Machine Learning
Chapter 2: Latent linear models
Part 2: The ICA algorithm

1 Introduction

The purpose of this assignment is to learn basic ICA method through the algorithm knokwn as FastICA [1]. The application to be developed exactly reproduces the example of page 410 of the textbook.

2 Summary of Independent Component Analysis

The algorithm in its version presented in class is restricted to the separation of one of the sources of the signal. Here, the straightforward details to extend the algorithm to multiple signals are explained. Readers interested in going deeper inside in ICA algorithms are encouraged to refer the comprehensive book *Independent Component Analysis*, by Aapo Hyvärinen.

2.1 Signal model

The ICA problem follows from the known as cocktail party problem, where several sources are mixed, and there is no prior knowledge that can be used to separate them. The signal model consists of a set of L sources $\mathbf{z}_i$ that are mixed through a matrix $\mathbf{W}$:

$$\mathbf{x}[n] = \mathbf{W}\mathbf{z}[n] + \varepsilon[n] \tag{1}$$

If the sources are signals transmitted from different positions in the space, the observation $\mathbf{x}[n]$ is a snapshot of dimension $D = L$ that can be thought as the signals registered in a set of sensors. Since both latent variables and

observations have the same dimension, factor matrix $\mathbf{W}$ is square, and it is often called propagation matrix.

2.2 Data whitening

The above model has added noise which is usually thermal in nature, so it can be approximated by a circularly symmetric Gaussian $\mathcal{N}(0, \mathbf{\Psi})$ of dimension D, which here is simplified to zero. The factor matrix is non orthonormal. Nevertheless, the data needs to be whitened, so its autocorrelation matrix is unitary. In order to whiten the data, a PCA is applied to its autocorrelation matrix. The PCA decomposition can be written as

$$\frac{1}{N}\sum_{n}\mathbf{x}[n]\mathbf{x}^{\top}[n] \approx \mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\top} \tag{2}$$

Let $\tilde{\mathbf{X}}$ contain all already whitened observation vectors $\tilde{\mathbf{x}} = \mathbf{B}\mathbf{x}[n]$ by multiplying them by matrix B. We obtain the values of this matrix by expressing its autocorrelation matrix as

$$\frac{1}{N}\mathbf{B}\mathbf{X}\mathbf{X}^{\top}\mathbf{B}^{\top} = \mathbf{I} \tag{3}$$

Since inside this matrix, we find the data autocorrelation matrix $\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\top}$, the above expression can be written as

$$\mathbf{B}\mathbf{X}\mathbf{X}^{\top}\mathbf{B}^{\top} = \mathbf{B}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\top}\mathbf{B}^{\top} = \mathbf{I} \tag{4}$$

For the second equality to be satisfied, we just need to set

$$\mathbf{B} = \mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}} \tag{5}$$

2.3 Probabilistic modelling of the latent variables

Latent variables are modelled with the objective of estimating their posteriors $p(\mathbf{z}[n]|\mathbf{x}[n], \boldsymbol{\theta})$. If zero mean Gaussian priors are chosen, then the linear combination of the latent variables or sources cannot be decomposed uniquely. Then, the criterion in ICA consists of forcing these priors to be independent non Gaussian distributions. The joint distribution is then

$$p(\mathbf{z}[n]) = \prod_{j=1}^{L} p_j(z_i[n]) \tag{6}$$

hence the name of Independent Component Analysis.

The variances of the sources are arbitrary. If these variances are chosen to be unitary, then the actual variances will be absorbed by $\mathbf{W}$ through a proper scaling.

2.4 Maximum Likelihood Estimation of ICA

We will assume hereinafter that the data $\mathbf{x}[n]$ has been whitened in order to avoid the notation $\tilde{\mathbf{x}}[n]$. For the case $L = D$ the factor or mixture matrix is square, and in absence of noise, from equation (1) follows

$$\mathbf{z}[n] = \mathbf{V}\mathbf{x}[n] \tag{7}$$

where $\mathbf{V} = \mathbf{W}^{-1}$ is the set of *recognition weights*.

