than at the mean, as for a Gaussian process. By considering departures from the ideal Gaussian noise model we will see how conventional techniques can under-perform and how more sophisticated (statistical-based) techniques can provide improved filtering.

We will now explore how this is simply another form of data reduction (or filtering) through projection onto a new set of axes or followed by data reconstruction through projection back into the original observation space. By reducing the number of axes (or dimensions) onto which we project our data, we perform a filtering operation (by discarding the projections onto axes that are believed to correspond to noise). By projecting from a dimensionally reduced space (into which the data has been compressed) back to the original space, we perform a type of interpolation (by adding information from a model that encodes some of our prior beliefs about the underlying nature of the signal or information derived directly from a observation data).

15.2 Matrix transformations as filters

The simplest filtering of a time series involves the transformation of a discrete one dimensional ($N = 1$) time series $\mathbf{x}[m]$, consisting of M sample points such that $\mathbf{x}[m] =$

[3] Therefore, no one-step prediction is possible. This type of noise can be generated in MATLAB with the `rand()` function.

[4] Throughout this chapter we shall use the ECG as a descriptive example because it has easily recognizable (and definable) features and contaminants.

[5] generated in MATLAB by the function `randn()`.

Qualities → Contaminant ↓	Frequency Range	Time duration
Electrical Powerline	Narrowband 16.6, 50 or 60 ± 2 Hz	Continuous
Movement Baseline Wander ($\sim \frac{1}{f^2}$)	Narrowband (< 0.5Hz)	Transient or Continuous
Muscle Noise ($\sim$ white)	Broadband	Transient
Non-powerline Electrical Interference	Narrowband (usually ≥ 100 Hz)	Transient or Continuous
Electrode pop from electrode pull	Narrowband ($\sim$ 1-10 Hz)	Transient (0.1 - 1 s)
Observation noise ($\sim \frac{1}{f}$)	Broadband	Continuous
Quantization noise ($\sim$ white & Gaussian)	Broadband	Continuous

Table 1: Contaminants on the ECG and their nature.

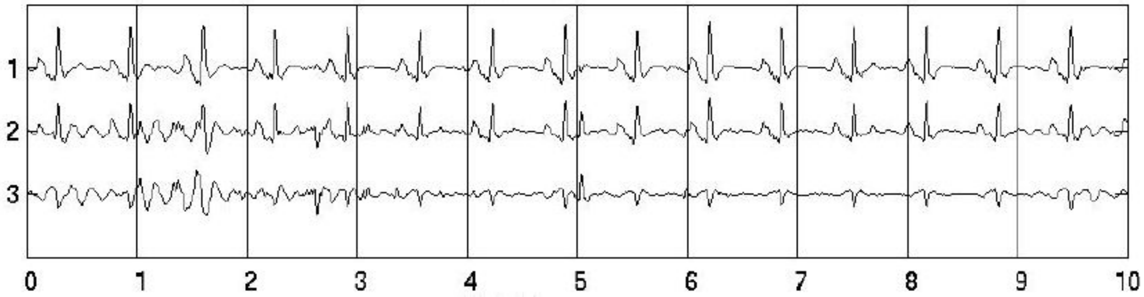

Figure 1: 10 seconds of 3 Channel ECG. Note the high amplitude movement artifact (at about 5 Hz) in the first two seconds and the 10^{th} second. Note also the QRS-like artifacts around 2.6 and 5.1 seconds. Both artifacts closely resemble real ECG phenomena; the former would trigger any ventricular fibrillation detector on channels 2 and 3, and the latter is almost indistinguishable from a ventricular ectopic beat on the same channels. The first artifact is due to muscle twitches (possibly stemming from either hypothermia or Parkinson's disease). The second artifact is due to electrode pop; a sudden tug on the electrodes used for channels 2 and 3.

$(x_1, x_2, x_3...x_M)^T$, into a new representation, $\mathbf{y} = (y_1, y_2, y_3...y_M)^T$. If $\mathbf{x}[m]$ $(t = 1, 2, ..., M)$ is a column vector[6] that represents a channel of ECG, then we can generalize this representation so that N channels of ECG $\mathbf{X}$, and their transformed representation $\mathbf{Y}$ are given by

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix}, \qquad \mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1N} \\ y_{21} & y_{22} & \cdots & y_{2N} \\ \vdots & \vdots & & \vdots \\ y_{M1} & y_{M2} & \cdots & y_{MN} \end{bmatrix} \tag{1}$$

Note that we will adopt the convention throughout this chapter (and in the accompanying laboratory exercises) that all vectors are written in lower-case bold and are column vectors, and all matrices are written in upper-case bold type. The M points of each of the N signal channels form $M \times N$ matrices (i.e. the signal is N-dimensional with M samples for each vector). An $(N \times N)$ transformation matrix $\mathbf{W}$ can then be applied to $\mathbf{X}$ to create the transformed matrix $\mathbf{Y}$ such that

$$\mathbf{Y^T} = \mathbf{W}\mathbf{X^T}. \tag{2}$$

The purpose of a transformation is to map (or *project*) the data into another space which serves to highlight different patterns in the data along different projection axes. To filter the data we discard the noise, or 'uninteresting' parts of the signal (which are masking the information we are interested in). This amounts to a dimensionality reduction, as we are discarding the dimensions (or subspace) that corresponds to the noise.

In general, transforms can be categorized as **orthogonal** or **biorthogonal** transforms. For orthogonal transformations, the transformed signal is same length (M) as the original and the energy of the data is unchanged. An example of this is the Discrete Fourier transform (DFT) where the same signal is measured along a new set of perpendicular axes corresponding to the coefficients of the Fourier series (see chapter 4). In the case of the DFT with $k = M$ frequency vectors, we can write Eq. 2 as $\mathbf{Y}_k = \sum_{n=1}^{N} \mathbf{W}_{kn}\mathbf{X}_n$ where $\mathbf{W}_{kn} = e^{-j2\pi kn/N}$, or equivalently

$$\mathbf{W} = \begin{bmatrix} e^{-j2\pi} & e^{-j4\pi} & \cdots & e^{-j2\pi N} \\ e^{-j4\pi} & e^{-j8\pi} & \cdots & e^{-j4\pi N} \\ \vdots & \vdots & & \vdots \\ e^{-j2\pi M} & e^{-j4\pi M} & \cdots & e^{-j2\pi MN} \end{bmatrix}. \tag{3}$$

For biorthogonal transforms, the angles between the axes may change and the new axes are not necessarily perpendicular. However, no information is lost and perfect reconstruction of the original signal is still possible (using $\mathbf{X}^T = \mathbf{W^{-1}}\mathbf{Y}^T$).

Transformations can be further categorized as as either **lossless** (so that the transformation can be reversed and the original data restored exactly) or as **lossy**. When a signal is filtered or compressed (through downsampling for instance), information is often lost and the transformation is not invertible. In general, lossy transformations involve a non-invertible transformation of the data using a transformation matrix that has at least one column set

[6] In `Matlab` the command `[M N]=size(x)` gives a dimension of $N = 1$ and a length equal to M for a column vector `x`.

to zero. Therefore there is an irreversible removal of some of the data N-dimensional data and this corresponds to a mapping to a lower number of dimensions ($p < N$).

In the following sections we will study two transformation techniques Principal Component Analysis (PCA) and Independent Component Analysis (ICA). Both techniques attempt to find an independent set of vectors onto which we can transform the data. The data that are projected (or mapped) onto each vector *are* the independent sources. The basic goal in PCA is to *decorrelate* the signal by projecting the data onto *orthogonal* axes. However, ICA results in a biorthogonal transform of the data and the axes are not necessarily orthogonal. Both PCA and ICA can be used to perform lossy or lossless transformations by multiplying the recorded (observation) data by a separation or *demixing* matrix. Lossless PCA and ICA both involve projecting the data onto a set of axes which are determined by the nature of the data, and are therefore methods of **blind source separation** (BSS). (*Blind* because the axes of projection and therefore the sources are determined through the application of an internal measure and without the use of any prior knowledge of the data structure.)

Once we have *discovered* the axes of the independent components in the data and have separated them out by projecting the data onto these axes, we can then use these techniques to filter the data. By setting columns of the PCA and ICA separation matrices that correspond to unwanted sources to zero, we produce non-invertible matrices[7]. If we then *force* the inversion of the separation matrix[8] and transform the data back into the original observation space, we can remove the unwanted source from the original signal. Figure 2 illustrates the BSS paradigm for filtering whereby we have N unknown sources in an unknown *source space* which are linearly mixed and transformed into an *observation space* in which they are recorded. We then attempt to discover (an estimate of) the sources, $\hat{\mathbf{Z}}$, or the inverse of the mixing matrix, $\mathbf{W} \approx \mathbf{A}^{-1}$, and use this to transform the data back into an estimate of our source space. After identifying the sources of interest and discarding those that we do not want (by altering the inverse of the demixing matrix to have columns of zeros for the unwanted sources), we reproject the data back into the observation space using the inverse of the altered demixing matrix, $\mathbf{W}_p^{-1}$. The resultant data $\mathbf{X}_{filt}$, is a filtered version of the original data $\mathbf{X}$.

We shall also see how the sources that we discover with PCA have a specific ordering according to the energy along each axis for a particular source. This is because we look for the axis along which the data has maximum variance (and hence energy or power[9]). If the signal to noise ratio (SNR) is greater than unity, the signal of interest is therefore confined to the first few components. However, ICA allows us to discover sources by measuring a relative cost function between the sources that is dimensionless. There is therefore no relevance to the order of the columns in the separated data and often we have to apply further signal-specific measures, or heuristics, to determine which sources are interesting.

[7] For example, a transformation matrix [1 0; 0 0] is **non-invertible**, or **singular** (`inv([1 0; 0 0]) = [Inf Inf; Inf Inf]` in `Matlab`) and multiplying a two dimensional signal by this matrix performs a simple reduction of the data by one dimension.

[8] Using a pseudo-inversion technique such as `Matlab`'s `pinv`; `pinv([1 0; 0 0]) = [1 0; 0 0]`.

[9] All are proportional to $\mathbf{x}^2$.

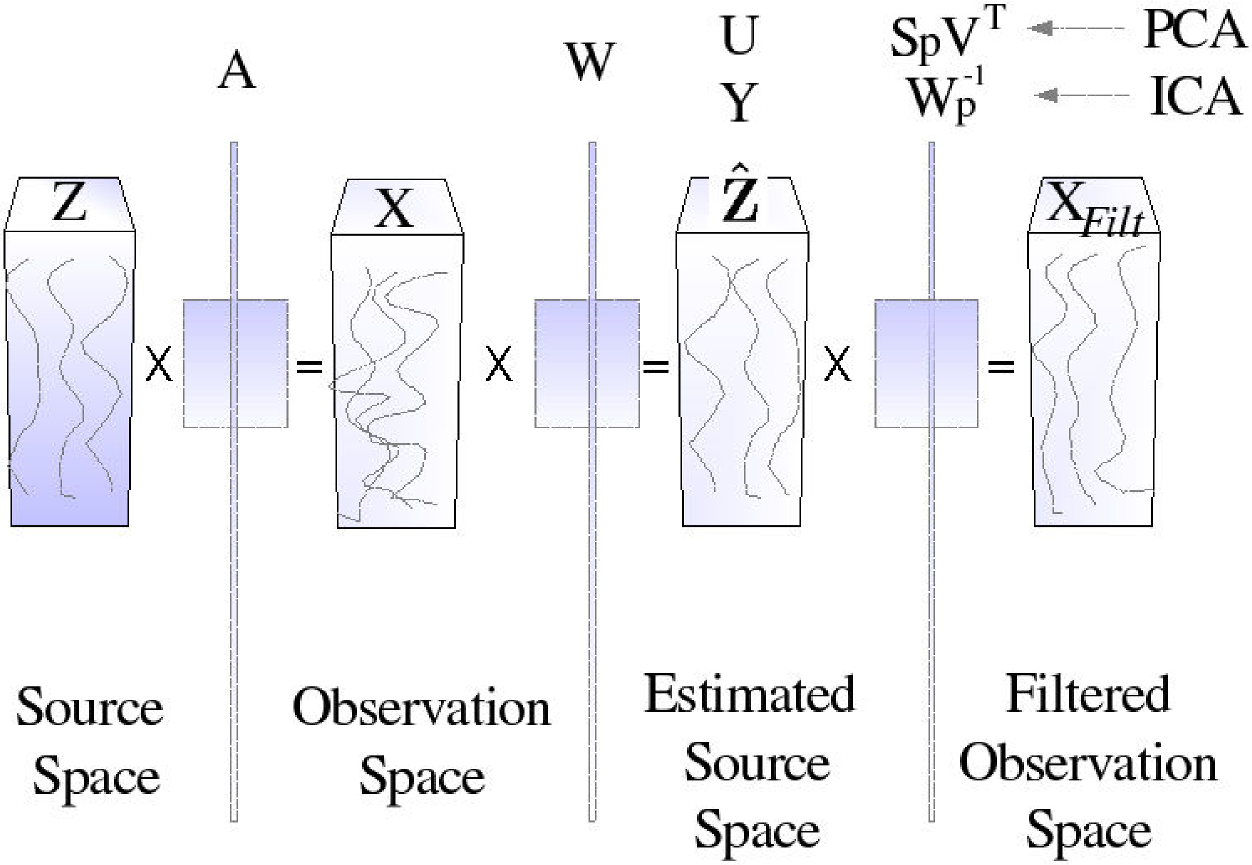

Figure 2: The general paradigm of Blind Source Separation for filtering. Given some unknown matrix of sources $\mathbf{Z}$ which is mixed by some linear stationary matrix of constants $\mathbf{A}$, our sources are projected from a *source space* to an *observation space* to give the observations, $\mathbf{X}$. These observations are then transposed back into an *estimated source space* in which the *estimates* of the sources, $\hat{\mathbf{Z}}$ are projected. We then reduce the dimensionality of the estimated source space, by discarding the estimates of the sources that correspond to noise or unwanted artifacts by setting $N-p$ columns of $\mathbf{W}^{-1}$ to zero (to give $\mathbf{W}_p^{-1}$) and reprojecting back into the observation space. The resulting matrix of filtered observations is $\mathbf{X}_{filt}$. The filtered observation space and original observation space are the same, but the data projected into them is filtered and unfiltered respectively. In the case of PCA, the sources are the columns of $\mathbf{U}$, and can be formed using $\mathbf{S}^{-1}$ and $\mathbf{V^T}^{-1}$ (see § 15.3.1, Eq. 4), but the transformation is not so straightforward. Reducing the dimensionality of $\mathbf{S}$ to have only p non-zero columns, the filtered observations can be reconstructed by evaluating $\mathbf{X}_{filt} = \mathbf{U}\mathbf{S}_p\mathbf{V^T}$. In the case of ICA, $\mathbf{X}$ can multiplied by the demixing matrix $\mathbf{W}$, to reveal the estimates of the sources, $\mathbf{Y} = \hat{\mathbf{Z}}$. Columns of $\mathbf{W}^{-1}$ can be set to zero to remove the 'noise' sources and the filtered data are reconstructed using $\mathbf{X}_{filt} = \mathbf{W}_p^{-1}\mathbf{Y}$

.

15.3 Principal Component Analysis

In the case of the Fourier transform, the **basis functions** or axes of the new representation are predefined and assumed to be independent, whereas with PCA the **representation**, or the basis vectors, are *found* in the data by looking for a set of axes that *are* independent. That is, the data undergoes a decorrelation using variance as the metric. Projections onto these axes, or basis vectors, are independent in a second order sense and are orthogonal (the dot product of the basis vectors, and the cross-correlation of the projections are close to zero).

