HST582J/6.555J/16.456J Biomedical Signal and Image Processing Spring 2005

Singular Value Decomposition &

Independent Component Analysis

G.D. Clifford

March 11, 2005

Introduction

In this chapter we will examine how we can generalize the idea of transforming a time series in 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 is projected onto a new set of axes that fulfill some statistical criterion, which imply independence, rather than a set of axes that represent discrete frequencies such as with the Fourier transform.

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 changes over time, then the *subspace* onto which the data is projected will change too.

Any projection onto another space is essentially a method for separating the data out into separate **sources** which will hopefully allow us to see important structure in a particular projection. 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 signal. 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 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 maximizing this measure. Since we *discover*, rather than define the the axes, this process is known as **blind source separation**. For PCA the measure we use to discover the axes is **variance** and leads to a set of orthogonal axes. For ICA this measure is based on 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 a distribution is. 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, we will eventually arrive at a Gaussian distribution.) 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 subspaces are discovered and what this means about our data. We shall also examine the assumptions we must make and what happens when these assumptions break down.

12.1 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[1], 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 (ACF) equal to an impulse[2]. 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

[1] it lowers the SNR!

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

electrocardiogram (ECG) is usually thought of as noise or artifact. 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 about 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 order to distinguish between random noise and such information-carrying contaminants we shall refer to these as *artifacts*. While noise is stochastic in nature, with the same characteristic spectrum over a wide range of frequencies, artifacts tend to be more deterministic in nature and confined to a specific band of the spectrum.

Table 1 illustrates the range of signal contaminants for the ECG[3]. We shall also examine the statistical qualities of these contaminants in terms of their probability distribution functions (PDFs) since the power spectrum of a signal 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 *linear Gaussian white noise*[4], this is often not the case. Noise is often correlated (with itself or sometimes the signal), 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). 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 basis functions followed by data reconstruction through projection back into the original space. By reducing the number of basis functions we filter the data (by discarding the projections onto axes that are believed to correspond to noise). By projecting from a reduced set of basis functions (onto 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 is derived from a data set).

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

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

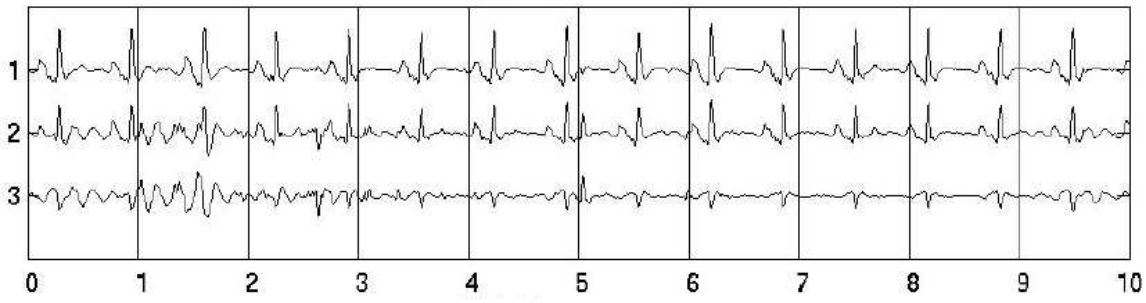

Figure 1: 10 seconds of 3 Channel ECG. Note the high amplitude movement artifact in the first two seconds and the 10^{th} second. Note also the QRS-like artifacts around 2.6 and 5.1 seconds

Qualities → Contaminant ↓	Frequency Range	Time duration
50 or 60HZ Powerline	Narrowband 50 or 60 ± 2 Hz	Continuous
Movement Baseline Wander	Narrowband (< 0.5Hz)	Transient or Continuous
Muscle Noise	Broadband	Transient
Electrical Interference	Narrowband	Transient or Continuous
Electrode pop	Narrowband	Transient
Observation noise	Broadband	Continuous
Quantization noise	Broadband	Continuous

Table 1: Contaminants on the ECG and their nature.

12.2 Matrix transformations as filters

The simplest filtering of a time series involves the transformation of a one dimensional ($N = 1$) time series of M points $\mathbf{x}(t) = (x_1, x_2, x_3 ... x_M)^T$ into a new representation, $\mathbf{y} = (y_1, y_2, y_3 ... y_M)^T$. If $\mathbf{x}(t)$ $(t = 1, 2, ..., M)$ is a column vector[5] 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 M-dimensional with N samples for each vector). A transformation matrix $\mathbf{W}$ can then be applied to $\mathbf{X}$ to create the transformed matrix $\mathbf{Y}$ such that

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

The purpose of a transformation is to highlight patterns in the data, identify interesting projections and (if we are filtering) discard the uninteresting parts of the signal (which are masking the information we are interested in). In general, transforms can be categorized as **orthogonal** or **biorthogonal** transforms. For orthogonal transformations, the transformed signal is same length as the original and the energy of the data is unchanged. An example of this is the Fourier transform where the same signal is measured along a new set of perpendicular axes corresponding to the coefficients of the Fourier series[6].

For biorthogonal transforms, lengths and angles 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 $\mathbf{X} = \mathbf{W}^{-1}\mathbf{Y}$).

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 mapping of the data into a representation using a transformation matrix that contains some zero entries, and therefore there is an irreversible removal of some of the data. Often this corresponds to a mapping to a lower number of dimensions.

In the following sections we will study two transformation techniques Singular Value Decomposition (SVD), and Independent Component Analysis (ICA). We shall see how SVD imposes an orthogonal transform, whereas ICA results in a biorthogonal transform on the

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

[6] there is also a class of wavelets, known as orthogonal wavelets that fall into the same class

data. Both techniques can which can be used to perform lossy or lossless transformations. Lossless SVD and ICA both involve projecting the data onto a set of axes which are determined by the nature of the data, and is 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.)

However, once we have *discovered* the basis functions of the independent axes 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 SVD and ICA separation matrices to zero that correspond to unwanted sources we produce noninvertible 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.

In the Fourier transform, the basis functions for the new representation are predefined and assumed to be independent, whereas in the SVD the representation the basis functions are *found* in the data by looking for a set of basis functions that *are* independent. That is, the data undergoes a decorrelation using variance as the metric (and therefore the basis functions are independent in a second order sense).

The basic idea in the application of PCA to a data set, is to find the components $\mathbf{y}_1$, $\mathbf{y}_2$,...,$\mathbf{y}_N$ so that they 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 $\mathbf{v}_1 = \arg\max_{\|\mathbf{v}\|=1} E\{v_1^T\mathbf{X})^2\}$. where $\mathbf{v}_1$ is the same length M as the data vector $\mathbf{X}$. 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 $\mathbf{y}_i = \mathbf{v}_i^T\mathbf{X}$, the projection of $\mathbf{X}$ onto each.

The basic goal in PCA is to decorrelate the signal by projecting the data onto orthogonal axes. Furthermore, we usually reduce the dimension of the data from N to p ($p < N$). 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 not contained in the first N components is assumed 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, is useful for visualizing the data.

In practice, the computation of the $\mathbf{v}_i$ can be simply accomplished using the sample covariance matrix $E\{\mathbf{X}\mathbf{X}^T\} = \mathbf{C}$. The $\mathbf{v}_i$ are the eigenvectors of $\mathbf{C}$ that correspond to the N largest eigenvalues of $\mathbf{C}$. A method for determining the eigenvalues in this manner

[7]For example, a transformation matrix [1 0; 0 0] is **noninvertible**, 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`

is known as Singular Value Decomposition (SVD), which is described below. SVD is also known as the (discrete) Karhunen-Loève transform, or the Hotelling transform (see Appendix 12.8.1).

12.2.1 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;

$$\mathbf{X} = \mathbf{U} * \mathbf{S} * \mathbf{V}^T \tag{3}$$

where $\mathbf{S}$ is a non-square matrix with zero entries everywhere, except on the leading diagonal with elements s_i 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**. The smaller the eigenvalues are, the less energy along the corresponding eigenvector there is. Therefore, the smallest eigenvalues are often considered to be due to noise. The columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{C}$ and the matrix $\mathbf{U}$ is the matrix of projections of $\mathbf{X}$ onto the eigenvectors of $\mathbf{C}$ [2]. If a truncated SVD of $\mathbf{X}$ is performed (i.e. we just retain the most significant p eigenvectors)[9], then the truncated SVD is 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 Figure 2).

