Freelance Project at Microbrain Biotech

Automate the characterization of electrical signals extracted through micro-electrode arrays during neural systems-on-a-chip experiments, and provide a health assessment of the neurons involved

September 2017 to December 2017

AVALOS Diana

Contents

List of Figures

1 The start-up MicroBrain Biotech

1.1 Start-up presentation

MicroBrain Biotech is a start-up that has been cofounded in 2014 by three researchers from the French National Center for Scientific Research (CNRS): Bernard BRUGG (neurobiologist expert in neuronal degeneration), Jean-Louis VIOVY (microfluidic expert) and Jean-Michel PEYRIN (neurobiologist experienced in the prion field). The start-up is lead by the CEO Bernadette BUNG, and the CTO Etienne JACOTOT (co-director of the Neuronal Stress and Aging team at INSERM).

In 2016, MicroBrain Biotech, the French National Center for Scientific Research (CNRS), The Curie Institute and Pierre et Marie Curie University (UPMC) signed an exclusive licensing agreement on the "Neural Diode". It is a microfluidic device able to control the architecture and connectivity of neurons, which provides reconstructed neural networks at the micron scale. The Neural Diode has proven its efficiency on preclinical models for Alzheimer and Parkinson so far [1]. The start-up targets the neuropharmacological Research and Development market, and is already working with the pharmaceutical groups Servier and Sanofi.

1.2 Start-up environment

While I was working on my project at MicroBrain Biotech, I was often in contact with the CEO Bernadette BUNG, and the CTO Etienne JACOTOT, for the implementation of this project. I also had the chance to work with biology and pharmacist engineers and trainees, and learn about the fabrication process of microfluidic devices, neural culture and neurons analysis through immuno-staining techniques. I was in charge of supervising a group of 5 interns, that where attending their first year of master's degree at the engineering school ECE. They performed the recordings on the microfluidic devices, and learned to extract parameters from the recorded signals, using python and its libraries. Learning to lead a team was a great experience, help them through their work, set goals, and supervise the advances of the project.

2 Presentation of the project

A neuron is a cell that receives, processes, and transmits information through electrical and chemical signals. These signal transmissions between neurons occur via specialized connections called synapses. Neurons connect to each other to form neural networks (Figure 1). They are major components of the central nervous system (brain and spinal cord), and the peripheral nervous system. Neurons are composed of dendrites that receive input signals, which are processed by the soma and information is transmitted through the axon in the form of action potentials (electrical impulses).

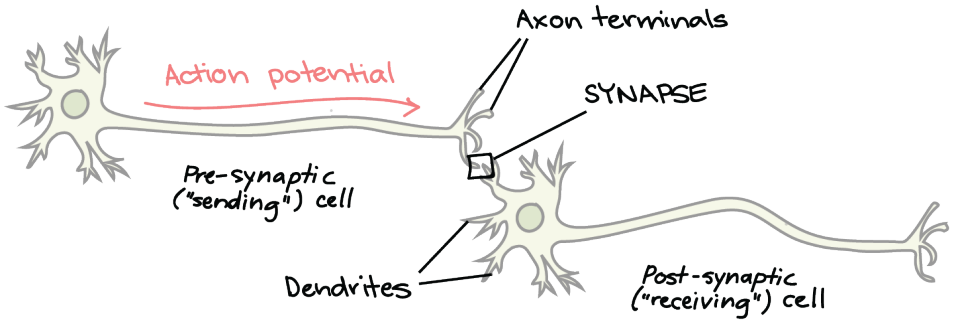


Figure 1: Scheme of the structure of 2 neurons and their synaptic connection

I am contributing to a project which goal is to evaluate the disturbance of neurotransmission with microfluidic-based reconstructed neuronal networks and microelectrode arrays. These disruptions in neuronal communication have been noticed in neurons presenting neurodegenerative diseases such as Alzheimer and Parkinson's [1]. My mission is designing and implementing the algorithm able to measure and learn the intrinsic characteristics of neurons in different states (for instance: healthy or presenting a specific neurodegenerative disease) and assess the efficiency of different medicines in term of recovery of the quality of neurotransmission.

3 State of the art

3.1 Biomedical Research and Nerve-On-Chip technology

The ultimate goal of biomedical research is to develop new and improved diagnosis and therapies. To achieve that, it is necessary to gain greater insight into mechanisms of human disease, at the molecular and cellular level. Recent advances in genetics, proteomics as well as in high-throughput screening have allowed the development of a vast number of compounds for various diseases. Once that their biological activity have been successfully tested in in-vitro assays, they are further tested in-vivo to evaluate their pharmacological and physiological profiles.

3.1.1 In vivo models

Since it is not possible to carry out fundamental research investigation on humans, disease models have been developed in animals such as mice and non-human primates, in order to test different treatments. Although great advances have been made and transgenic animals (that experienced specific gene alterations) successfully reconstitute disease manifestations and phenotypes that are apparently similar to those observed in humans, several studies show that the underlying molecular mechanisms can actually differ greatly between animals and humans with the same disease phenotype [2-4]. These observations prevent us to extrapolate the results of preclinical drug testing in animals to humans. This explains why we notice a significant gap between drug development and drug approval [5], where compounds that show promise in animal testing fail during human clinical studies: only 11.3% of the compounds that enter clinical studies receive approval for marketing [6]. In addition, it is nearly impossible to identify precisely cellular and molecular contributors to a disease in whole-animal models.

3.1.2 In vitro models

To identify the mechanisms at stake at a cellular and molecular level when testing a new drug therapy, investigators work with in-vitro 2D cell culture models. They use cultured human cells: primary cells (isolated directly from human or animal tissue), established cell lines (cells that have been continually passaged over a long period of time) [7] and, more recently, derivatives of induced pluripotent stem cells (iPSCs) [8]. iPSCs present the advantage of proliferating indefinitely, as well as giving rise to every other cell type in the body (such as neurons, heart, pancreatic, and liver cells). Even though cells cultured on standard culture dishes exhibit differentiated cell functions, they commonly fail to mimic tissue- and organ-level structures and functions that are central to disease etiology and usually only provide a broad indication of the compound efficacy and toxicity. Thus, there is a great need to develop alternative experimental systems containing living human cells that recapitulate accurately cell functions and physiology in vitro.

3.1.3 Organs-on-chip

To overcome these limitations, organs-on chips have been developed: they consist of microfluidic devices able to mimic the architecture and physiology more accurately than conventional methods [9]. They are able to provide controlled drug release systems, static and dynamic mechanical stresses, spatiotemporal distributions of biochemical cues,... Spatiotemporal control of cell growth and proliferation is monitored, as well as recreation of specialized tissue-tissue interfaces, physicochemical microenvironments, vascular perfusion characteristics of specific living organs and analysis at the level of intercellular communications [10-12]. All these properties enable to provide a realistic determination of drug's pharmacokinetics, pharmacodynamics and toxicity profiles [13]. Researchers aim to engineer models of diseases of the heart, lung, intestine, liver, kidney, cartilage, skin and vascular, endocrine, musculoskeletal, and nervous systems, as well as models of infectious diseases and cancer [14].

3.1.4 Nerve-on-chip

Neural systems-on-a-chip, also called microfluidic-based reconstructed neural network, are organs-on-chip specialized in the growth of connections between populations of neurons. They enable in-vitro study of structure-function relationship of the brain circuitry, in a minimalistic environment free of the in-vivo increased complexity networks. This technology allows building models to study particular aspects of neurological diseases, disorders, and injuries (NDDIs) [15], implement high-throughput assays for drug discovery [16], and create biological neuronal computers on-chip [17]. Furthermore, studies have shown that in-vitro neural networks share the same network characteristics as their in-vivo counterparts [18-19]. In a nerve-on-chip device, different kinds of neurons are grown in distinct chambers linked by channels in which neurons can connect via their axons. So far, a maximum of 3 different populations have been connected [20], but this technology is still at its beginnings and the complexity is increasing towards the reconstruction of complete neuronal pathways. The structure of the nerve on chip is represented in Figure 2 .

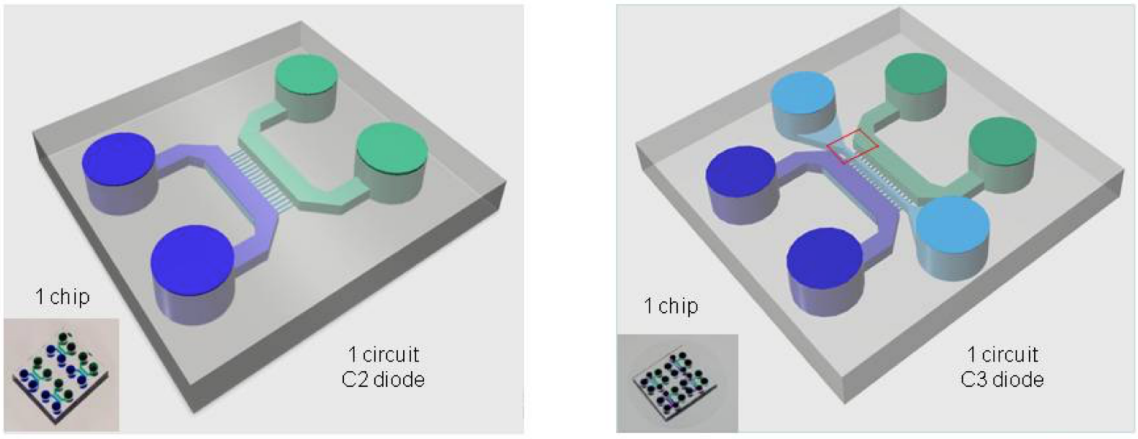


Figure 2: Structure of the neuronal diode for 2 (left) or 3 (right) different kinds of neurons

3.2 The neuronal diode

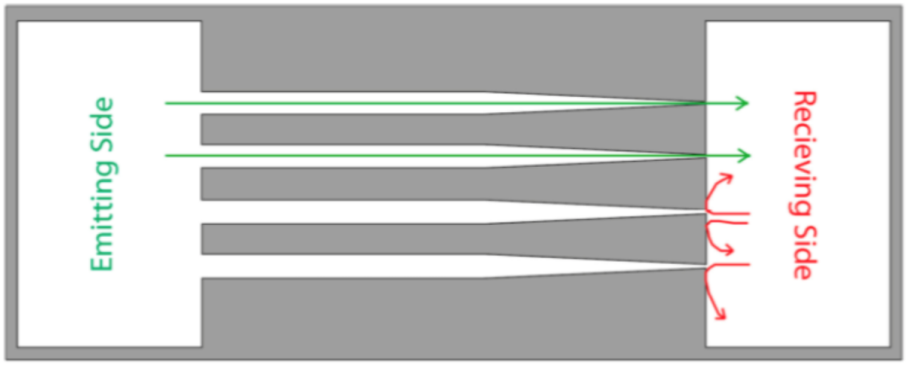


Figure 3: Unilateral axon path between 2 microfluidic chamber

The neural diode developed by MicroBrain Biotech is a type of nerve-on chip. Their innovation relies on the unilateral connections between neuron chambers obtained thanks to the funnel shape of the microfluidic channels connecting two separated neuron chambers (Figures 3, 4) [21]. This technology allows a better control of the architecture of the reconstructed neural network. This neural diode, coupled with micro electrodes arrays and an adequate algorithm could constitute a powerful tool to screen and evaluate the synaptico-toxicological effect as well as the synapto-protective potentials of several drugs. MicroBrain Biotech develop the neural diode for 2 or 3 different kind of neurons .

Figure 5 shows the images from the neural diode in black and white and with fluorescent imaging to highlight the structure and grosth of cortical and hippocampal mice neurons. Recordings for the algorithm were performed on this type of neurons in these neural diodes.