The basic idea in the application of PCA to a data set, is to find the component vectors $\mathbf{y}_1$, $\mathbf{y}_2$,...,$\mathbf{y}_N$ that explain the maximum amount of variance possible by N linearly transformed components. PCA can be defined in an intuitive way using a recursive formulation. The direction of the first principal component $\mathbf{v}_1$ is found by passing over the data and attempting to maximize the value of $\mathbf{v}_1 = \arg\max_{\|\mathbf{v}\|=1} E\{(\mathbf{v}_1^T\mathbf{X})^2\}$ where $\mathbf{v}_1$ is the same length M as the data $\mathbf{X}$. Thus the first principal component is the projection on the direction in which the variance of the projection is maximized. Each of the remaining $N-1$ principal components are found by repeating this process in the remaining orthogonal subspace (which reduces in dimensionality by one for each new component we discover). The principal components are then given by $\mathbf{y}_i = \mathbf{v}_i^T\mathbf{X}$ $(i = 1, ..., N)$, the projection of $\mathbf{X}$ onto each $\mathbf{v}_i$. This transformation of the columns of $\mathbf{X}$ onto $\mathbf{v}_i^T$, to give $\mathbf{y}_i$ is also known as the (discrete) Karhunen-Loève transform, or the Hotelling transform, a derivation of which is given in appendix 15.9.1).

Although the basic goal in PCA is to *decorrelate* the data by performing an orthogonal projection, we often reduce the dimension of the data from N to p $(p < N)$ to remove unwanted components in the signal. It can be shown that the PCA representation is an optimal linear dimension reduction technique in the mean-square sense [1]. One important application of this technique is for noise reduction, where the data contained in the last $N-p$ components is assumed to be mostly due to noise. Another benefit of this technique is that a projection into a subspace of a very low dimension, for example two or three, can be useful for visualizing multidimensional or higher order data.

In practice, the computation of the $\mathbf{v}_i$ can be simply accomplished using the sample covariance matrix $\mathbf{C} = \mathbf{X}^T\mathbf{X}$. The $\mathbf{v}_i$ are the eigenvectors of $\mathbf{C}$ (an $M \times M$ matrix) that correspond to the N eigenvalues of $\mathbf{C}$. A method for determining the eigenvalues in this manner is known as Singular Value Decomposition (SVD), which is described below.

15.3.1 Method of SVD

To determine the principal components of a multi-dimensional signal, we can use the method of Singular Value Decomposition. Consider a real $M \times N$ matrix $\mathbf{X}$ of observations which may be decomposed as follows;

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{4}$$

where $\mathbf{S}$ is an $M \times N$ non-square matrix with zero entries everywhere, except on the leading diagonal with elements s_i ($= S_{MN}$, $M = N$) arranged in descending order of magnitude. Each s_i is equal to $\sqrt{\lambda_i}$, the square root of the eigenvalues of $\mathbf{C} = \mathbf{X}^T\mathbf{X}$. A stem-plot of these values against their index i is known as the **singular spectrum** or **eigenspectrum**. The smaller the eigenvalue, the smaller the total energy is that is projected along the corresponding eigenvector. Therefore, the smallest eigenvalues are often considered to be associated with eigenvectors that describe the noise in the signal[10]. The columns of $\mathbf{V}$ form an $N \times N$ matrix of column vectors, which are the eigenvectors of $\mathbf{C}$. The $M \times M$ matrix $\mathbf{U}$ is the matrix of projections of $\mathbf{X}$ onto the eigenvectors of $\mathbf{C}$ [2]. A truncated SVD of $\mathbf{X}$ can be performed such that only the the most significant (p largest) eigenvectors are retained. In practice choosing the value of p depends on the nature of the data, but is often taken to be the *knee* in the eigenspectrum (see §15.3.3) or the value where $\sum_{i=1}^{p} s_i > \alpha \sum_{i=1}^{N} s_i$ and α is some fraction ≈ 0.95. The truncated SVD is then given by $\mathbf{Y} = \mathbf{U}\mathbf{S}_p\mathbf{V}^T$ and the columns of the $M \times N$ matrix $\mathbf{Y}$ are the noise-reduced signal (see Fig, 3 and the practical example given in § 15.3.3).

A routine for performing SVD is as follows:

1. Find the N non-zero eigenvalues, λ_i of the matrix $\mathbf{C} = \mathbf{X^T X}$ and form a non-square diagonal matrix $\mathbf{S}$ by placing the square roots $s_i = \sqrt{\lambda_i}$ of the N eigenvalues in descending order of magnitude on the leading diagonal and setting all other elements of $\mathbf{S}$ to zero.
2. Find the orthogonal eigenvectors of the matrix $\mathbf{X^T X}$ corresponding to the obtained eigenvalues, and arrange them in the same order. this ordered collection of column-vectors forms the matrix $\mathbf{V}$.
3. Find the first N column-vectors of the matrix $\mathbf{U}$: $\mathbf{u}_i = s_i^{-1}\mathbf{X}\mathbf{v}_i$ $(i = 1 : N)$. Note that s_i^{-1} are the elements of $\mathbf{S}^{-1}$.
4. Add the rest of $M - N$ vectors to the matrix $\mathbf{U}$ using the Gram-Schmidt orthogonalization process (see appendix 15.9.2).

15.3.2 Eigenvalue decomposition - a worked example

To find the singular value decomposition of the matrix

$$\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{5}$$

first we find the eigenvalues, λ, of the matrix

$$\mathbf{C} = \mathbf{X^T X} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

[10] This, of course, is not true if the noise energy is comparable or larger than the signal of interest.

in the usual manner by letting

$$\mathbf{Cv} = \lambda \mathbf{v} = 0 \tag{6}$$

so $(\mathbf{C} - \lambda \mathbf{I}) = 0$ and

$$\begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix} = 0$$

Evaluating this determinant and solving this characteristic equation for λ, we find $(2-\lambda)^2 - 1 = 0$, and so $\lambda_1 = 3$ and $\lambda_2 = 1$. Next we note the number of non-zero eigenvalues of the matrix $\mathbf{X^TX}$ (two in this case). Then we find the orthonormal eigenvectors of the matrix X^TX corresponding to the non-zero eigenvalues (λ_1 and λ_2) by solving for $\mathbf{v_1}$ and $\mathbf{v_2}$ using λ_1 and λ_2 and in $(\mathbf{C} - \lambda \mathbf{I})\mathbf{v} = 0$ ...

$$\mathbf{v_1} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}, \mathbf{v_2} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix}, \tag{7}$$

forming the matrix

$$\mathbf{V} = [\mathbf{v_1 v_2}] = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix} \tag{8}$$

where $\mathbf{v_1}$ and $\mathbf{v_2}$ are normalized to unit length. Next we write down the singular value matrix $\mathbf{S}$ which is a diagonal matrix composed of the square roots of the eigenvalues of $\mathbf{C} = \mathbf{X^TX}$ arranged in descending order of magnitude.

$$\mathbf{S} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \sqrt{(\lambda_1)} & 0 \\ 0 & \sqrt{(\lambda_2)} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \sqrt{3} & 0 \\ 0 & \sqrt{1} \\ 0 & 0 \end{bmatrix}. \tag{9}$$

Recalling that the inverse of a matrix $\mathbf{B}$ with elements b_{ij} is given by

$$\mathbf{B}^{-1} = \frac{1}{det\|\mathbf{B}\|} \begin{bmatrix} b_{22} & -b_{21} \\ -b_{12} & b_{11} \end{bmatrix} \tag{10}$$

and so

$$\mathbf{S}^{-1} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{3} \end{bmatrix} \tag{11}$$

and we can find the first two columns of $\mathbf{U}$, using the relation

$$\mathbf{u_1} = s_1^{-1}\mathbf{Xv_1} = \frac{\sqrt{3}}{3} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{6}}{3} \\ \frac{\sqrt{6}}{6} \\ \frac{\sqrt{6}}{6} \end{bmatrix}$$

and

$$\mathbf{u_2} = s_2^{-1}\mathbf{Xv_2} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}.$$

Using the Gram-Schmitd process (see appendix 15.9.2) we can calculate the third and remaining orthogonal column of $\mathbf{U}$ to be

$$\mathbf{u}_3 = \begin{bmatrix} \frac{\sqrt{3}}{3} \\ -\frac{\sqrt{3}}{3} \\ -\frac{\sqrt{3}}{3} \end{bmatrix}.$$

Hence

$$\mathbf{U} = [\mathbf{u}_1\mathbf{u}_2\mathbf{u}_3] = \begin{bmatrix} \frac{\sqrt{6}}{3} & 0 & \frac{\sqrt{3}}{3} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{2}}{2} & \frac{-\sqrt{3}}{3} \\ \frac{\sqrt{6}}{6} & \frac{-\sqrt{2}}{2} & \frac{-\sqrt{3}}{3} \end{bmatrix}$$

and the singular value decomposition of the matrix $\mathbf{X}$ is

$$\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \mathbf{USV^T} = \begin{bmatrix} \frac{\sqrt{6}}{3} & 0 & \frac{\sqrt{3}}{3} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{2}}{2} & \frac{-\sqrt{3}}{3} \\ \frac{\sqrt{6}}{6} & \frac{-\sqrt{2}}{2} & \frac{-\sqrt{3}}{3} \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} \end{bmatrix}$$

15.3.3 SVD filtering - a practical example using the ECG

We will now look at a more practical (and complicated) illustration. SVD is a commonly employed technique to compress and/or filter the ECG. In particular, if we align N heartbeats, each M samples long, in a matrix (of size $M \times N$), we can compress the matrix down (into an $M \times p$) matrix, using only the first $p << N$ principal components. If we then reconstruct the data by inverting the reduced rank matrix, we effectively filter the original data.

Fig. 3a is a set of 8 heart beat waveforms recorded from a single ECG lead recorded with a sampling frequency $F_s = 200$ Hz, which have been divided into one-second segments centered on their R-peaks (maximum values), and placed side-by-side to form a 200×8 matrix. The data set is therefore 8-dimensional and an SVD will lead to 8 eigenvectors. Fig. 3b is the eigenspectrum obtained from SVD[11]. Note that most of the *power* is contained in the first eigenvector. The *knee* of the eigenspectrum is at the second principal component. Fig. 3c is a plot of the reconstruction (filtering) of the data using just the first eigenvector[12]. Fig. 3d is the same as Fig. 3c, but the first two eigenvectors have been used to reconstruct the data. The data in Fig. 3d is therefore noisier than that in Fig. 3c.

Note that S derived from a full SVD (using Matlab's function `svd()`) is an **invertible** matrix, and no information is lost if we retain all the principal components. In other words, we recover the original data by performing the multiplication $\mathbf{USV}^T$. However, if we perform a truncated SVD (using `svds()`) then the inverse of S (`inv(S)`) does not exist. The transformation that performs the filtering is **non-invertible** and information is lost because S is **singular**.

[11] In Matlab: `[U S V]=svd(data); stem(diag(S))`.

[12] In Matlab: `[U S V]=svds(data,1); mesh(U*S*V')`.

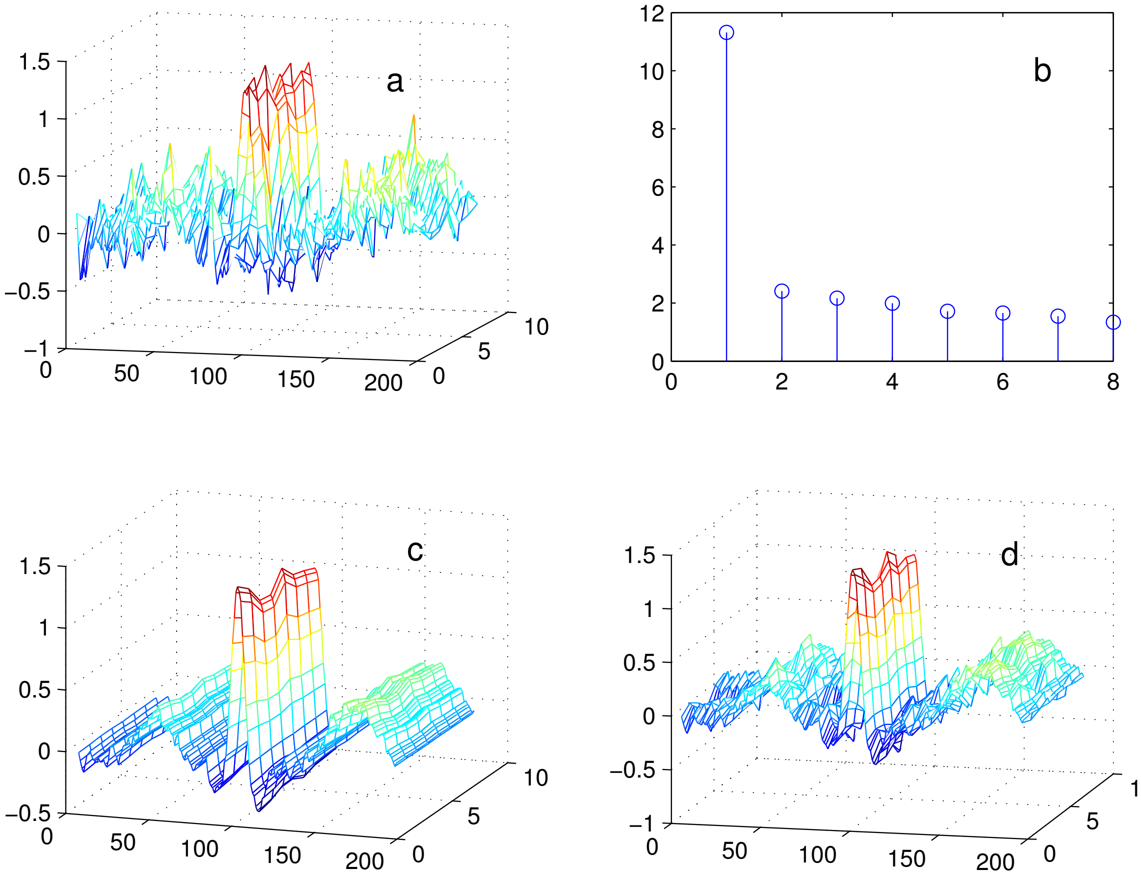

Figure 3: SVD of eight R-peak aligned P-QRS-T complexes; a) in the original form with a large amount of in-band noise, b) eigenspectrum of decomposition, c) reconstruction using only the first principal component, d) reconstruction using only the first two principal components.

From a data compression point of view, SVD is an excellent tool. If the eigenspace is known (or previously determined from experiments), then the N-dimensions of data can in general be encoded in only p-dimensions of data. So for M sample points in each signal, an $M \times N$ matrix is reduced to an $M \times p$ matrix. In the above example, retaining only the first principal component, we achieve a compression ration of $8 : 1$. Note that the data are encoded in the $\mathbf{U}$ matrix and so we are only interested in the first p columns. The eigenvalues and eigenvectors are encoded in $\mathbf{S}$ and $\mathbf{V}$ matrices, and therefore an additional p scalar values are required to encode the relative energies in each column (or signal source) in $\mathbf{U}$. Furthermore, if we wish to encode the *eigenspace* onto which the data in $\mathbf{U}$ is projected, we require an additional N^2 scalar values (the elements of $\mathbf{V}$).

It should be noted that the eigenvectors are likely to change[13], based upon heart-rate dependent beat-to-beat morphology changes (because the cardiac conduction speed changes at different heart rates) and the presence of abnormal beats.

In order to find the global eigenspace for all beats, we need to take a large, representative set of heartbeats[14] and perform SVD upon this [3]. Projecting each new beat onto these *globally derived* basis vectors results in a filtering of the signal that is essentially equivalent to passing the P-QRS-T complex through a set of trained weights of a multi-layer perceptron (MLP) neural network (see [4] & appendix 15.9.4). Abnormal beats or artifacts erroneously detected as normal beats will have abnormal eigenvalues (or a highly irregular structure when reconstructed by the MLP). In this way, beat classification can be performed. It should be noted however, that in order to retain all the subtleties of the QRS complex, at least $p = 5$ eigenvalues and eigenvectors are required (and another five for the rest of the beat). At a sampling frequency of F_s Hz and an average beat-to-beat interval of $\mathrm{RR}^{\mathrm{av}}$ (or heart rate of $60/\mathrm{RR}^{\mathrm{av}}$) the compression ratio is $F_s \cdot \mathrm{RR}^{\mathrm{av}} \cdot (\frac{M-p}{p}) : 1$ where M is the number of samples in each segmented heart beat.