A routine for performing SVD is as follows:

1. Find the non-zero eigenvalues, λ_i of the matrix $\mathbf{X^T X}$ and arrange them in descending order.
2. Find the orthogonal eigenvectors of the matrix $\mathbf{X^T X}$ corresponding to the obtained eigenvalues, and arrange them in the same order to form the column-vectors of the matrix $\mathbf{V}$
3. Form a diagonal matrix $\mathbf{S}$ placing the square roots $s_i = \sqrt{\lambda_i}$ of the $p = min\{M, N\}$ first eigenvalues of the matrix $\mathbf{X^T X}$ found in step (1) in descending order on the leading diagonal.
4. Find the first column-vectors of the matrix $\mathbf{U}$: $\mathbf{u}_i = \mathbf{s}_i^{-1}\mathbf{X}\mathbf{v}_i$ $(i = 1 : p)$.
5. Add to the matrix $\mathbf{U}$ the rest of $M - p$ vectors using the Gram-Schmidt orthogonalization process (see appendix 12.8.2).

[9] 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 12.2.3) or as the value where $\sum_{i=1}^{p} s_i > \alpha \sum_{i=1}^{N} s_i$ and α is some fraction ≈ 0.95

12.2.2 Eigenvalue decomposition - a worked example

To find the singular value decomposition of the matrix

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

first we find the eigenvalues, λ, of the matrix

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

in the usual manner by letting

$$\mathbf{Cv} = \lambda \mathbf{v} = 0 \tag{5}$$

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 nonzero eigenvalues, r, of the matrix $\mathbf{X^T X}$: $r = 2$. Then we find the orthonormal eigenvectors of the matrix $X^T X$ corresponding to the eigenvalues λ_1 and λ_2 by solving for $\mathbf{v_1}$ and $\mathbf{v_2}$ using λ_1 and λ_2 and in $(\mathbf{C} - \lambda \mathbf{I})\mathbf{v} = 0$ ...

$$\mathbf{v_1} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}, \mathbf{v_2} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix}, \tag{6}$$

forming the matrix

$$\mathbf{V} = [\mathbf{v_1 v_2}] = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix} \tag{7}$$

where

$$and$$

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 $\mathbf{C} = \mathbf{X^T X}$ arranged in descending order of magnitude.

$$\mathbf{S} = \begin{bmatrix} \sqrt{3} & 0 \\ 0 & \sqrt{1} \\ 0 & 0 \end{bmatrix} \tag{8}$$

Next we find the first two columns of $\mathbf{U}$, (noting that $\mathbf{S}^{-1} = \frac{1}{det||\mathbf{S}||}[s_{22} \; -s_{21}; -s_{12} \; s_{11}]$ $= \frac{1}{\sqrt{3}}[1 \; 0; 0 \; \sqrt{3}]$) using the relation

$$\mathbf{u_1} = \mathbf{s}_1^{-1} \mathbf{X v_1} = \frac{\sqrt{3}}{3} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{6}}{3} \\ \frac{\sqrt{6}}{6} \\ \frac{\sqrt{6}}{6} \end{bmatrix}$$

and

$$\mathbf{u_2} = \mathbf{s}_2^{-1}\mathbf{X}\mathbf{v_2} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}.$$

Using the Gram-Schmitd process (see appendix 12.8.2) 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

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

12.2.3 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 m heartbeats, each j samples long, in a matrix (of size $m \times j$), we can compress it down (into an $m \times p$) matrix, using only the first $p << m$ principal components. If we then reconstruct the data by inverting the reduced rank matrix, we effectively filter the original data.

Fig. 2a is a set of 64 heart beat waveforms which have been cut into one second segments (200Hz data), aligned by their R-peaks and placed side-by-side to form a 8×200 matrix. The data is therefore 8-dimensional and an SVD will lead to 8 eigenvectors. Fig. 2b is the eigenspectrum after SVD[10]. Note that most of the *power* is contained in the first eigenvector. The *knee* of the eigenspectrum is at the second principal component. Fig. 2c is a plot of the reconstruction (filtering) of the data using just the first eigenvector[11]. Fig. 2d is the same as Fig. 2c, but the first two eigenvectors have been used to reconstruct the data. The data in Fig. 2d is therefore noisier than that in Fig. 2c.

Note that $\mathbf{S}$ derived from a full SVD (using Matlab's SVD()) is an **invertible** matrix, and no information is lost if we retain all the principal components. However, if we perform a

[10] [U S V]=SVD(DATA); STEM(DIAG(S))
[11] [U S V]=SVDS(DATA,1); MESH(U*S*V')

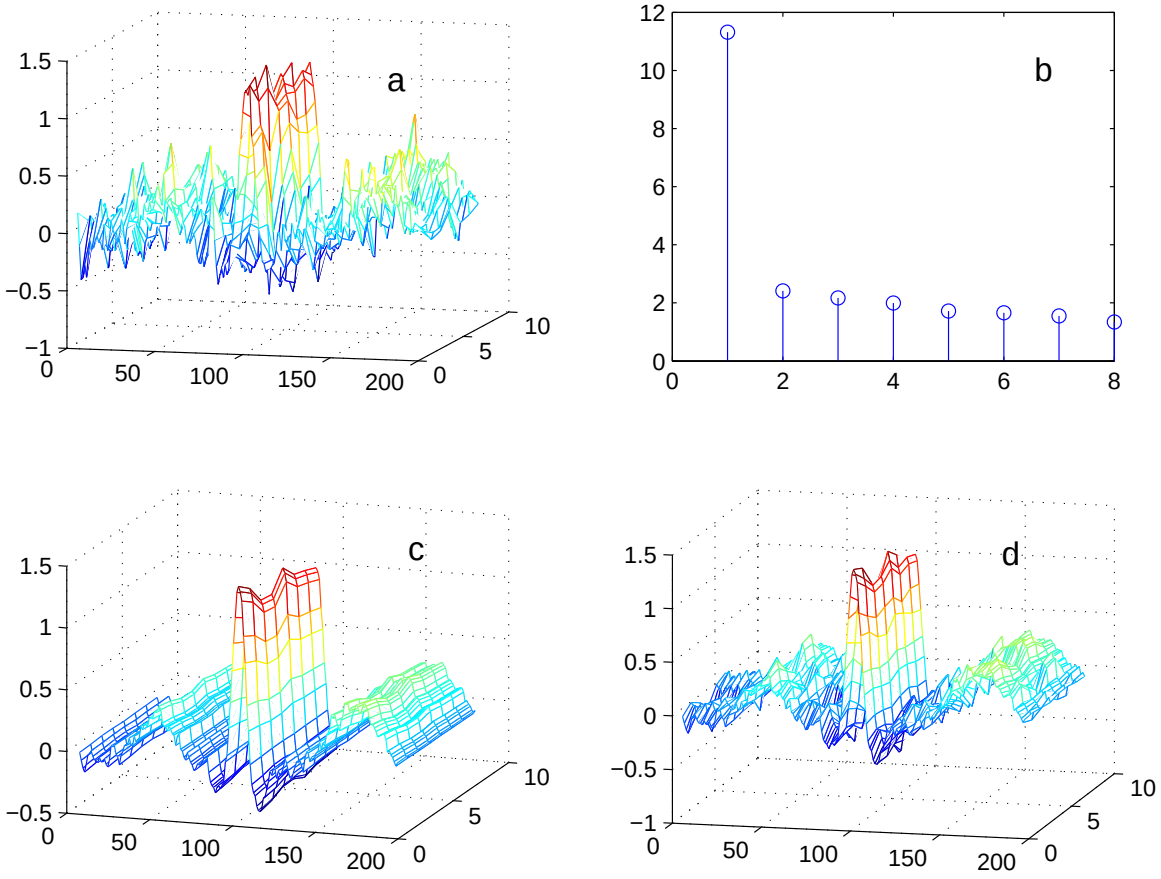

Figure 2: SVD of eight R-peak aligned P-QRS-T complexes; a) in the original form with a large amount of in-band noise, b) eigenspectrum of decomposition, c) reconstruction using only the first principal component, d) reconstruction using only the first two principal components.

truncated SVD (using SVDS()) then the inverse (INV(S)) does not exist. The transformation that effects the filtering is **noninvertible** and information is lost because **S** is **singular**.

From a compression point of view, SVD is an excellent tool. In the above example we have taken 8 heartbeats and compressed them down to 24 numbers; the 8 eigenvalues along the first eigenvector for each beat, the 8-dimensional eigenvector and the eight time stamps that represent the *fiducial point* of each beat. I.e. a compression ratio of 256:24 = 8:3. If the orthogonal subspace is the same for every group of beats, we can simply take any new set of heartbeats and project them onto the first principal component we have derived here. However, it should be noted that the eigenvectors are likely to change, based upon the noise in the signal. Therefore, the correct way to do this is to take a large, representative set of heartbeats and perform SVD upon this [3]. Projecting each new beat onto these *globally derived* basis functions leads to 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 [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 5 eigenvectors are required (and another five for the rest of the beat). At a sampling rate of F_s Hz and an average beat-to-beat interval of $\mathrm{RR}^{\mathrm{av}}$ The compression ratio is therefore $(\frac{60}{\mathrm{RR}^{\mathrm{av}}})\cdot(\frac{m-p}{p}) : 1$ using p principal components. (The first quotient rescales the compression ratio by the average RR interval, since increases in heart rate and hence increases in the average RR interval lead to a reduction in the compression factor. The second quotient is the ratio of the number of principal components considered noise to the number used to encode the signal.)