Since the observation model is deterministically determined by (1), the likelihood of the data can be written as

$$p_x(\mathbf{W}\mathbf{z}[n]) \tag{8}$$

The Papoulis theorem states that $p_y(\mathbf{y}) = p_x(\mathbf{x}) \left|\frac{\partial \mathbf{x}}{\partial \mathbf{y}}\right|$. Applying it to the previous distribution leads to

$$p_x(\mathbf{W}\mathbf{z}[n]) = p_z(\mathbf{z}[n]) \left|\mathbf{W}^{-1}\right| = p_z(\mathbf{W}\mathbf{x}[n]) \left|\mathbf{V}\right| \tag{9}$$

and then, the observed data negative log-likelihood (NLL) can be written as

$$\frac{1}{N} \log p(\mathbf{X}|\mathbf{V}) = \log |\det(\mathbf{V})| + \frac{1}{N} \sum_{j=1}^{L} \sum_{n=1}^{N} \log p_j(\mathbf{v}_j \mathbf{x}[n]) \tag{10}$$

We will force vectors $\mathbf{v}_i$ to be orthonormal. Hence, the first term of expression (10) are a constant that can be dropped in the expression to be optimized. We define the NNL as

$$NLL(\mathbf{V}) = \sum_{j=1}^{L} \mathbb{E}[G_j(\mathbf{v}_j \mathbf{x}[n])] \tag{11}$$

where $G_j(z_j) = -\log p_j(z_j)$.

The objective here is to minimize the NLL subject to the orthonormalization of $\mathbf{V}$:

$$\begin{aligned} L(\mathbf{v}) &= \sum_{j=1}^{L} \mathbb{E}[G_j(\mathbf{v}_j \mathbf{x}[n])] \\ \text{subject to: } & \mathbf{v}^\top \mathbf{v} = 1 \end{aligned} \tag{12}$$

Applying the positive Lagrange multiplier λ the optimization can be expressed as a minimization without constraints:

$$f(\mathbf{v}) = \sum_{j=1}^{L} \mathbb{E}[G_j(\mathbf{v}_j \mathbf{x}[n])] + \lambda(1 - \mathbf{v}^\top \mathbf{v}) \tag{13}$$

The optimization of this functional leads to the recursion (see class notes)

$$\begin{aligned} \mathbf{v}^* &= \mathbb{E}[\mathbf{x} g(\mathbf{v}^\top \mathbf{x})] - \mathbb{E}[g'(\mathbf{v}^\top \mathbf{x})]\mathbf{v} \\ \mathbf{v}^* &\longrightarrow \frac{\mathbf{v}^*}{||\mathbf{v}^*||} \end{aligned} \tag{14}$$

where $g\cdot$ and $g'(\cdot)$ are respectively the first and second derivatives of function $G(\cdot)$.

2.5 Computation of all ICA components

The previous section shows how to compute one of the recognition weights $\mathbf{v}$. Since these vectors should be forced to be orthogonal, the recursion to find them consists on just adding the orthogonalization of the successive components. The algorithm can be written as:

1. Update vector i:

$$\mathbf{v}_i \rightarrow \mathbb{E}[\mathbf{x} g(\mathbf{v}_i^\top \mathbf{x})] - \mathbb{E}[g'(\mathbf{v}_i^\top \mathbf{x})]\mathbf{v}_i$$

2. Force orthogonality with respect to all previously updated vectors

$$\mathbf{v}_i \rightarrow \mathbf{v}_i - \sum_{j=1}^{i-1} < \mathbf{v}_i, \mathbf{v}_j > \mathbf{v}_j$$

3. Force the normalization of the vector

$$\mathbf{v} \longrightarrow \frac{\mathbf{v}}{||\mathbf{v}||}$$

This recursion should be repeated until the convergence of the algorithm.