15.4 Independent Component Analysis for source separation and filtering

Using SVD we have seen how we can separate a signal into a subspace that is *signal* and a subspace that is essentially *noise*. This is done by assuming that only the eigenvectors associated with the p largest eigenvalues represent the signal, and the remaining $(M - p)$ eigenvalues are associated with the noise subspace. We try to maximize the independence between the eigenvectors that span these subspaces by requiring them to be orthogonal. However, the differences between signals and noise are not always clear, and orthogonal subspaces may not be the best way to differentiate between the constituent sources in a measured signal.

In this section we will examine how choosing a measure of independence other than variance can lead to a more effective method for separating signals. A particularly intuitive illustration of the problem of source separation through discovering independent sources, is known as the *Cocktail Party* Problem.

[13] Since they are based upon the morphology of the beats, they are also lead-dependent.

[14] That is, $N >> 8$.

15.4.1 Blind Source Separation; the Cocktail Party Problem

The *Cocktail Party Problem* is a classic example of Blind Source Separation (BSS), the separation of a set of observations into the constituent underlying (statistically independent) **source** signals. The Cocktail Party Problem is illustrated in Fig. 4. If each of the J voices you can hear at a party are recorded by N microphones, the recordings will be a matrix composed of a set of N vectors, each of which is a (weighted) linear superposition of the J voices. For a discrete set of M samples, we can denote the sources by an $J \times M$ matrix, $\mathbf{Z}$, and the N recordings by an $N \times M$ matrix $\mathbf{X}$. $\mathbf{Z}$ is therefore transformed into the observables $\mathbf{X}$ (through the propagation of sound waves through the room) by multiplying it by a $N \times J$ mixing matrix $\mathbf{A}$ such that[15] $\mathbf{X^T} = \mathbf{AZ^T}$. (Recall Eq. 2 in §15.2.) Figure 4 illustrates this paradigm where sound waves from $J = 3$ independent speakers ($\mathbf{z}_1$, $\mathbf{z}_2$, and $\mathbf{z}_3$, left) are superimposed (center), and recorded as three mixed source vectors with slightly different phases and volumes at three spatially separated but otherwise identical microphones.

In order for us to 'pick out' a voice from an ensemble of voices in a crowded room, we must perform some type of BSS to recover the original sources from the observed mixture. Mathematically, we want to find a demixing matrix $\mathbf{W}$, which when multiplied by the recordings $\mathbf{X^T}$, produces an estimate $\mathbf{Y^T}$ of the sources $\mathbf{Z^T}$. Therefore $\mathbf{W}$ is a set of weights (approximately[16]) equal to $\mathbf{A^{-1}}$. One of the key methods for performing BSS is known as **Independent Component Analysis** (ICA), where we take advantage of (an assumed) linear independence between the sources.

An excellent interactive example of the cocktail party problem can be found at

`http://www.cis.hut.fi/projects/ica/cocktail/cocktail_en.cgi`

The reader is encouraged to experiment with this URL at this stage. Initially you should attempt to mix and separate just two different sources, then increase the complexity of the problem adding more sources. Note that the relative phases and volumes of the sources differ slightly for each recording (microphone) and that the separation of the sources may change in order and volume (amplitude). This is known as the *permutation and scaling problem* for ICA (see § 15.8.1).

15.4.2 Higher order independence: ICA

Independent Component Analysis is a general name for a variety of techniques which seek to uncover the **independent** source signals from a set of observations that are composed of **linear** mixtures of the underlying sources. Consider $\mathbf{X_{jn}}$ to be a matrix of J observed

[15] Note that $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ are row matrices, for consistency with the PCA formulation, and so we take the transpose in the ICA formulation. Note also that in standard ICA notation,$\mathbf{X} = \mathbf{AS}$, where **X** and **S** are row matrices and **S** are the sources. However, to avoid confusion with the PCA notation, we **S** is denoted $\mathbf{Z^T}$.

[16] Depending on the performance details of the algorithm used to calculate $\mathbf{W}$.

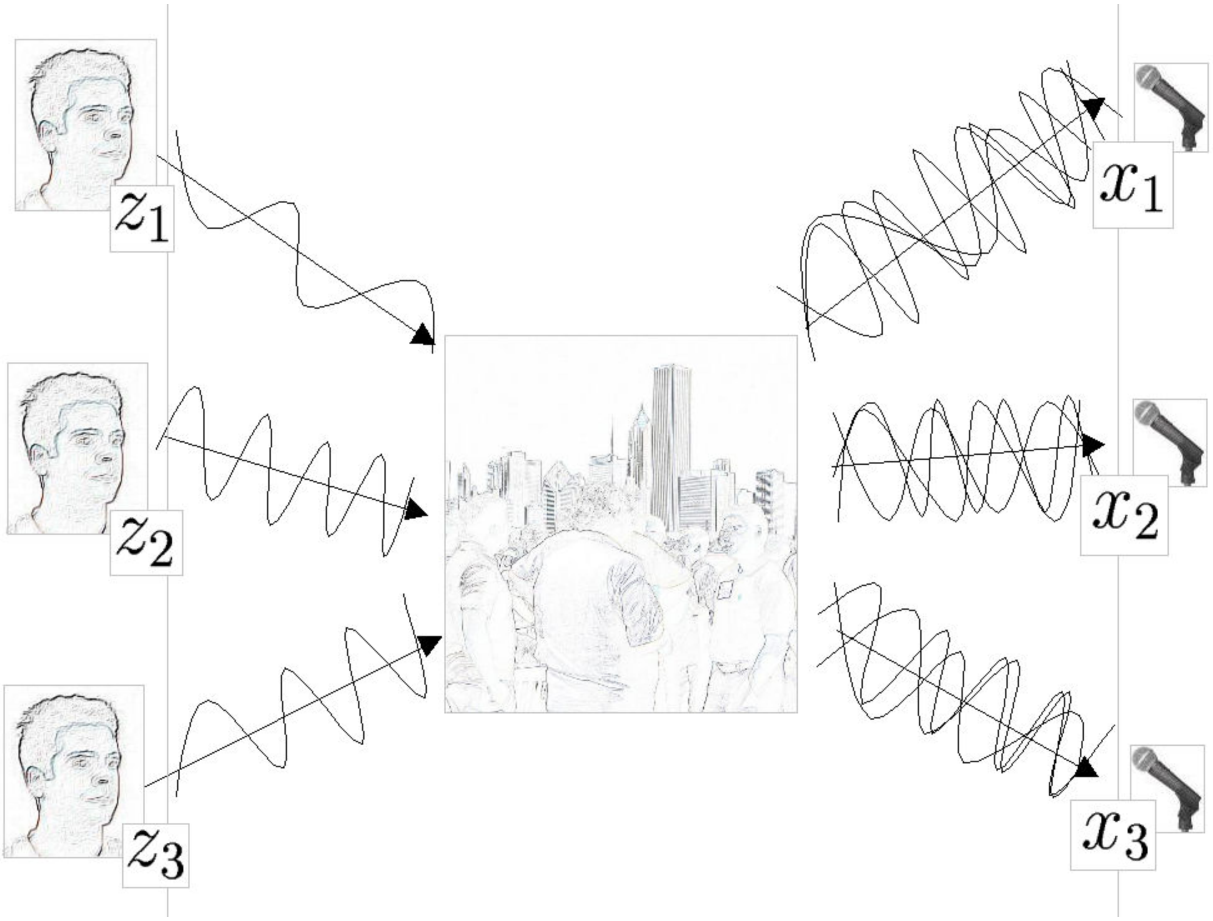

Figure 4: The Cocktail Party Problem: sound waves from $J = 3$ independent speakers ($\mathbf{z}_1$, $\mathbf{z}_2$ and $\mathbf{z}_3$ left) are superimposed at a *cocktail party* (center), and are recorded as three mixed source vectors, $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ on $N = 3$ microphones (right). The $M \times J$ observations (or recordings), $\mathbf{X}^\mathbf{T}$ of the underlying sources, $\mathbf{Z}^\mathbf{T}$, are a linear mixture of the sources, such that $\mathbf{X}^\mathbf{T} = \mathbf{A}\mathbf{Z}^\mathbf{T}$, where $\mathbf{A}$ is a $J \times N$ linear mixing matrix. An estimate $\mathbf{Y}^\mathbf{T}$, of the $M \times J$ sources $\mathbf{Z}^\mathbf{T}$, is made by calculating a demixing matrix $\mathbf{W}$, which acts on $\mathbf{X}^\mathbf{T}$ such that $\mathbf{Y}^\mathbf{T} = \mathbf{W}\mathbf{X}^\mathbf{T} = \hat{\mathbf{Z}}^\mathbf{T}$ and $\mathbf{W} \approx \mathbf{A}^{-1}$.

random vectors, $\mathbf{A}$ a $N \times J$ mixing matrix and $\mathbf{Z}$, the J (assumed) source vectors such that

$$\mathbf{X^T} = \mathbf{AZ^T} \tag{12}$$

Note that here we have chosen to use the transposes of $\mathbf{X}$ and $\mathbf{Z}$ to retain dimensional consistency with the PCA formulation in § 15.3, Eq. 4. ICA algorithms attempt to find a separating or demixing matrix $\mathbf{W}$ such that

$$\mathbf{Y^T} = \mathbf{WX^T} \tag{13}$$

where $\mathbf{W} = \hat{\mathbf{A}}^{-1}$, an approximation of the inverse of the original mixing matrix, and $\mathbf{Y^T} = \hat{\mathbf{Z}}^{\mathbf{T}}$, an $M \times J$ matrix, is an approximation of the underlying sources. These sources are assumed to be statistically independent (generated by unrelated processes) and therefore the joint PDF is the product of the densities for all sources:

$$P(Z) = \prod p(z_i) \tag{14}$$

where $p(z_i)$ is the PDF of the i^{th} source and $P(Z)$ is the joint density function.

The basic idea of ICA is to apply operations to the observed data $\mathbf{X^T}$, or the de-mixing matrix, $\mathbf{W}$, and measure the independence between the output signal channels, (the columns of $\mathbf{Y^T}$) to derive estimates of the sources, (the columns of $\mathbf{Z^T}$). In practice, iterative methods are used to maximize or minimize a given cost function such as **mutual information**, **entropy** or the fourth order moment, **kurtosis**, a measure of non-Gaussianity (see § 15.4). We shall see later how entropy-based cost functions are related to kurtosis and therefore all of the cost functions are a measure of non-Gaussianity to some extent[17]. From the **Central Limit Theorem**[5], we know that the distribution of a sum of independent random variables tends toward a Gaussian distribution. That is, a sum of two independent random variables usually has a distribution that is closer to Gaussian than the two original random variables. In other words, independence *is* non-Gaussianity. In ICA, if we wish to find *independent* sources, we must find a demixing matrix $\mathbf{W}$ that maximizes the non-Gaussianity of each source. It should also be noted at this point that determining the number of sources in a signal matrix is outside the scope of this chapter[18], and we shall stick to the convention $J \equiv N$, the number of sources equals the dimensionality of the signal (the number of independent observations). Furthermore, in conventional ICA, we can never recover more sources than the number of independent observations ($J \ngtr N$), since this is a form of interpolation and a model of the underlying source signals would have to be used. (In terms of §15.2, we have a subspace with a higher dimensionality than the original data[19].)

The essential difference between ICA and PCA is that PCA uses variance, a second order moment, rather than higher order statistics (such as the fourth moment, kurtosis) as a

[17] The reason for choosing between different cost functions is not always made clear, but computational efficiency and sensitivity to outliers are among the concerns; see § 15.5. The choice of cost function also determines whether we uncover sub- or super-Gaussian sources; see §15.6.

[18] See articles on **relevancy determination** [6, 7].

[19] There are methods for attempting this type of analysis; if there are more sensors than sources, the data are over-determined. If there are less sensors than sources, then the problem is under-determined, but it is still possible to extract sources under certain conditions by exploiting known properties of the sources, such as their dynamics. See [8, 9, 10, 11, 12, 13, 14, 15].

metric to separate the signal from the noise. Independence between the projections onto the eigenvectors of an SVD is imposed by requiring that these basis vectors be orthogonal. The subspace formed with ICA is not necessarily orthogonal and the angles between the axes of projection depend upon the exact nature of the data used to calculate the sources.

The fact that SVD imposes orthogonality means that the data has been decorrelated (the projections onto the eigenvectors have zero covariance). This is a much weaker form of independence than that imposed by ICA[20]. Since independence implies uncorrelatedness, many ICA methods constrain the estimation procedure such that it always gives uncorrelated estimates of the independent components. This reduces the number of free parameters, and simplifies the problem.

Gaussianity

We will now look more closely at what the kurtosis of a distribution means, and how this helps us separate component sources within a signal by imposing independence. The first two moments of random variables are well known; the *mean* and the *variance*. If a distribution is Gaussian, then the mean and variance are sufficient characterize variable. However, if the PDF of a function is not Gaussian then many different signals can have the same mean and variance. (For instance, all the signals in Fig. 6 have a mean of zero and unit variance.

Recall from earlier chapters that the mean (central tendency) of a random variable x, is defined to be

$$\mu_x = E\{x\} = \int_{-\infty}^{+\infty} x p_x(x) \mathrm{d}x \tag{15}$$

where $E\{\}$ is the expectation operator, $p_x(x)$ is the probability that x has a particular value. The variance (second central moment), which quantifies the spread of a distribution is given by

$$\sigma_x^2 = E\{(x-\mu_x)^2\} = \int_{-\infty}^{+\infty} (x-\mu_x)^2 p_x(x) \mathrm{d}x \tag{16}$$

and the square root of the variance is equal to the standard deviation, σ, of the distribution. By extension, we can define the N^{th} central moment to be

$$\upsilon_n = E\{(x-\mu_x)^n\} = \int_{-\infty}^{+\infty} (x-\mu_x)^n p_x(x) \mathrm{d}x \tag{17}$$

The third moment of a distribution is known as the *skew*, ζ, and characterizes the degree of asymmetry about the mean. The skew of a random variable x is given by $\upsilon_3 = \frac{E\{(x-\mu_x)^3\}}{\sigma^3}$. A positive skew signifies a distribution with a tail extending out toward a more positive value and a negative skew signifies a distribution with a tail extending out toward a more negative (see Fig. 5a).

[20] Orthogonality implies independence, but independence does not necessarily imply orthogonality.

The fourth moment of a distribution is known as *kurtosis* and measures the relative peakedness of flatness of a distribution with respect to a Gaussian (normal) distribution. See Fig. 5b. It is defined in a similar manner to be

$$\kappa = \upsilon_4 = \frac{E\{(x-\mu_x)^4\}}{\sigma^4} \tag{18}$$

Note that the kurtosis of a Gaussian is equal to 3 (whereas the first three moments of a distribution are zero)[21]. A distribution with a positive kurtosis (> 3 in Eq. (20)) is termed *leptokurtic* (or super-Gaussian). A distribution with a negative kurtosis (< 3 in Eq. (20)) is termed *platykurtic* (or sub-Gaussian). Gaussian distributions are termed *mesokurtic*. Note also that skewness and kurtosis are normalized by dividing the central moments by appropriate powers of σ to make them dimensionless.

These definitions are however, for continuously valued functions. In reality, the PDF is often difficult or impossible to calculate accurately and so we must make empirical approximations of our sampled signals. The standard definition of the mean of a vector $\mathbf{x}$ with M values ($\mathbf{x} = [x_1, x_2, ..., x_M]$) is

$$\hat{\mu}_x = \frac{1}{M}\sum_{i=1}^{M} x_i$$

the variance of $\mathbf{x}$ is given by

$$\hat{\sigma}^2(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M}(x_i - \hat{\mu}_x)^2$$

and the skewness is given by

$$\hat{\zeta}(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M}\left[\frac{x_i - \hat{\mu}_x}{\hat{\sigma}}\right]^3. \tag{19}$$

The empirical estimate of kurtosis is similarly defined by

$$\hat{\kappa}(x) = \frac{1}{M}\sum_{i=1}^{M}\left[\frac{x_i - \hat{\mu}_x}{\hat{\sigma}}\right]^4 \tag{20}$$

Fig. 6 illustrates the time series, power spectra and distributions of different signals and noises found in the ECG recording. From left to right: (i) the underlying Electrocardiogram signal, (ii) additive (Gaussian) observation noise, (iii) a combination of muscle artifact (MA) and baseline wander (BW), and (iv) powerline interference; sinusoidal noise with f ≈33Hz ± 2Hz. Note that all the signals have significant power contributions within the frequency of interest (< 40Hz) where there exists clinically relevant information in the ECG. Traditional filtering methods therefore cannot remove these noises without severely distorting the underlying ECG.