12.2.4 Signal separation - a general approach

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.

The term *noise* generally means the stochastic or non-periodic[12]) part of the signal we wish to remove in order to analyze the periodic contribution to the signal. However, this term can be misleading, and sometimes the noise is information carrying (if it is not white), transient and can even be deterministic.

[12]There are no spikes in the power spectrum for this section of the signal. Periodic noise (sometimes called artifact) is often *in-band* or non-stationary and therefore cannot be removed by traditional filtering techniques; see the sections on ICA for further discussion of this.

So far we have looked at techniques that deal well with **linear**, **stationary**, **white**, **Gaussian** noise. However, since many signals are nonlinear, nonstationary, colored or non-Gaussian, we should consider how these techniques deal with such signals and use alternative methods when they fail. In the next section we will consider all the sources in a set of observations to be signals (or source components), rather than the conventional signal/noise paradigm and use a higher order moment-based statistical technique for extracting out each source (one or more of which may represent what we conventionally think of as *noise*).

12.3 Blind Source Separation; the *Cocktail Party* Problem

The **Cocktail Party Problem** is the often-quoted illustration for 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. 3. If each of the i voices you can hear at a party are recorded by j microphones, the recordings will be a matrix composed of a set of j vectors, each of which is a (weighted) linear superposition of the i voices. For a discrete set of n samples, we can denote the sources by an $n \times i$ matrix, $\mathbf{Z}$, and the j recordings by an $n \times j$ matrix $\mathbf{X}$. $\mathbf{Z}$ is therefore transformed into the observables $\mathbf{X}$ (through the propagation of sound waves through he room) by multiplying it by a $j \times i$ mixing matrix $\mathbf{A}$ such that $\mathbf{X_{ni}} = \mathbf{A_{ji}Z_{jn}}$.

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 $\mathbf{X}$, produces an estimate $\mathbf{Y}$ of the sources $\mathbf{Z}$. Therefore $\mathbf{W}$ is a set of weights (approximately[13]) 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 the 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 of the sources differ slightly for each recording (microphone) and that the separation of the sources may change in order and volume. The reason for this is explained in the following sections.

[13] depending on the performance details of the algorithm used to calculate $\mathbf{W}$

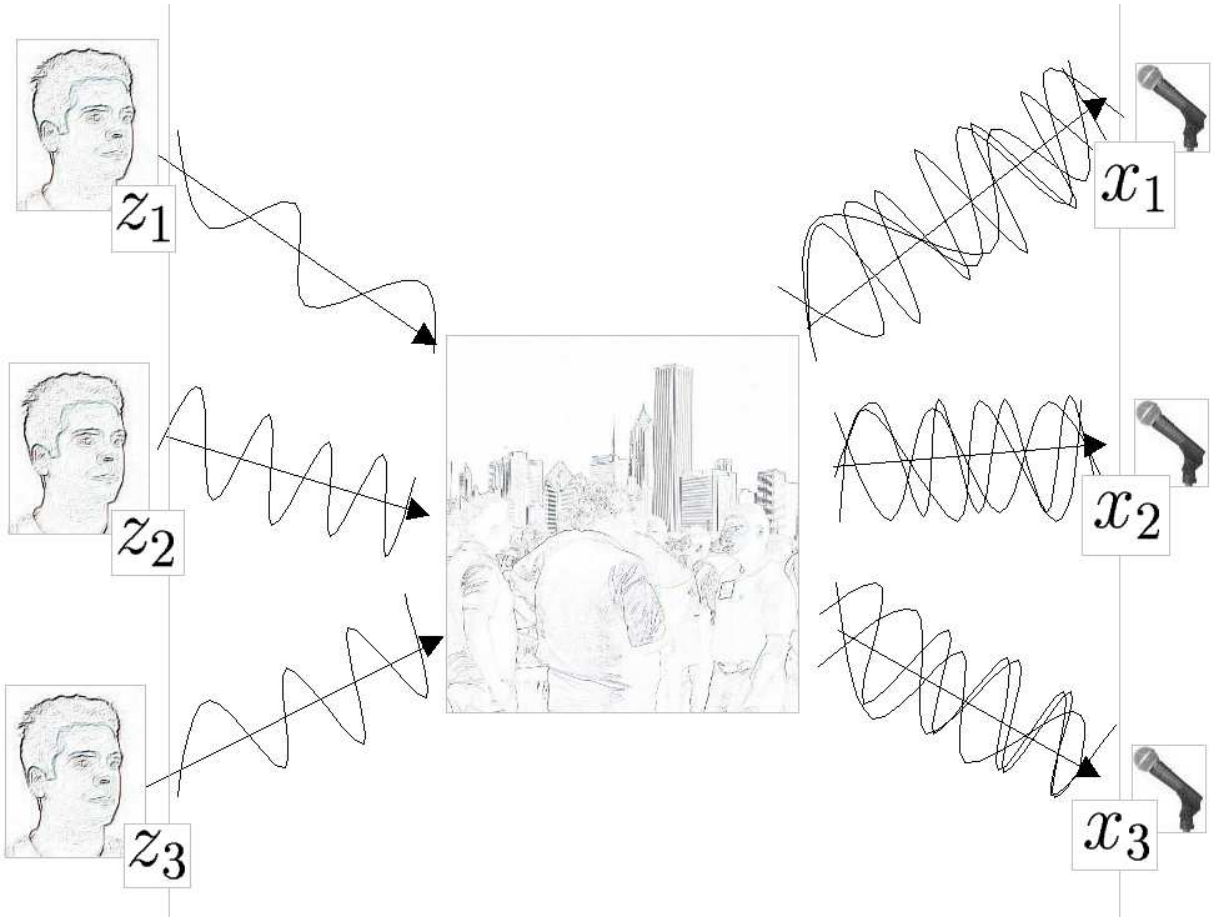

Figure 3: The Cocktail Party Problem: sound waves from M independent speakers (z_1, z_2 and z_3 left) are superimposed at a *cocktail party* (center), and are recorded as three mixed source vectors, x_1, x_2 and x_3 on J microphones (right). The recordings, $\mathbf{X}$ of the sources, $\mathbf{Z}$, are an n-sample linear mixture of the sources, such that $\mathbf{X_{nj}} = \mathbf{A_{ji}Z_{ni}}$, where $\mathbf{A_{ji}}$ is a $J \times M$ linear mixing matrix. An estimate $\mathbf{Y}$, of the sources $\mathbf{Z}$, is made by calculating a demixing matrix $\mathbf{W}$, which acts on $\mathbf{X}$ such that $\mathbf{Y} = \mathbf{WX} = \hat{\mathbf{Z}}$ and $\mathbf{W} \approx \mathbf{A^{-1}}$.

12.4 Independent Component Analysis

Independent Component Analysis is a general name for a variety of techniques which seek to uncover the **independent** signals 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 $j = J$ observed random vectors, $\mathbf{A_{ij}}$ a $M \times J$ mixing matrix and $\mathbf{Z_{in}}$, the $i = M$ (assumed) source vectors such that

$$\mathbf{X} = \mathbf{AZ} \tag{9}$$

ICA algorithms attempt to find a separating or demixing matrix $\mathbf{W}$ such that

$$\mathbf{Y} = \mathbf{WX} \tag{10}$$

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

$$P(Z) = \prod p(z_i) \tag{11}$$

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

The basic idea is to apply operations to the data or the mixing matrix and measure the independence between the output signal channels, (the columns of $\mathbf{Y}$) to derive estimates of the sources, (the columns of $\mathbf{Z}$). 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**[14]). 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[15]. 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. Therefore, in ICA, if we wish to find *independent* sources, we must find a demixing matrix $\mathbf{W}$ that maximizes the non-Gaussianity of $y_i = w_i^T z.$ for all i, where i is the number of sources. I.e., independence *is* non-Gaussianity. 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[16], and we shall stick to the convention $i \equiv m$, 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 independent observations ($i \not> m$), since this is a form of interpolation and a model of the underlying source signals would have to be used. (In terms of §12.2, we have a subspace with a higher dimensionality than the original data[17].)

[14] defined as $kurt(\mathbf{y}) = E\{\mathbf{y}^4\} - 3(E\{\mathbf{y}^2\})^2$ where $E\{\mathbf{y}\}$ is the expectation of the vector $\mathbf{y}$, a column of $\mathbf{Y}$. See § 12.4.1

[15] 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 § 12.5.