2.6 Prior models

If we decide that our distribution are super-Gaussians, a good choice is to use the zero mean, unit variance logistic distribution

$$p(z) = \frac{e^{-\frac{\pi z}{\sqrt{3}}}}{\frac{\sqrt{3}}{\pi}\left(1 + e^{-\frac{\pi z}{\sqrt{3}}}\right)}$$

that is differentiable $\forall x$. In order to use it in the ICA procedure, we need to compute its logarithm $G(\cdot)$ and the first and second derivatives of this function.

$$G(z) = \log p(z) = -2 \log \cosh\left(\frac{\pi z}{2\sqrt{3}}\right) - \log \frac{4\sqrt{3}}{\pi} \tag{15}$$

Since the variance of the prior is arbitrary, we can change this function by

$$G(z) = \log \cosh(z) \tag{16}$$

and hence its derivatives are

$$\begin{aligned} g(z) &= \tanh(z) \\ g'(z) &= 1 - \tanh^2(z) \end{aligned} \tag{17}$$

3 Simulation

3.1 Data

The following script produces four sources of data similar to the example in the textbook.

```
x=(-14:14)/29;
z1=kron(ones(1,18),x); z1=z1(1:500);
```

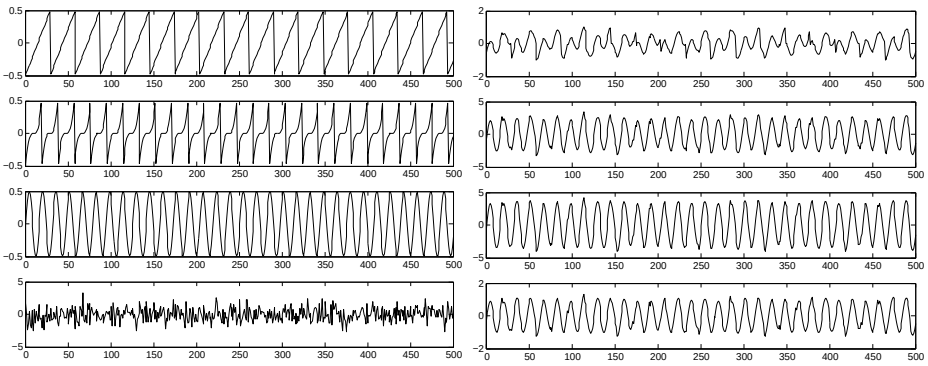

Fig. 1: Left panel: Sources of the example. Right panel: Signals mixed through the propagation matrix $\mathbf{W}$.

```
x=(-9:9)/25;
z2=kron(ones(1,28),x.^3)/0.1;z2=z2(1:500);
z3=sin(2*pi*32*(0:499)/500)/2;
z4=randn(1,500);
W=[1 1 0.5 0.2;0.2 1 1 0.5;0.2 1 1.4 0.4; 1 0.3 0.5 0.5];
X=W'*[z1;z2;5*z3;0.01*z4];
```

The source signals $z_i[n]$ are mixed through matrix $\mathbf{W}$, that has the expression

$$\mathbf{W} = \begin{pmatrix} 1 & 1 & 0.5 & 0.2 \\ 0.2 & 1 & 1 & 0.5 \\ 0.2 & 1 & 1.4 & 0.4 \\ 1 & 0.3 & 0.5 & 0.5 \end{pmatrix}$$

The signals are a sawtooth, a square sawtooth raised to the third power, a sinusoidal and a white noise realization. Figure show the source signals and the observations.

3.2 Implementation of the FastICA algorithm

- Use the algorithm described above to separate the four signals of the example. Initialize matrix $\mathbf{V}$ randomly with a standard Gaussian distribution for each component.

In order to orthonormalize components bfv_i you can use the Gram Smith function

```
function V=gramsmith(V)
for i=1:size(V,2)
    V(:,i)=V(:,i)- V(:,1:i-1)*V(:,1:i-1)'*V(:,i);
    V(:,i)=V(:,i)/norm(V(:,i));
end
```

- Draw a plot of the obtained components.
- Does the algorithm suffer from local minima? Try an explanation of this phenomenon.
- Is it possible to properly estimate the noise? Why?
- Draw an estimate of the evolution of the log likelihood.
- A common improvement of the algorithm consists of the modification of the data so the autocorrelation of each of the mixed signals with respect to time is $r_{x_i}[\tau] = \delta(\tau)$, so signals are white in time. Find a method to whiten signals $x_i[n]$ in time.
- Repeat the above experiments. Do you find any difference?
- Explain alternative methods to improve the presented algorithm. Cite all used references about ICA methods.

References

[1] Oja E. Hyvärinen, A., "Independent component analysis: Algorithms and applications," *Neural Networks*, vol. 13, no. 4–5, pp. 411–430, 2000.