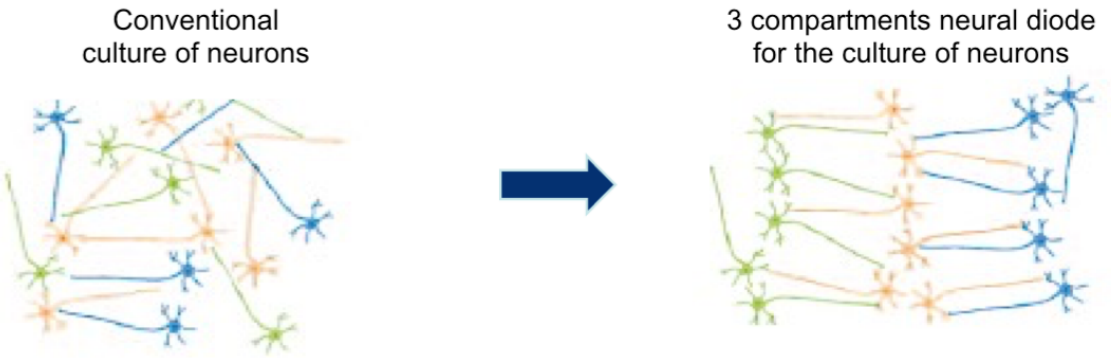


Figure 4: Comparison of conventional neuronal cell culture and neuronal cell culture in the neural diode

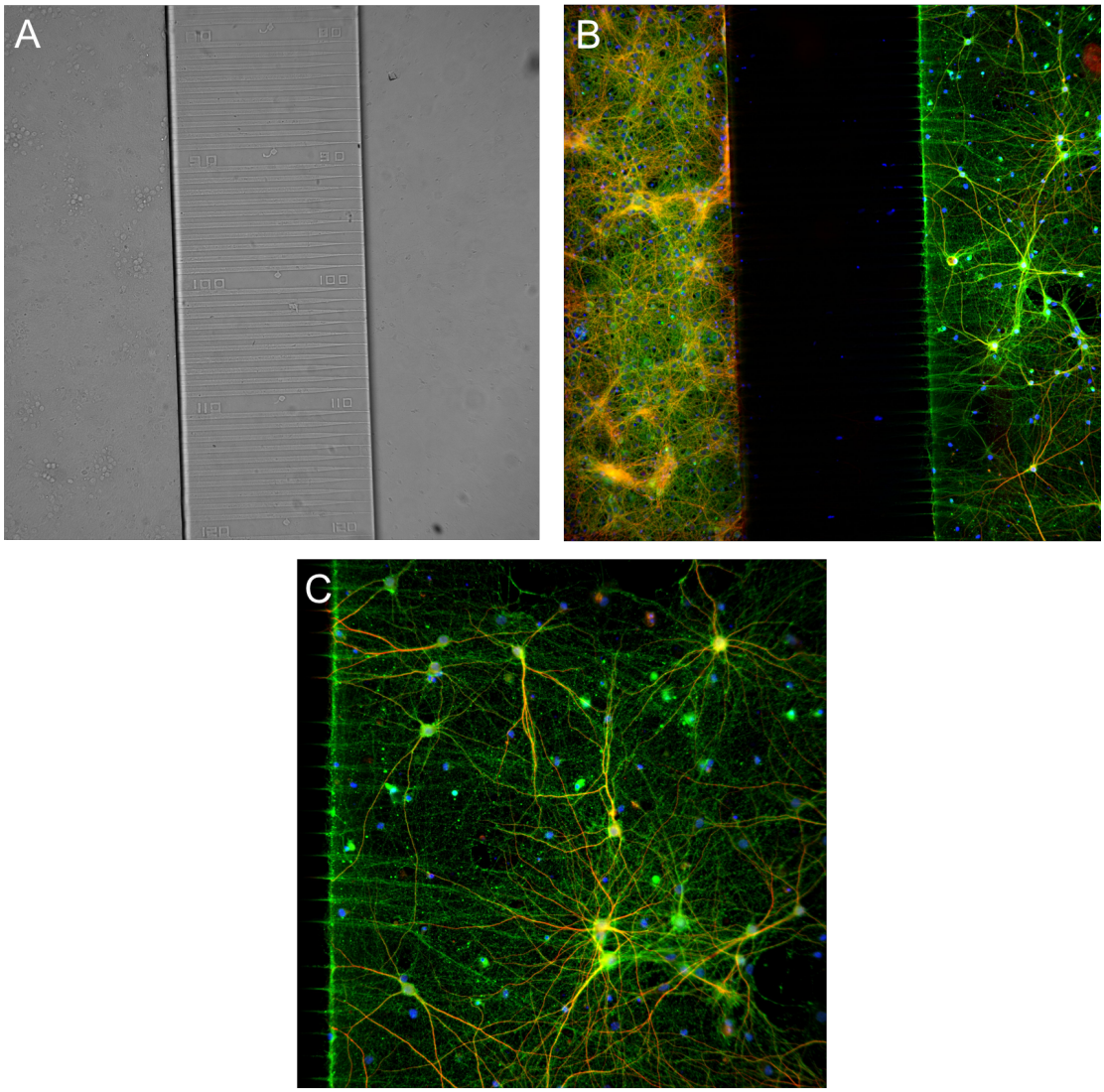


Figure 5: Pictures of neurons in the neural diode. A: black and white picture of the neural diode. Funnel shape micro-channels are clearly visible. (A x10) B and C: Fluorescent images of the mice neurons in the neural diode (B is x10 and C is x20). Hippocampal neurons are on the left side and Cortical neurons are on the right side. Blue color stains the neuron soma, green stains dendrites and axons and red marks dendrites. Pictures provided by Lorraine Pinot

3.3 Alteration of the trans-synaptic connections in Alzheimer's disease

3.3.1 Alzheimer's principal characteristics and neuropathological changes

According to the World Health Organization [28] there are worldwide 33 million people with Alzheimer's Disease (AD). It is the most common cause of dementia and it may contribute to 70% of its cases.

This disease affects mostly elderly people. Its symptoms include forgetfulness, losing track of the time, becoming lost in familiar places, and increasing difficulty in communication (troubles with words, in speaking and writing). Behavioral and personality changes as also experienced, together with troubles in judgement-making, concentration, realization of familiar tasks. As the dementia progresses, symptoms can go as far as having difficulties recognizing relatives [29].

On the cellular level, AD is a neurodegenerative disease characterized by extra-cellular deposits of β-amyloid peptides (Aβ) and intracellular filamentous aggregates of hyper phosphorylated Tau proteins. Histopathological studies show that these hallmarks spread along disease evolution [30]. Aβ was first sequenced and recognized as a potential marker of Alzheimer's disease in 1984 [31], and identified as the major component of the extracellular plaques that define this major form of dementia in 1985 [32]. Since then it has been extensively studied because regulating its production might prevent AD.

Aβ is the produce of a cleavage of the transmembrane protein APP (amyloid precursor protein). APP can undergo cleavage down one by the enzyme α-secretase that produces sAPPα fragment or it can be cleaved down by β- and γ-secretases predominates, and Aβ is formed (with 40 or 42 amino acids :Aβ40 and Aβ42). In AD, This peptide aggregates over time to produce senile plaques, if it is not degraded by enzymes. The balance of production versus degradation and clearance determines levels of Aβ aggregates present. [33-34]

3.3.2 Memantine treatment for AD

Nowadays, the only 2 medication classes approved specifically for the treatment of AD are the cholinesterase inhibitors (donepezil, rivastigmine, and galantamine) and memantine. Memantine is a glutamatergic N-methyl-d-aspartate (NMDA) receptor antagonist, which means it acts on the glutamatergic neurons and blocks its NMDA receptors (an NMDA receptor is a glutamate and ion channel protein receptor that is activated when glycine and glutamate bind to it, and it becomes permeable to Ca^{2+}). Memantine is widely used for AD treatment, as it presents statistically significant improvement in several AD-oriented outcome measures, whether alone or combined with other treatments as cholinesterase inhibitors [38-40] A recent study [41] showed that chronic treatment of AD with memantine reduces the amount of Aβ oligomers in animal models of AD, through inhibition of the formation of different types of Aβ aggregates and disaggregation of preformed Aβ fibrils.

3.3.3 Aβ participation in the dying-back pattern of degeneration in AD

There is ample evidence [35-36] that neurons affected in AD follow a dying-back pattern of degeneration, where abnormalities in synaptic and axonal functions long precede somatic cell death. More

precisely, pre-synaptic loss precedes axonal fragmentation, post-synaptic spine alterations and somatic cell death [37]. This observation raised the question whether Aβ and Tau protein local abnormalities could trigger a degenerative process through trans-synaptic alterations. Even if the underlying processes causing a distant breakdown are still uncertain, a recent study [1] have showed that axons are partly resistant to axonal Aβ exposure, while somatic Aβ treatments trigger an anterograde degeneration signal in axons, inducing a dying-back pattern. Through the reconstruction of oriented cortico-hippocampal network with a microfluidic device containing funnel-shaped micro-channels, the research team demonstrated that somato-dendritic deposits of Aβ on cortical neurons trigger a rapid cortical presynaptic disconnection together with a glutamate-dependent postsynaptic hippocampal tau-phosphorylation before any axonal and somatic cortical degeneration.

3.3.4 Characterization of the trans-synaptic alterations in the dying-back process of neurons affected by Aβ

Since evidence proved the existence of a dying back pattern of degeneration in Alzheimer disease, the algorithm that I am developing could help characterize the trans-synaptic alterations related to this phenomenon. The algorithm is constructed on neurophysiological data collected on cortico-hippocampal networks built in funnel-shaped micro-channels microfluidic-devices. Neural signals are collected for spontaneous activity, then Aβ aggregates are deposited in the chamber containing cortex neurons, and after Memantine is applied into the hippocampal compartment. These 3 types of neural signals(healthy, +Aβ, +Aβ+Memantine) are analyzed and compared in order to extract the main characteristics of trans-synaptic alterations. The microfluidic neural network coupled micro-electrode arrays to record the neural signals, together with an algorithm able to analyse these neurophysiological signals, offers an efficient way to screen and evaluate the synaptico-toxicological and synaptico-protective potential of several drugs.

3.4 Neural Signal Processing

Several parameters can be extracted and analyzed from our electrophysiological recordings. To know which are the important parameters to take into account, several articles handling neural data were analyzed, as well as the python librairies available for representation and analysis of electrophysiological data and professional softwares for analysis of neurophysiological data[22].

There are many libraries available for representation and analysis of electrophysiological data. Among the popular ones there are Elephant (Electrophysiology Analysis Toolkit) [23], Spykeviewer [24-25] and OpenElectrophy [26]. All these librairies are open-source and implemented in the Python programming language. They are built upon the Neo framework [27].

3.4.1 Effects of pharmacological agents on neurophysiological data

The algorithm being developed in this project aims at characterizing the effects of pharmacological agents on the synaptic transmission between two different kinds of neurons. Therefore, it is essential to establish the state-of-the-art in this domain. In the first experiment to assess the effects or pharmacological agents, effects of several pharmacological agents were described in a qualitative fashion: acceleration of existing burst patterns, decreased burst frequency, electrophysiological

activity stopped,... [42] . Then, with implementation of new techniques, chemically induced toxic effects were measured with more precision, and measures of extracellular electrophysiology led to specific estimations of the firing activity, number of bursts and burst duration, the number of spikes in a burst, the percentage of spikes in a burst, the inter-spike and inter-burst intervals,... [43-48]

3.4.2 Neural stimulation

In-vivo recordings of neuronal activity are characterized by a high degree of irregularity. In-vitro, the complexity of the system is decreased, as the neurons studied are isolated from the huge brain network, but spontaneous neural activity is still irregular. However, a constant stimulus is able to generate spike trains that look regular upon common parameters. Indeed, it has been observed that isolated neurons react in a very reliable and reproducible manner to a fluctuating input current, and so can neurons in sensory cortex in vivo when driven by a strong time-dependent signal [49]. Recording neural activity generated upon a given stimulus might help compare differences in neural activity on different experimental conditions (by adding specific drugs into an in-vitro neural network).

Several research studies have tested simulation and micro-electrode arrays (MEA) measurements of in-vitro neural networks. For instance, Kapucu et al. [43] applied stimulation on single channel MEA with 3 different kinds of waveforms. The ISIs were set to at least 30 ms for the spikes in bursts and 300 ms for the individual spikes. Simulation was applied during 60 seconds, with 10 bursts in total with mixed waveforms, and 4 spikes between bursts. Background noise was also added to the stimulation signal.