[16] See articles on **relevancy determination** [6, 7]

[17] There are methods for attempting this type of analysis; see [8, 9, 10, 11, 12, 13, 14, 15]

The essential difference between ICA and PCA is that PCA uses variance, a second order moment, rather than kurtosis, a fourth order moment, as a 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 therefore not necessarily orthogonal and the angles between the subspace 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[18]. 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.

12.4.1 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 a set of values are well known; the *mean* and the *variance*. These are often sufficient to coarsely characterize a time series. However, they are not unique and one often forgets that the form of the distribution is often required (usually assumed to be Gaussian).

The third moment of a distribution is known as the *skew* and characterizes the degree of asymmetry about the mean. The skew of a random variable x is given by

$$skew(x) = \frac{1}{N}\sum_{j=1}^{N}\left[\frac{x_j - \overline{x}}{\sigma}\right]^3 \tag{12}$$

where σ is the standard deviation, $\overline{x}$ is the mean and N the number of data points. A positive skew signifies a distribution with a tail extending out toward a more positive x and a negative skew signifies a distribution with a tail extending out toward a more negative x (see Fig. 4a).

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. 4b. It is defined by

$$kurt(x) = \frac{1}{N}\sum_{j=1}^{N}\left[\frac{x_j - \overline{x}}{\sigma}\right]^4 \tag{13}$$

where the -3 term makes the value zero for a normal distribution. Note that some conventions add a -3 correction factor to the above equation to make a Gaussian have zero

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

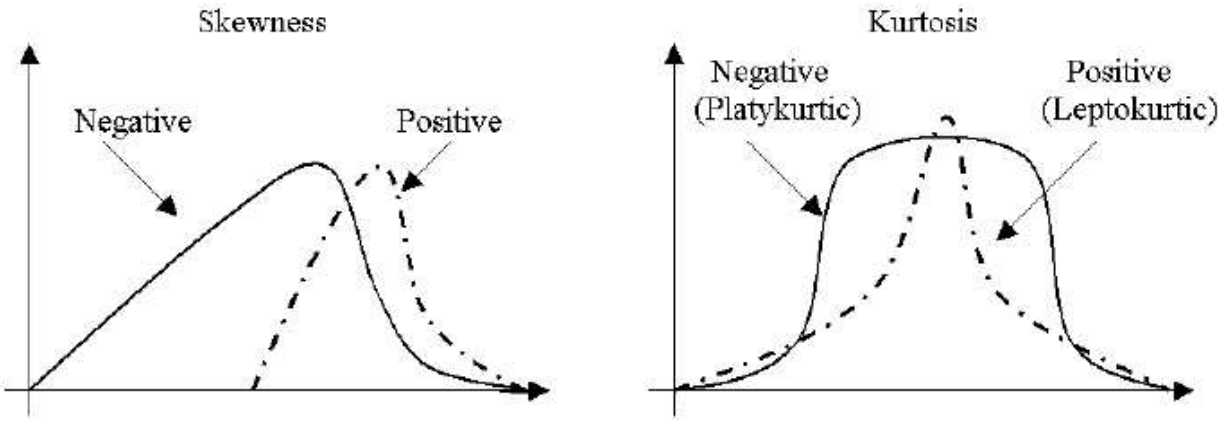

Figure 4: Distributions with third and fourth moments [skewness, (a) and kurtosis (b) respectively] that are significantly different from normal (Gaussian).

kurtosis. This is the convention in Numerical Recipes in C, but Matlab uses the convention without the -3 term and therefore Gaussian distributions have a $kurt = 3$. We shall adopt the Matlab convention in these notes.

A distribution with a positive kurtosis (> 3 in Eq. (13)) is termed *leptokurtic* (or super Gaussian). A distribution with a negative kurtosis (< 3 in Eq. (13)) is termed *platykurtic* (or sub Gaussian). Gaussian distributions are termed *mesokurtic*. Note also that in contrast to the mean and variance, the skew and kurtosis are dimensionless quantities.

Fig. 5 illustrates the 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 $f \approx 33Hz \pm 2Hz$. 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.

12.5 ICA algorithms

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])

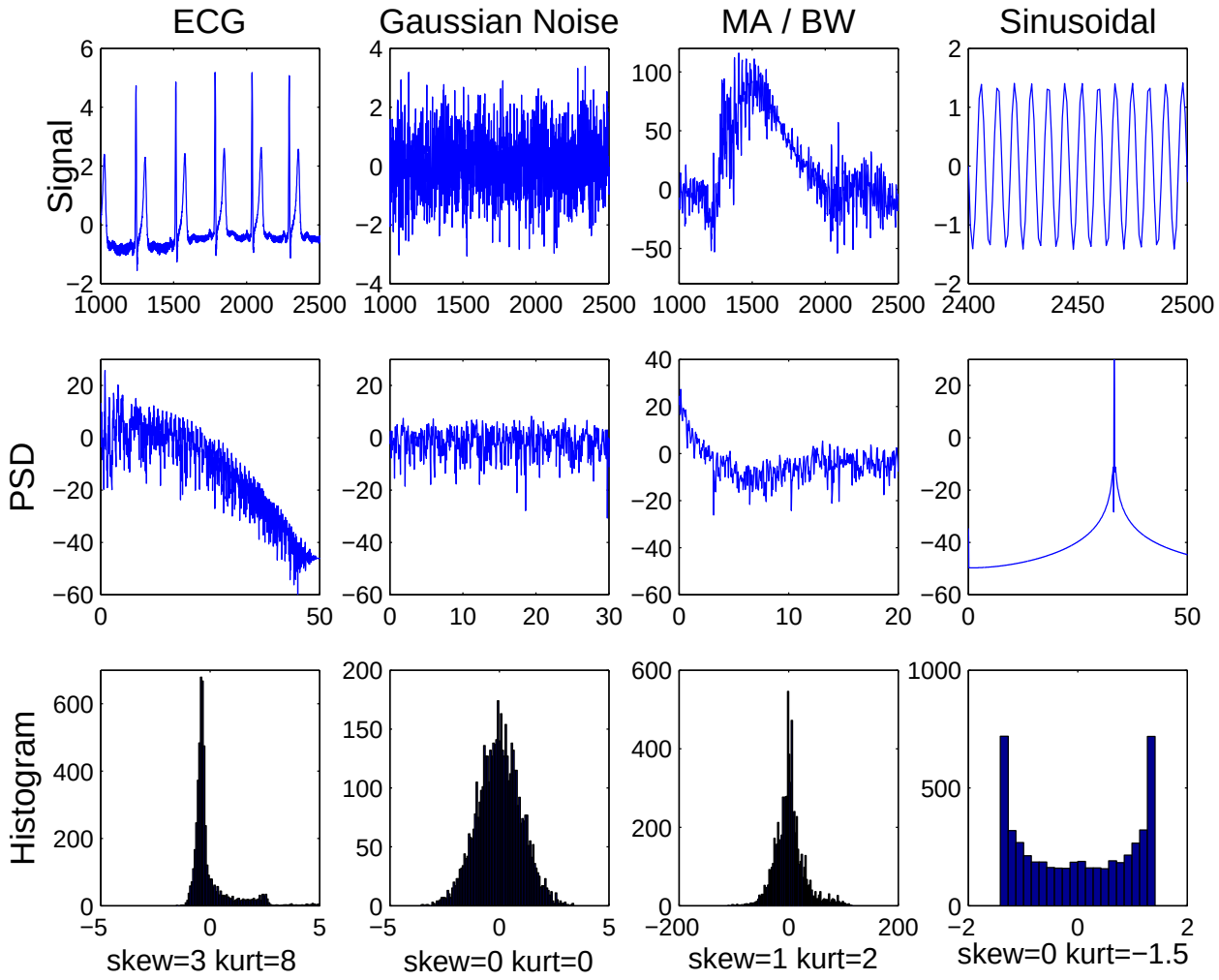

Figure 5: time Series, power spectra and distributions of different signals and noises found on the ECG. From left to right: (i) the underlying Electrocardiogram signal, (ii) additive (Gaussian) observation noise, (iii) a combination of muscle artifact (MA) and baseline wander (BW), and (iv) powerline interference; sinusoidal noise with $f \approx 33Hz \pm 2Hz$.

- Higher order moments and cumulant (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])
- Nonlinear PCA (Karhunen *et al.* [25, 26], Oja *et al.* [27])

Although all these methods use a measure of non-Gaussianity, the actual implementation can involve either a manipulation of the output data, $\mathbf{Y}$, or a manipulation of the demixing matrix, $\mathbf{W}$. We will now examine a selection of these methods that use these techniques.

