DAMC Miniproject 1: Data exploration and clustering

Group 20 - Imler Théo, Freundler Nicolas, Freundler Frederic

October 15, 2017

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

1 Introduction

In a brain-computer experiment, a non human primate used a pole to move a robot arm. An electrode was implanted in the monkey's motor cortex and recorded the surrounding electrical activity for 30 seconds with a sampling frequency of 30 kHz. The signal was then filtered in order to get rid of low-frequency oscillations and high-frequency noise. Each electrical activity exceeding a certain voltage threshold was then labeled as a spike and a 2 ms length window with 64 time points was extracted around it. After further signal processing and *principal component analysis (PCA)* were performed on the data, were were left with the `spikes.mat` dataset file, containing a `spikes` matrix of 6000 samples (spikes x time) and a `spikesPCA` matrix, displaying the spikes' projections after PCA (spikes x principal components), displaying 3 distinct features.

Whereas decoding the monkey's intended movements involves supervised learning and is not part of this project's scope, we however focused on discriminating multi-neurons signals in order to find how many different types of neurons were involved in the monkey's movements. The assumption was made that neuron cells from the same population would elicit firing signals with the same profile. Therefore, in order to segregate each population based on its characteristics signals, an unsupervised learning approach was used.

2 Methods

This project was done using `MATLAB`. At first, we plotted 10 random spikes from the `spikes` matrix, in order to estimate the possible number of different neuron cells populations. Then, we focused on the features from `spikesPCA` and computed their statistical distributions using the `histogram` and `boxplot` commands. With the informations provided by the plots, we deducted the separability of the cells populations based on their features.

Afterwards, we used the `plotmatrix` command on the `spikesPCA` features, which creates scatter plots of the 3 features in order to detect some probable clusters. Finally, we used the *k-means* clustering method - the `kmeans` function of `MATLAB` - in order to segregate k populations of neurons, based on their features. This method aims to group a set of objects in clusters such that the objects of one cluster are more similar to each other than those in the other clusters. Mathematically, it means that the within-cluster sum of square distances is minimized:

$$argmin(C) \sum_{i=1}^{k} \sum_{s \in C_i} ||s - \mu_i||^2$$

It is important to notice that we chose to set the parameter `k` of the *k-means* method to 3, since we deducted 3 possible clusters from our observations (see section 3 for further information on this choice).

We then used a `gplotmatrix` command on the `spikesPCA` features, which creates features scatter plots as with the `plotmatrix` command, but takes into account the number of groups of distributions to display - in this case, 3, coming from the indexation created by the *k-means* method.

We also tried to segregate different numbers of clusters with the *k-means* method, to test the influence of increasing the k number of clusters. In this scope, we especially considered the consequential changes in the `gplotmatrix` display - that is, the visual balance between under/over-clustering - and the change in the within-cluster sums of squared errors - provided as an output by the `kmeans` function. Finally, we used the `evalclusters` function in order to find the optimal number of clusters based on four internal criterions:

- **Calinski-Harabasz** criterion (Variance Ratio Criterion - VRC): aims to minimize the ratio of overall between-cluster variance and the overall within-cluster variances;

- **Silhouette** criterion: allows to inspect the optimal number k of clusters for a given clustering method (here, *k-means*) based on the resemblance (silhouette value) of within-clusters samples and their dissemblance to samples from other clusters;
- **Gap** criterion: from what we understood from the `MATLAB` documentation, this criterion compares error measurements of a certain k number of clusters with that expected from a Monte Carlo sampled distribution, and is very time consuming;
- **Davies-Bouldin** criterion is based on a ratio of within-cluster and between-cluster distances, meaning that the optimal number of k clusters will be determined by the lowest Davies-Bouldin ratio - that is, the within-cluster distances are small and the between-cluster distances are big.

After observing these visual and mathematical possibilities, we draw conclusions on the optimal choice of clusters for our neuron cells population.