[21] The proof of this is left to the reader, but noting that the general form of the normal distribution is $p_x(x) = \frac{e^{-(x-\mu_x^2)/2\sigma^2}}{\sigma\sqrt{2\pi}}$, and $\int_{-\infty}^{\infty} e^{-ax^2}dx = \sqrt{\pi/a}$ should help (especially if you differentiate the integral twice). Note also then, that the above definition of kurtosis (and Eq. (20)) sometimes has an extra -3 term to make a Gaussian have zero kurtosis, such as in Numerical Recipes in C. Note that Matlab uses the convention without the -3 term and therefore Gaussian distributions have a $\kappa = 3$. This convention is used in the laboratory assignment that accompanies these notes.

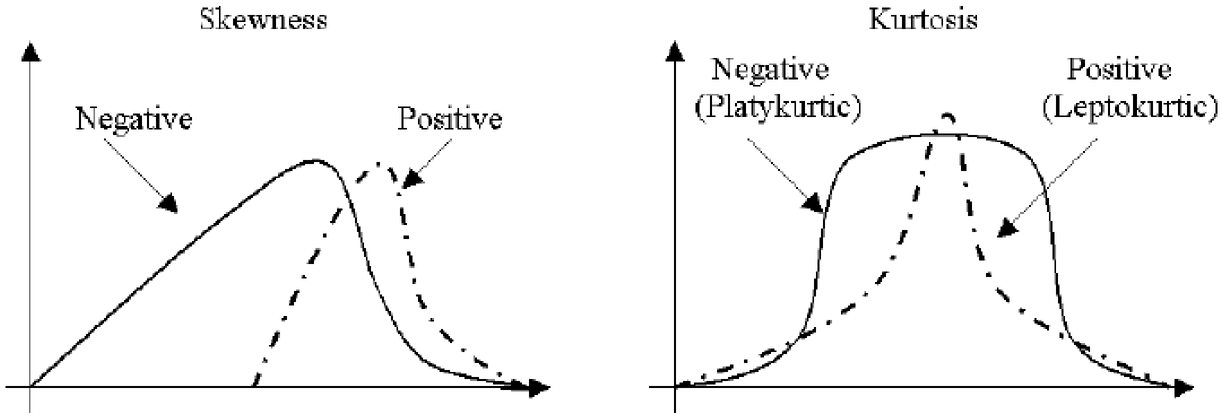

Figure 5: Distributions with third and fourth moments [skewness, (a) and kurtosis (b) respectively] that are significantly different from normal (Gaussian).

15.4.3 ICA for removing noise on the ECG

Figure 7 illustrates the effectiveness of ICA in removing artifacts from the ECG. Here we see 10 seconds of 3 leads of ECG before and after ICA decomposition (upper and lower graphs respectively). The upper plot (a) is the same data as in Fig. 1. Note that ICA has separated out the observed signals into three specific sources; 1b) The ECG, 2b) High kurtosis transient (movement) artifacts, and 2c) Low kurtosis continuous (observation) noise. In particular, ICA has separated out the *in-band* QRS-like spikes that occurred at 2.6 and 5.1 seconds. Furthermore, time-coincident artifacts at 1.6 seconds that distorted the QRS complex, were extracted, leaving the underlying morphology intact.

Relating this to the cocktail party problem, we have three *'speakers'* in three locations. First and foremost we have the series of cardiac depolarization/repolarization events corresponding to each heartbeat, located in the chest. Each electrode is roughly equidistant from each of these. Note that the amplitude of the third lead is lower than the other two, illustrating how the cardiac activity in the heart is not spherically symmetrical. Another source (or 'speaker') is the perturbation of the contact electrode due to physical movement. The third 'speaker' is the Johnson (thermal) observation noise.

However, we should not assume that ICA is a panacea to remove all noise. In most situations, complications due to lead position, a low signal-noise ratio, and positional changes in the sources cause serious problems. Section 15.8 addresses many of the problems in employing ICA, using the ECG as a practical illustrative guide.

It should also be noted that the ICA decomposition does not necessarily mean the relevant clinical characteristics of the ECG have been preserved (since our interpretive knowledge of the ECG is based upon the observations, not the sources). Therefore, in order to reconstruct the original ECGs in the absence of noise, we must set to zero the columns of the demixing matrix that correspond to artifacts or noise, then invert it and multiply by the

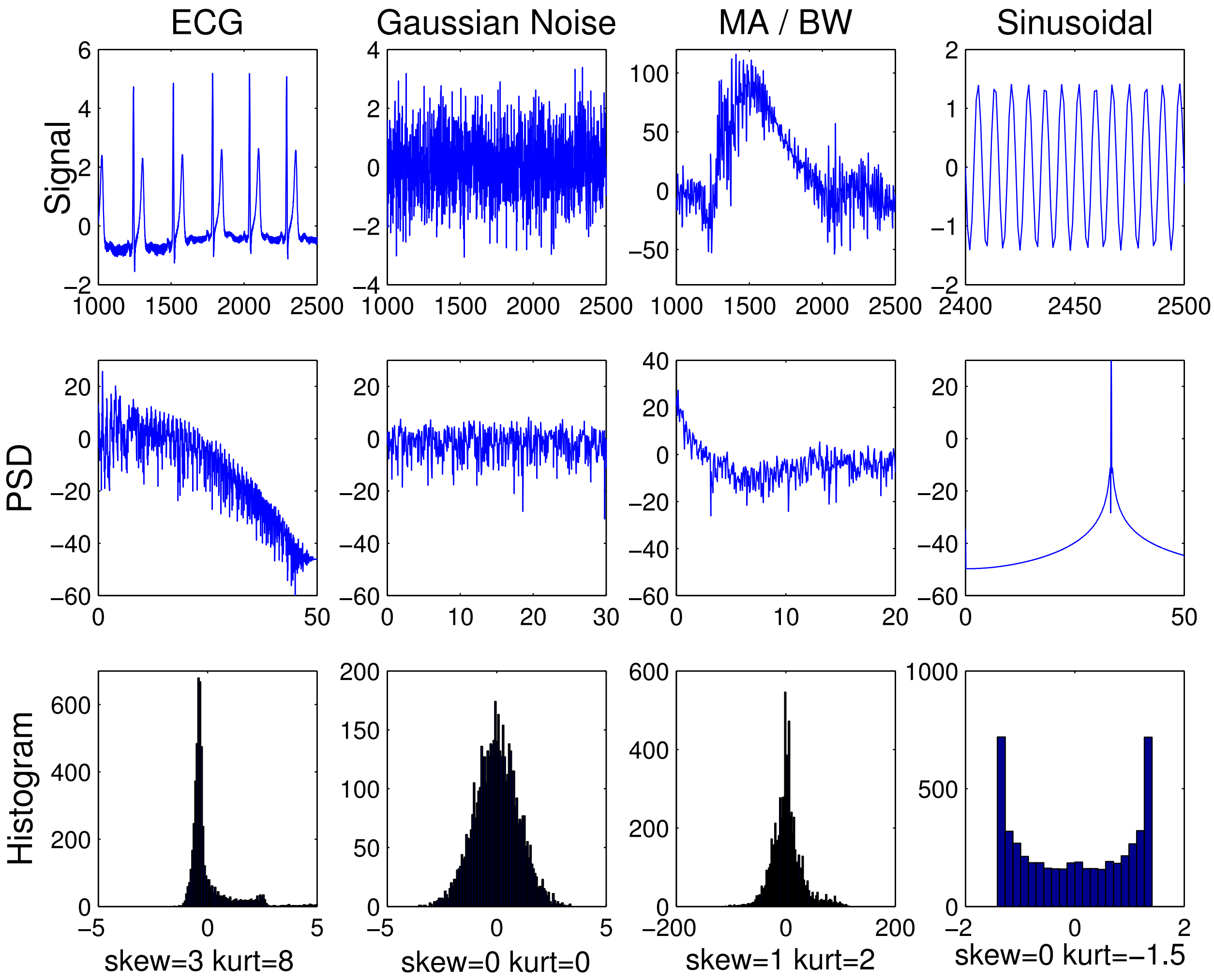

Figure 6: time Series, power spectra and distributions of different signals and noises found on the ECG. From left to right: (i) the underlying Electrocardiogram signal, (ii) additive (Gaussian) observation noise, (iii) a combination of muscle artifact (MA) and baseline wander (BW), and (iv) powerline interference; sinusoidal noise with $f \approx 33Hz \pm 2Hz$.

decomposed data to 'restore' the original ECG observations (see Fig. 2. An example of this procedure using the data in Fig. 1 and Fig. 7 are presented in Fig. 8. In terms of Fig. 2 and our general ICA formalism, the estimated sources $\hat{\mathbf{Z}}$ (Fig. 7b) are recovered from the observation $\mathbf{X}$ (Fig. 7a) by estimating a demixing matrix $\mathbf{W}$. It is no longer obvious which lead the underlying source (signal 1 in Fig. 7b) corresponds to.In fact, this source does not correspond to any clinical lead at all, just some transformed combination of leads. In order to perform a diagnosis on this lead, the source must be projected back into the observation domain by inverting the demixing matrix $\mathbf{W}$. It is at this point that we can perform a removal of the noise sources. Columns of $\mathbf{W}^{-1}$ that correspond to noise and/or artifact (signal 2 and signal 3 on Fig. 7b in this case) are set to zero ($\mathbf{W}^{-1} \rightarrow \mathbf{W}_p^{-1}$), where the number of non-noise sources, $p = 1$, and the filtered version of each clinical lead of $\mathbf{X}$, is reconstructed in the observation domain using $\mathbf{X}_{filt} = \mathbf{W}_p^{-1}\mathbf{Y}$ to reveal a cleaner 3-lead ECG (Fig. 8).

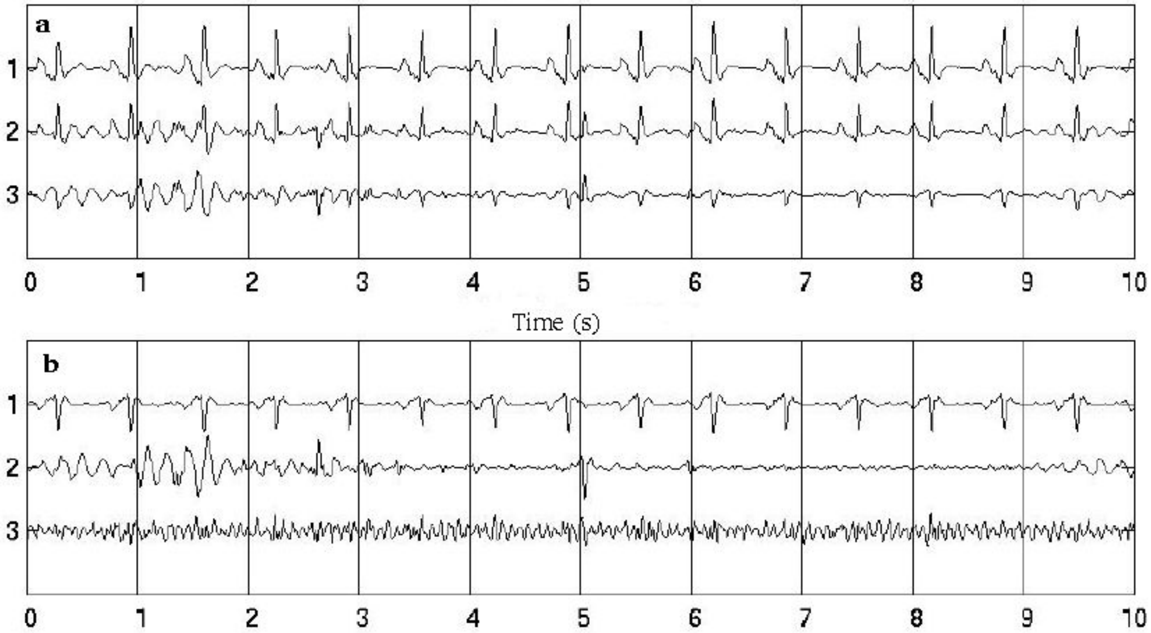

Figure 7: 10 seconds of 3 Channel ECG **a**) before ICA decomposition and **b**) after ICA decomposition. Plot **a** is the same data as in Fig. 1. Note that ICA has separated out the observed signals into three specific sources; 1 **b**) The ECG, 2 **b**) High kurtosis transient (movement) artifacts, and 2 **c**) Low kurtosis continuous (observation) noise.

15.5 Different methods for performing ICA - choosing a cost function

Although the basic idea behind ICA is very simple, the actual implementation can be formulated from many perspectives:

- **Maximum likelihood** (MacKay [16], Pearlmutter & Parra [17], Cardoso [18], Girolami & Fyfe [19])

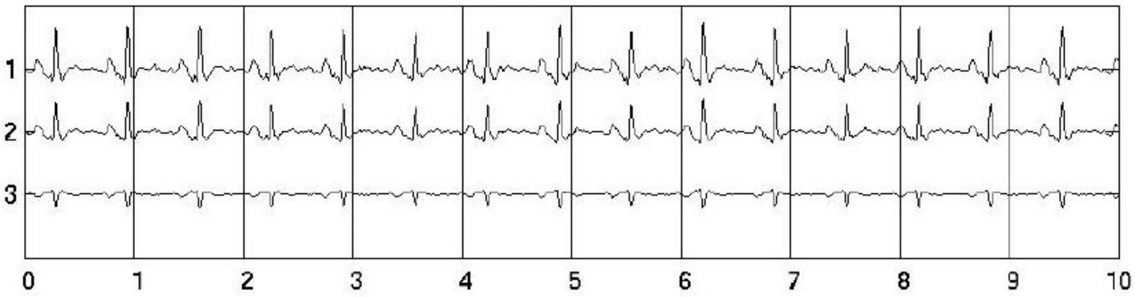

Figure 8: 10 seconds of data (from Fig. 1) after ICA decomposition, (see Fig 7) and reconstruction with noise channels set to zero.

- **Higher order moments and cumulants** (Comon [20], Hyvärinen & Oja [21],)
- **Maximization of information transfer** (Bell & Sejnowski [22], Amari *et al.* [23]; Lee *et al.* [24])
- **Negentropy maximization** (Girolami & Fyfe [19])
- **Non-linear PCA** (Karhunen *et al.* [25, 26], Oja *et al.* [27])

All the above methods use separation metrics (or cost functions) that are essentially equivalent to measuring the non-Gaussianity of the estimated sources. The actual implementation can involve either a manipulation of the output data, $\mathbf{Y}$, or a manipulation of the demixing matrix, $\mathbf{W}$. In the remainder of section 15.5 we will examine three common cost functions, *negentropy*, *mutual information* and the *log likelihood*. A method for using these cost functions to determine the elements of $\mathbf{W}$, *gradient descent*, (or ascent) which is described in section 15.5.3 and appendix 15.9.4.