3.4.3 Summary of the parameters to analyse and extract from the electrophysiological data

Plots:
* Signal plots
* Interspike interval histograms
* Rate histograms
* Peristimulus time histograms
* Peristimulus time rasters
* Joint peristimulus histograms
* Autocorrelograms
* Crosscorrelogramss
* Rasterplots
* Burst analysis
* Spectral densities and spectrograms

Parameters:
* Cumulative activity
* Instantaneous frequency
* Interspike intervals versus time
* Poincare maps of interspike intervals
* Coherence analysis
* Regularity analysis

* Reverse correlation analysis
* Spike and burst dynamics (intra and extra burst spike rate, burst duration, mean burst interval duration, spikes per bursts, probability of spike detection)
* Information theory: Entropy and Mutual information
* Coefficient of the discrete Fourier Transform (linked with the Power Spectral Density)
* Variance, Mean, Extremums of the signal, Analysis of the derivative of the signal,...

3.4.4 The Neo Framework

Neo provides a standardize way for organizing and saving data from electrophysiological recordings. These recordings are converted into specific object models with a determined structure. The Neo initiative of a consistent and homogeneous framework could help neuroscience make faster progresses, as data and functions to analyze, visualize and generate data are exploitable by all. Neo enables simplified exchange of tools developed for neural data and facilitates efficient assessments of the quality and reproducibility of studies.

Neo data objects build on the quantities package, which in turn builds on NumPy by adding support for physical dimensions. Thus Neo objects behave just like normal NumPy arrays, but with additional metadata, checks for dimensional consistency and automatic unit conversion.[27] Neo is deliberately limited to represention of data, with no functions for data analysis or visualization. The latter are performed by the librairies using the Neo package. Neo is adapted to a wide variety of data formats, and can be used for intracellular and extracellular electrophysiology and EEG data with support for multi-electrodes (for example tetrodes).

3.4.5 Libraries for electrophysiological data based on the Neo framework

Several librairies using Neo package are available (Elephant, Spykeviewer and OpenElectrophy to start with). Giving a look at all the specificities and characteristics of each would enable to assess the most relevants to study.

Elephant (Electrophysiology Analysis Toolkit) focuses on generic analysis functions for spike train data and time series recordings from electrodes, such as the local field potentials (LFP) or intracellular voltages. Elephant is the direct successor to Neurotools and maintains ties to complementary projects such as OpenElectrophy and Spykeviewer. Spykeviewer functions enable the implementation of common neuroscientific plots (e.g. rasterplots, peristimulus time histograms, spectograms, auto and cross-correlograms, signal plots, interspike interval histograms, spike density estimation). Custom plugins for other analyses or plots can be easily created and modified. OpenElectophy has also modules to simplify data and analysis sharing, functions to analyse neural signals, a GUI for exploring and viewing datasets, plus an offline spike sorting tool.

3.4.6 Investigation of these libraries for analysis of electrophysiological data

I am supervising a group of 5 ECE engineering students. Their project is the development of tools for the analysis of the recorded neural signals. I asked them to implement functions to extract properties of the signal, and suggested them to investigate the available python librairies. Their goal is to investigate more in depth the possibilities offered by these libraries, implement

the fuctions available and extract information on our recordings. This project is challenging as they are not very familiar with coding, and even less with Python. I provided them with the recordings in the HD5F format. They converted the data into Neo objects in order to use python librairies mentioned above. I am monitoring the advances and steps of their project and helping them through the difficulties encountered.

3.5 Study on Burst analysis

3.5.1 Introduction

Neurons communicate through the emission of action potentials that can be emitted within brief bursts of high frequency discharge. These spike bursts constitute an important element in synaptic plasticity and information processing in the central nervous system. Neurons generate bursts to increase the reliability of the communication and avoid synaptic transmission failure [50]. Furthermore, they might provide more precise information than single action potentials and convey selective communication between neurons based on resonance frequencies [51].

The bursting behavior of neurons have been considered by several researchers as the most important property for analysing the dynamics of electrical activity during the development of neuronal networks [52] and for the investigation of the modification of neuronal networks which are induced by alteration of the physiological environment [53,54] (such as chemical exposure and electrical stimuli).

Efficient burst detection is a long-standing challenge. Indeed, bursts are difficult to identify because bursting activities or patterns vary with physiological conditions or external stimuli. Many researchers have investigated this topic and have published articles presenting new ways to detect bursts in spike train. However, many of these methods depend on the parameters assumed for burst detection [54-58]. Among the articles that I investigated, the algorithm developed in 2008 by Chen et al. [60] and the algorithm developed in 2002 by Tam [61] seemed to me the most relevant ones to implement. Indeed, we seek an algorithm that is auto-adaptive [60-62] to the intrinsic properties of the spike train and able to extract different bursting patterns.

3.5.2 Methods

Bursts detection and analysis is based on parameters such as the study of the inter-spike intervals ISI (time between 2 spikes), within bursts and outside bursts, the intra-burst periods (sum of the ISI within a burst), and inter-burst periods (sum of the ISI between bursts).

The algorithm developed by Chen et al. in 2008 [60] is auto-adaptive to the spike trains analysed, and its development is based on the 'mean interspike interval' (MISI). The logic of the algorithm is easy to understand and implement.

Another auto-adaptive method developed by Tam in 2002 [61] seems really interesting too, and relies on a local analysis of the ISI. The logic of the algorithm is simple, however its imple-

mentation seems more complicated and relevant for online analysis or signals that might vary in long-time scaled recordings. The auto-adaptive methoddeveloped by Kapucu [62] is also worth analyzing but it is much more complicated to implement, with a lot of data processing. Therefore, the selected method is the one developed by Chen et al. [60]

3.5.3 Implementation of the bursts detection algorithm

1. Define the Mean Inter-Spike Interval (MISI), with ISI^n the n_{th} interspike interval

$$MISI = \sum_{n=1}^{N-1} \frac{ISI_n}{N-1} \tag{1}$$

2. Bursts are defined as high frequency spike episodes. The ISI within a burst is inferior to the MISI. All the ISI smaller than the mean MISI are collected into the list L:
For all the ISI:

$$\begin{aligned} If \;\; ISI_n < MISI: \\ L = L + SI_n \end{aligned}$$

3. List L mean is given by ML:

$$ML = \sum_{n=1}^{N-1} \frac{L_n}{N-1} \tag{2}$$

4. A burst is defined as two or more successive ISIs of the original sequence ISI! with a mean value smaller than ML. When this condition is verified, the ISIs are collected in the burst sequence B.

$$\begin{aligned} If \quad & (ISI_n, ISI_{n-1}) < ML: \\ & If \quad ISI_n \in ML: \\ & \qquad B = B + ISI_{n+1} \\ & Else \\ & \qquad B = B + ISI_n + ISI_{n-1} \end{aligned} \tag{3}$$

3.6 Feature Selection

3.6.1 Introduction

A feature is an individual measurable property of the process being observed. The bigger the number of features, the more complicated it is to extract significant parameters to perform an efficient analysis of the data. The need for efficient feature selection algorithms has emerged for several reasons. First, it aims at reducing the size of the problem, and consequently reducing the compute time and space to run our algorithms. Furthermore, when a classification algorithm follows feature selection, the accuracy of the classifier is improved by removing irrelevant features

that would increase the likelihood of overfitting to noisy data. At last, a better understanding of the data and its sorting into categories is obtained by focusing on the meaningful parameters.

Feature selection aims at extracting meaningful features that enable an efficient sorting of the observed processes into several categories. This sorting into classes is solved by the implementation of a machine-learning algorithm for classification, which is explained in the next chapter. An efficient feature selection method should be able to select the features that contain the maximum discrimination about the categories, while avoiding redundant information brought by highly correlated features. The advantage of feature selection is that it doesn't alter the original representation of the features, but merely select a subset of them, offering the advantage of interpretability. An alternative to feature selection is feature extraction. This process consists on creating new features from the original ones, with the extraction of instructive and non-redundant information.

Finding the optimal set of features is a real challenge. This chapter provides a summary of the analysis of several feature selection methods, as well as the feature extraction methods.

3.6.2 Methods for Feature Selection

Directly evaluating all the subsets of features (2^N) in order to find the best combination of features becomes an NP-hard problem as the number N of features grows. Several approaches have been developed for this purpose. Here, the focus is made on supervised learning (classification) where the class labels are known beforehand. Feature selection methods are classified in 3 categories: Filter methods, Wrapper methods and at last Embedded methods that combine the two previous approaches.

Filter methods rank the features based on how 'useful' a feature is likely to be for classification, which means the efficiency to discriminate between the different classes. Then, an arbitrary amount of the best features is picked for the algorithm. Filter techniques look at the intrinsic properties of the data. Among the most common used filter methods we find the Pearson's Correlation, the Mutual Information and the Relief algorithms. All these methods provide a ranking of the best features according to their criteria. In the Ensemble method, all these rankings can be combined together to create a score, and generate an optimal raking of the features with more robust results. Once that the ranking has been established, low-scoring features are discarded.

Filter methods are fast, scalable and present a better computational complexity than wrapper methods. Nonetheless, the weakness of these methods relies in the fact that they do not take into account that different features can be highly discriminant but redundant, and that some features are not likely to be chosen alone, but combined together they are optimal for class separation. Moreover, they ignore the interactions with the classifier (search in the feature subset space rather than in the hypothesis space). Features work together, and more complex methods might be required.

Wrapper methods are so called because they wrap up the classifier in the feature selection algorithm. Features are not assumed to be independent and so advantages may be gained from looking at their combined effect. These methods measure the efficacy of a set of features extracted

from the feature space, looking at the classification of the data. Looking for every possible combination (like Exhaustive Search methods) requires a lot of resources, so some heuristic search methods have been developed to optimize the selected set of features.

First, there are the deterministic methods. Among them, the Sequential Forward Selection starts with the feature with the best performance. Then, this feature is tested with all other features as a pair and the best performing pair is chosen and so on, incrementally adding other features. The Sequential Backward Elimination on the opposite starts with all the features, and incrementally removes the features. If excluding a feature increases the classification result, the feature is removed.

These are also randomized methods such as evolutionary algorithms (Genetic Algorithm or Particle Swarm Optimization). These algorithms study different subsets of features in order to optimize the classification result. They are less prone to local optima, but they present a higher risk of overfitting the data than deterministic methods.

Common drawbacks of wrapper techniques are that they are computationally intensive and they present a high risk of overfitting than filter methods. For each new set of features, the model is trained on a subset and tested on another one to test the accuracy of the classification. Then, cross-validation is performed, to avoid high accuracy but poor generalization power of the classifier.

Embedded methods combine filter and wrapper methods. Features are ranked and then wrapper methods are applied to the best-ranked features (for instance: Ranked Forward Search). Since a first selection of the features is made, embedded methods are less computationally intensive than wrapper methods alone, and are also more efficient than filter methods alone because correlations and redundancy of features is taken into account.

More detailed explanations of feature selection algorithms are provided in the articles of Chandrashekar et al. (2014) [64] and Saeys et al. (2007)[65], as well as these Wikipedia pages [78].

3.6.3 Dimensionality reduction

An alternative to feature selection is dimensionality reduction. This process involves projecting the high dimensional sample onto a space of lower dimensionality that carries sufficient information. The resulting features are a combination of the former features. Dimensionality reduction of the input space might result in a loss of information for the discrimination of the classes. The challenge relies on preserving as much relevant information as possible.

The study of dimensionality reduction method is restricted here to techniques that don't need to preserve the topological properties of the input space. Among these techniques there are the Principal component analysis (PCA), Kernel PCA, Linear or Quadratic discriminant analysis (LDA, QDA) and Generalized discriminant analysis (GDA), just to name a few.

Principal component analysis (PCA) and Kernel PCA are unsupervised approaches, which mean that the classes in which the observations belong are not taken into account. In the contrary, Fisher's linear discriminant and the methods resulting from this technique (LDA, QDA, FDA,

RDA, GDA) take into account those classes, and constitute supervised approaches. They perform the classification at the same time than the dimensionality reduction.

Through eigenvalue decomposition of the covariance matrix of the data, PCA creates a new set of features resulting of linear combinations of the original features. This orthogonal transformation produces new features (namely the principal components) sorted by the amount of variance captured, which are uncorrelated. The first principal components are the ones with the highest variance, and therefore they present more chances to separate efficiently the data.

Vapnik-Chervonenkis theory maintains that mappings that take us into a higher dimensional space than the dimension of the input space provide us with greater classification power. The Kernel trick enables to benefit from high dimension whilst avoiding high computation. Therefore, Kernel PCA is PCA applied to a high dimensional feature space using the "kernel trick". A Kernel function ϕ projects the data points in some nonlinear feature space of higher dimension. Points can be separated in a non-linear way and different kernel function can be used as the Gaussian, Hyperbolic and Tangent ones. Kernel PCA can give a good re-encoding of the data when it lies along a non-linear manifold. However, the kernel matrix is NxN, so kernel PCA will have difficulties if we have lots of data points.