3 Results and discussion

From the 10 random plotted spikes (see fig.1), we observed that whereas every signal seems to go up and then down around an initial *resting-state* level, some display a large negative pit before stabilising towards that level, some show the same behaviour but with a smaller pit and others seem to display sort of a positive hill in place of a negative pit. From that, we estimated that there could be 3 different neuron cells populations. Concerning figure 2, whereas the histograms gives more information about the type of distribution and the features separability - here, we observe 3 distinct distributions - the boxplots tells us more about the mean value, which we can estimate from the median and the outliers. Whereas feature 1 is twice more dispersed as feature 2, both display distributions that look alike and two distinct peaks. The corresponding boxplots confirm their resembling distributions. Feature 3's distribution, on the other hand, is tightly packed around 0 and resembles the normal distribution, with a lot of outliers (red dots). Plotting scatter plots of these 3 features with the `plotmatrix` features allowed us to estimate again 3 distinct populations (see fig.3).

Using these observations, we went on to cluster the spikes using the `kmeans` function, setting the number of clusters to *k=3*. The scatter plots made after clustering using the `gplotmatrix` (see fig.4) function revealed a visually satisfying segregation of the samples, and the different mean profiles of the 3 different clusters (see fig.5) confirmed the assumption made from the 10 random spikes that there are 3 different kinds of profile. Furthermore, repeating the clustering 10 times using the `Replicate` option of the `kmeans` function didn't yield any significant change, displaying "`Best total sum of distances = 4090.45`" for each replicate, moving the clusters' centroids of a few ten thousandths between replicates - for instance `[-1.0820 -0.3103 0.0276]` to `[ -1.0812 -0.3104 0.0284]`, having no impact on the visual segregation (`gplotmatrix`). These slight but not significant changes could come from the change in local minima for each replicate of the *k-means* clustering method, since the initial conditions change for each one. Another variation source could be the converging tolerance of the method which might not be small enough.

Then, we investigated the effect of increasing the number of clusters, from 1 to 10. The obvious observation that can be made from the consequential group scatter plot is that an over-clustering occurs when going above *k=3* clusters (see fig.6 for an example with *k=5*): visually, obvious clusters are separated where they shouldn't. In a more mathematical approach, we plotted the within-cluster sum of point-to-centroid distances, provided as an output by the `kmeans` function, for the different number of clusters (see fig.7). This metrics allows us to use any number of clusters (yet not bigger

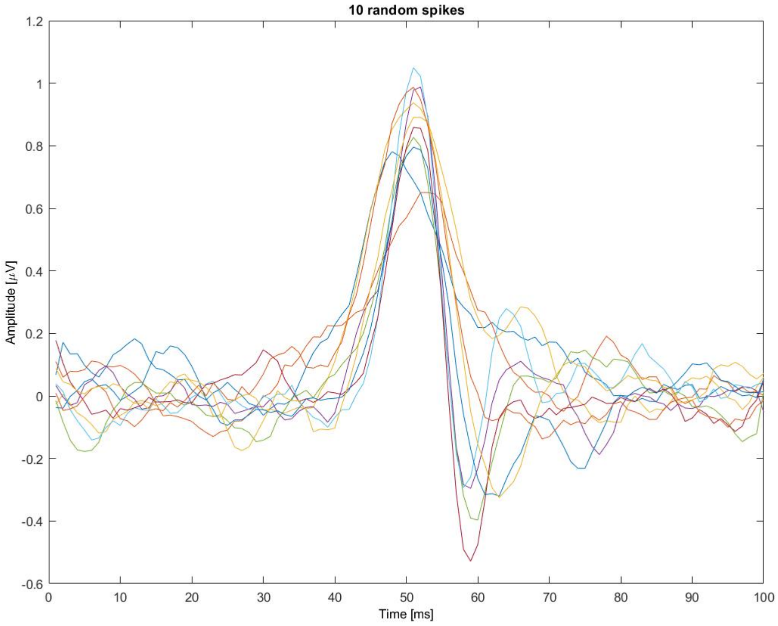

Figure 1: Plot of 10 random spikes in order to guess how many different neuron cells population there could be in our dataset.

than data number). There are two things we want to minimize : the sum of the point-to-centroid distance sums (considered as an error measurement) and the number of cluster (a big number gives a big complexity, until the preposterous case of *k=6000*, where the error is null because each sample is its own cluster centroid). Regarding that, 3 was a good guess, but 2 and 4 could have been good enough considering the error measurement only. However, completed by our initial guess, 3 is the best one. Furthermore, the drastic decrease of the error in figure 7 seemingly stabilises after *k=3*, implying that there is no significant improvement in the clustering method for larger k's.