15.5.1 Negentropy instead of kurtosis as a cost function

Although kurtosis is theoretically a good measure of non-Gaussianity, kurtosis is disproportionately sensitive to changes in the distribution tails. Therefore, other measures of independence are often used. Another important measure of non-Gaussianity is given by **negentropy**. Negentropy is often a more robust (outlier insensitive) method for estimating the fourth moment. Negentropy is based on the information-theoretic quantity of (differential) entropy. The more random (i.e. unpredictable and unstructured the variable is) the larger its entropy. More rigorously, entropy is closely related to the coding length of the random variable, in fact, under some simplifying assumptions, entropy is the coding length of the random variable. The entropy H of a discrete random variable y_i with probability distribution $P(y_i)$ is defined as

$$H(y) = -\sum_i P(y_i) \log_2 P(y_i). \tag{21}$$

This definition can be generalized for continuous-valued random variables and vectors, in which case it is called differential entropy. The differential entropy H of a random vector $\mathbf{y}$ with a probability density function $P(\mathbf{y})$ is defined as

$$H(\mathbf{y}) = -\int P(\mathbf{y}) \log_2 P(\mathbf{y}) \mathbf{d}\mathbf{y}. \tag{22}$$

A fundamental result of information theory is that a Gaussian variable has the largest entropy among all random variables of equal variance [28]. This means that entropy could be used as a measure of non-Gaussianity. In fact, this shows that the Gaussian distribution is the "most random" or the least structured of all distributions. Entropy is small for distributions that are clearly concentrated on certain values, i.e., when the variable is clearly clustered, or has a PDF that is very "spiky".

To obtain a measure of non-Gaussianity that is zero for a Gaussian variable and always non-negative[22], we can use a slightly modified version of the definition of differential entropy, called **negentropy**. Negentropy, $\mathcal{J}$, is defined as follows

$$\mathcal{J}(\mathbf{y}) = H(\mathbf{y}_G) - H(\mathbf{y}) \tag{23}$$

where $\mathbf{y}_G$ is a Gaussian random variable of the same covariance matrix as $\mathbf{y}$. Negentropy is always non-negative, and is zero if and only if $\mathbf{y}$ has a Gaussian distribution. Negentropy has the additional interesting property that it is constant for a particular vector which undergoes an invertible linear transformation, such as in the ICA mixing-demixing paradigm.

The advantage of using negentropy, or, equivalently, differential entropy, as a measure of non-Gaussianity is that it is well justified by statistical theory. In fact, negentropy is in some sense the optimal estimator of non-Gaussianity, as far as statistical properties are concerned. The problem in using negentropy is, however, that it is difficult to compute in practice. Estimating negentropy using the definition above would require an estimate (possibly non-parametric) of the probability density function. Therefore, simpler approximations of negentropy are used. One such approximation actually involves kurtosis:

$$\mathcal{J}(y) \approx \frac{1}{12} E\{y^3\}^2 + \frac{1}{48}\kappa(y)^2 \tag{24}$$

but this suffers from the problems encountered with kurtosis. Another estimate of negentropy involves entropy:

$$\mathcal{J}(y) \approx \left[E\{g(y)\} - E\{g(\vartheta)\}\right], \tag{25}$$

where ϑ is a zero mean unit variance Gaussian variable and the function g is some non-quadratic function which leads to the approximation always being non-negative (or zero if y has a Gaussian distribution). g is usually taken to be $\frac{1}{\alpha}\ln\cosh(\alpha y)$ or $g(y) = -e^{-\frac{y^2}{2}}$ with α some constant $(1 \leq \alpha \leq 2)$. If $g(y) = y$, Eq. 25 degenerates to the definition of kurtosis.

$\mathcal{J}(y)$ is then the cost function we attempt to minimize between the columns of $\mathbf{Y}$. We will see how to minimize a cost function to calculate the demixing matrix in section 15.6.

[22] Therefore, separation of the independent components is achieved by attempting to make negentropy as close to zero as possible (and hence making the sources maximally non-Gaussian).

15.5.2 Mutual Information based ICA

Using the concept of differential entropy, we define the mutual information (MI) $\mathcal{I}$ between M (scalar) random variables, y_i, $i = 1...M$ as follows

$$\mathcal{I}(y_1, y_2, ..., y_M) = \sum_{i=1}^{M} H(y_i) - H(\mathbf{y}). \tag{26}$$

MI is a measure of the (in-) dependence between random variables. MI is always non-negative, and zero if and only if the variables are statistically independent. MI therefore takes into account the whole dependence structure of the variables, and not only the covariance (as is the case for PCA).

Note that for an invertible linear transformation $\mathbf{Y^T} = \mathbf{WX^T}$,

$$\mathcal{I}(y_1, y_2, ..., y_M) = \sum_{i=1}^{M} H(y_i) - H(x_i) - \log_2 \|\mathbf{W}\|. \tag{27}$$

If we constrain the y_i to be uncorrelated and have unit variance $E\{\mathbf{y^T y}\} = \mathbf{W}E\{\mathbf{x^T x}\}\mathbf{W^T}$ $= \mathbf{I}$. This implies that $\|\mathbf{I}\| = 1 = (\|\mathbf{W}E\{\mathbf{x^T x}\}\mathbf{W^T}\|) = \|\mathbf{W}\|\|E\{\mathbf{x^T x}\}\|\|\mathbf{W^T}\|$ and hence $\|\mathbf{W}\|$ must be constant. If y_i has unit variance, MI and negentropy differ only by a constant, and a sign;

$$\mathcal{I}(y_1, y_2, ..., y_m) = c - \sum_{i=1}^{M} \mathcal{J}(y_i) \tag{28}$$

where c is a constant. This shows the fundamental relationship between MI and negentropy and hence with kurtosis.

Since MI is a measure of the (mutual) information between two functions, finding a $\mathbf{W}$ which minimises $\mathcal{I}$ between the columns of $\mathbf{Y^T}$ in the transformation $\mathbf{Y^T} = \mathbf{WX^T}$ leads to method for determining the independent components (sources) in our observations $\mathbf{X^T}$.

15.5.3 Maximum Likelihood

Independent component analysis can be thought of as a statistical modeling technique that makes use of **latent variables** to describe a probability distribution over the observables. This is known as **generative latent variable modeling** and each source is found by deducing its corresponding distribution. Following MacKay [16], we can model the J observable vectors $\{x_j\}_{j=1}^{J}$ as being generated from latent variables $\{z_i\}_{i=1}^{N}$ via a linear mapping $\mathbf{W}$ with elements[23] w_{ij}. To simplify the derivation we assume the number of sources equals the number of observations ($N = J$), and the data are then defined to be, $D = \{\mathbf{X}\}_{m=1}^{M}$, where M is the number of samples in each of the J observations. The latent variables are assumed to be independently distributed with marginal distributions $P(z_i) \equiv p_i(z_i)$, where p_i denotes the assumed probability distributions of the latent variables.

[23] Note that the transpose of w_{ij} is written w_{ji}.

Given $\mathbf{A} \equiv \mathbf{W}^{-1}$, the probability of the observables $\mathbf{X}$ and the hidden variables $\mathbf{Z}$ is

$$\begin{aligned} P(\{\mathbf{X}\}_{m=1}^{M}, \{\mathbf{Z}\}_{m=1}^{M}|\mathbf{W}) &= \prod_{m=1}^{M} \left[P(\mathbf{x}^m|\mathbf{z}^m, \mathbf{W})P(\mathbf{z}^m)\right] & (29) \\ &= \prod_{m=1}^{M} \left[\left(\prod_{j=1}^{J}(x_j^m - \sum_i w_{ji}z_i^m)\right)\left(\prod_i p_i(z_i^m)\right)\right]. & (30) \end{aligned}$$

Note that for simplicity we have assumed that the observations $\mathbf{X}$ have been generated without noise[24]. If we replace the term $(x_j - \sum_i w_{ji}z_i)$ by a (noise) probability distribution over x_j with mean $\sum_i w_{ji}z_i$ and a small standard deviation, the identical algorithm results [16].

To calculate w_{ij}, the elements of $\mathbf{W}$ we can use the method of gradient descent which requires the optimization of a dimensionless objective function $\text{Ł}(\mathbf{W})$, of the parameters (the weights). The sequential update of the elements of the mixing matrix, w_{ij}, are then computed as

$$\Delta w_{ij} = \eta \frac{\partial \text{Ł}}{\partial w_{ij}} \tag{31}$$

where η is the learning rate[25].

The cost function $\text{Ł}(\mathbf{W})$ we wish to minimize to perform ICA (to maximize independence) is the **log likelihood function**

$$\text{Ł}(\mathbf{W}) = \log_2\left(P(\mathbf{X}|\mathbf{W})\right) = \log_2\left(\prod_{m=1}^{M} P(\mathbf{x}^m|\mathbf{W})\right) \tag{32}$$

which is the log of the product of the (independent) factors. Each of the factors is obtained by *marginalizing* over the latent variables, which can be shown ([16], appendix 15.9.3) to be equal to

$$\text{Ł}(\mathbf{W}) = \log_2 |\det \mathbf{W}| + \sum_{\imath} \log_2 p_i(w_{ij}^{-1}\mathbf{x}_j). \tag{33}$$

15.6 Gradient descent to find the de-mixing matrix W

In order to find $\mathbf{W}$ we can iteratively update its elements w_{ij}, using gradient descent or ascent on the objective function $\text{Ł}(\mathbf{W})$. To obtain a maximum likelihood algorithm, we find the gradient of the log likelihood. This turns out to be

$$\frac{\partial}{\partial w_{ij}} \log_2 P(\mathbf{x}^m|\mathbf{A}) = a_{ji} + x_j z_i \tag{34}$$

$\text{Ł}(\mathbf{W})$ can be used to 'guide' a gradient ascent of the elements of $\mathbf{W}$ and maximise the log likelihood of each source. If we choose $\mathbf{W}$ so as to ascend this gradient, we obtain

[24] This leads to the Bell-Sejnowski algorithm [16, 22].

[25] Which can be fixed or variable to aid convergence depending on the form of the underlying source distributions.

the learning algorithm from Bell and Sejnowski [22] ($\Delta\mathbf{W} \propto [\mathbf{W^T}]^{-1} + \mathbf{zx^T}$). A detailed mathematical analysis of gradient descent/ascent and its relationship to PCA and neural networks are given in appendix 15.9.4. (Treat this as optional reading).

In general, the learning algorithm for ICA can be summarized as a linear mapping, such that $\mathbf{Y^T} = \mathbf{WX^T}$, followed by a non-linear mapping $y_i \rightarrow f(y_i)$. f is a non-linear function that helps the elements of $\mathbf{W}$ converge to values that give maximal statistical independence between the rows of $\mathbf{Y}$. In practice, the choice of the non-linearity, $f(y_i)$, in the update equations for learning $\mathbf{W}$ is heavily dependent on the distribution of the underlying sources. For example, if we choose a traditional $\tanh$ non-linearity ($f(y_i) = -\tanh(\beta y_i)$), with β a constant initially equal to unity, then we are implicitly assuming the source densities are heavier tailed distributions than a Gaussian ($p_i(z_i) \propto 1/\cosh(z_i) \propto 1/(e^{z_i} + e{-}z_i)$, $z_i = f(y_i)$, with $f = -tanh(y_i)$). Varying β reflects our changing beliefs in the underlying source distributions. In the limit of large β, the non-linearity becomes a step function and $p_i(z_i)$ becomes a biexponential distribution ($p_i(z_i) \propto e^{-|z|}$). As β tends to zero, $p_i(z_i)$ approach more Gaussian distributions.

If we have no non-linearity, $f(y_i) \propto -y_i$, the we are implicitly assuming a Gaussian distribution on the latent variables. However, it is well known [4, 29] that without a non-linearity, the gradient descent algorithm leads to second order decorrelation. That is, we perform the same function as PCA. Equivalently, the Gaussian distribution on the latent variables is invariant under rotation of the latent variables, so there is no information to enable us to find a preferred alignment of the latent variable space. This is one reason why conventional ICA is only able to separate non-Gaussian sources. See [16], [30] and appendix 15.9.3 for further discussion on this topic.

15.7 Applications of ICA

Apart from the example given in §15.4.3, ICA has been used to perform signal separation in many different domains. These include:

- Blind source separation; Watermarking, Audio [31, 32], ECG, (Bell & Sejnowski [22], Barros *et al.* [33], McSharry *et al.* [13]), EEG (Mackeig *et al.* [34, 35],).
- Signal and image denoising (Hyvärinen - [36]), medical (fMRI - [37]) ECG & EEG (Mackeig *et al.* [34, 35])
- Modeling of the hippocampus and visual cortex (Lörincz, Hyvärinen [38])
- Feature extraction and clustering, (Marni Bartlett, Girolami, Kolenda [39])
- Compression and redundancy reduction (Girolami, Kolenda, Ben-Shalom [40])

Each particular domain involves subtle differences in the statistical structure of the sources in the data which affect the particular choice of the ICA algorithm. Pre-processing steps (sometimes including the application of PCA) are extremely important too. However, we do not have the space to go into detail for each of these applications and the reader is encouraged to explore the above references at this point.

15.8 Limitations of ICA

While ICA is a powerful technique with few assumptions on the nature of the observations and the underlying sources, it must be remembered that ICA does have some intrinsic limitations.

15.8.1 The permutation and scaling problem

Recall Eq. (12), $\mathbf{X^T} = \mathbf{AZ^T}$. We may inset an arbitrary matrix $\mathbf{B}$ and its inverse $\mathbf{B}^{-1}$ such that

$$\mathbf{X^T} = \mathbf{ABB}^{-1}\mathbf{Z^T} \tag{35}$$

and Eq. (12) remains unchanged. The mixing matrix is now $\mathbf{AB}$ and the sources are now $\mathbf{B}^{-1}\mathbf{Z^T}$ with a different column order and a different scaling. Since we only know $\mathbf{X}$, we can only solve jointly for the mixing matrix and the sources and an infinite number of (equally valid) pairs are possible. Therefore, the estimates of the sources may appear in an arbitrary (column) order (which change with small changes in the observations), and with arbitrary scaling, which has no relation to the amplitude or energy in the underlying sources.

Another way to think about this problem, is that we derive the estimate of the demixing matrix ($\mathbf{W} \approx \mathbf{A}^{-1}$) by optimising a cost function *between* the columns of the estimate of the sources $\mathbf{\hat{Z}^T}$. This cost function measures independence in a manner that is amplitude independent. (Recall that kurtosis is a dimensionless quantity.) In order to mitigate for this problem, some ICA algorithms order the sources in terms of kurtosis and scale them to have unit variance. To preserve the original amplitude of the source, it is possible to invert the transformation, retaining only a single source, and reconstruct each source back in the observation domain. Therefore an accurate knowledge of certain features or properties of the underlying sources (such as distinguishing morphological oscillations in the time domain or the exact value of the kurtosis) is required to identify a particular source in the columns of $\mathbf{\hat{Z}^T}$.

15.8.2 Stationary Mixing

ICA assumes a linear **stationary** mixing model (the mixing matrix is a set of constants independent of the changing structure of the data over time). However, for many applications this is only true from certain observation points or for very short lengths of time. For example, consider the earlier case of noise on the ECG. As the subject inhales, the chest expands and the diaphragm lowers. This causes the heart to drop and rotate slightly. If we consider the mixing matrix $\mathbf{A}$ to be composed of a stationary component $\mathbf{A}^s$ and a non-stationary component $\mathbf{A}^{ns}$ such that $\mathbf{A} = \mathbf{A}^{s} + \mathbf{A}^{ns}$ then $\mathbf{A}^{ns}$ is equal to some constant (α) times one of the rotation matrices[26] such as

$$\mathbf{A}^{ns}(\theta) = \alpha \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta)t & -\sin(\theta)t \\ 0 & \sin(\theta)t & \cos(\theta)t \end{bmatrix},$$

where $\theta = 2\pi f_{resp}$ and f_{resp} is the frequency of respiration[27]. In this case, α will be a function of θ, the angle between the different sources (the electrical signals from muscle contractions and those from cardiac activity), which itself is a function of time. This is only true for small values of α, and hence a small angle θ, between each source. This is a major reason for the differences in effectiveness of ICA for source separation for different lead configurations.

