Density Transformations and Independent Component Analysis

6.1 Density Transformations (2 points)

Note: Let $f(\mathbf{x}) = f(x_1, \ldots, x_n)$ be a function of $\mathbf{x} \in \Omega$ and assume we make a change of variables to a new coordinate system by a mapping $\mathbf{u} = \mathbf{u}(\mathbf{x}) = (u_1(\mathbf{x}), \ldots, u_n(\mathbf{x}))$, whose inverse mapping $\mathbf{x} = \mathbf{x}(\mathbf{u}) = (x_1(\mathbf{u}), \ldots, x_n(\mathbf{u}))$ exists. As we change the coordinate system, the integral over f changes according to

$$\int_\Omega f(\mathbf{x})\mathbf{dx} = \int_{u(\Omega)} f(\mathbf{x}(\mathbf{u})) \left|\det \frac{\partial \mathbf{x}(\mathbf{u})}{\partial \mathbf{u}}\right| \mathbf{du} = \int_{u(\Omega)} f(\mathbf{x}(\mathbf{u})) \frac{1}{\left|\det \frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|}\mathbf{du},$$

where $\frac{\partial \mathbf{x}(\mathbf{u})}{\partial \mathbf{u}}$ is the Jacobi matrix, which is the matrix of the partial derivatives

$$\frac{\partial \mathbf{x}(\mathbf{u})}{\partial \mathbf{u}} = \begin{pmatrix} \frac{\partial x_1(\mathbf{u})}{\partial u_1} & \cdots & \frac{\partial x_1(\mathbf{u})}{\partial u_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_n(\mathbf{u})}{\partial u_1} & \cdots & \frac{\partial x_n(\mathbf{u})}{\partial u_n} \end{pmatrix}$$

and whose determinant $\det \frac{\partial \mathbf{x}(\mathbf{u})}{\partial \mathbf{u}} = (\det \frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}})^{-1}$ is called the Jacobi determinant (also "functional determinant" or "Jacobian"). The absolute value of the Jacobi determinant at a point $\mathbf{u}_0$ corresponds to the factor by which the function $\mathbf{x}$ expands or shrinks volumes near $\mathbf{u}_0$.

Conservation of Probability

Consider the density of a random variable x to be $p_x(x) = e^{-x},\ x \geq 0$. Now define a change of variable from x to u as $u = u(x) = e^{-x}$.

- Calculate $p_u(u)$ using the Jacobi determinant.

6.2 Random Number Generation (4 points)

If $F_x(x)$ is the *cumulative distribution function* (cdf) of a random variable x, then the random variable $z = F_x(x)$ is uniformly distributed on the interval [0,1]. This result allows generation of random variables having a desired distribution from uniformly distributed random numbers.

1. First, the cdf of the desired density is computed, and then
2. the inverse transformation z^{-1} is determined.

The pdf of a *Laplace* distribution with location parameter μ (= mean), and scale parameter $b > 0$ (variance = $2b^2$) is given by

$$p(x) = \frac{1}{2b}\, exp\left(-\frac{|x-\mu|}{b}\right),$$

- Following the procedure described above, give a formula for generating samples of a scalar random variable with a Laplacian distribution from uniformly distributed samples.
- Implement your procedure for verification and generate 500 samples for a Laplacian random vector $\mathbf{X}$ with a specific mean 1 and variance 2.

6.3 ICA (4 points)

This exercise requires you to implement the Infomax Principle for Independent Component Analysis (ICA) and apply natural gradient learning. The required data files sounds.zip can be downloaded from the ISIS platform.

Initialization

- Load the sound files `sound1.dat` and `sound2.dat` (packed in `sounds.zip`). Each of the $N = 2$ sources is sampled at at 8192 Hz and contains $p = 18000$ samples. In Matlab you can use `soundsc` to play them.
- Create a random $N \times N$ mixing matrix $\mathbf{A}$ and mix the sources: $\mathbf{x} = \mathbf{A}\mathbf{s}$
- Permute the columns of the $N \times p$ data matrix $\mathbf{x}$ randomly.
- Calculate the correlations between the sources and the mixtures: $\rho_{\mathbf{s},\mathbf{x}} = \frac{cov(\mathbf{s},\mathbf{x})}{\sigma_{\mathbf{s}}\sigma_{\mathbf{x}}}$
  In Matlab you can use `corr` or `corrcoef`.
- Center the data to zero mean.
- Initialize the unmixing matrix $\mathbf{W}$ with random values.

Optimization

- Use the logistic function for the transformation $\hat{f}$ and calculate the *natural gradient* (see lecture notes) and update matrix $\mathbf{W}$ to implement an *online* learning procedure.
- Choose a suitable learning rate η and apply it to the data to unmix the sources.

Hint: Implement the matrix formulation of the algorithm. This should reduce your code for this part to one loop (over the samples) and a few lines.

Results

The recovered signals (estimated sources) are given by: $\hat{\mathbf{s}} = \mathbf{W}\mathbf{x}$

- Play and plot the original sounds, the mixed sources (before and after permutation) and the recovered signals.
- Calculate the correlations (as above) between the true sources and the estimations.

Total points: 10