These unsupervised dimensionality reduction methods are recommended if the classifier algorithm chose afterwards assumes uncorrelated features, and if the next step of the algorithm is an unsupervised classification. It is important to consider that these techniques don't take into account the classes of the observations, which can jeopardize the efficiency of the classification subsequently.

3.6.4 Dimensionality reduction with classification

There are also dimensionality reduction techniques that are supervised, and take into account the classes of the data. Consequently, classification happens at the same time than dimensionality reduction.

Fisher's linear discriminant defines the separation between two distributions to be the ratio of the variance between the classes to the variance within the classes. It projects high-dimensional data onto a line and performs classification in this one-dimensional space.
Linear discriminant analysis (LDA) and Quadratic Discriminant Analysis (QDA) perform a projection of the data on N-1 dimensions (higher accuracy than Fisher's discriminant), that maximizes the separation between several classes.

Both LDA and QDA are based on the class conditional distribution P(X—y) of the data X for each class y. P(X—y) is modeled as a multivariate Gaussian distribution with density:

$$P(X|y=k) = \frac{1}{(2\pi)^n |\sum_k|^{1/2}} \exp(\frac{-1}{2}(X-\mu_k)^t |\sum_k|^{-1}(X-\mu_k)) \quad (4)$$

From the training data, we extract for each class k the class mean μ_k and the covariance matrix $\sum_k$. LDA and QDA make the assumption that the data is normally distributed, so the data has to be standardized beforehand. LDA assumes that all covariance matrices $\sum_k$ are equal, which means that LDA can only learn linear boundaries. In the case of QDA, there are no such assumptions, leading to quadratic decision surfaces.

3.7 Supervised classification and regression algorithms

There are two main classes of algorithms: Classification and Regression algorithms.

Classification is the process of identifying to which class a new data observation belongs. We differentiate the supervised procedure, in which the algorithm knows the classes of the data, and the unsupervised procedure, in which the algorithm doesn't know the number and parameters of the classes, and involve grouping the data based on inherent similarities of the features (step also known as clustering).

We seek a supervised algorithm able to differentiate several classes (studied case: spontaneous activity of neurons, neurons+ amyloid beta, neurons+ amyloid beta +memantine), that classifies an unseen data sample into one of the known classes.

However, we also seek an algorithm able to deliver a result in terms of likelihood of belonging to each class, and not explicitly the class the data sample is more likely to belong too. For instance, if 0 is the status of a healthy neuron and 1 is a neuron with Alzheimer disease, we would like to estimate the synaptic transmission recovery of a neuron with AD that received a given treatment (for example: memantine) on the 0 to 1 scale. In order to obtain this kind of result, we could look at regression algorithms instead of classification algorithms, or we could modify algorithms designed for classification in order to output a likelihood of belonging to each class (ideas explained further in the algorithms description).

These prerequisites narrow our research to:
-Naive Bayes
-LDA
-Simple Linear Regression
-Multivariate Linear Regression
-Logistic Regression
-Perceptron
-QDA
-Support Vector Machines
-Decision Trees:CART, RForest
-K-Nearest-Neighbors
-GaussianProcessClassifier

The following chapters describes the logic process behind these different algorithms, so we can get a better insight of how they work, their levels of complexity, strengths and weaknesses, and the different assumptions made on the data.

3.7.1 Naive Bayes Algorithm

Bayes' Theorem provides a way to estimate the probability of a piece of data belonging to a given class, given the prior knowledge of the dataset. Naive Bayes algorithm is based on Bayes' theorem, it is straightforward to understand and implement, and present the advantage of being highly scalable. Despite its oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations.

Bayes'Theorem provides a way that we can calculate the probability of a hypothesis h given our prior knowledge d, which is called the posterior probability:

$$P(h|d) = \frac{P(d|h) * P(h)}{P(d)} \tag{5}$$

After calculating the posterior probability for a number of different hypotheses, you can select the hypothesis with the highest probability: the maximum a posteriori (MAP) hypothesis.

$$MAP(h) = max(P(h|d)) \tag{6}$$

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector $\{x_1, ..., x_n\}$, Bayestheorem states the following relationship:

$$P(y|x_1, ..., .x_n) = \frac{P(x_1, ..., x_n|y) * P(y)}{P(x_1, ..., x_n)} \tag{7}$$

Using the naive independence assumption that:

$$P(x_i|y, x_1, ..x_{i-1}, x_{i+1}, ..x_n) = P(x_i|y) \tag{8}$$

This relationship is simplified to:

$$P(y|x_1, ..., .x_n) = \frac{P(y) * \prod_{i=1}^{N} P(x_i|y)}{P(x_1, ..., x_n)} \tag{9}$$

The Naive Bayes algorithm computes the distribution of each class (based on the mean μ and standard deviation σ) in the feature space of N dimensions, based on known observations. For an unseen observation X, it computes conditional probabilities of class values given the value of the features. Multiplying conditional probabilities of each feature together gives the probabilities $P(X|y)$ of a data instance X belonging to each class y. The class reaching the highest probability is attributed to the data instance. For our model, we will just keep the class probabilities.

Assuming a Gaussian distribution of our dataset, the likelihood of having x_i the feature i of the observation X belonging to the class $y = k$(mean μ_{ki}, std σ_{ki}) is:

$$P(x_i|y = k) = \frac{1}{\sqrt{2\pi\sigma_{ki}}} exp(-\frac{(X - \mu_{ki}^2)}{2\sigma_{ki}^2}) \tag{10}$$

This Naive Bayes model assumes that all the features are independents and that the data is normally distributed. These are strong assumptions but the model still present robust results with real complex data.

3.7.2 K-Nearest Neighbors

The K-Nearest Neighbors algorithm stores the entire dataset. When a prediction is required, the k-most similar records to the new sample are found. The classes of the neighbors are extracted, and the most frequent class of these neighbors is assigned to the sample.

An adaptation of this algorithm would be computing the probability of belonging to each class for a new sample, based on the classes representation across the neighbors and its similarity degree to its neighbors. The similarity between the records can be computed in many different ways. The most common would be computing the euclidian distance. The euclidian distance of a sample X $(x_1, ..x_n)$ and a sample Y$(y_1, ..y_n)$ of n dimensions (namely the features) is calculated as the square root of the sum of the squared differences between the two vectors X and Y.

$$Euclidian \quad distance(X, Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{11}$$

Neighbors of a data sample are the closest data instances. First, the distance between each record in the dataset to the new piece of data needs to be calculated. Records are scored and the k closest are selected for the class prediction of the new data sample. (get_neighbors fct)

3.7.3 Classification And Regression Trees

Both classification and regression trees could be used for our algorithm. The classification tree is a good method to assess the algorithm classification of different data samples into several classes. The Regression tree could rather be used to assess the recovery of the neural signal that have AD (amyloid beta) with memantine.

KEEP ON WRITING

3.7.4 Parameters to take into account in the algorithm implementation

Several parameters have to be considered when designing and implementing an algorithm.

First, we have to choose a good trade-off between the **bias and the variance**. The bias reflects the difference between the expected classification and the real one. A low bias means that the learning algorithm fits the data well. But if the algorithm is too flexible in order to present a good bias, it can lead to a high variance: it will fit each training data set differently, which means the algorithm over-fits the training sets and is not generalizable.

Another parameter to take into account is the **complexity of the algorithm regarding the amount of training data**. For a great accuracy, it is desirable to design a complex algorithm with low bias and high variance. However, if it reaches high levels of complexity, it will only be learnable from a very large amount of training data. If this is not possible, concessions have to be made on the algorithm complexity.

Furthermore, there is the problem of the **dimensionality of the input space**. The learning problem has a difficulty proportional to the number of dimensions of the input vectors. Dimensionality reduction through feature selection or feature extraction can simplify the problem and lead to better classification results.

Finally, algorithms can make different assumptions on the data, which have to be considered to reach a good performance. For instance, many algorithms assume a Gaussian distribution of the data (like the ones using distance functions or neural networks with weights).

3.7.5 Scaling the features

To have an efficient learning algorithm, it is preferable to have input features scaled to similar ranges. Most of the machine-learning algorithms require scaled data, like methods using weight inputs or distances. Two popular methods are normalization and standardization.

Normalization aims at rescaling the data so it belongs to the range between 0 and 1. Features are normalized independently. The minimum and maximum values for each features are computed and then data is normalized:

$$normalized \quad value = \frac{value - min}{max - min} \tag{12}$$

Standardization is a rescaling technique that refers to centering the distribution of the data on 0 and the standard deviation to 1. Together, the mean and the standard deviation can be used to summarize a normal distribution. Therefore, this technique assumes that the data conforms to a Gaussian distribution. The mean describes the central tendency for a collection of numbers. The standard deviation (std) describes the average spread of values from the mean.

$$standard \quad value = \frac{value - mean}{std} = \frac{value - \frac{\sum_{i=1}^{N} values_i}{count(values)}}{\sqrt{\frac{\sum_{i=1}^{N} (values - mean)^2}{count(values) - 1}}} \tag{13}$$

Here, the method implemented is the standardization of the data.

3.7.6 Performance metrics

The estimation of the classification algorithm predictions quality is a required phase in the implementation of the algorithm. After the training phase, the algorithm is tested on unknown data. Its class predictions are compared to the real classes. Several standard measures exist to summarize the efficiency of these predictions.

For classification problems:

Among the performance metrics, there is the classification **accuracy ratio**. It is a ratio of the number of correct predictions out of all predictions that were made:

$$accuracy = \frac{correct \quad predictions}{total \quad predictions} * 100 \tag{14}$$

Another popular way to calculate the performance is to use the **confusion matrix**, which provides a summary of all of the predictions made compared to the expected actual values. The results are presented in a matrix with counts in each cell. The counts of actual class values are summarized horizontally, whereas the counts of predictions for each class values are presented vertically.

For regression problems:

Other techniques are more specific to regression problems such as the Mean Absolute Error (MAE) that calculates the average of the absolute error values, and the Root Mean Squared Error (RMSE) that calculates the square root of the mean of the squared differences between actual outcomes and predictions.

3.7.7 Evaluation of the performance of the algorithms

To evaluate the performance of the model, we split the data into a train and a test dataset (usually 2/3 of the data for training and 1/3 for testing). During the training phase, the model remembers every observation it is shown (which corresponds to the train set). Afterwards, it must evaluate its performance on unseen data. The model makes predictions on the test set and evaluate the predictions against the expected results.

This evaluation technique of a single train-test set split is very fast. It is ideal for large datasets where there is strong evidence than both splits are representative of the underlying problem, and when the investigated algorithm is slow to train. The downside of this technique is that it can have a high variance. The difference in the train and test datasets can lead to meaningful differences in the estimate of accuracy. Preferably, we want to ensure that the estimate of accuracy is stable independently of the configuration.

K-fold cross validation is a method to estimate the performance of the model with less variance than a single train-test split. The dataset is split in k-parts, called folds. The algorithm is trained on k-1 folds and tested on the held back fold. The mean and variance are computed from the different performance scores. The final result gives a more reliable estimate as it has been trained and tested multiple times on different data. The choice of k must allow the size of each test partition to be large enough to be a reasonable sample of the problem, whilst allowing enough repetitions of the train-test evaluation of the algorithm to provide a fair estimate of the algorithms performance on unseen data. Usually, k values are included between 3 and 10.

Leave-one-out cross validation is a cross validation with the test set containing only one sample, and k is set to the number of observations in the dataset. This technique provides a better estimate of accuracy on unseen data, but it is also a computationally more expensive procedure than k-fold cross validation.

Repeated random test-train splits is another variation of the k-fold cross validation with a random train/test split. The process can be repeated as many times as needed, and provides a

reduction in variance of the estimated performance compared to the k-fold cross validation. Its main drawback is the introduction of redundancy into the evaluation, as the repetitions may include much of the same data in the train and test sets.

To test the evaluation of the model, K-fold cross validation is implemented. If results are not satisfying enough, Leave-one-out cross validation and Repeated random test-train splits will be considered.