15.8.3 The assumption of independence

The sources (columns of $\mathbf{Z^T}$) mixed by $\mathbf{A}$ are assumed to statistically independent. That is, they are generated from some underlying processes that are unrelated. In the cocktail party problem, this is trivially obvious; each speaker is not modulating their words as a function of any other words being spoken at the same time. However, in the case of the ECG noise/artifact removal, this is sometimes not true. When a monitored subject suddenly increases their activity levels, artifacts from muscle movements can manifest on the ECG. Sometimes, there will be a significant changes in heart rate or beat morphology as a result of the activity change. The muscle artifact and beat morphology change are no longer independent. If the relationship is strong enough, then ICA will not be able to separate the sources.

15.8.4 Under- or over-determined representations and relevancy determination

Throughout these notes we have assumed that the number of sources is exactly equal to the number of observations. However, this is rarely true. In the case of the cocktail party, we usually have two microphones (ears) and more than two independent sources (all the other speakers in the room plus any ambient noises such as music). Our representation is therefore under-determined and we need to modify the standard ICA formulation to deal with this. See [41, 9, 10] for an analysis of this problem.

Conversely, we may have more observations than sources, as in the case of a 12-lead ECG. Apart from the problem of determining which sources are relevant, the actual estimate of each source will change depending on how many sources are assumed to be be in the mixture (observation). Therefore, an accurate determination of the number of sources can prove to be important. See Roberts *et al.* [6, 7] and Joho *et al.* [8] for further discussions on this topic.

[26] See Eq. 118 in appendix 15.9.6.

[27] Assuming an idealized sinusoidal respiratory cycle.

15.9 Summary and further reading

In this chapter we have explored how we can apply a transformation to a set of observations in order to project them onto a set of axes that are in some way more informative than the observation space. This is achieved by defining some contrast function between the data in the projected subspace which is essentially a measure of independence. If this contrast function is second order (variance) then we perform decorrelation through PCA. If the contrast function is fourth order and therefore related to Gaussianity, then we achieve ICA. The cost function measured between the estimated sources that we use in the iterative update of the demixing matrix (and the manner in which we update it to explore the weight space) encodes our prior beliefs as to the non-Gaussianity (kurtosis) of the source distributions. The data projected onto the independent (source) components is as statistically independent as possible. We may then select which projections we are interested in and, after discarding the uninteresting components, invert the transformation to effect a filtering of the data.

ICA covers an extremely broad class of algorithms, as we have already seen. Lee *et al.* [42] show that different theories recently proposed for Independent Component Analysis (ICA) lead to the same iterative learning algorithm for blind separation of mixed independent sources. This is because all the algorithms attempt to perform a separation onto a set of basis vectors that are in some way **independent**, and that the independence can always be recast as a departure from Gaussianity.

However, the concept of blind source separation is far more broad than this chapter reveals. ICA has been the fertile meeting ground of statistical modeling [43], PCA [44], neural networks [45], Independent Factor Analysis [46], Wiener filtering [11, 47, 48], wavelets [49, 47, 50], hidden Markov modeling [51, 7, 52], Kalman filtering [53] and non-linear dynamics [14, 54]. Many of the problems we have presented in this chapter have been addressed by extending the ICA model with these tools. Although these concepts are outside the scope of this course, they are currently the focus of ongoing research. For further reading on ICA and related research, the reader is encouraged to browse the following URLs:

`http://www.cnl.salk.edu/`   `http://www.inference.phy.cam.ac.uk/mackay/ica.pdf`
`http://web.media.mit.edu/~paris/ica.html` `http://www.robots.ox.ac.uk/~sjrob/`

Acknowledgements and Caveats

These notes have evolved over the last 10 years and lean heavily on the work of Te-Won Lee and David Mackay in the latter half of the chapter. Many suggestions and criticisms were gratefully received from Julie Greenberg and John Fisher. The appendices are drawn from my doctoral thesis which relied on input from Lionel Tarassenko and Steve Roberts. Any conceptual errors and remaining mistakes (or any I have subsequently introduced) are entirely my own. Please feel free to email me with any errors you may notice or areas of confusion. I can't guarantee I'll answer, but I would appreciate all feedback to help improve these notes. Please feel free to borrow from them according to the Creative Commons Licensing and giving the relevant citation(s).

References

[1] Jolliffe IT. Principal Component Analysis. New York: Springer-Verlag, 1988.

[2] Golub GH, Van Loan CF. Matrix Computations. 2^{nd} edition. Oxford: North Oxford Academic, 1983.

[3] Moody GB, Mark RG. QRS morphology representation and noise estimation using the Karhunen-Loève transform. Computers in Cardiology 1989;269–272.

[4] Clifford GD, Tarassenko L. One-pass training of optimal architecture auto-associative neural network for detecting ectopic beats. IEE Electronic Letters Aug 2001; 37(18):1126–1127.

[5] Trotter HF. An elementary proof of the central limit theorem. Arch Math 1959; 10:226–234.

[6] Penny W, Roberts S, Everson R. ICA: Model order selection and dynamic source models. In Roberts SJ, Everson R (eds.), Independent Component Analysis: Principles and Practice. Cambridge University Press, 2001; .

[7] Choudrey RA, Roberts SJ. Bayesian ICA with hidden Markov model sources. In International Conference on Independent Component Analysis. Nara, Japan, 2003; 809–814.

[8] Joho M, Mathis H, Lambert R. Overdetermined blind source separation: Using more sensors than source signals in a noisy mixture. In Proc. International Conference on Independent Component Analysis and Blind Signal Separation. Helsinki, Finland, 2000; 81–86.

[9] Lee T, Lewicki M, Girolami M, Sejnowski T. Blind source separation of more sources than mixtures using overcomplete representations. IEEE Sig Proc Lett April 1999; 4(4).

[10] Lewicki MS, Sejnowski TJ. Learning overcomplete representations. Neural Computation 2000;12(2):337–365.

[11] Benaroya L, Donagh LM, Bimbot F, Gribonval R. Non negative sparse representation for Wiener based source separation with a single sensor. Acoustics Speech and Signal Processing 2003 Proceedings ICASSP 03 2003 IEEE International Conference on 6-10 April 2003;6:VI–613–16 vol.6. ISSN 1520-6149.

[12] Clifford GD, McSharry PE. A realistic coupled nonlinear artificial ECG, BP, and respiratory signal generator for assessing noise performance of biomedical signal processing algorithms. Proc of SPIE International Symposium on Fluctuations and Noise 2004;5467(34):290–301.

[13] McSharry PE, Clifford GD. A comparison of nonlinear noise reduction and independent component analysis using a realistic dynamical model of the electrocardiogram. Proc of SPIE International Symposium on Fluctuations and Noise 2004;5467(09):78–88.

[14] James CJ, Lowe D. Extracting multisource brain activity from a single electromagnetic channel. Artificial Intelligence in Medicine May 2003;28(1):89–104.

[15] Broomhead DS, King GP. Extracting qualitative dynamics from experimental data. Physica D 1986;20:217–236.

[16] MacKay DJC. Maximum likelihood and covariant algorithms for independent component analysis, -unpublished, 1996, updated, 2002. http://www.inference.phy.cam.ac.uk/mackay/abstracts/ica.html.

[17] Pearlmutter BA, Parra LC. Maximum likelihood blind source separation: A context-sensitive generalization of ICA. In Mozer MC, Jordan MI, Petsche T (eds.), Advances in Neural Information Processing Systems, volume 9. The MIT Press, 1997; 613.

[18] Cardoso J. Infomax and maximum likelihood for blind source separation. IEEE Signal Processing Letters April 1997;4(4):112–114.

[19] Girolami M, Fyfe C. Negentropy and kurtosis as projection pursuit indices provide generalised ICA algorithms. In A. C, Back A (eds.), NIPS-96 Blind Signal Separation Workshop, volume 8. 1996; .

[20] Comon P. Independent component analysis, a new concept? Signal Processing 1994; 36:287–314.

[21] Hyvärinen A, Oja A. A fast fixed point algorithm for independent component analysis. Neural Computation 1997;9:1483–1492.

[22] Bell AJ, Sejnowski TJ. An information-maximization approach to blind separation and blind deconvolution. Neural Computation 1995;7(6):1129–1159.

[23] Amari S, Cichocki A, Yang HH. A new learning algorithm for blind signal separation. In Touretzky DS, Mozer MC, Hasselmo ME (eds.), Advances in Neural Information Processing Systems, volume 8. The MIT Press, 1996; 757–763.

[24] Lee TW, Girolami M, Sejnowski TJ. Independent component analysis using an extended infomax algorithm for mixed sub-Gaussian and super-Gaussian sources. Neural Computation 1999;11(2):417–441.

[25] Karhunen J, Joutsensalo J. Representation and separation of signals using nonlinear PCA type learning. Neural Networks 1994;7:113–127.

[26] Karhunen J, Wang L, Vigario R. Nonlinear PCA type approaches for source separation and independent component analysis, 1995.

[27] Oja E. The nonlinear PCA learning rule and signal separation – mathematical analysis, 1995.

[28] Cover T, Thomas J. Elements of Information Theory. John Wiley and Sons, 1991.

[29] Bourlard H, Kamp Y. Auto-association by multilayer perceptrons and singular value decomposition. Biol Cybern 1988;(59):291–294.

[30] Bishop C. Neural Networks for Pattern Recognition. New York: Oxford University Press, 1995.

[31] Toch B, Lowe D, Saad D. Watermarking of audio signals using ICA. In Third International Conference on Web Delivering of Music, volume 8. 2003; 71–74.

[32] Kwon OW, Lee TW. Phoneme recognition using ICA-based feature extraction and transformation. IEEE Trans on Signal Processing 2004;84(6):1005–1019. ISSN 0165-1684.

[33] Barros A, Mansour A, Ohnishi N. Adaptive blind elimination of artifacts in ECG signals. In Proceedings of I and ANN. 1998; .

[34] Makeig S, Bell AJ, Jung TP, Sejnowski TJ. Independent component analysis of electroencephalographic data. In Touretzky DS, Mozer MC, Hasselmo ME (eds.), Advances in Neural Information Processing Systems, volume 8. The MIT Press, 1996; 145–151.

[35] Jung TP, Humphries C, Lee TW, Makeig S, McKeown MJ, Iragui V, Sejnowski TJ. Extended ICA removes artifacts from electroencephalographic recordings. In Jordan MI, Kearns MJ, Solla SA (eds.), Advances in Neural Information Processing Systems, volume 10. The MIT Press, 1998; .

[36] Hyvärinen A. Sparse code shrinkage: Denoising of nongaussian data by maximum likelihood estimation. Neural Computation 1999;11(7):1739–1768.

[37] Hansen LK. ICA of fMRI based on a convolutive mixture model. In Ninth Annual Meeting of the Organization for Human Brain Mapping (HBM 2003), NewYork, 2003 June. 2003; .

[38] Lörincz A, Póczos B, Szirtes G, Takács B. Ockham's razor at work: Modeling of the 'homunculus'. Brain and Mind 2002;3:187–220.

[39] Bartlett M, Movellan J, Sejnowski T. Face recognition by independent component analysis. IEEE Transactions on neural networks 2002;13(6):1450–1464.

[40] Ben-Shalom A, Dubnov S, Werman M. Improved low bit-rate audio compression using reduced rank ic a instead of psychoacoustic modeling. In IEEE International Conference on Acoustics, Speech and Signal Pr ocessing. IEEE, 2003; .

[41] Roweis ST. One microphone source separation. In Proc. Neural Information Processing Systems (NIPS). Denver, Colo, USA, 2000; 793–799.

[42] Lee TW, Girolami M, Bell AJ, Sejnowski TJ. A unifying information-theoretic framework for independent component analysis, 1998.

[43] Lee TW, Lewicki MS, Sejnowski TJ. ICA mixture models for unsupervised classification of non-Gaussian classes and automatic context switching in blind signal separation. IEEE Transactions on Pattern Analysis and Machine Intelligence 2000; 22(10):1078–1089.

[44] Karhunen J, Pajunen P, Oja E. The nonlinear PCA criterion in blind source separation: Relations with other approaches. Neurocomputing 1998;22:520.

[45] Amari S, Cichocki A, Yang H. Recurrent neural networks for blind separation of sources. In Proc. Int. Symp. NOLTA. 1995; 37–42.

[46] Attias H. Independent factor analysis. Neural Comput 1999;11(4):803–851. ISSN 0899-7667.

[47] Portilla J, Strela V, Wainwright M, Simoncelli E. Adaptive wiener denoising using a Gaussian scale mixture model in the wavelet domain. October 2001; 37–40.

[48] Joho M, Mathis H, Moschytz G. An fft-based algorithm for multichannel blind deconvolution. In IEEE International Symposium on Circuits and Systems, number 4. Orlando, FL, 1999; 203–206.

[49] Roberts S, Roussos E, Choudrey R. Hierarchy, priors and wavelets: structure and signal modelling using ICA. Signal Process 2004;84(2):283–297. ISSN 0165-1684.

[50] Simoncelli EP. Bayesian denoising of visual images in the wavelet domain. In Müller P, Vidakovic B (eds.), Bayesian Inference in Wavelet Based Models. New York: Springer-Verlag, Spring 1999; 291–308.

[51] Penny W, Everson R, Roberts S. Hidden Markov independent components analysis. In Girolami M (ed.), Independent Components Analysis. Kluwer Academic Publishers, 2000; .

[52] Penny W, Roberts S, Everson R. Hidden Markov independent components for biosignal analysis. In Proceedings of MEDSIP-2000, International Conference on Advances in Medical Signal and Information Processing. 2000; .

[53] Everson R, Roberts SJ. Particle filters for nonstationary ICA. In Roberts SJ, Everson R (eds.), Independent Component Analysis: Principles and Practice. Cambridge University Press, 2001; 280–298.

[54] Valpola H, Karhunen J. An unsupervised ensemble learning method for nonlinear dynamic state-space models. Neural Comput 2002;14(11):2647–2692. ISSN 0899-7667.

[55] Cardoso J, Laheld B. Equivariant adaptive source separation. IEEE Trans on Signal Processing 1996;44(12):3017–3030.

[56] Clifford GD, Tarassenko L, Townsend N. Fusing conventional ECG QRS detection algorithms with an auto-associative neural network for the detection of ectopic beats. In 5^{th} International Conference on Signal Processing. IFIP, Beijing, China: World Computer Congress, August 2000; 1623–1628.

[57] Tarassenko L, Clifford GD, Townsend N. Detection of ectopic beats in the electrocardiogram using an auto-associative neural network. Neural Processing Letters Aug 2001;14(1):15–25.

[58] Golub GH. Least squares, singular values and matrix approximations. Applikace Matematiky 1968;(13):44–51.

[59] Bunch J, Nielsen C. Updating the singular value decomposition. Numer Math 1978; (31):111–129.

Appendix A:

15.9.1 Karhunen-Loéve or Hotelling Transformation

The Karhunen-Loéve transformation maps vectors $\mathbf{x}^n$ in a d-dimensional space $(x_1, ..., x_d)$ onto vectors $\mathbf{z}^n$ in an p-dimensional space $(z_1, ..., z_p)$, where $p < d$.

The vector $\mathbf{x}^n$ can be represented as a linear combination of a set of d orthonormal vectors $\mathbf{u}_i$

$$\mathbf{x} = \sum_{i=1}^{d} z_i \mathbf{u}_i \tag{36}$$

Where the vectors $\mathbf{u}_i$ satisfy the orthonormality relation

$$\mathbf{u}_i \mathbf{u}_j = \delta_{ij} \tag{37}$$

in which δ_{ij} is the Kronecker delta symbol.

