Data Analysis and Model Classification
Guidesheet I: Data exploration and clustering

Ruslan Aydarkhanov Bastien Orset Julien Rechenmann
Ricardo Chavarriaga José del R. Millán

October 2, 2017

Data exploration

Any scientific study involves the collection of data. But in most cases, this proves to be time consuming and expensive, which leads to rather small datasets being collected.A good way to start the data exploration is to 'play around' with it, i.e. look at plots and statistical parameters to get a feeling for the data.

For this course, we have stored our datasets as MATLAB files (.mat). You can download the dataset `spikes.mat`. The variable `spikes` contains extracted after epoching `spikes` $\leftarrow$ (spikes $\times$ times) and their projections after principle component analysis in the variable `spikesPCA` $\leftarrow$ (spikes $\times$ principal components) (you will learn PCA later in the course). After loading the data into the MATLAB workspace, the fastest way to get a visual impression of the data distribution is to plot it. We are going to cover three different plot types here.

- **2D plot** To plot the shape of a single spike you can use command `plot(spikes(1,:))`. Use `hold on` to plot a spike in one color and then another spike with different color in the same figure. Or you can plot a set of spikes together.
- **Scatter plot** With function `scatter` you can visualize 2 principal components in 2D, one of the components given in `x` and another one given in `y`.
- **Histogram** A histogram is a representation of the frequency of observations in a certain range, called bin. With `histogram(x)` the program automatically choses the bins, but you can control that number with the command `histogram(x,nbins)`. The input x is a vector. With `hold on` you can overlay the histogram plots of two features.
- **Other** You can check yourself other useful plotting functions provided by MATLAB, e.g. `plotmatrix, gplotmatrix, boxplot, scatter3, imagesc` ...

Hands on

- After loading the dataset identify the size of the feature vector and how many samples it contains.
- Plot spikes (either some random ones, or all of them in one plot) and observe the spike shapes. Do they all look the same? Could you guess how many different neurons there could be?
- Pick two features of the PCA-transformed dataset and visualize them.
- Use the histogram function on different features to see their statistical distributions.
- Create boxplots and compare them with the histograms on the same feature. Which plot gives you more information about: the type of distribution, the mean values, outliers, feature separability?
- Try to plot transformed spikes `plotmatrix(spikesPCA)`. Can you now detect some probable clusters?

k-means clustering

Based on the similarity of the features, we want to cluster the samples into classes (= different neurons in this case). One simple yet effective method is the *k-means* clustering algorithm.

The algorithm assigns a set of samples $s_1, s_2, \ldots, s_n$ to k different clusters $C_1, C_2, \ldots, C_k$ so that the within-cluster sum of square distances is minimized:

$$\underset{C}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{s \in C_i} \|s - \mu_i\|^2 \qquad \text{with} \qquad \mu_i = \text{mean of } C_i$$

The algorithm walks through the following steps:

1. **Initialization** Determines the starting centroids of the clusters. This can be done in many ways. The easiest is to randomly pick from the observations.
2. **Assignment** Every sample gets assigned to the closest centroid according to the chosen norm (hence nearest neighbor).
3. **Update centroids** Update the centroids to be the mean of the assigned samples.

Steps 2 and 3 are repeated until there is no change in assignment anymore.

Hands on

- Use the MATLAB function `kmeans` with the k you guessed before to do the clustering on transformed data `spikesPCA`. Use the resulting vector of assignments `idx` to visualize the results with the MATLAB command `gplotmatrix(spikesPCA,[],idx)`.
- Plot the spike profile for your different neurons, i.e. the mean of the spikes for each cluster. How different are your extracted spike profiles? Can you justify your chosen number of neurons?
- Repeat the clustering multiple times and look at the plots. Do you notice any change? Can you explain why or why not?
- Repeat the process with different number of clusters k. How are the plots changing? The `kmeans` function also provides as an output the within-cluster sum of squared errors. How does it evolve with the number of clusters? Does this metric allow to choose the number of clusters? According to you, what would be the optimal number?
- There is a MATLAB function that allows to evaluate clustering algorithms and find optimal number of clusters `evalclusters` based on an internal criterion. Try to play with the *criterion* parameter. Which of the available criteria give the most reasonable result? Look at the actual formulas in the documentation and think why some of them are performing better on the given data.