When the performance of the algorithm is satisfying and it is chosen to be the algorithm implemented in the model, it is retrained on the entire dataset and ready for operational use. [79-80]

3.7.8 Algorithm implementation

Even though I have been studying how to implement all the algorithms mentioned above, their implementation into the Scikit-Learn Python library might be way faster and optimized. The idea is to test these different algorithms with scikit learn to determine which ones works best on the problem at hand, and implement ourselves the ones with the best results to tune their parameters and increase their accuracy.

First, the data is divided into a training and testing set. Then, the algorithms are trained on the training set: they learn from the data containing observations whose category membership is known. Afterwards, the accuracy of the classification is tested on the testing set (the class membership is not given, and the predictions of the algorithms are compared to the real membership). Cross-validation is performed to ensure the learning algorithms are not overfitting to the training set. Some parameters might need to be tuned for a better fitting to the data.

4 Data recording

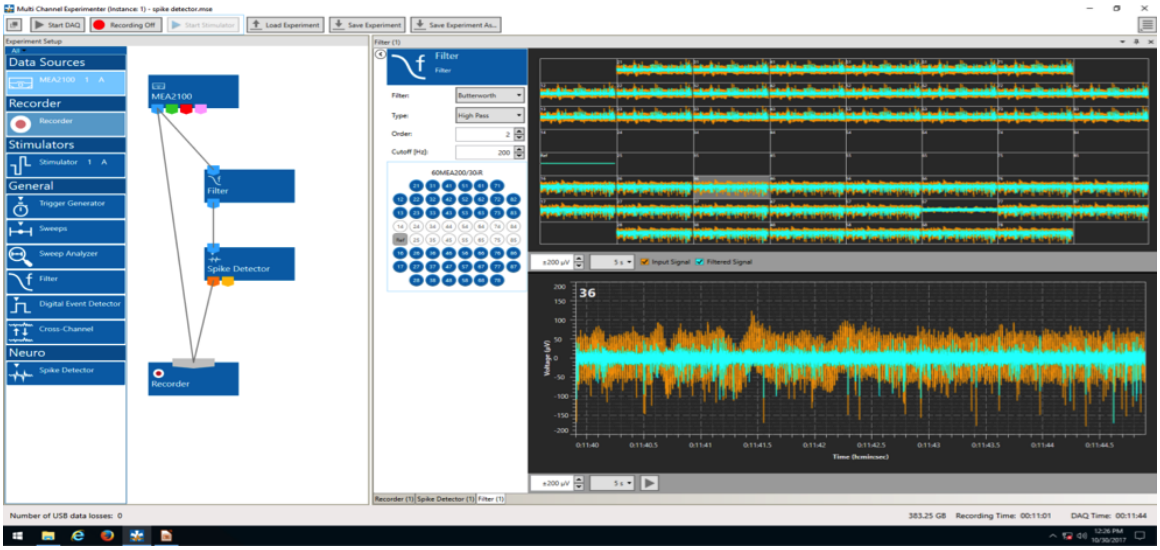


Figure 6: Screenshot of the Multi Channel System Explorer Software during the recording of MEA5 Abeta 90min 10nm

Figure 6 is a screenshot of the interface of the Multi Channel System Explorer Software, used for the recording of the data. On the right upper panel, we can see the electrophysiological activity of all the active channels, and below we observe a bigger representation of the selected channel. Just on the left, there are buttons to define which are the active channels. On the left panel, we notice the functions applied to the recorded data. The data is filtered (what kind of filter?) and the data goes through a spike detector while it is recorded. (note: when I extract the data I do not extract the detected spikes performed by the spike detector)

5 Algorithm development

5.1 Calendar for the algorithm implementation

1. Define the problem
   Investigate and characterize the problem in order to better understand the goals of the project.
2. Bibliography
   Summary of the state-of-the-art in the field and gathering of all experiments and analysis that are similar to the project
3. Analyze Data
   Use descriptive statistics and visualization to better understand the data available.
4. Prepare Data
   Use data transforms in order to better expose the structure of the prediction problem to modeling algorithms. Extract meaningful features.
5. Evaluate Algorithms
   Design a test harness to evaluate a number of standard algorithms on the data and select the top few to investigate further.
6. Improve Results
   Use algorithm tuning and ensemble methods to get the most out of the algorithms on the data.
7. Finalize the model, make predictions and present results.
8. Provide an interface for easily analyzing the data

5.2 Algorithm Development: Programming Language and Libraries

Python is a general purpose interpreted programming language used both for R&D and in production. It is easy to learn and use primarily because the language focuses on readability. Furthermore, it is a dynamic language, very suited to interactive development and quick prototyping with the power to support the development of large applications. Its ecosystem is growing and it benefits from an excellent library support. It is also widely used for machine learning and data science. Moreover, this language is open source and usable commercially under the BSD license, and it is supported by multiple systems and platforms.

Among the ecosystem of Python libraries, there is **Scipy**, specialized in mathematics, science and engineering. It contains modules such as **NumPy** to work efficiently with data in arrays, **Matplotlib** for the creation of 2D charts and plots and also **Pandas** which possesses tools and data structures to organize and analyze the data. **Scikit-learn** is also a remarkable library specialized in machine-learning algorithms for classification, regression and clustering. It provides tools for related tasks such as evaluating models, tuning parameters and pre-processing data.

The algorithm is developed in **Python 2.7** instead of Python 3, in case we need to use some libraries that haven't been updated yet for Python 3. I developed the algorithm in Python with the help of the **Spyder IDE** (Integrated Development Environment), which is considered as one of the best free IDE for Python.

5.3 Data Extraction

5.3.1 Data conversion to HDF5

The data extracted from the experiments through the devices commercialized by MultiChannel Systems (MCS) are in their own special format. The data needs to be converted into a more conventional format. The software Multi Channel DataManager (only compatible with Windows and downloaded from MCS website) converts the data in the HDF5 format (.h5 files). This format is widely used for big databases, and the data is easily transferred in Numpy arrays or Pandas dataframes for further processing.

5.3.2 Data conversion to Numpy arrays and Pandas dataframes

The conversion of HDF5 files into a format that can be handled easily in Python (through Numpy or Pandas) requires handling the **HDF5 library** and also the **McsPyDataTools library** (provided by MCS).

McsPyDataTools is a library able to extract data from the HDF5 files obtained through the Multi Channel DataManager software. Indeed, MCS has structured its HDF5 files in their own specific way. However, their library is not fully functional and even though their tutorial is followed (see **McsPyDataNotebook.ipynb** a Jupyter Notebook file), several errors appear and some elements of the files are impossible to extract. (see **read_MCS.py** file)

The alternative is using the HDF5 library. For each new HDF5 file, I run **print_entire_h5_file_structure.py** file, and the full structure of the file is displayed. Then, I run the specific functions **extract_analogs.py** and **extract_segments.py** to extract Analog Stream and Segment Stream data. Figure 7 represents the global architecture of HDF5 MCS files. Figure 8 shows an example of the data structure of HDF5 files from MCS recordings. The example is extracted from 2016-04-15T14-40-38McsRecording.h5 with print_entire_h5_file_structure.py

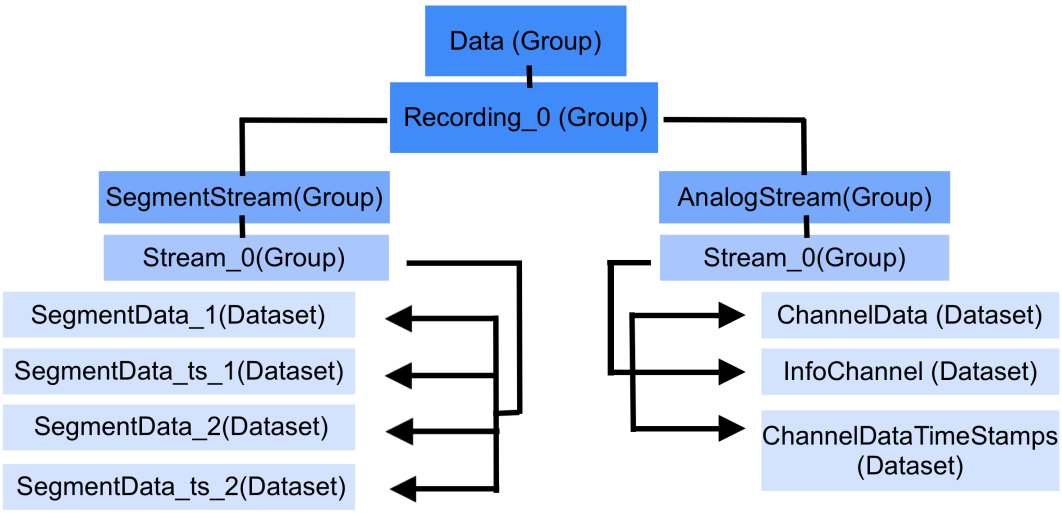


Figure 7: Structure of the HDF5 file built by MCS

```
<HDF5 file "2016-04-15T14-40-38McsRecording.h5" (mode r)> (File) /
    Data (Group) /Data
        Recording_0 (Group) /Data/Recording_0

            AnalogStream (Group) /Data/Recording_0/AnalogStream
                Stream_0 (Group) /Data/Recording_0/AnalogStream/Stream_0
                    ChannelData (Dataset) /Data/Recording_0/AnalogStream/Stream_0/ChannelData     len = (37, 4060000)
                    InfoChannel (Dataset) /Data/Recording_0/AnalogStream/Stream_0/InfoChannel     len = (37,)
                    ChannelDataTimeStamps (Dataset) /Data/Recording_0/AnalogStream/Stream_0/ChannelDataTimeStamps     len = (1, 3)

            SegmentStream (Group) /Data/Recording_0/SegmentStream
                Stream_0 (Group) /Data/Recording_0/SegmentStream/Stream_0
                    SegmentData_26 (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SegmentData_26     len = (61, 1777)
                    SegmentData_7 (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SegmentData_7     len = (61, 3)
                    SegmentData_24 (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SegmentData_24     len = (61, 4165)
                    SegmentData_ts_7 (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SegmentData_ts_7     len = (1, 3)
                    SegmentData_23 (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SegmentData_23     len = (61, 2720)
                    SegmentData_ts_23 (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SegmentData_ts_23     len = (1, 2720)
                    SegmentData_ts_26 (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SegmentData_ts_26     len = (1, 1777)
                    SegmentData_ts_18 (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SegmentData_ts_18     len = (1, 32)
                    SegmentData_ts_19 (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SegmentData_ts_19     len = (1, 6240)
                    SegmentData_19 (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SegmentData_19     len = (61, 6240)
                    SegmentData_18 (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SegmentData_18     len = (61, 32)
                    SegmentData_ts_24 (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SegmentData_ts_24     len = (1, 4165)
                    InfoSegment (Dataset) /Data/Recording_0/SegmentStream/Stream_0/InfoSegment     len = (37,)
                    SourceInfoChannel (Dataset) /Data/Recording_0/SegmentStream/Stream_0/SourceInfoChannel     len = (37,)
```

Figure 8: Structure of the HDF5 file 2016-04-15T14-40-38McsRecording.h5 from MCS recordings extracted with print_entire_h5_file_structure.py

5.3.3 5 minutes data extraction from the files

The data analyzed in this project are the first 5 minutes of all the recorded files. What motivated this choice is that we want all the files having the same weight in the analysis of the data and the design of the machine learning algorithm. Also, the data is saved with its metadata and extracted

parameters in PKL files. PKL files are created by pickle, a Python module that enables objects to be serialized to files on disk and deserialized back into the program at runtime. It contains a byte stream that represents the objects. Saving the data and its parameters makes further analysis easier, without having to extract all the parameters all the time there is a new analysis added on to the project.