This transformation can be regarded as a simple rotation of the coordinate system from the original x's to a new set of coordinates represented by the z's. The z_i are given by

$$z_i = \mathbf{u}_i^T \mathbf{x} \tag{38}$$

Suppose that only a subset of $p \leq d$ basis vectors $\mathbf{u}_i$ are retained so that we use only p coefficients of z_i. The remaining coefficients will be replaced by constants b_i so that each vector $\mathbf{x}$ is approximated by the expression

$$\widetilde{\mathbf{x}} = \sum_{i=1}^{p} z_i \mathbf{u}_i + \sum_{i=p+1}^{d} b_i \mathbf{u}_i \tag{39}$$

The *residual* error in the vector $\mathbf{x}^n$ introduced by the dimensionality reduction is given by

$$\mathbf{x}^n - \widetilde{\mathbf{x}}^n = \sum_{i=p+1}^{d} (z_i - b_i) \mathbf{u}_i \tag{40}$$

We can then define the best approximation to be that which minimises the sum of the squares of the errors over the whole data set. Thus we minimise

$$\xi_p = \frac{1}{2} \sum_{n=1}^{N} \sum_{i=p+1}^{d} (z_i - b_i)^2 \tag{41}$$

If we set the derivative of ξ_p with respect to b_i to zero we find

$$b_i = \frac{1}{N} \sum_{n=1}^{N} z_i^n = \mathbf{u}_i^T \bar{\mathbf{x}} \tag{42}$$

Where we have defined the vector $\bar{\mathbf{x}}$ to be the mean vector of the N vectors,

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^n \tag{43}$$

We can now write the sum-of-squares-error as

$$\begin{aligned}\xi_p &= \frac{1}{2}\sum_{n=1}^{N}\sum_{i=p+1}^{d}(\mathbf{u}_i^T(\mathbf{x}^n - \bar{\mathbf{x}}))^2 \\ &= \frac{1}{2}\sum_{n=1}^{N}\mathbf{u}_i^T\mathbf{C}\mathbf{u}_i \end{aligned} \tag{44}$$

Where $\mathbf{C}$ is the covariance matrix of the set of vectors $\mathbf{x}^n$ and is given by

$$\mathbf{C} = \sum_{n}(\mathbf{x}^n - \bar{\mathbf{x}})(\mathbf{x}^n - \bar{\mathbf{x}})^T \tag{45}$$

It can be shown (see Bishop [30]) that the minimum occurs when the basis vectors satisfy

$$\mathbf{C}\mathbf{u}_i = \lambda_i\mathbf{u}_i \tag{46}$$

so that they are eigenvectors of the covariance matrix. Substituting (46) into (44) and making use of the orthonormality relation of (37), the error criterion at the minimum is given by

$$\xi_p = \frac{1}{2}\sum_{i=p+1}^{d}\lambda_i \tag{47}$$

Thus the minimum error is obtained by choosing the $d-p$ smallest eigenvalues, and their corresponding eigenvectors, as the ones to discard.

Appendix B:

15.9.2 Gram-Schmidt Orthogonalization Theorem

If $\{\mathbf{x_1}, ..., \mathbf{x_m}\}$ is a linearly independent vector system in the vector space with scalar product F, then there exists an orthonormal system $\{\varepsilon_1, ..., \varepsilon_m\}$, such that

$$span\{\mathbf{x_1}, ..., \mathbf{x_m}\} = span\{\varepsilon_1, ..., \varepsilon_m\}. \tag{48}$$

This assertion can be proved by induction. In the case $m = 1$, we define $\varepsilon_1 = \mathbf{x_1}/\|\mathbf{x_1}\|$ and thus $span\{\mathbf{x_1}\} = span\{\varepsilon_1\}$. Now assume that the proposition holds for $m = i-1$, i.e., there exists an orthonormal system $\{\varepsilon_1, ..., \varepsilon_{i-1}\}$, such that $span\{\mathbf{x_1}, ..., \mathbf{x_{i-1}}\} = span\{\varepsilon_1, ..., \varepsilon_{i-1}\}$. Then consider the vector

$$\mathbf{y_i} = \lambda_1\varepsilon_1 + ... + \lambda_{i-1}\varepsilon_{i-1} + \mathbf{x_i}, \tag{49}$$

choosing the coefficients $\lambda_\nu, (\nu = 1 : i-1)$ so that $\mathbf{y_i} \perp \varepsilon_\nu(\nu = 1 : i-1)$, i.e. $(\mathbf{y_i}, \varepsilon_\nu) = 0$. This leads to the $i-1$ conditions

$$\begin{aligned} \lambda_\nu(\varepsilon_\nu, \varepsilon_\nu) + (\mathbf{x_i}, \varepsilon_\nu) &= 0, \\ \lambda_\nu &= -(\mathbf{x_i}, \varepsilon_\nu) \quad (\nu = 1 : i-1). \end{aligned} \tag{50}$$

Therefore,

$$\mathbf{y_i} = \mathbf{x_i} - (\mathbf{x_i}, \varepsilon_1)\varepsilon_1 - ... - (\mathbf{x_i}, \varepsilon_{i-1})\varepsilon_{i-1}. \tag{51}$$

Now we choose $\varepsilon_i = \mathbf{y_i}/\|\mathbf{y_i}\|$. Since $\varepsilon_\nu \in span\{\mathbf{x_1}, ..., \mathbf{x_{i-1}}\}(\nu = 1 : i-1)$, we get, by the construction of the vectors $\mathbf{y_i}$ and ε_i, $(\varepsilon_i \in span\{\mathbf{x_1}, ..., \mathbf{x_i}\})$. Hence

$$span\{\varepsilon_1, ..., \varepsilon_i\} \subset span\{\mathbf{x_1}, ..., \mathbf{x_i}\}. \tag{52}$$

From the representation of the vector $\mathbf{y_i}$ we can see that $\mathbf{x_i}$ is a linear combination of the vectors $\varepsilon_1, ..., \varepsilon_i$. Thus

$$span\{\mathbf{x_1}, ..., \mathbf{x_i}\} \subset span\{\varepsilon_1, ..., \varepsilon_i\} \tag{53}$$

and finally,

$$span\{\mathbf{x_1}, ..., \mathbf{x_i}\} = span\{\varepsilon_1, ..., \varepsilon_i\} \tag{54}$$

An example

Given a vector system $\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}\}$ in $\mathcal{R}^4$, where
$\mathbf{x_1} = [1\ 0\ 1\ 0]^T$, $\mathbf{x_2} = [1\ 1\ 1\ 0]^T$, $\mathbf{x_3} = [0\ 1\ 0\ 1]^T$,
such that $\mathbf{X} = [\mathbf{x_1 x_2 x_3}]^\mathbf{T}$ we want to find such an orthogonal system $\{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$, for which

$$span\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}\} = span\{\varepsilon_1, \varepsilon_2, \varepsilon_3\}.$$

To apply the orthogonalization process, we first check the system $\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}\}$ for linear independence. Next we find

$$\varepsilon_1 = \mathbf{x_1}/\|\mathbf{x_1}\| = \left[\frac{1}{\sqrt{2}}\ 0\ \frac{1}{\sqrt{2}}\ 0\right]^T. \tag{55}$$

For $\mathbf{y_2}$ we get

$$\mathbf{y_2} = \mathbf{x_2} - (\mathbf{x_2}, \varepsilon_1)\varepsilon_1 = [1\ 1\ 1\ 0]^T - \sqrt{2}\left[\frac{1}{\sqrt{2}}\ 0\ \frac{1}{\sqrt{2}}\ 0\right]^T = [0\ 1\ 0\ 0]^T. \tag{56}$$

Since $\|\mathbf{y_2}\| = 1$, $\varepsilon_2 = \mathbf{y_2}/\|\mathbf{y_2}\| = [0\ 1\ 0\ 0]^T$. The vector $\mathbf{y_3}$ can be expressed in the form

$$\mathbf{y_3} = \mathbf{x_3} - (\mathbf{x_3}, \varepsilon_1)\varepsilon_1 - (\mathbf{x_3}, \varepsilon_2)\varepsilon_2 = [0\ 1\ 0\ 1]^T - 0 \cdot \left[\frac{1}{\sqrt{2}}\ 0\ \frac{1}{\sqrt{2}}\ 0\right]^T - 1 \cdot [0\ 1\ 0\ 0]^T = [0\ 0\ 0\ 1]^T. \tag{57}$$

Therefore,

$$\varepsilon_3 = \mathbf{y_3}/\|\mathbf{y_3}\| = [0\ 0\ 0\ 1]^T. \tag{58}$$

Appendix D:

15.9.3 Maximum Likelihood through gradient ascent

Recalling that for scalars $\int dz\delta(x-wz)f(s) = \frac{1}{|w|}f(x/w)$ and adopting a conventional index summation such that $w_{ji}z_i^m \equiv \sum_i w_{ji}z_i^m$, a single factor in the likelihood is given by

$$\begin{aligned} P(\mathbf{x}^m|\mathbf{A}) &= \int_{-\infty}^{\infty} d^N\mathbf{z}^m P(\mathbf{x}^m|\mathbf{z}^m,\mathbf{A})P(\mathbf{z}^m) && (59)\\ &= \mathbf{x}^m \prod_j \delta(\mathbf{x}_j^m - a_{ji}\mathbf{z}_i^m)\prod_i p_i(\mathbf{z}_i^m) && (60)\\ &= \frac{1}{|\det\mathbf{A}|}\prod_i p_i(a_{ij}^{-1}\mathbf{x}_j) && (61)\\ & && (62)\end{aligned}$$

which implies

$$\log_2 P(\mathbf{x}^m|\mathbf{A}) = \log_2\det\mathbf{A} + \sum_\imath \log_2 p_i(a_{ij}\mathbf{x}_j). \qquad (63)$$

Noting that $\mathbf{W} = \mathbf{A}^{-1}$,

$$\log_2 P(\mathbf{x}^m|\mathbf{A}) = \log_2\det\mathbf{W} + \sum_\imath \log_2 p_i(w_{ij}\mathbf{x}_j). \qquad (64)$$

To find the gradient of the log likelihood we define

$$\begin{aligned} \frac{\partial}{\partial a_{ji}}\log_2\det\mathbf{A} &= a_{ij}^{-1} = w_{ij} && (65)\\ \frac{\partial}{\partial a_{ji}}a_{kl}^{-1} &= -a_{kj}^{-1}a_{il}^{-1} = -w_{kj}w_{il} && (66)\\ \frac{\partial}{\partial w_{ij}}g &= -a_{jl}\left(\frac{\partial}{\partial a_{kl}}g\right)a_{li} && (67)\\ & && (68)\end{aligned}$$

with g some arbitrary function, w_{ij} representing the elements of $\mathbf{W}$, $y_i \equiv w_{ij}x_j$ and $f(y_i) \equiv d\log_2 p_i(y_i)/dy_i$. g indicates in which direction y_i needs to change to make the probability of the data greater. Using equations 66 and 67 we can obtain the gradient of a_{ji}

$$\frac{\partial}{\partial a_{ji}}\log_2 P(\mathbf{x}^m|\mathbf{A}) = -w_{ij} - w_i f_{y'} w_{i'j} \qquad (69)$$

where i' is a dummy index. Alternatively, we can take the derivative with respect to a_{ij}

$$\frac{\partial}{\partial w_{ij}}\log_2 P(\mathbf{x}^m|\mathbf{A}) = a_{ji} + x_j z_i \qquad (70)$$

If we choose $\mathbf{W}$ so as to ascend this gradient, we obtain the exact learning algorithm from Bell and Sejnowski [22] ($\Delta\mathbf{W} \propto [\mathbf{W}^T]^{-1} + \mathbf{z}\mathbf{x}^T$). A detailed mathematical analysis of gradient descent/ascent and its relationship to neural networks and PCA are given in appendix 15.9.4.

The problem of non-covariance

It should be noted that the **principle of covariance** (consistent algorithms should give the same results independently of the units in which the quantities are measured) is not always true. One example is the popular steepest descent rule (see Eq. 31) which is dimensionally inconsistent; the left hand side has dimensions of $[w_i]$ and the right hand side has dimensions $[w_i]$ ($\text{Ł}(\mathbf{W})$ is dimensionless).

One method for alleviating this problem is to precondition the input data (scaling it between ± 1). Another method is to decrease η at a rate of $1/n$ where n is the number of iterations through the backpropagation of the updates of the w_i. The Munro-Robbins theorem ([30] p.41) shows that the parameters will asymptotically converge to the maximum likelihood parameters since each data point receives equal weighting. If n is held constant then one is explicitly solving a weighted maximum likelihood problem with an exponential weighting of the data and the parameters will not converge to a limiting value.

The algorithm would be covariant if $\Delta w_i = \eta \sum_{i'} \mathbf{G}_{ii'} \frac{\partial \text{Ł}}{\partial w_i}$, where $\mathbf{G}$ is a curvature matrix with the $(i, i')^{th}$ element having dimensions $[w_i w'_i]$. It should be noted that the differential of the metric for the gradient descent is not linear (as it is for a least square computation), and so the space on which we perform gradient descent is non-Euclidean. In fact, one must use the *natural* [23] or *relative* [55] gradient. See [16] and [23] for further discussion on this topic.

Appendix D:

15.9.4 Gradient descent, neural network training and PCA

This appendix is intended to give the reader a more thorough understanding of the inner workings of gradient descent and learning algorithms. In particular, we will see how the weights of a neural network are essentially a matrix that we train on some data by minimising or maximising a cost function through gradient descent. If a multi-layered perceptron (MLP) neural network is auto-associative (it has as many output nodes as input nodes), then we essentially have the same paradigm as Blind Source Separation. The only difference is the cost function.

This appendix describes relevant aspects of gradient descent and neural network theory. The error back-propagation algorithm is derived from first principles in order to lay the ground-work for gradient descent training of an auto-associative neural network.

The neuron

The basic unit of a neural network is the neuron. It can have many inputs x and its output value, y, is a function, f, of all these inputs. Figure 9 shows the basic architecture of a neuron with three inputs.

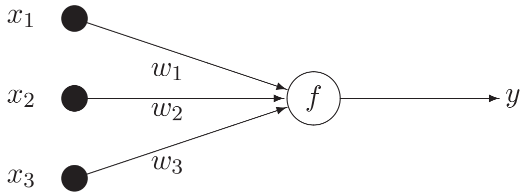

Figure 9: The basic neuron

For a linear unit the function, f, is the linear weighted sum of the inputs, sometimes known as the activation, a, in which case the output is given by

$$y = a = \sum_i w_i x_i \tag{71}$$

(Be careful not to confuse the activation a with the elements of the mixing matrix $\mathbf{A} = a_{ij} = \hat{\mathbf{W}}^{-1}$.) For a non-linear unit, a non-linear function f, is applied to the linearly weighted sum of inputs. This function is often the sigmoid function defined as

$$f_a(a) = \frac{1}{1+e^{-a}} \tag{72}$$

The output of a non-linear neuron is then given by

$$y = f_a\{(\sum_i w_i x_i)\} \tag{73}$$

If the outputs of one layer of neurons are connected to the inputs of another layer, a neural network is formed.

Multi-layer networks

The standard MLP consists of three layers of nodes, the layers being interconnected via synaptic weights w_{ij} and w_{jk} as shown in Figure 10. The input units simply pass all of the input data, likewise the non-linear output units of the final layer receive data from each of the units in the hidden layer. Bias units (with a constant value of 1.0), connect directly via bias weights to each of the neurons in the hidden and output layers (although these have been omitted from the diagram for clarity).

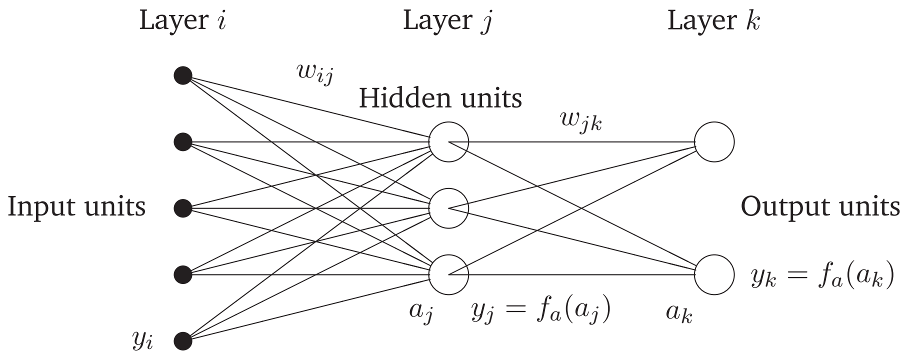

Figure 10: Schematic of a 5-3-2 multi-layer perceptron. Bias units and their weights are omitted for clarity.