In the end, we evaluated the potential optimal number of clusters k with the `evalclusters` function of `MATLAB`, following the four internal criterion described in section 2. The tests yielded the following respective results: `[Calinski-Harabasz, Silhouette, Gap, Davies-Bouldin] = [3, 2, 7, 2]`. The Gap criterion, which gives an aberrant number of *k=7* clusters, is not well suited for our simulation for several reasons: one is that it is consuming too much time for such a small dataset. Another is that it simulates a Monte Carlo sampling, and we are not familiar with the underlying concepts. The Silhouette and Davies-Bouldin criteria, based on simple geometric notions, give an optimal number of *k=2* clusters, which is mathematically comfortable but does not really satisfy the visual distribution of the features. Eventually, Calinski-Harabasz seems to yield the most adapter *k=3* number of clusters. Not only does it satisfy our visual intuition, but it is also based on variances analysis, as explained in section 2. Since the *principal components analysis* of the spikes (`spikesPCA`) displays three **uncorrelated** features, the variances analysis is relevant in this case and seems adapted to the *k-means* clustering method, which is also based on variances analysis.

In the end, we concluded that segregating our populations of neuron cells in three clusters was the best compromise between minimising the error and having a plausible number of clusters.

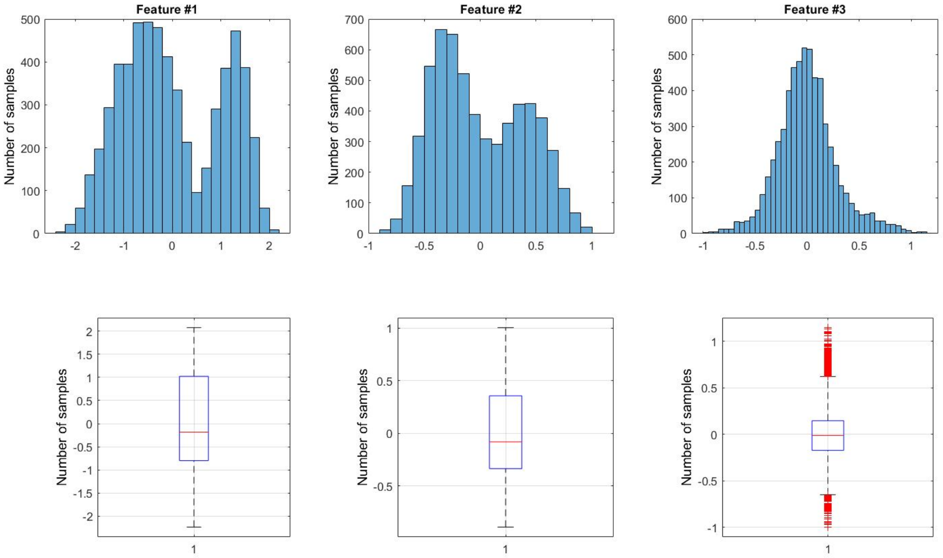

Figure 2: **Top:** Histogram distribution of each feature. **Bottom:** Boxplots distribution of the same 3 features.