5.3.4 Data files

Here are presented the files considered for the analysis:

Index	category	subcategory	file	signature	date	info	raw_data_path	extracted	pkl
0	amyloid	15	2016-04-15T14-40-38McsR…	2016-04-15T14-40-38	2016-04-15	+ amyloidB 10nm 15min cortex	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
1	memanti		2016-04-15T13-42-38McsR…	2016-04-15T13-42-38	2016-04-15	+ memantine 1µm ds cotex	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
2	unknown		2016-04-15T12-55-37McsR…	2016-04-15T12-55-37	2016-04-15	+stim	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
3	healthy		2016-04-15T12-22-47McsR…	2016-04-15T12-22-47	2016-04-15	actSp	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
4	unknown		2016-04-15T14-13-44McsR…	2016-04-15T14-13-44	2016-04-15	postwaching	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
5	healthy		2017-10-16T13-02-03McsR…	2017-10-16T13-02-03	2017-10-16	actSp	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
6	healthy		2017-10-16T13-52-11MEA2…	2017-10-16T13-52-11	2017-10-16	actSp 2	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
7	amyloid	15	2017-10-16T13-35-49MEA1…	2017-10-16T13-35-49	2017-10-16	amyloidB 15min 10nm	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
8	amyloid	90	2017-10-16T14-35-45MEA1…	2017-10-16T14-35-45	2017-10-16	amyloidB 90min 10nm	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
9	memanti	5	2017-10-16T14-19-30MEA2…	2017-10-16T14-19-30	2017-10-16	memantine 5min 1um	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
10	healthy		2017-10-23T11-30-27MEA3 act sp.h5	2017-10-23T11-30-27	2017-10-23	actSp	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
11	healthy		2017-10-23T10-47-04MEA4 act sp sans CO2.h5	2017-10-23T10-47-04	2017-10-23	actSp 2 sans CO2 MEA 4	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
12	amyloid	40	2017-10-23T12-31-28MEA3 Ab 10nM 40min.h5	2017-10-23T12-31-28	2017-10-23	amyloidB 10nM 40min	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
13	amyloid	90	2017-10-23T13-16-17MEA3 Ab 10nM 90min.h5	2017-10-23T13-16-17	2017-10-23	amyloidB 10nM 90min	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
14	memanti	15	2017-10-23T12-12-11MEA4 Mem 1uM.h5	2017-10-23T12-12-11	2017-10-23	memantine 1uM 15min	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
15	healthy		2017-10-30T10-26-54MEA5 Act sp.h5	2017-10-30T10-26-54	2017-10-30	actSp	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
16	healthy		2017-10-30T10-48-59MEA6 Act sp.h5	2017-10-30T10-48-59	2017-10-30	actSp 2	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
17	amyloid	45	2017-10-30T11-37-10MEA5 Abeta 55 min.h5	2017-10-30T11-37-10	2017-10-30	amyloidB 45min	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
18	amyloid	90	2017-10-30T12-15-21MEA5 Abeta 90 min.h5	2017-10-30T12-15-21	2017-10-30	amyloidB 90min	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…
19	memanti	15	2017-10-30T11-20-57MEA6 Mem 15 min.h5	2017-10-30T11-20-57	2017-10-30	memantine 15 min	/Volumes/ DIANAAVALOS/…	True	/Volumes/ DIANAAVALOS…

Figure 9: File structuration

Figure 9 is a screenshot of the file structure within the algorithm. For each file, the category is extracted: actSp, amiloydB, memantine, DMSO, BIC, BIA, NR and actSp, NR and amyloidB. The subcategory extracts the time in minute of incubation of the neurons with the pharmacological agent. The complete filename is extracted, and its signature (a specific number proper to each file). All the important info is saved, in case other parameters are relevant but are not yet considered in the analysis. The raw data path is saved, to find the file easily in the computer folders. If the parameters and features have already been extracted from the file, extracted mentions "True", and the parameters extracted are saved in a pickle file which path is mentioned in pkl column.

5.3.5 Mapping of the channels

When extracting the data from the HDF5 files, the channel numbering is not the same than on the Multi Channel Experimenter interface. Channel names are mapped with the function channel_ordering() in neural_data.py. The type of neuron is also extracted, so we know if the channel was recording cortex or hippocampal neurons.

labels	neurontype	channelID ▲
47	hippocampe	0
48	hippocampe	1
46	hippocampe	2
38	hippocampe	4
37	hippocampe	5
28	hippocampe	6
36	hippocampe	7
27	hippocampe	8
17	hippocampe	9
26	hippocampe	10
16	hippocampe	11
Ref	Ref	14
13	cortex	18
23	cortex	19
12	cortex	20
22	cortex	21
33	cortex	22
21	cortex	23
32	cortex	24
31	cortex	25

Figure 10: Extract of the channel referencing. Labels are the name displayed on the screen when recording is performed through the Multi Channel System Explorer Software

5.4 Data visualization and data smoothing

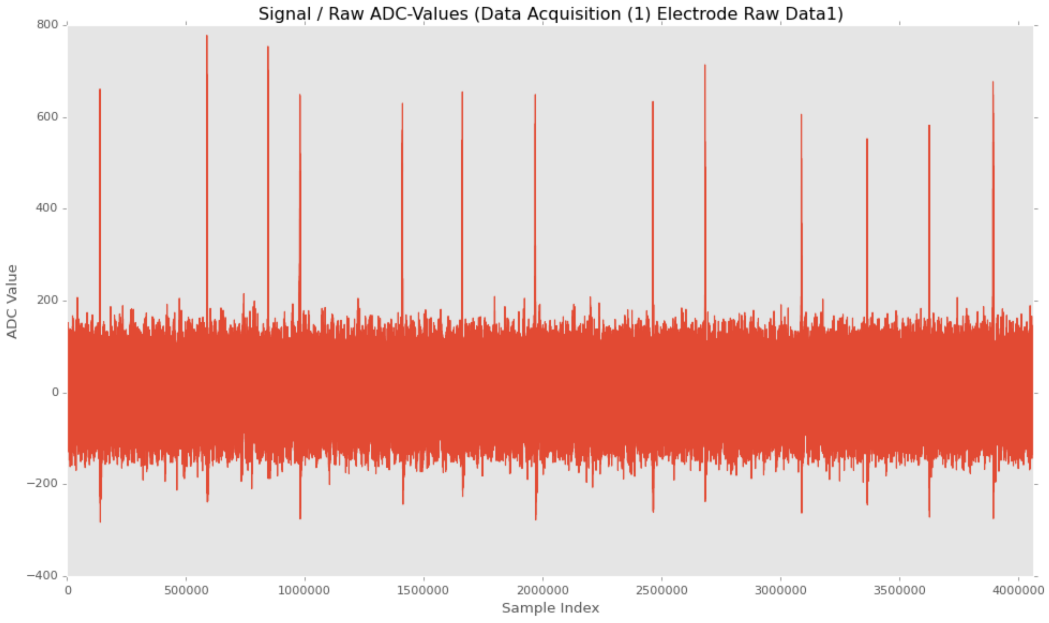


Figure 11: Visualization of channel 1 from healthy20160415

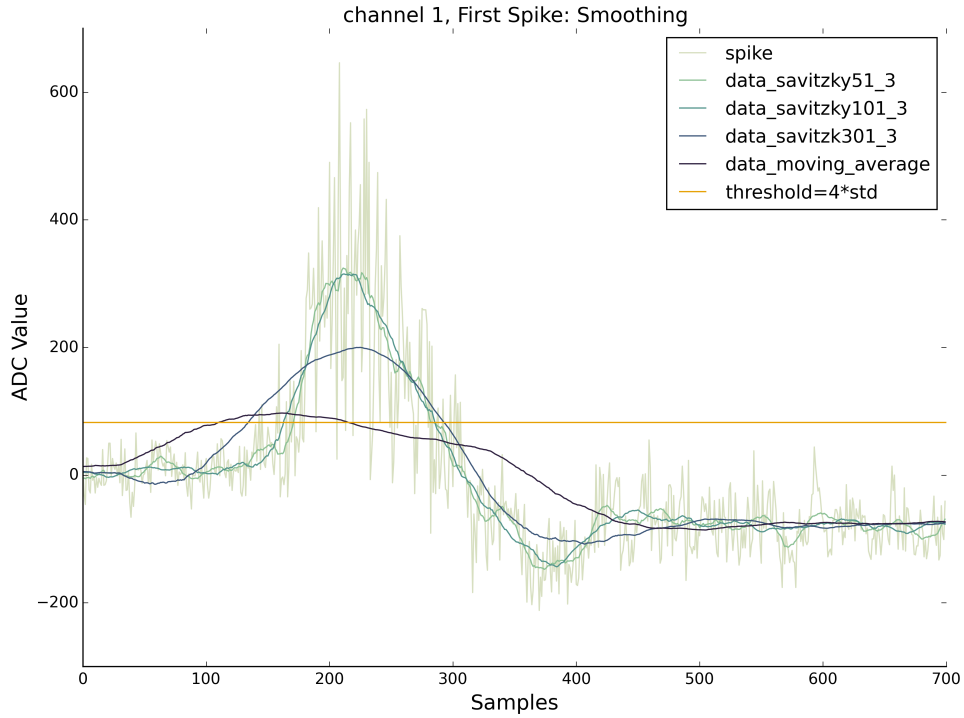


Figure 12: Optimization of the method for smoothing the signal

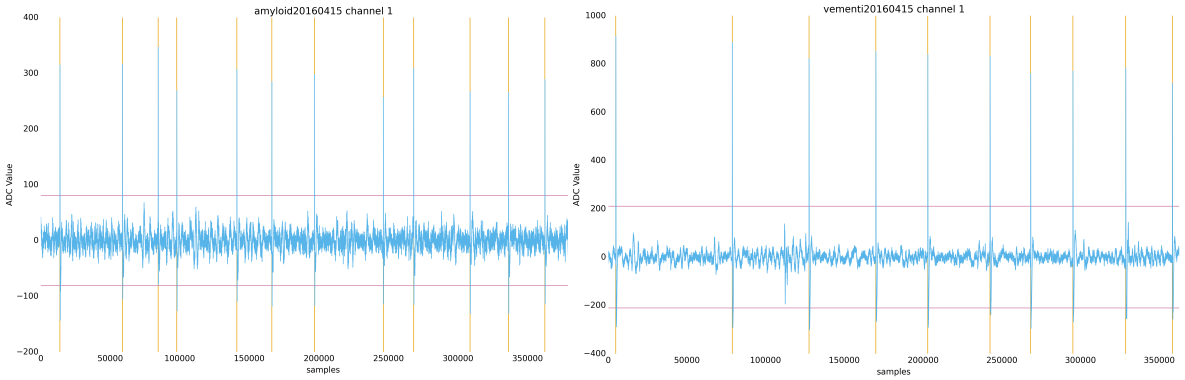
The following figures have been realized from channel 1 of the file amyloid20160415.h5 with **plots_Savitzki_comparison.py** and **extract_data.py**

Figure 11 represents the data extracted from one channel of recordings. The signal is downsampled by a factor 10 (faster signal processing). In order to simplify the spike detection, the signal is smoothed (Fig. reffig:smoothing). That means, increasing the signal-to-noise ratio without greatly distorting the signal. An interesting study from Guin et al. (2007) 81 compares two methods, which are the Savitzki-Golay filter and the Moving Average.

The moving average filter operates by averaging a number of points across a moving window, in a recursive fashion. The input parameters of this function are the data to filter and the number of samples included in the moving window. It is an unweighted moving average filter that works as a convolution with an horizontal line (all the coefficient are equal to 1/(sample number).

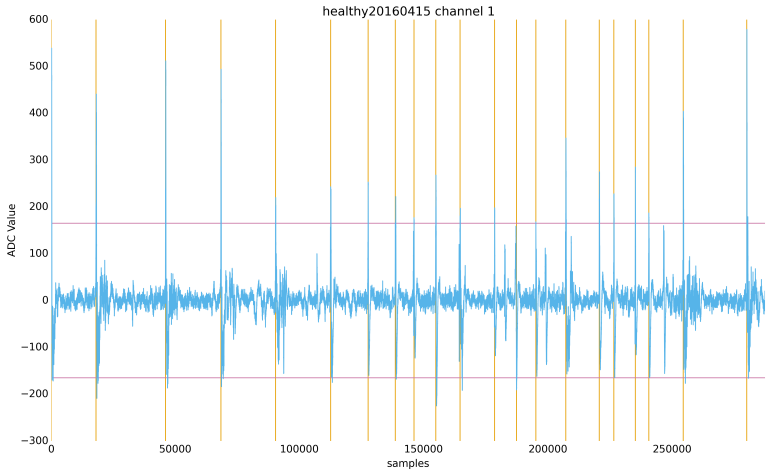
The Saviztky-Golay filter is based on the least squares polynomial fitting across a moving window within the data in the time domain. This function takes as input the polynomial order and the number of samples in the moving window. A polynomial order of degree 2 takes into account the curvature, and a degree 3 considers the inflexion points. In theory, the higher the polynomial order, the bigger the smoothing without attenuation of data features. The size of the window has to be big enough so the smoothing is efficient: otherwise, the polynomial curve passes through all the sampled points, and there is no smoothing. On the other hand, a big window will generate an important smoothing and reduce significant details.