Learning algorithm - Gradient descent

The input data used to train the network, now defined as y_i for consistency of notation, is fed into the network and propagated through to give an output y_k given by

$$y_k = f_a(\sum_j w_{jk} f_a(\sum_i w_{ij} y_i)) \tag{74}$$

Note that our x's (observations/recordings) are now y_i's and our sources are y_j (or y_k if multi-layered network is used). During training, the target data or desired output, t_k, which is associated with the training data, is compared to the actual output y_k. The weights, w_{jk} and w_{ij} , are then adjusted in order to minimise the difference between the propagated output and the target value. This error is defined over all training patterns, N, in the training set as

$$\xi = \frac{1}{2} \sum_N \sum_k (f_a(\sum_j w_{jk} f_a(\sum_i w_{ij} y_i^p)) - t_k^p)^2 \tag{75}$$

Note that in the case of ML-BSS the target is one of the other output vectors (source estimates) and the error function ξ, is the log likelihood. We must therefore sum ξ over all the possible pairs of output vectors.

Error back-propagation

The squared error, ξ, can be minimised using the method of gradient descent [30]. This requires the gradient to be calculated with respect to each weight, w_{ij} and w_{jk}. The weight update equations for the hidden and output layers are given as follows:

$$w_{jk}^{(\tau+1)} = w_{jk}^{(\tau)} - \eta \frac{\partial \xi}{\partial w_{jk}} \tag{76}$$

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} - \eta \frac{\partial \xi}{\partial w_{ij}} \tag{77}$$

The full derivation of the calculation of the partial derivatives, $\frac{\partial \xi}{\partial w_{ij}}$ and $\frac{\partial \xi}{\partial w_{jk}}$, is given in appendix 15.9.5. Using equations 116 and 108 we can write:

$$w_{jk}^{(\tau+1)} = w_{jk}^{(\tau)} - \eta \delta_k y_j \tag{78}$$

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} - \eta \delta_j y_i \tag{79}$$

where η is the learning rate and δ_j and δ_k are given below:

$$\delta_k = (y_k - t_k) y_k (1 - y_k) \tag{80}$$

$$\delta_j = \sum_k \delta_k w_{jk} y_j (1 - y_j) \tag{81}$$

For the bias weights, the y_i and y_j in the above weight update equations are replaced by unity.

Training is an iterative process (repeated application of equations 78 and 79) but, if continued for too long, the network starts to fit the noise in the training set and that will have a negative effect on the performance of the trained network on test data. The decision on when to stop training is of vital importance but is often defined when the error function (or it's gradient) drops below some predefined level. The use of an independent validation set is often the best way to decide on when to terminate training. (See Bishop [30] p262 for more details.) However, in the case of an auto-associative network, no validation set is required and the training can be terminated when the ratio of the variance of the input and output data reaches a plateau. (See [56, 57].)

Auto-associative networks

An auto-associative network has as many output nodes as input nodes and can be trained using an objective cost function measured between the inputs and outputs; the target data are simply the input data. Therefore, no labelling of the training data is required. An auto-associative neural network performs dimensionality reduction from D to p dimensions ($D > p$) and then projects back up to D dimensions, as shown in figure 11. PCA, a standard linear dimensionality reduction procedure is also a form of unsupervised learning [30]. In fact, the number of hidden-layer nodes ($\dim(y_j)$) is usually chosen to be the same as the number of principal components, p, in the data (see § 15.3.1), since (as we shall see later) the first layer of weights performs PCA if trained with a linear activation function. The full derivation of PCA, given in appendix 15.9.1, shows that PCA is based on minimising a sum-of-squares error cost function.

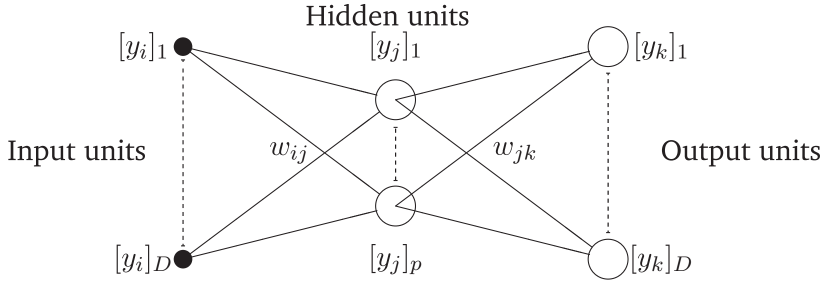

Figure 11: Layout of a D-p-D auto-associative neural network.

Network with linear hidden layer and output units

Since $y_k = a_k$

$$\frac{\partial y_k}{\partial a_k} = 1 \tag{82}$$

the expression for δ_k reduces to

$$\delta_k = \frac{\partial \xi}{\partial a_k} = \frac{\partial \xi}{\partial y_k}.\frac{\partial y_k}{\partial a_k} = (y_k - t_k) \tag{83}$$

Similarly for δ_j:

$$\delta_j = \frac{\partial \xi}{\partial a_j} = \frac{\partial \xi}{\partial a_k}.\frac{\partial a_k}{\partial y_j}.\frac{\partial y_j}{\partial a_j} = \sum_k \delta_k w_{jk} \tag{84}$$

Linear activation functions perform PCA

The two layer auto-associative MLP with linear hidden units and a sum-of-squares error function used in the previous section, learns the principal components for that data set. PCA can of course be performed and the weights can be calculated directly by computing a matrix pseudo-inverse [30], and this shall reduce 'training time' significantly. Consider Eq. (75) where the activation function is linear ($f_a = 1$) for the input and hidden layers;

$$\xi = \frac{1}{2}\sum_{n=1}^{N}\sum_{j=1}^{p}(\sum_{i=0}^{D} y_i^n w_{ij} - t_j^n)^2 \tag{85}$$

where p is the number of hidden units. If this expression is differentiated with respect to w_{ij} and the derivative is set to zero the usual equations for least-squares optimisation are given in the form

$$\sum_{n=1}^{N}(\sum_{i'=0}^{D} y_i^n w_{i'j} - t_j^n) y_i^n = 0 \tag{86}$$

which is written in matrix notation as

$$(\mathbf{Y^T Y})\mathbf{W^T} = \mathbf{Y^T T} \tag{87}$$

$\mathbf{Y}$ has dimensions $N \times D$ with elements y_i^n where N is the number of training patterns and D the number of input nodes to the network (the length of each ECG complex in our examples given in the main text). $\mathbf{W}$ has dimension $p \times D$ and elements w_{ij} and $\mathbf{T}$ has dimensions $N \times p$ and elements t_j^n. The matrix $(\mathbf{Y^TY})$ is a square $p \times p$ matrix which may be inverted to obtain the solution

$$\mathbf{W^T} = \mathbf{Y^\dagger T} \tag{88}$$

where $\mathbf{Y}^\dagger$ is the $(p \times N)$ pseudo-inverse of $\mathbf{Y}$ and is given by

$$\mathbf{Y}^\dagger = (\mathbf{Y^TY})^{-1}\mathbf{Y^T} \tag{89}$$

Note that in practice $(\mathbf{Y^TY})$ usually turns out to be near-singular and SVD is used to avoid problems caused by the accumulation of numerical roundoff errors.

Consider N training patterns presented to the auto-associative MLP with i input and k output nodes $(i = k)$ and $j \leq i$ hidden nodes. For the n^{th} $(n = 1...N)$ input vector x_i of the $i \times N$ $(N \geq i)$ real input matrix, $\mathbf{X}$, formed by the N (i-dimensional) training vectors, the hidden unit output values are

$$h_j = f(\mathbf{W}_1 x_i + w_{1b}) \tag{90}$$

where $\mathbf{W}_1$ is the input-to-hidden layer $i \times j$ weight matrix, w_{1b} is a rank-j vector of biases and f is an activation function. The output of the auto-associative MLP can then be written as

$$y_k = \mathbf{W}_2 h_j + w_{2b} \tag{91}$$

where $\mathbf{W}_2$ is the hidden-to-output layer $j \times k$ weight matrix and w_{2b} is a rank-k vector of biases. Now consider the singular value decomposition of X, given by [2] as $\mathbf{X_i} = \mathbf{U_iS_iV_i^T}$ where $\mathbf{U}$ is an $i \times i$ column-orthogonal matrix, S is an $i \times N$ diagonal matrix with positive or zero elements (the singular values) and $\mathbf{V}^T$ is the transpose of an $N \times N$ orthogonal matrix. The best rank-j approximation of $\mathbf{X}$ is given by [58] as $\mathbf{W}_2 h_j = \mathbf{U_jS_jV_j^T}$ where

$$h_j = \mathbf{FS_jV_j^T} \qquad \text{and} \tag{92}$$

$$\mathbf{W_2} = \mathbf{U_jF^{-1}} \tag{93}$$

with $\mathbf{F}$ being an arbitrary non-singular $j \times j$ scaling matrix. $\mathbf{U}_j$ has $i \times j$ elements, $\mathbf{S}_j$ has $j \times j$ elements and $\mathbf{V}^T$ has $j \times N$ elements. It can be shown that [29]

$$\mathbf{W}_1 = \alpha_1^{-1}\mathbf{FU}_j^\mathbf{T} \tag{94}$$

where $\mathbf{W}_1$ are the input-to-hidden layer weights and α is derived from a power series expansion of the activation function, $f(x) \approx \alpha_0 + \alpha_1 x$ for small x. For a linear activation function, as in this application, $\alpha_0 = 0$, $\alpha_1 = 1$. The bias weights given in [29] reduce to

$$\begin{aligned} w_{1b} &= -\alpha_1^{-1}\mathbf{FU}_j^T \mu_X = -\mathbf{U}_j^T \mu_X, \\ w_{2b} &= \mu_X - \alpha_0 \mathbf{U}_j \mathbf{F}^{-1} = \mu_X \end{aligned} \tag{95}$$

where $\mu_X = \frac{1}{N}\sum_N x_i$, the average of the training (input) vectors and $\mathbf{F}$ is here set to be the $(j \times j)$ identity matrix since the output is unaffected by the scaling. Using equations

(90) to (95)

$$\begin{aligned} y_k &= \mathbf{W}_2 h_j + w_{2b} \qquad (96) \\ &= \mathbf{U_j}\mathbf{F}^{-1} h_j + w_{2b} \\ &= \mathbf{U_j}\mathbf{F}^{-1}(W_1 x_i + w_{1b}) + w_{2b} \\ &= \mathbf{U_j}\mathbf{F}_1^{-1}\mathbf{F}\mathbf{U}_j^T x_i - \mathbf{U}_j\mathbf{F}^{-1}\mathbf{U}_j^T \mu_X + \mu_X \end{aligned}$$

giving the output of the auto-associative MLP as

$$y_k = \mathbf{U_j}\mathbf{U_j^T}(\mathbf{X} - \mu_X) + \mu_X. \qquad (97)$$

Equations (93), (94) and (95) represent an analytical solution to determine the weights of the auto-associative MLP 'in one pass' over the input (training) data with as few as $Ni^3 + 6Ni^2 + O(Ni)$ multiplications [59]. We can see that $\mathbf{W_1} = w_{ij}$ is the matrix that rotates the each of the data vectors $x_i^n = y_i^n$ in $\mathbf{X}$ into the hidden data y_i^p, which are our p underlying sources. $\mathbf{W_2} = w_{jk}$ is the matrix that transforms our sources back into the observation data (the target data vectors $\sum_N t_i^n = \mathbf{T}$). If $p < N$, we have discarded some of the possible information sources and effected a filtering. In terms of PCA, $\mathbf{W_1} = \mathbf{S}_p\mathbf{V^T} = \mathbf{U}_p\mathbf{U}_p^{\mathbf{T}}$ (where $\mathbf{U}_p$ denotes using the p^{th} most significant components) and in terms of BSS, $\mathbf{W_1} = \mathbf{W}_p$. Note also that $\mathbf{W_2} = \mathbf{W_1^T}$.

Appendix F:

15.9.5 Derivation of error back-propagation

The error ξ is given over all input patterns p by:

$$\xi = \frac{1}{2}\sum_n \sum_k (y_k^n - t_k^n)^2 \qquad (98)$$

Which may be written as:

$$\xi = \frac{1}{2}\sum_n \sum_k (f_a(\sum_j w_{jk} f_a(\sum_i y_i w_{ij})) - t_k^n)^2 \qquad (99)$$

To calculate the update rules the gradient of the error with respect to the weights w_{ij} and w_{jk} must be calculated. The update equations (100) (101) are given below, η is the learning rate.

$$w_{jk}^{(\tau+1)} = w_{jk}^{(\tau)} - \eta\frac{\partial\xi}{\partial w_{jk}} \qquad (100)$$

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} - \eta\frac{\partial\xi}{\partial w_{ij}} \qquad (101)$$

The calculation of the gradients is performed using simple chain rule partial differentiation.

$$\frac{\partial \xi}{\partial w_{jk}} = \frac{\partial \xi}{\partial y_k}.\frac{\partial y_k}{\partial a_k}.\frac{\partial a_k}{\partial w_{jk}} \tag{102}$$

The input to the output units a_k is given by

$$a_k = \sum_j y_j w_{jk} \tag{103}$$

From (98) and (103) we may write

$$\frac{\partial \xi}{\partial y_k} = (y_k - t_k), \qquad \frac{\partial a_k}{\partial w_{jk}} = y_j \tag{104}$$

Since y_k is defined as

$$y_k = f_a(a_k) = \frac{1}{1 + e^{-a_k}} \tag{105}$$

We may write

$$\frac{\partial y_k}{\partial a_k} = \frac{\partial}{\partial a_a}(\frac{1}{1 + e^{-a_k}}) = \frac{e^{-a_k}}{(1 + e^{-a_k})^2} = y_k(1 - y_k) \tag{106}$$

Hence

$$\frac{\partial \xi}{\partial w_{jk}} = (y_k - t_k)y_k(1 - y_k)y_j \tag{107}$$

Therefore we may write the w_{jk} update as

$$w_{jk}^{(\tau+1)} = w_{jk}^{(\tau)} - \eta \delta_k y_j \tag{108}$$

Where

$$\delta_k = \frac{\partial \xi}{\partial y_k}.\frac{\partial y_k}{\partial a_k} = \frac{\partial \xi}{\partial a_k} = (y_k - t_k)y_k(1 - y_k) \tag{109}$$

In order to calculate the w_{ij} update equation the chain rule is applied several times, hence

$$\frac{\partial \xi}{\partial w_{ij}} = \frac{\partial \xi}{\partial a_j}.\frac{\partial a_j}{\partial w_{ij}} \tag{110}$$

$$\frac{\partial \xi}{\partial w_{ij}} = \frac{\partial \xi}{\partial a_k}.\sum_k(\frac{\partial a_k}{\partial y_j}).\frac{\partial y_j}{\partial a_j}.\frac{\partial a_j}{\partial w_{ij}} \tag{111}$$

From (103)

$$\frac{\partial a_k}{\partial y_j} = \sum_k w_{jk} \tag{112}$$

The input to the hidden units is given by

$$a_j = \sum_i y_i w_{ij} \tag{113}$$

Hence

$$\frac{\partial a_j}{\partial w_{ij}} = y_i \tag{114}$$

By symmetry from (106) we have

$$\frac{\partial y_j}{\partial a_j} = y_j(1 - y_j) \tag{115}$$

Therefore from Eq.s (109),(112),(114) and (115) the update equation for the w_{ij} weights is given by

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} - \eta \delta_j y_i \tag{116}$$

Where

$$\delta_j = \frac{\partial \xi}{\partial a_j} = \sum_k \delta_k w_{jk} y_j (1 - y_j) \tag{117}$$

Appendix G:

15.9.6 Orthogonal rotation matrices

The classical (orthogonal) three-dimensional rotation matrices are

$$\mathbf{R_x}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}, \mathbf{R_y}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \mathbf{R_z}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{118}$$

where $\mathbf{R_x}(\theta)$, $\mathbf{R_y}(\theta)$ and $\mathbf{R_z}(\theta)$ produce rotations of a 3-D signal about the x-axis, y-axis and z-axis respectively.