12.5.1 Maximum Likelihood and gradient descent

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], if we 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 W_{ij}. Note that the transpose of W_{ij} is written W_{ji}. To simplify the derivation we assume the number of sources equals the number of observations ($N = J$), and the data is then defined to be, $D = \{\mathbf{X}^{(n)}\}_{n=1}^{N}$, where N is the number of samples in each of the J observations. The latent variables are assumed to be independently distributed with marginal distributions $P(z_i|\mathcal{H}) \equiv p_i(z_i)$ where $\mathcal{H}$ denotes the assumed form of this model and p_i the assumed probability distributions of the latent variables.

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

$$P(\{\mathbf{X}^{(n)}\}_{n=1}^{N}, \{\mathbf{Z}^{(n)}\}_{n=1}^{N}|\mathbf{W}, \mathcal{H}) = \prod_{n=1}^{N} \left[P(\mathbf{x}^{(n)}|\mathbf{z}^{n}, \mathbf{W}\mathcal{H})P(\mathbf{z}^{(n)}|\mathcal{H})\right] \quad (14)$$

$$= \prod_{n=1}^{N} \left[\left(\prod_{j} \delta(x_j^{(n)} - \sum_{i} W_{ji} z_i^{(n)}) \right) \left(\prod_{i} p_i(z_i^{(n)}) \right) \right] \quad (15)$$

Note that for simplicity we have assumed that the observations $\mathbf{X}$ have been generated without noise[19]. If we replace the term $\delta(x_j^{(n)} - \sum_i W_{ji} z_i^{(n)})$ by a (noise) probability distribution over $x_j^{(n)}$ with mean $\sum_i W_{ji} z_i^{(n)}$ 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 a dimensionless objective function of the parameters Ł($\mathbf{W}$) to be optimized. The

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

sequential update of the parameters w_i are then computed as

$$\Delta w_i = \eta \frac{\partial Ł}{\partial w_i} \tag{16}$$

where η is the learning rate[20].

The quantity $Ł(\mathbf{W})$ we choose to 'guide' the gradient descent of the elements of $\mathbf{W}$. The quantity we choose for ICA (to maximize independence) is the **log likelihood function**

$$\log_e \left(P(D|\mathbf{W}, \mathcal{H}) \right) = \log_e \left(\prod_{\mathbf{n}} P(\mathbf{z}^{(\mathbf{n})}|\mathbf{W}, \mathcal{H}) \right) \tag{17}$$

which is the natural log of a product of the (independent) factors each of which is obtained by *marginalizing* over the latent variables.

OPTIONAL**: (the rest of this subsection is for the avid reader only)**

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

$$\begin{aligned} P(\mathbf{x}^{(n)}|\mathbf{W}, \mathcal{H}) &= \int d^N \mathbf{z}^{(n)} P(\mathbf{x}^{(n)}|\mathbf{z}^{(n)}, \mathbf{W}) P(\mathbf{z}^{(n)}) & (18) \\ &= \mathbf{x}^{(n)} \prod_j \delta(x_j^{(n)} - W_{ji} z_i^{(n)}) \prod_i p_i(z_i^{(n)}) & (19) \\ &= \frac{1}{|\det \mathbf{W}|} \prod_i p_i(W_{ij}^{-1} x_j) & (20) \\ & & (21) \end{aligned}$$

which implies

$$\log_e P(\mathbf{z}^{(n)}|\mathbf{W}, \mathcal{H}) = \log_e \det \mathbf{W} + \sum_i \log_e p_i(W_{ij} x_j). \tag{22}$$

We shall then define

$$\begin{aligned} \frac{\partial}{\partial W_{ji}} \log_e \det \mathbf{W} &= W_{ij}^{-1} = A_{ij} & (23) \\ \frac{\partial}{\partial W_{ji}} W_{kl}^{-1} &= -W_{kj}^{-1} W_{il}^{-1} = -A_{kj} V_{il} & (24) \\ \frac{\partial}{\partial A_{ij}} g &= -W_{jl} \left(\frac{\partial}{\partial W_{kl}} g \right) W_{li} & (25) \\ & & (26) \end{aligned}$$

[20]which can be fixed or variable to aid convergence depending on the form of the underlying source distributions

with g some arbitrary function, $a_i \equiv A_{ij}x_j$ and $f_a(a_i) \equiv d\log_e p_i(a_i)/da_i$. f_a indicates in which direction a_i needs to change to make the probability of the data greater. Using equations 24 and 25 we can obtain the gradient of W_{ji}

$$\frac{\partial}{\partial W_{ji}} \log_e P(\mathbf{x}^{(n)}|\mathbf{W}, \mathcal{H}) = -A_{ij} - a_i \vartheta_{i'} A_{i'j} \tag{27}$$

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

$$\frac{\partial}{\partial A_{ij}} \log_e P(\mathbf{x}^{(n)}|\mathbf{A}, \mathcal{H}) = W_{ji} + x_j \vartheta_i \tag{28}$$

If we choose $\mathbf{W}$ so as to ascend this gradient, we obtain precisely the 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 12.8.3. (Treat this as optional reading).

In practice, the choice of the nonlinearity, ϕ, 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 $\tanh$ nonlinearity ($\phi_i(a_i) = -\tanh(\beta a_i)$), with β a constant initially equal to unity, then we are implicitly assuming the source densities are heavier tailed distributions than Gaussian ($p_i(z_i) \propto 1/\cosh(z_i) \propto 1/(e^{z_i} + e{-}z_i)$, $z_i = f_a(a_i)$, with $f_a = -tanh(a_i)$). Varying β reflects our changing beliefs in the underlying source distributions. In the limit of large β, the nonlinearity 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 nonlinearity, ${}_a(a_i) = -\kappa a_i$, the we are implicitly assuming a Gaussian distribution on the latent variables. However, it is well known [28, 4] that without a nonlinearity, the gradient descent algorithm leads to second order decorrelation. That is, we perform the same function as SVD. 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.

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. 16) which is dimensionally inconsistent; the left hand side has dimensions of $[w_i]$ and the right hand side has dimensions $[w_i]$ ($L(\mathbf{W})$ 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 ([29] 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 $\Delta w_i = \eta \sum_{i'} \mathbf{G}_{ii'} \frac{\partial L}{\partial w_i}$, where $\mathbf{G}$ is a curvature matrix with the i, i' 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 not Euclidian. In fact, one must use the *natural* [23] or *relative* [30] gradient. See [16] and [23] for further discussion on this topic.

12.5.2 Negentropy as a cost function

Although kurtosis is theoretically a good measure of non-Gaussianity, it 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 is of a discrete random variable Y is defined as

$$H(Y) = -\sum_i P(Y = a_i) \log_e P(Y = a_i) \tag{29}$$

where the a_i are the possible values of Y. This very well known definition can be generalized for continuous-valued random variables and vectors, in which case it is often called differential entropy. The differential entropy H of a random vector $\mathbf{y}$ with density $f(\mathbf{y})$ is defined as

$$H(\mathbf{y}) = -\int f(\mathbf{y}) \log_e f(\mathbf{y}) \mathrm{d}\mathbf{y}. \tag{30}$$

A fundamental result of information theory is that a Gaussian variable has the largest entropy among all random variables of equal variance. 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 "spikey".

To obtain a measure of non-Gaussianity that is zero for a Gaussian variable and always nonnegative, one often uses a slightly modified version of the definition of differential entropy, called **negentropy**. Negentropy J is defined as follows

$$J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y}) \tag{31}$$

where $\mathbf{y}_{gauss}$ is a Gaussian random variable of the same covariance matrix as $\mathbf{y}$. Due to the above-mentioned properties, negentropy is always non-negative, and it is zero if and only

if $\mathbf{y}$ has a Gaussian distribution. Negentropy has the additional interesting property that it is invariant for invertible linear transformations.

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 computationally very difficult. Estimating negentropy using the definition would require an estimate (possibly nonparametric) of the PDF. Therefore, simpler approximations of negentropy are very useful. One such approximation actually involves kurtosis:

$$J(y) \approx \frac{1}{12} E\{y^3\}^2 + \frac{1}{48} kurt(y)^2 \tag{32}$$

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

$$J(y) \approx \left[E\{G(y)\} - E\{G(\eta)\}\right], \tag{33}$$

where η is a zero mean unit variance Gaussian variable and the functions G_i are some non-quadratic functions which lead to the approximation always being non-negative (or zero if y has a Gaussian distribution). G_i is usually taken to be $\frac{1}{a_1}\log_e \cosh(a_1 u)$ or $G(u) = -e^{-\frac{u^2}{2}}$ with $1 \leq a_1 \leq 2$. If G(y)=y, this degenerates to the definition of kurtosis.