I decided to implement and apply both methods to the data (Fig. 12). It is clear that the



(a) amyloid20160415

(b) vementi20160415



(c) healthy20160415

Figure 13: Spike detection from the first recording (2016-04-15) Signal is in blue, threshold lines in pink (4*std) and spikes detected in yellow.

Saviztky-Golay filter provides a better preservation of the data features, which are more attenuated by the moving average filter.

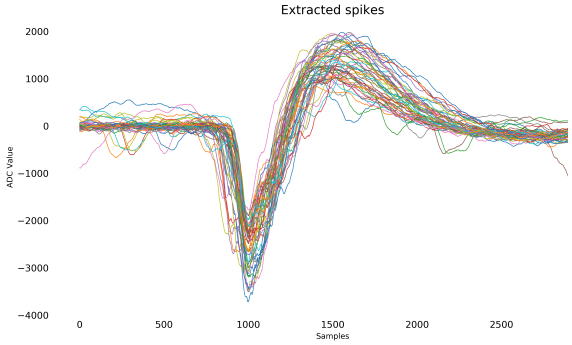
The next step is the tuning of the filter. The polynomial order is set at 3, because no huge difference was noticed in the result by increasing the complexity of the filter. Regarding the moving average, I applied the Savitzky-Golay filter at order 3 with different window sizes (300, 100 at 50 samples per window) (Fig. ??). 100 samples by window seemed a good trade-off between smoothing and accuracy.

5.5 Spikes detection and Extraction

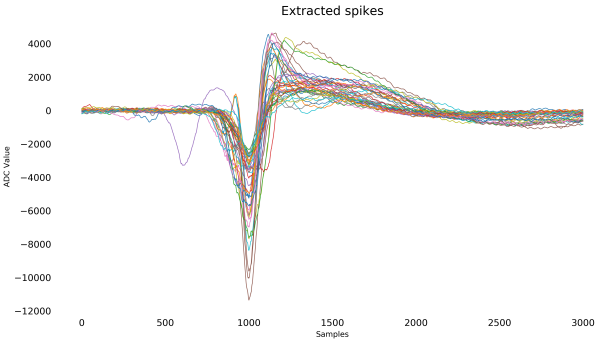
In Fig 13, signal is represented in blue and threshold line in pink. Detected spikes are marked with yellow vertical lines. The threshold is defined as the standard deviation multiplied by a factor 4. (Several factors were tried, and 4 appeared to be the more efficient for spike detection). The threshold is therefore auto adapted to the properties of the signal. Whenever the threshold is crossed, a spike is detected. However, some noise remaining in the signal can lead

to a double detection of the spike. A simple function was implemented to remove these artifacts: threshold_crossed_selection function.

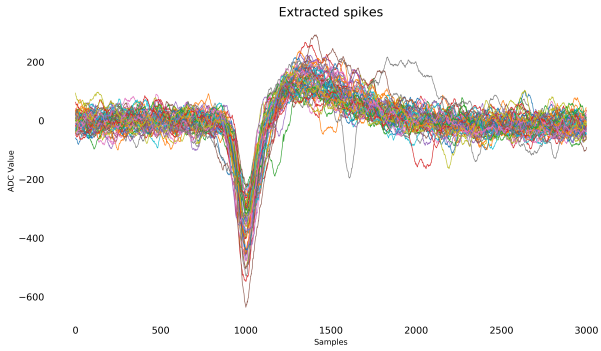
When the data crosses the threshold at a given sample point, we look for the maximum in the local area. A given number of samples are extracted before and after this sample point where the maximum is detected.



(a) 2017-10-16 Healthy



(b) 2017-10-16 Amyloid B + Memantine



(c) 2017-10-16 Amyloid B

Figure 14: Spike extraction from the recordings on the 16 of October 2017, for channel channel 53 (Cortex)

Most of the time the spikes extracted in the Cortex have a higher quality than the ones recorded in the Hippocampus, that often look like noise. Moreover, the shape of the spikes extracted in the Cortex and the Hippocampus are very different. The spikes extracted on the Hippocampus should be aligned with their maximum. A change to add in the algorithm would be change the alignment of the spikes depending on the type of channel considered. A good question to investigate would be understanding this difference in shape from the different neuron types. In theory, we would expect the action potentials to have the same global shape, regarding the ion channels activation in the action potential emission (

5.6 Neural_data class

For each file, an instance Neural_data is created. The class has the following properties:

Class Neural_data

Methods:
-__init__
-extract_data_all_channels_fast
-processdata -spikes_extraction_all_channels
-extract_features_all_spikes
Attributes:
-raw_data_path
-signature
-data
-sampling_frequency
-time
-threshold
-threshold_crossed
-spikes_max_index
-spikes_extracted

The instance is created, the data recorded from every channel is extracted, as well as the sampling frequency and the time vector. The data and time arrays are downsampled by a factor 10, because the amount of data is very important to process and such precision is not needed. The function **savitzky_golay()** applies a Savitzky-Golay filter to smooth the data. The window of the filter is set to 101, and the polynomial degree is set to 3. Spikes are detected with the **detect_threshold_crossing()** function. When the signal crosses a given threshold set to 4 times the standard deviation of the signal, the value of the sample point is saved. Finally, the function **spikes_extraction_all_channels()** extracts all the spikes. It looks for the maximum value of the signal in the neihborhood of the point where the threshold was crossed. 1000 samples before and 2000 samples after this point are extracted for the spike. All the spikes are therefore aligned to their maximum. The function **plot_full_channel_threshold_crossed_lines()** plots the signal of the channel with the threshold limit and the detected spikes. The function **plot_extracted_spikes()** plots the extracted spikes aligned with their maximum value.

5.6.1 Parameters to analyze in neural signals: in spike extracted and whole signal

The characterization of the signal through feature extraction is essential to understand how the distribution is within a class, and then between different classes, in order to obtain an efficient separation with the classification algorithm. Indeed, later we will be able to analyze the differences between healthy neurons and neurons that have developed Alzheimer disease. features_of_extracted_spikes.py is a function to extract significant features (or parameters) from the spikes. The function takes as input the extracted spikes of the signal of one channel, and also the timing of the emission of these spikes.
The features extracted for a spike are:
-The maximum amplitude: amp_max
-The minimum amplitude: amp_min
-The mean value: spike_mean: -The variance of the spike: spike_var
-The sample number where the max amplitude is reached (spikes aligned with the maximum value, so it should be the same for all spikes): index_max
-The sample number where the min amplitude is reached: index_min

-The distance between the 2 extrema (distance between index_max and index_min)
-The peak-to-peak amplitude
-The maximum reached by the derivative of the signal: derivate_max
-The sum of the absolute value of each point of the derivate: derivate_sum
-The peak2peak value of the 3 most significant peaks: top3p2p
-The absolute value of the 10 first coefficients from the Fourier Transform, which are the one that carry the most information (absolute value to get rid of the imaginary component): fft_r
The interspike interval (ISI), which is the distance between consecutive spikes, is also computed from the timing of the spike emission.

5.6.2 Observation of a signal characteristics through the analysis of its feature parameters

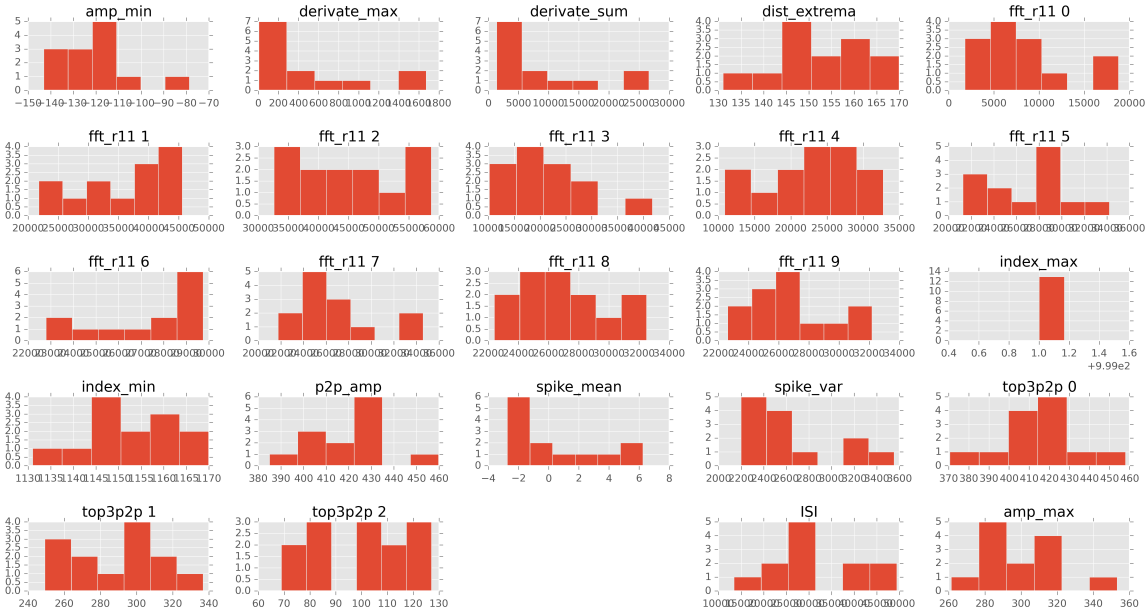


Figure 15: Histogram of the features extracted from the spikes of one channel

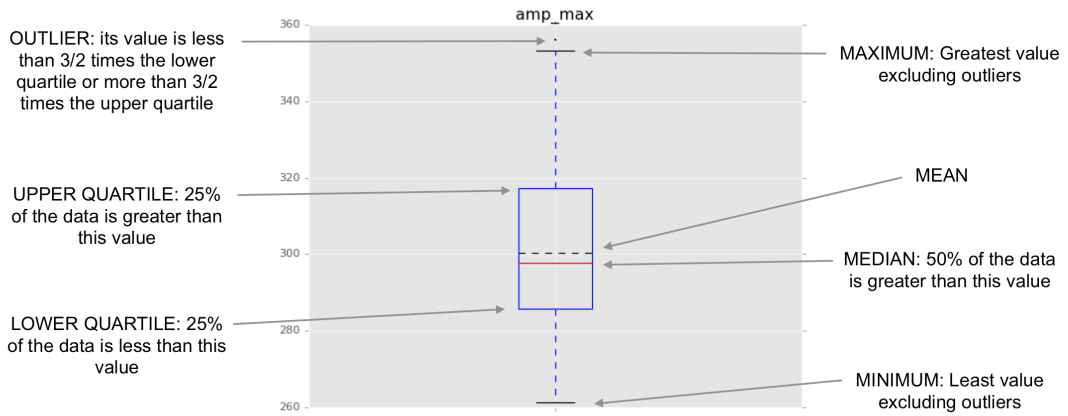


Figure 16: How to read a boxplot

Boxplots are a great tool to visualize the distribution of the data. The red line represents the median: half the scores are greater than or equal to this value and half are less. The blue box is the inter-quartile range; it represents 50% of the scores of the group. The upper limit of the blue box is the upper quartile: 25% of the data is greater than this value. The lower limit of the blue box is the lower quartile: 25% of the data is less than this value. The black dot line represents the mean. The black horizontal lines outside the box indicate the greatest and lowest values, excluding the outliers. A data point is considered an outlier when its value is less than 3/2 times the lower quartile or more than 3/2 times the upper quartile.

