Machine Intelligence 2 **Exercise Sheet 5**
SoSe 2020, Obermayer/Kashef due: 2020-05-28 at 23:55

Independent Component Analysis: Infomax

This exercise is about implementing the *Infomax Principle* for Independent Component Analysis (ICA) using gradient based learning. The files `sound1.dat` and `sound2.dat` in `sounds.zip` contain recordings of two acoustic sources. Your implementation should contain the following steps:

Exercise 5.1: Initialization (2 points)

(a) Load the sound files. Each of the $N = 2$ sources is sampled at 8192 Hz and contains $p = 18000$ samples.

(b) Create a random (& invertible[1]) $N \times N$ mixing matrix $\underline{\mathbf{A}}$ and mix the sources:

$$\underline{\mathbf{x}}^{(\alpha)} = \underline{\mathbf{A}}\,\underline{\mathbf{s}}^{(\alpha)} \qquad \alpha = 1, \ldots, p$$

(c) Remove the temporal structure by permuting randomly the columns of the $N \times p$ data matrix $\underline{\mathbf{X}} = (\underline{\mathbf{x}}^{(1)}, \ldots, \underline{\mathbf{x}}^{(p)})$. Use this shuffled data in all subsequent steps.

(d) Calculate the correlations between the sources and the mixtures: $\rho_{s_i,x_j} = \dfrac{\mathrm{cov}(s_i, x_j)}{\sigma_{s_i}\sigma_{x_j}}$, with covariance in the numerator and standard deviations in the denominator.

(e) Center the data s.t. that each observed variable x_i has zero mean.

(f) Initialize the unmixing matrix $\underline{\mathbf{W}}$ with random values.

Exercise 5.2: Optimization (4 points)

Implement a *matrix version* of the ICA *online* learning algorithm that iterates as often as required over the training data. For $\hat{f}$ use the logistic function (see lecture slides). This should reduce your code for this part to a few lines. Implement two variants of this learning algorithm:

(a) Compute the update matrix $\Delta\underline{\mathbf{W}}$ using the standard gradient.

(b) Compute the update matrix $\Delta\underline{\mathbf{W}}$ using the *natural gradient* as described in the lecture.

(c) Find a suitable learning rate ε that decays exponentially (but sufficiently slowly: e.g. $\varepsilon_0 = 0.01$, $\varepsilon_{t+1} = 0.9999\varepsilon_t$), and apply both gradient algorithms to the data (after it has been shuffled and centered) for unmixing the sources.

Hint: :

You can start by implementing the component-wise update of the weights before figuring out the matrix version. The only advanatage of the matrix version is to speed up your implementation and to practice how to "vectorize" your implementation. You can use the component-wise implementation as a means to verify your implementation.

[1] You can simply check that the matrix is invertible and re-create it if it's not.

Exercise 5.3: Results (4 points)

(a) Plot & Play (i) the original sounds (e.g. use `scipy.io.wavfile` to save playable files), (ii) the mixed sources (before and after the data permutation), and (iii) the recovered signals (estimated sources) $\underline{\hat{\mathbf{s}}} = \underline{\mathbf{W}}\,\underline{\mathbf{x}}$ using the unpermuted data.

(b) Calculate the correlations (as above) between the true sources and the estimations.

(c) For every 1000$^{\text{th}}$ update, plot $||\Delta\underline{\mathbf{W}}||_F^2 := \sum_{i=1,j=1}^{N}(\Delta w_{ij})^2$ (i.e. the square of the Frobenius norm) to compare the convergence speed for the two gradient methods. Whiten your data before applying ICA and compare the learning speeds again. Describe the differences between the two variants of the learning algorithm.

(d) Plot the density of the mixed, unmixed, and true signals & interpret your results.

total: 10 points