12.5.3 Mutual Information based ICA

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

$$I(y_1, y_2, ..., y_m) = \sum_{i=1}^{m} H(y_i) - H(\mathbf{y}). \tag{34}$$

MI is an intuitive measure of the (in-) dependence between random variables and is equivalent to the Kullback-Leibler divergence between the joint density $f(\mathbf{y})$ and the product of its marginal densities [31]. 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, like PCA and related methods.

Note that for an invertible linear transformation $\mathbf{Y} = \mathbf{AX}$,

$$I(y_1, y_2, ..., y_m) = \sum_{i=1} H(y_i) - H(\mathbf{x}) - \log_e \|\mathbf{A}\|. \tag{35}$$

If we constrain the y_i to be uncorrelated and have unit variance then $E\{\mathbf{yy^T}\} = \mathbf{A}E\{\mathbf{xx^T}\}\mathbf{A}^T$ $= \mathbf{I}$ and this implies that $\|\mathbf{I}\| = 1 = (\|\mathbf{A}E\{\mathbf{xx^T}\}\mathbf{A}^T\|) = \|\mathbf{A}\|\|E\{\mathbf{xx^T}\|\|\mathbf{A^T}\|$ and this implies that $\|\mathbf{A}\|$ must be constant. Moreover, for y_i of unit variance, entropy and negentropy differ only by a constant, and a sign. Thus,

$$I(y_1, y_2, ..., y_m) = c - \sum_{i=1} J(y_i) \tag{36}$$

where c is a constant that does not depend on $\mathbf{A}$. This shows the fundamental relationship between negentropy and MI.

12.5.4 Fourth order cumulant based ICA algorithm; JADE

The JADE (Joint Approximate Diagonalization of Eigen-matrices) algorithm developed by Cardoso [32] is summarized as follows:

1. **Initialization:** Estimate a whitening matrix using SVD $\theta = \mathbf{V}\mathbf{S}^{\frac{1}{2}}\mathbf{V}^{\mathbf{T}}$ and premultiply the observations by this: $\mathbf{X} \rightarrow \theta\mathbf{X}$.
2. **Form the statistics:** Estimate a maximal set $\{Q_X\}$ of the cumulant matrix

$$Q_x(M) = E\{(x^T M z)xx^T\} - C_x tr(MC_x) - C_x M_x - C_x M^T C_x, \tag{37}$$

   where z is an $n \times 1$ random vector, M is any $n \times n$ matrix, $tr()$ denotes the trace of a matrix and $\mathbf{C_x} = \mathbf{E}(\mathbf{X^T X})$ is the covariance matrix.
3. **Optimize orthogonal contrast**: Find a rotation matrix $\mathbf{R}$ such that the cumulant matrix is as diagonal as possible.
4. **Separate**: Estimate the mixing matrix $\mathbf{A}$ as $\mathbf{W} = \mathbf{R}\theta^{-1}$ and the sources as $\mathbf{Z} = \mathbf{R}^{-1}\mathbf{X}$.

12.6 Applications of ICA

ICA has been used to perform signal separation in many different domains. These include:

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

Another example of source separation is maternal/fetal ECG separation from an abdominal recording (which contains both the mother and the fetal ECG signals superimposed). We will, however, leave an analysis of this to the laboratory assignment, and as a simpler example, we will study the problem of transient noise on the ECG.

12.6.1 ICA for removing noise on the ECG

Figure 6 provides an excellent illustration of the power of ICA to remove 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.

Figure 6: 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.

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.

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

Figure 7: Data from Fig. 1 after ICA decomposition, (Fig 6) and reconstruction. See text for explanation.

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 decomposed data to "restore' the original ECG observations. An example of this using the data in Fig. 1 and Fig. 6 are presented in Fig. 7.

12.7 Limitations of ICA

12.7.1 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 above 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[21] 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[22]. 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

[21] see Eq. 108 in Appendix 12.8.5

[22] assuming an idealized sinusoidal respiratory cycle

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.

12.8 Summary and further reading

The basic idea in this chapter has been to explain how we can apply a transformation to a set of observations in order to project them onto a set of basis functions 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 decorrealtion through PCA. If the contrast function is fourth order and therefore related to Gaussianity, then we achieve ICA. The manner in which we iteratively estiamte the demixing matrix using a Gaussian-related cost function encodes our prior beliefs as to the Gaussianity (kurtosis) of the source distributions. The data projected onto the source components is as statistically independent as possible. We may then select which projections we are interested in and, after discarding the uniteresting 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.* [39] 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 functions 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 [40], PCA [41], neural networks [42], Independent Factor Analysis Wiener filtering [11], wavelets [43, 44, 45], hidden Markov modeling [46, 7, 47], Kalman filtering [48] and nonlinear dynamics [14, 49]. 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 interesting research. For further reading on ICA and related research, the reader is encouraged to browse at the following URLs:

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

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, 2001. URL `citeseer.ist.psu.edu/penny01ica.html`.

[7] Choudrey RA, Roberts SJ. Bayesian ica with hidden markov model sources. In International Conference on Independent Component Analysis. Nara, Japan, 2003; 809–814. URL `www.robots.ox.ac.uk/ parg/publications.html`.

[8] Joho M, Mathis H, Lambert R. Overdetermined blind source separation: Using more sensors than source signals in a noisy mixture, 2000. URL `citeseer.ist.psu.edu/joho00overdetermined.html`.

[9] Lee T, Lewicki M, Girolami M, Sejnowski T. Blind source separation of more sources than mixtures using overcomplete representations, 1998. URL `citeseer.ist.psu.edu/lee98blind.html`.

[10] Lewicki MS, Sejnowski TJ. Learning overcomplete representations. Neural Computation 2000;12(2):337–365. URL `citeseer.ist.psu.edu/lewicki98learning.html`.

[11] Single SW. Non negative sparse representation for wiener based source. URL `citeseer.ist.psu.edu/702512.html`.

[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, 1996. http://www.inference.phy.cam.ac.uk/mackay/abstracts/ica.h tml.

[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. URL citeseer.ist.psu.edu/pearlmutter97maximum.html.

[18] Cardoso J. Infomax and maximum likelihood for blind source separation. IEEE Signal Processing Letters April 1997;4(4):112–114. URL citeseer.ist.psu.edu/cardoso97infomax.html.

[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. URL citeseer.ist.psu.edu/bell95informationmaximization.html.

[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. URL citeseer.ist.psu.edu/amari96new.html.

[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. URL citeseer.ist.psu.edu/karhunen95nonlinear.html.

[27] Oja E. The nonlinear PCA learning rule and signal separation – mathematical analysis, 1995. URL citeseer.ist.psu.edu/oja95nonlinear.html.

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

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

[30] Cardoso J, Laheld B. Equivariant adaptive source separation, 1996. URL citeseer.ist.psu.edu/cardoso96equivariant.html.

[31] Cardoso JF. Entropic contrasts for source separation. In Haykin S (ed.), Adaptive Unsupervised Learning. 1996; .

[32] Cardoso JF. Multidimensional independent component analysis. Proc ICASSP 1998; Seattle.

[33] 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.

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

[35] 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. URL citeseer.ist.psu.edu/makeig96independent.html.

[36] 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; URL citeseer.ist.psu.edu/article/jung97extended.html.

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

[38] 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; URL http://208.164.121.55/hbm2003/abstract/abstract840.htm.

[39] Lee TW, Girolami M, Bell AJ, Sejnowski TJ. A unifying information-theoretic framework for independent component analysis, 1998. URL citeseer.ist.psu.edu/lee99unifying.html.

[40] 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. URL citeseer.ist.psu.edu/9536.html.

[41] Karhunen J, Pajunen P, Oja E. The nonlinear PCA criterion in blind source separation: Relations with other approaches. Neurocomputing 1998;22:520. URL citeseer.ist.psu.edu/karhunen98nonlinear.html.

[42] Amari S, Cichocki A, Yang H. Recurrent neural networks for blind separation of sources, 1995. URL citeseer.ist.psu.edu/amari95recurrent.html.

[43] 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.

[44] Portilla J, Strela V, Wainwright M, Simoncelli E. Adaptive wiener denoising using a gaussian scale mixture model in the wavelet domain; 37–40. URL citeseer.ist.psu.edu/569459.html.

[45] 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. URL citeseer.ist.psu.edu/simoncelli99bayesian.html.

[46] Penny W, Everson R, Roberts S. Hidden markov independent components analysis, 2000. URL citeseer.ist.psu.edu/penny00hidden.html.

[47] Stephen WP. Hidden markov independent components for biosignal analysis. URL citeseer.ist.psu.edu/397426.html.

[48] Everson R, Roberts S. Particle filters for nonstationary ica, 2001. URL citeseer.ist.psu.edu/everson01particle.html.

[49] 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.

[50] 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.

[51] 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.

