Seminar Week 3: Pre-Processing

Please note there is **more** here than you may possibly handle in one hour so please do not worry. This is for you to work with in your own time.

Q&A session

I will be answering any question you may have on the lectures so far.

Getting familiar with toolboxes

Your assignments will involve writing some code and using machine learning toolboxes. You should start familiarising yourself with your platform of choice. In what follows I will typically assume either Matlab or Python. If you are considering a career in Data Science / Machine Learning, you are probably better off focusing on Python and/or R rather than Matlab.

Python users: In the seminars (including this one) and the first assignment, I will typically provide data stored in a Matlab mat file. You should make sure you know how to load the data in Python. You will need to use function `loadmat` from package `scipy.io`. For more information, read this excellent tutorial.

Histograms and density estimation

Generate 50 random values from a normal (Gaussian) distribution with mean and standard deviation of your choice. In Matlab, use function `normrnd`. In Python, use function: `numpy.random.normal`.

- Create a histogram to visualize the data. In a separate figure plot both the histogram with its area normalized to 1, and the actual distribution you generated your data from (Matlab: use `normpdf`; Python: use `scipy.stats.norm`). Try varying the number of bins to see how this impacts your results.
- Using a standard Gaussian function, use kernel density estimation to estimate the probability distribution. How does using different values of h compare to using a different number of bins for your histogram? If you feel confident, write your own kernel density estimation function. If not (that's OK!), use the following toolboxes. If you are a **Matlab user**, download the kernel density estimator (kde.m) from the Mathworks File Exchange (URL) and compare its output to your own. Some extra material can be found here. If you are a **Python user**, a number of options are available to you (SciPy, Scikit-learn). An excellent resource is: this tutorial. See also the following URL for pointers.

Pre-processing

Here, we will look into simple normalisation, whitening and PCA. We will be working with two-dimensional data as it is much easier to visualise, however, it is important that any function you write can be generalised to an arbitrary number of dimensions. Download the data file **`sem9_q1_data.mat`** from Study Direct and use the following methods to pre-process the data (**Python users: the dictionary entry is 'data'**). For each of these questions, you can consider writing your own code but it's OK if you don't.

- Simple normalisation: Use simple normalisation and plot the original and normalised data on the same graph.
- Whitening: Use the whitening method described in lecture and plot the original and whitened data on the same graph.
- PCA: Use Principal Component Analysis to reduce the dimension of the data to 1 (i.e. use one principal component). What happens if you use two principal components? A good resource here on PCA is: this tutorial (**Python-based**).