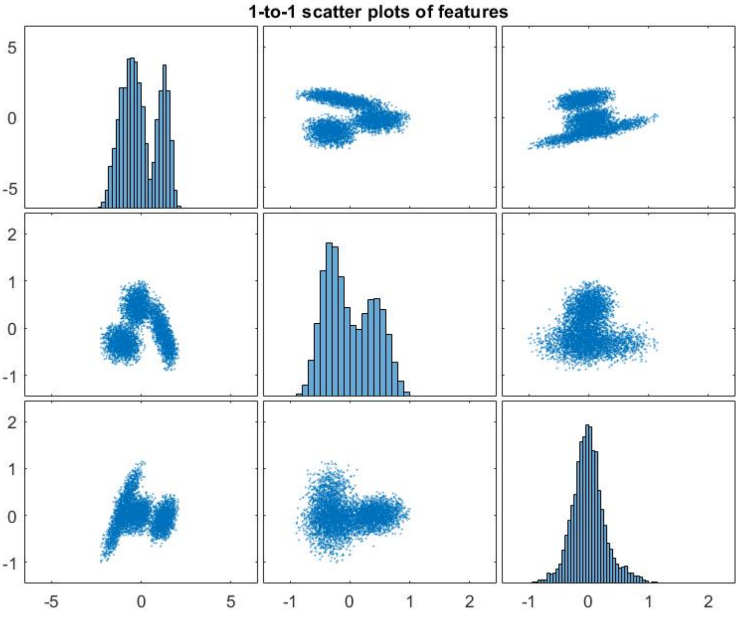

Figure 3: Scatter plots of the 3 features, revealing 3 potential clusters.

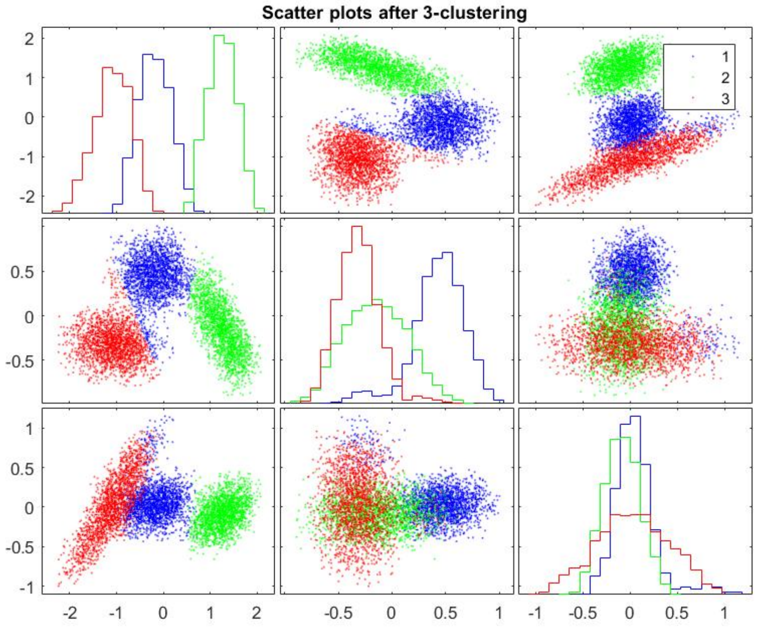

Figure 4: Scatter plots after 3-clustering.