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

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

Appendix A:

12.8.1 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 $\mathbf{u}_i$

$$\mathbf{x} = \sum_{i=1}^{d} z_i \mathbf{u}_i \tag{38}$$

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

$$\mathbf{u}_i \mathbf{u}_j = \delta_{ij} \tag{39}$$

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

$$z_i = \mathbf{u}_i^T \mathbf{x} \tag{40}$$

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

$$\widetilde{\mathbf{x}} = \sum_{i=1}^{p} z_i \mathbf{u}_i + \sum_{i=p+1}^{d} b_i \mathbf{u}_i \tag{41}$$

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

$$\mathbf{x}^n - \widetilde{\mathbf{x}}^n = \sum_{i=p+1}^{d} (z_i - b_i) \mathbf{u}_i \tag{42}$$

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

$$E_p = \frac{1}{2} \sum_{n=1}^{N} \sum_{i=p+1}^{d} (z_i - b_i)^2 \tag{43}$$

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

$$b_i = \frac{1}{N} \sum_{n=1}^{N} z_i^n = \mathbf{u}_i^T \bar{\mathbf{x}} \tag{44}$$

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

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^n \tag{45}$$

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

$$\begin{aligned} E_p &= \frac{1}{2}\sum_{n=1}^{N}\sum_{i=p+1}^{d}(\mathbf{u}_i^T(\mathbf{x}^n - \bar{\mathbf{x}}))^2 \\ &= \frac{1}{2}\sum_{n=1}^{N}\mathbf{u}_i^T\mathbf{\Sigma}\mathbf{u}_i \end{aligned} \tag{46}$$

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

$$\mathbf{\Sigma} = \sum_n (\mathbf{x}^n - \bar{\mathbf{x}})(\mathbf{x}^n - \bar{\mathbf{x}})^T \tag{47}$$

It can be shown (see Bishop [29]) that the minimum occurs when the basis vectors satisfy

$$\mathbf{\Sigma}\mathbf{u}_i = \lambda_i\mathbf{u}_i \tag{48}$$

so that they are eigen vectors of the covariance matrix. Substituting (48) into (46) and making use of the orthonormality relation of (39), the error criterion at the minimum is given by

$$E_p = \frac{1}{2}\sum_{i=p+1}^{d}\lambda_i \tag{49}$$

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

Appendix B:

12.8.2 Gram-Schmidt Orthogonalization Theorem

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

$$span\{\mathbf{x_1}, ..., \mathbf{x_m}\} = span\{\varepsilon_1, ..., \varepsilon_m\}. \tag{50}$$

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

$$\mathbf{y_i} = \lambda_1\varepsilon_1 + ... + \lambda_{i-1}\varepsilon_{i-1} + \mathbf{x_i}, \tag{51}$$

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

$$\begin{aligned} \lambda_\nu(\varepsilon_\nu, \varepsilon_\nu) + (\mathbf{x_i}, \varepsilon_\nu) &= 0, \\ \lambda_\nu &= -(\mathbf{x_i}, \varepsilon_\nu) \quad (\nu = 1 : i-1). \end{aligned} \tag{52}$$

Therefore,

$$\mathbf{y_i} = \mathbf{x_i} - (\mathbf{x_i}, \varepsilon_1)\varepsilon_1 - ... - (\mathbf{x_i}, \varepsilon_{i-1})\varepsilon_{i-1}. \tag{53}$$

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

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

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

$$span\{\mathbf{x_1}, ..., \mathbf{x_i}\} \subset span\{\varepsilon_1, ..., \varepsilon_i\} \tag{55}$$

and finally,

$$span\{\mathbf{x_1}, ..., \mathbf{x_i}\} = span\{\varepsilon_1, ..., \varepsilon_i\} \tag{56}$$

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

$$\varepsilon_1 = \mathbf{x_1}/\|\mathbf{x_1}\| = \left[\frac{1}{\sqrt{2}}\ 0\ \frac{1}{\sqrt{2}}\ 0\right]^T. \tag{57}$$

For $\mathbf{y_2}$ we get

$$\mathbf{y_2} = \mathbf{x_2} - (\mathbf{x_2}, \varepsilon_1)\varepsilon_1 = [1\ 1\ 1\ 0]^T - \sqrt{2}\left[\frac{1}{\sqrt{2}}\ 0\ \frac{1}{\sqrt{2}}\ 0\right]^T = [0\ 1\ 0\ 0]^T. \tag{58}$$

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

$$\mathbf{y_3} = \mathbf{x_3} - (\mathbf{x_3}, \varepsilon_1)\varepsilon_1 - (\mathbf{x_3}, \varepsilon_2)\varepsilon_2 = [0\ 1\ 0\ 1]^T - 0 \cdot \left[\frac{1}{\sqrt{2}}\ 0\ \frac{1}{\sqrt{2}}\ 0\right]^T - 1 \cdot [0\ 1\ 0\ 0]^T = [0\ 0\ 0\ 1]^T. \tag{59}$$

Therefore,

$$\varepsilon_3 = \mathbf{y_3}/\|\mathbf{y_3}\| = [0\ 0\ 0\ 1]^T. \tag{60}$$

Appendic C:

12.8.3 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 essentailly a matrix that we train on some data by minmising 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 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 8 shows the basic architecture of a neuron with three inputs.

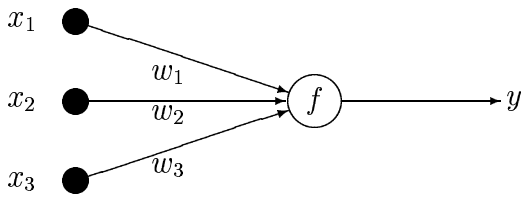

Figure 8: The basic neuron

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

$$y = a = \sum_i w_i x_i \tag{61}$$

For a nonlinear unit, a nonlinear function f, is applied to the linearly weighted sum of inputs. This function is often the sigmoid function defined as

$$f_a(a) = \frac{1}{1 + e^{-a}} \tag{62}$$

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

$$y = f_a\{(\sum_i w_i x_i)\} \tag{63}$$

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 9. 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).

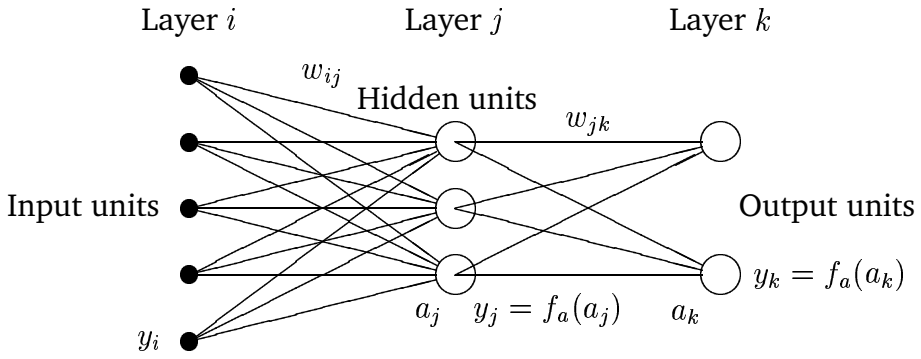

Figure 9: Schematic of a 5-3-2 multi-layer perceptron. Bias units and weights are omitted for clarity.

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

$$y_k = f_a(\sum_j w_{jk} f_a(\sum_i w_{ij} y_i)) \tag{64}$$

Note that our x's 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

$$E = \frac{1}{2}\sum_{N}\sum_{k}(f_a(\sum_{j} w_{jk} f_a(\sum_{i} w_{ij} y_i^p)) - t_k^p)^2 \tag{65}$$

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

Error back-propagation

The squared error, E, can be minimised using the method of gradient descent [29]. 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:

$$w_{jk}^{(\tau+1)} = w_{jk}^{(\tau)} - \eta\frac{\partial E}{\partial w_{jk}} \tag{66}$$

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} - \eta\frac{\partial E}{\partial w_{ij}} \tag{67}$$

The full derivation of the calculation of the partial derivatives, $\frac{\partial E}{\partial w_{ij}}$ and $\frac{\partial E}{\partial w_{jk}}$, is given in Appendix 12.8.4. Using equations 106 and 98 we can write:

$$w_{jk}^{(\tau+1)} = w_{jk}^{(\tau)} - \eta\delta_k y_j \tag{68}$$

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} - \eta\delta_j y_i \tag{69}$$

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

$$\delta_k = (y_k - t_k)y_k(1 - y_k) \tag{70}$$

$$\delta_j = \sum_{k}\delta_k w_{jk} y_j(1 - y_j) \tag{71}$$

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 68 and 69) 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 [29] p262 for more details). However, in the case of an auto-asscociative network, no validation set is required and the training can be terminated when the ratio of the variance of the input and ouput data reaches a plateau. (See [50, 51]).

