Lecturer: Henning Sprekeler
Assistant: Owen Mackwood

Slow Feature Analysis

April 28, 2016

The solutions for these exercises (comprising source code, discussion and interpretation as an IPython Notebook) should be handed in by **May 5 8am** through the Moodle interface. The solutions will be discussed in the tutorial on May 9. **If you have any questions or find problems, please send an email to owen.mackwood@bccn-berlin.de. We will post questions and answers of general interest on Moodle.**

Exercise 1: Slow Feature Analysis on two independent signals

This will be the first exercise where we apply Slow Feature Analysis (SFA) to investigate signals. To begin with, we focus on linear SFA transformations on a 2-dimensional time-dependent signal.

1. Write a function that generates a random signal of length T (number of samples) that has an average frequency spectrum corresponding to PSD($\mathbf{k}$) $\sim exp(-\frac{||k||^2}{2\epsilon})$. To compute $\mathbf{k}$, you may use the function `fftfreq`. The function should accept the parameters ϵ and T. You may rely on the function *gaussian_spectrum_1D* from the new *helper.py* file that is provided on Moodle.

2. Use the function from the task above to generate two independent 1D-signals with a PSD($\mathbf{k}$) where $\epsilon = \frac{1}{\tau^2}$. Each signal should have the same length of T=1000 samples, but substantially different τ-values, e.g. $\tau_1 = 30$ and $\tau_2 = 100$. Normalise the signals, so that both have zero mean and unit variance. Plot both signals into a single figure. How does τ influence the characteristics of the signals?

3. We will now perform linear SFA to see whether we can generate a slow signal by linearly combining both signals. In order to do so, first join them into a single 2x1000 matrix X. Compute the joint covariance matrix of both signals $C = cov(X)$ as well as the matrix of second moments of the temporal derivative $\dot{C} = \dot{X}\dot{X}^T/(T-1)$ (Hint: use the function `diff` for the derivative). Both matrices C and $\dot{C}$ will have a dimensionality of 2x2. Next, to perform SFA, we need to solve the Generalised Eigenvalue Problem of both matrices: $\dot{C}\vec{w} = \lambda C\vec{w}$. Use the *scipy.linalg*-function *eig* which provides as output the eigenvalues (as a vector) and eigenvectors (as a matrix, where the columns represent the vectors in the same order as the eigenvalues). Normalise the eigenvectors to length 1.

   Hint: `Numpy` knows two matrix-like objects, *matrix* and *array*, that can be generated and cast into each other with commands by the same name. If you want to do a true matrix multiplication, you need to either use `dot(array1,array2)` or `matrix1*matrix2` depending on the type you use.

4. The eigenvectors determine how the input signals are mixed by the SFA to generate the new (slow) signals. What kind of mixture do you ideally expect given how the

data was generated? What does the empirical mixture look like, judging from the extracted eigenvectors?

5. Repeat task 3 with different signal sample sizes (ranging logarithmically from $10 - 10'000$ in 20 steps) and store the resulting normalised eigenvectors (Hint: write another function that accepts as input the sample size and outputs the normalised SFA-eigenvectors). Plot the four individual components of the eigenvectors against the number of samples. How does the SFA-mixture change depending on the length of the input signal? Can you give an intuition why?

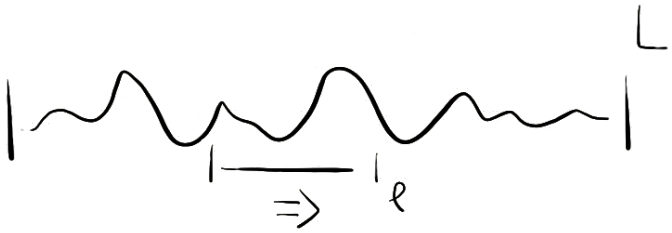
6. How do the eigenvalues depend on the τ's that you chose (if you are unsure, try different combinations of τ_1 and τ_2. Also, longer signals will reduce errors in the numerical relationship).

Exercise 2: SFA on a high-dimensional correlated signal

In this task, we will use linear SFA to learn 1D-receptive fields. We first generate a large "1D-image" and then slide a small visual field across this image. In this way we simulate a slowly moving input to the receptive field of a hypothetical visual cell.

1. Generate a 1D-image of length $L = 10'000$ pixels with the same power spectral density as above, but with $\epsilon = 0.04$ (use the function from exercise 1 on the present sheet).

2. Extract a time-dependent receptive field input by sliding a window of length $l = 100$ pixels across the entire image. The portion of the image that is covered by the window represents the current input to the visual cell. Starting from the left, the window is shifted to the right by one pixel in each time step. The width of the window determines the dimensionality of the receptive field input. Generate a matrix X containing the complete receptive field input recorded while the window is sliding over the image. It should have a dimensionality of 100x9900: 100 variables describing the current image, each with 9900 samples over time. Make sure that the average input to each pixel of the receptive field is zero, i.e. each row of the matrix should have zero-mean.



3. Compute the covariance matrix C of the high-dimensional signal X as well as the matrix $\dot{C}$ of the second moments of its temporal derivative $\dot{X}$. Solve the Generalised Eigenvalue Problem $\dot{C}\vec{w} = \lambda C\vec{w}$ using the function *scipy.linalg.eig*. There will be 100 eigenvalues and as many eigenvectors with the a dimensionality of 100 represented in a matrix W.

4. Sort the eigenvectors according to the magnitude of the corresponding eigenvalues. Extract the eigenvectors that correspond to the five slowest SFA-components, i.e. the ones with the smallest eigenvalues. Plot these eigenvectors.

5. How do you interpret the shape of the eigenvectors? In which way do the characteristics of the eigenvectors correspond to the intentions underlying SFA? Discuss the properties of the eigenvectors at the boundaries of the receptive fields (Hint: think about how these boundaries react when sliding over edges in the image). How does the shape of these eigenvectors relate to the structure of the receptive fields in primary visual cortex V1?

6. As a last step we will use the extracted SFA-eigenvectors W to project the original signal (which was generated in the pixel-basis) onto the new SFA-basis. Determine the signal in the SFA-basis Y by multiplying the eigenvectors with the signal ($Y = W^T X$). Plot the five slowest components of the SFA-signal into a single graph (you may add a y-offset to each component so they don't overlay). Did the *Slow* Feature Analysis succeed?