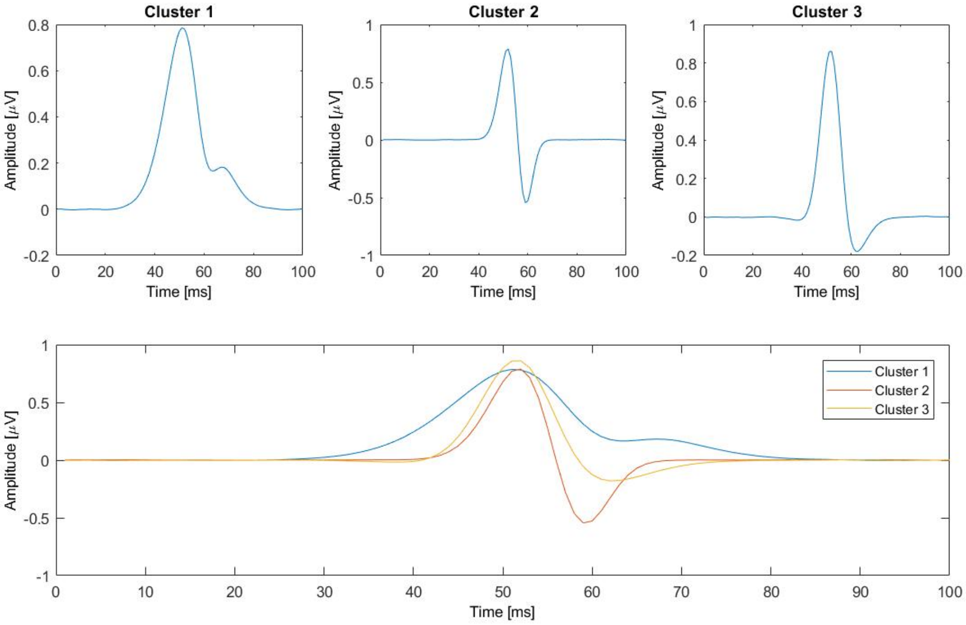

Figure 5: Mean spikes profiles of the 3 clusters.

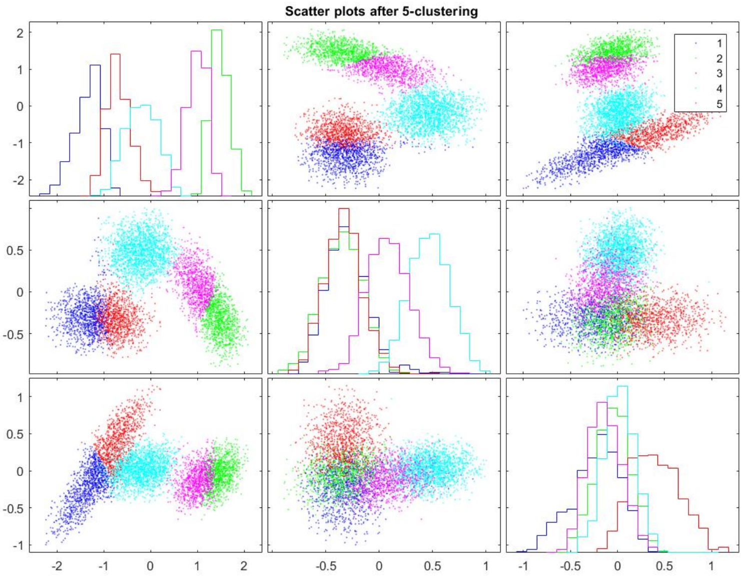

Figure 6: Over-clustering with *k=5* clusters.

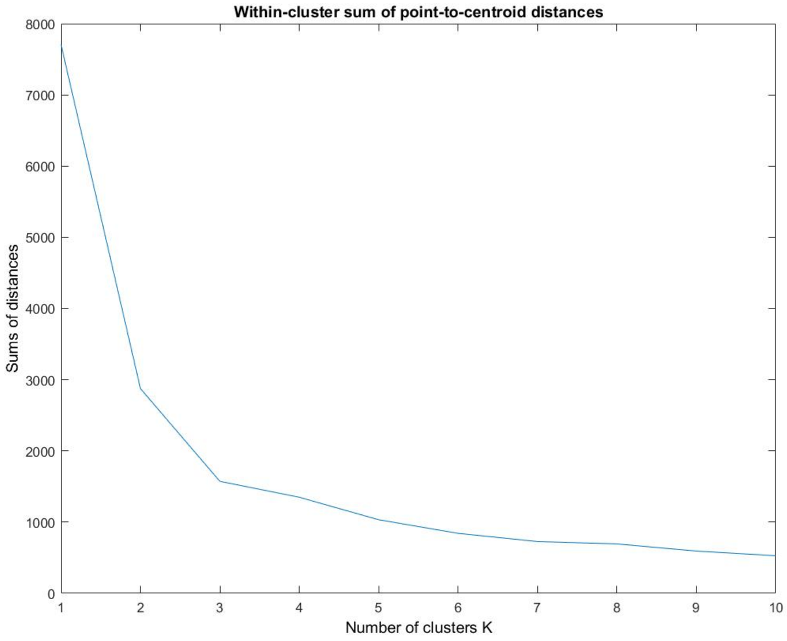

Figure 7: Sum of within-cluster sum of point-to-point distances for 1 to 10 clusters.