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 ouputs; the target data is 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 10. PCA, a standard linear dimensionality reduction procedure is also a form of unsupervised learning [29]. 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 § 12.2.1), 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 12.8.1, shows that PCA is based on minimising a sum-of-squares error cost function.

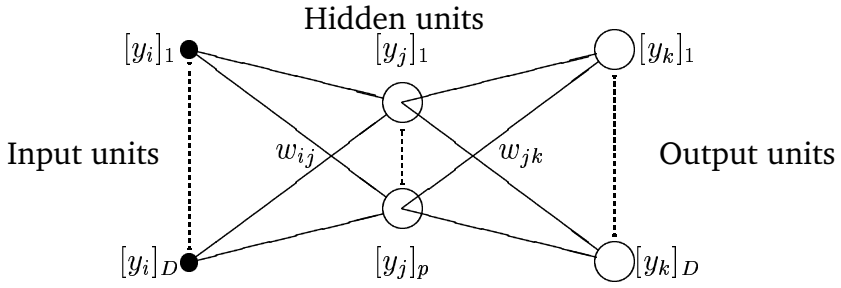

Figure 10: Layout of a D-p-D auto-associative neural network.

Network with linear hidden layer and output units

Since $y_k = a_k$

$$\frac{\partial y_k}{\partial a_k} = 1 \tag{72}$$

the expression for δ_k reduces to

$$\delta_k = \frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial y_k}.\frac{\partial y_k}{\partial a_k} = (y_k - t_k) \tag{73}$$

Similarly for δ_j:

$$\delta_j = \frac{\partial E}{\partial a_j} = \frac{\partial E}{\partial a_k}.\frac{\partial a_k}{\partial y_j}.\frac{\partial y_j}{\partial a_j} = \sum_k \delta_k w_{jk} \tag{74}$$

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 [29], and this shall reduce 'training time' significantly. Consider Eq. (65) where the activation function is linear ($f_a = 1$) for the input and hidden layers;

$$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{j=1}^{p}(\sum_{i=0}^{D} y_i^n w_{ij} - t_j^n)^2 \tag{75}$$

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

$$\sum_{n=1}^{N}(\sum_{i'=0}^{D} y_i^n w_{i'j} - t_j^n)y_i^n = 0 \tag{76}$$

which is written in matrix notation as

$$(\mathbf{Y^T Y})\mathbf{W^T} = \mathbf{Y^T T} \tag{77}$$

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). W has dimension $p \times D$ and elements w_{ij} and T has dimensions $N \times p$ and elements t_j^n. The matrix $(y^T y)$ is a square $p \times p$ matrix which may be inverted to obtain the solution

$$\mathbf{W^T} = \mathbf{Y^\dagger T} \tag{78}$$

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

$$\mathbf{Y^\dagger} = (\mathbf{Y^T Y})^{-1}\mathbf{Y^T} \tag{79}$$

Note that in practice $(\mathbf{Y^T Y})$ usually turns out to be near-singular 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

$$h_j = f(\mathbf{W}_1 x_i + w_{1b}) \tag{80}$$

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

$$y_k = \mathbf{W}_2 h_j + w_{2b} \tag{81}$$

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 $\mathbf{X_i} = \mathbf{U_i S_i V_i^T}$ where 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 [52] as $\mathbf{W}_2 h_j = \mathbf{U_j S_j V_j^T}$ where

$$h_j = \mathbf{H S_j V_j^T} \qquad \text{and} \tag{82}$$

$$\mathbf{W_2} = \mathbf{U_j H^{-1}} \tag{83}$$

with $\mathbf{H}$ 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 [28]

$$\mathbf{W}_1 = \alpha_1^{-1} \mathbf{H U_j^T} \tag{84}$$

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 [28] reduce to

$$\begin{aligned} w_{1b} &= -\alpha_1^{-1} \mathbf{H U}_j^T \mu_X = -\mathbf{U}_j^T \mu_X, \\ w_{2b} &= \mu_X - \alpha_0 \mathbf{U}_j \mathbf{H}^{-1} = \mu_X \end{aligned} \tag{85}$$

where $\mu_X = \frac{1}{N} \sum_N x_i$, the average of the training (input) vectors and $\mathbf{H}$ is here set to be the $(j \times j)$ identity matrix since the output is unaffected by the scaling. Using equations (80) to (85)

$$\begin{aligned} y_k &= \mathbf{W}_2 h_j + w_{2b} \\ &= \mathbf{U_j H^{-1}} h_j + w_{2b} \\ &= \mathbf{U_j H^{-1}} (W_1 x_i + w_{1b}) + w_{2b} \\ &= \mathbf{U_j H_1^{-1-1} H U_j^T} x_i - \mathbf{U_j H^{-1} U_j^T} \mu_X + \mu_X \end{aligned} \tag{86}$$

giving the output of the auto-associative MLP as

$$y_k = \mathbf{U_j U_j^T} (\mathbf{X} - \mu_X) + \mu_X. \tag{87}$$

Equations (83), (84) and (85) 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 [53]. We can see that $\mathbf{W_1} = \mathbf{W_{ij}}$ is the matrix that rotates the each of the data vectors $x_i^n = y_i^n$ in $\mathbf{X}$ into the hidden data $y_i^p{}_j$, which are out p underlying sources. $\mathbf{W_2} = \mathbf{W_{jk}}$ is the matrix that tranforms 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, $\mathbf{W_1} = \mathbf{S V^T} = \mathbf{U U^T}$ and in terms of BSS, $\mathbf{W_1} = \mathbf{W}$.

Appendix D:

12.8.4 Derivation of error back-propagation

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

$$E = \frac{1}{2}\sum_n \sum_k (y_k^n - t_k^n)^2 \tag{88}$$

Which may be written as:

$$E = \frac{1}{2}\sum_n \sum_k (f_a(\sum_j w_{jk} f_a(\sum_i y_i w_{ij})) - t_k^n)^2 \tag{89}$$

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 (90) (91) are given below, η is the learning rate.

$$w_{jk}^{(\tau+1)} = w_{jk}^{(\tau)} - \eta \frac{\partial E}{\partial w_{jk}} \tag{90}$$

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} - \eta \frac{\partial E}{\partial w_{ij}} \tag{91}$$

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

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial y_k} . \frac{\partial y_k}{\partial a_k} . \frac{\partial a_k}{\partial w_{jk}} \tag{92}$$

The input to the output units a_k is given by

$$a_k = \sum_j y_j w_{jk} \tag{93}$$

From (88) and (93) we may write

$$\frac{\partial E}{\partial y_k} = (y_k - t_k), \qquad \frac{\partial a_k}{\partial w_{jk}} = y_j \tag{94}$$

Since y_k is defined as

$$y_k = f_a(a_k) = \frac{1}{1 + e^{-a_k}} \tag{95}$$

We may write

$$\frac{\partial y_k}{\partial a_k} = \frac{\partial}{\partial a_a}(\frac{1}{1 + e^{-a_k}}) = \frac{e^{-a_k}}{(1 + e^{-a_k})^2} = y_k(1 - y_k) \tag{96}$$

Hence

$$\frac{\partial E}{\partial w_{jk}} = (y_k - t_k)y_k(1 - y_k)y_j \tag{97}$$

Therefore we may write the w_{jk} update as

$$w_{jk}^{(\tau+1)} = w_{jk}^{(\tau)} - \eta\delta_k y_j \tag{98}$$

Where

$$\delta_k = \frac{\partial E}{\partial y_k}.\frac{\partial y_k}{\partial a_k} = \frac{\partial E}{\partial a_k} = (y_k - t_k)y_k(1 - y_k) \tag{99}$$

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

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial a_j}.\frac{\partial a_j}{\partial w_{ij}} \tag{100}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial a_k}.\sum_k(\frac{\partial a_k}{\partial y_j}).\frac{\partial y_j}{\partial a_j}.\frac{\partial a_j}{\partial w_{ij}} \tag{101}$$

From (93)

$$\frac{\partial a_k}{\partial y_j} = \sum_k w_{jk} \tag{102}$$

The input to the hidden units is given by

$$a_j = \sum_i y_i w_{ij} \tag{103}$$

Hence

$$\frac{\partial a_j}{\partial w_{ij}} = y_i \tag{104}$$

By symmetry from (96) we have

$$\frac{\partial y_j}{\partial a_j} = y_j(1 - y_j) \tag{105}$$

Therefore from (99),(102),(104) and (105) the update equation for the w_{ij} weights is given by

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} - \eta\delta_j y_i \tag{106}$$

Where

$$\delta_j = \frac{\partial E}{\partial a_j} = \sum_k \delta_k w_{jk} y_j(1 - y_j) \tag{107}$$

Appendix E:

12.8.5 Orthogonal rotation matrices

The classical (orthogonal) three-dimensional rotation matrices are

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

(108)

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.