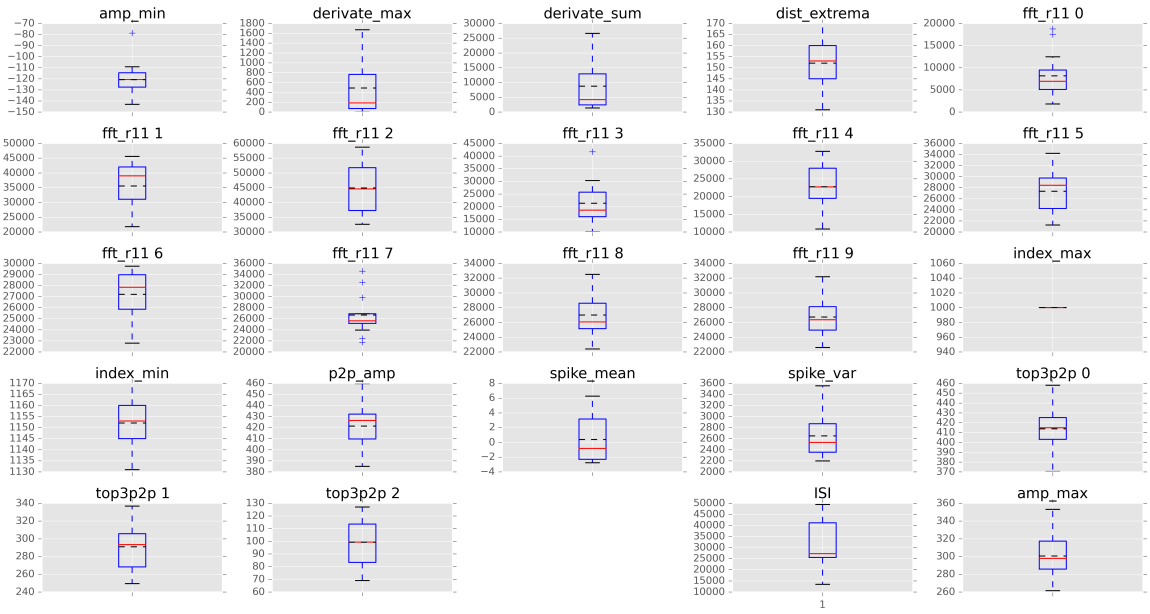


Figure 17: Boxplot of the features extracted from the spikes of one channel

5.7 Implementation of the Naive Bayes Algorithm

-The data is split into a train and a test set (67%-33%) by the function **splitDataset()** that takes as input the dataset and the split ratio.
- **summarizeByClass()** is applied to the training set.
- **separateByClass()** separates the training dataset instances by class value so that we can calculate statistics for each class. -**summarize()** collects the mean and the standard deviation for each feature, by class value. -**getPredictions()** calls **predict()** which calls **calculateClassProbabilities()**. It computes the Gaussian Probability Density Function (eq. 15) for each observation X of the test set, with each class y=k (mean μ_k standard deviation σ_k) thanks to the values extracted by **summarizeByClass()**.
Then **predicts()** assigns to each observation the class with which it has the highest probability.
Finally, **getAccuracy()** compares the predicted classes with the real ones and give the assignment results as a ratio in %.
All the code is written in the file **classify3spikes.py**.

5.8 Comparison of three signals

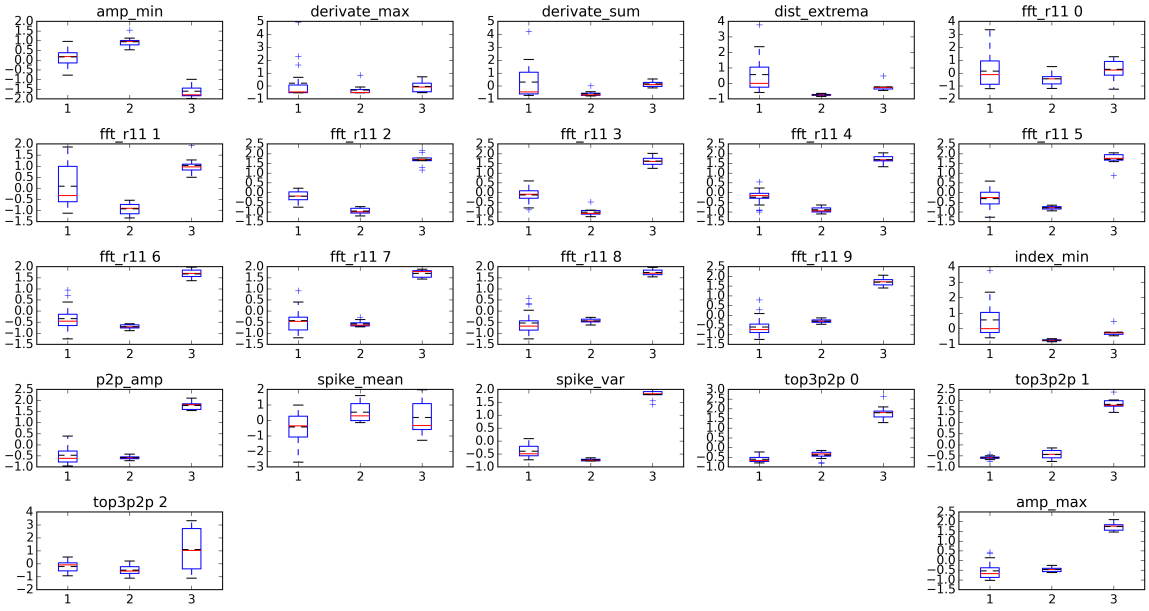


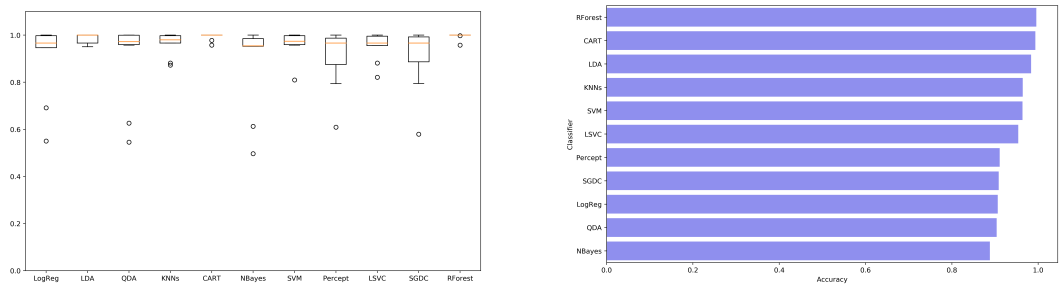
Figure 18: Comparison of the different feature parameters between distinct spikes.
1:healthy20160415, 2:amyloid20160415, 3: vementi20160415

5.9 Bars and Boxplot diagrams

In order to assess which algorithm will be the best for a given problem, we design a test harness that we use to evaluate several machine learning algorithms. The test harness involves splitting the dataset into a training and testing set, performing 10-fold cross validation to test each machine learning algorithm to evaluate, and measure performance.

Performance of the machine learning algorithms is assessed for all the files together and for each day of experience independently. Observing the classification of the different categories for one day of experiment is the first step to measure if there are enough difference in the recorded data to have an algorithm able to differentiate several categories. Generalizing this process to all the experiments over several days helps measuring the consistency of the data collected. In the following figures, we can observe the bar and box plot diagrams for the days of experiment independently, and then for all the files together .

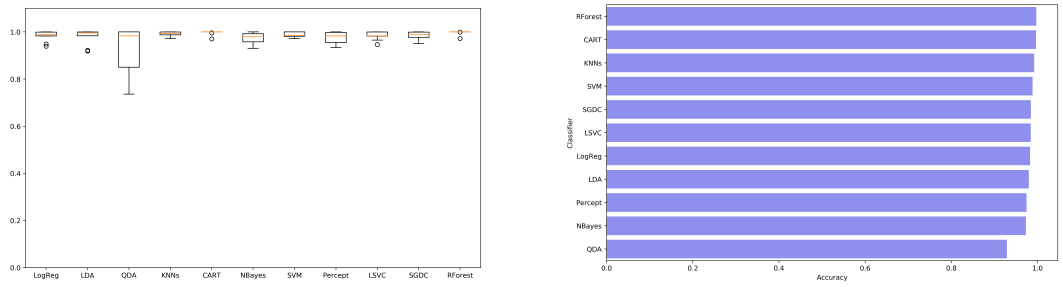
In the following diagrams, we have bar diagrams, to see the average performance of each kind of algorithm. Among the algorithm tested we have Random Forest (RForest), CART (Classification and Regression Trees), Linear Discriminant Analysis (LDA), K-nearest neighbors (KNN), Support Vector Machines (SVM), LSVC, Perceptron (Percept), SGDC, Logistic Regression (LogReg) , Quadratic Analysis Discriminant (QDA) and Naive Bayes (NBayes). models.append(('LSVC', LinearSVC())) models.append(('SGDC', SGDClassifier()))



(a) 2017-10-30 Algorithm comparison Boxplot

(b) 2017-10-30 Algorithm comparison Bars

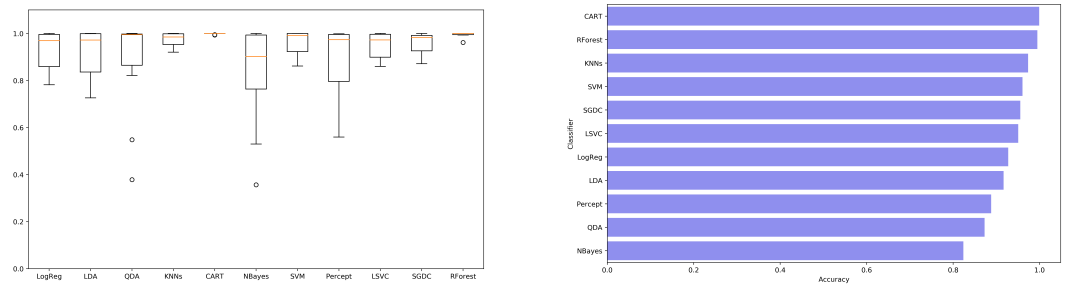
Figure 19: 2017-10-30 Algorithm comparison



(a) 2017-10-23 Algorithm comparison Boxplot

(b) 2017-10-23 Algorithm comparison Bars

Figure 20: 2017-10-23 Algorithm comparison



(a) 2017-10-16-Algorithm comparison Boxplot

(b) 2017-10-16-Algorithm comparison Bars

Figure 21: 2017-10-16 Algorithm comparison

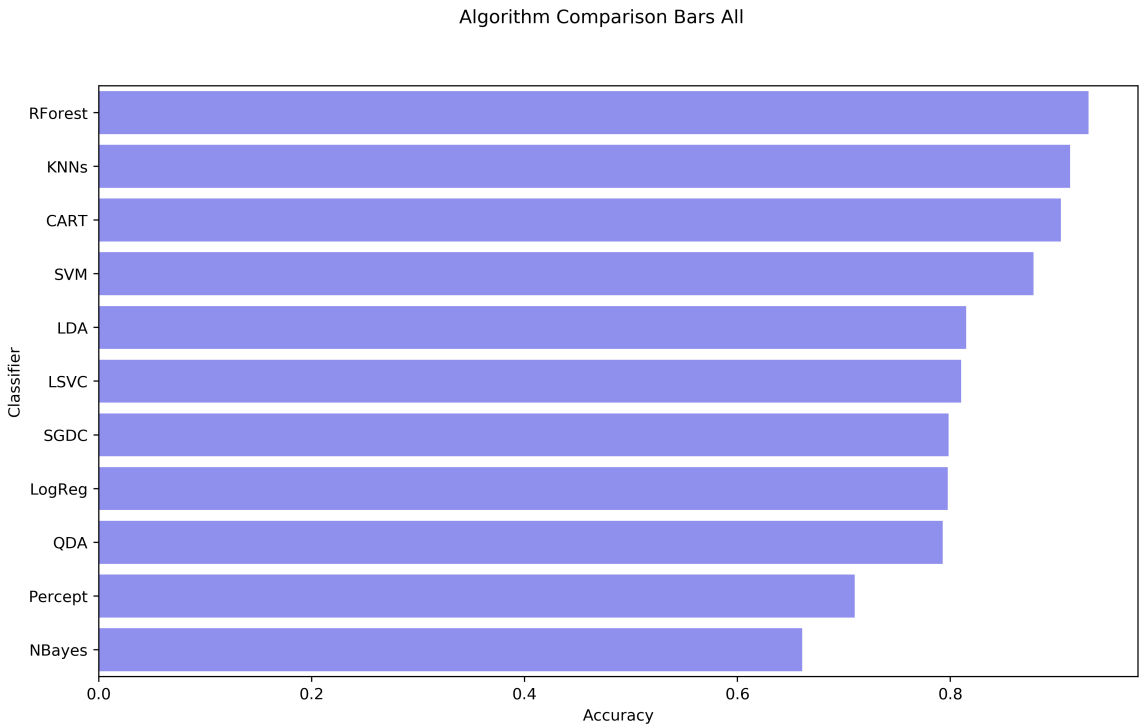


Figure 22: Comparison of the accuracy of different algorithms

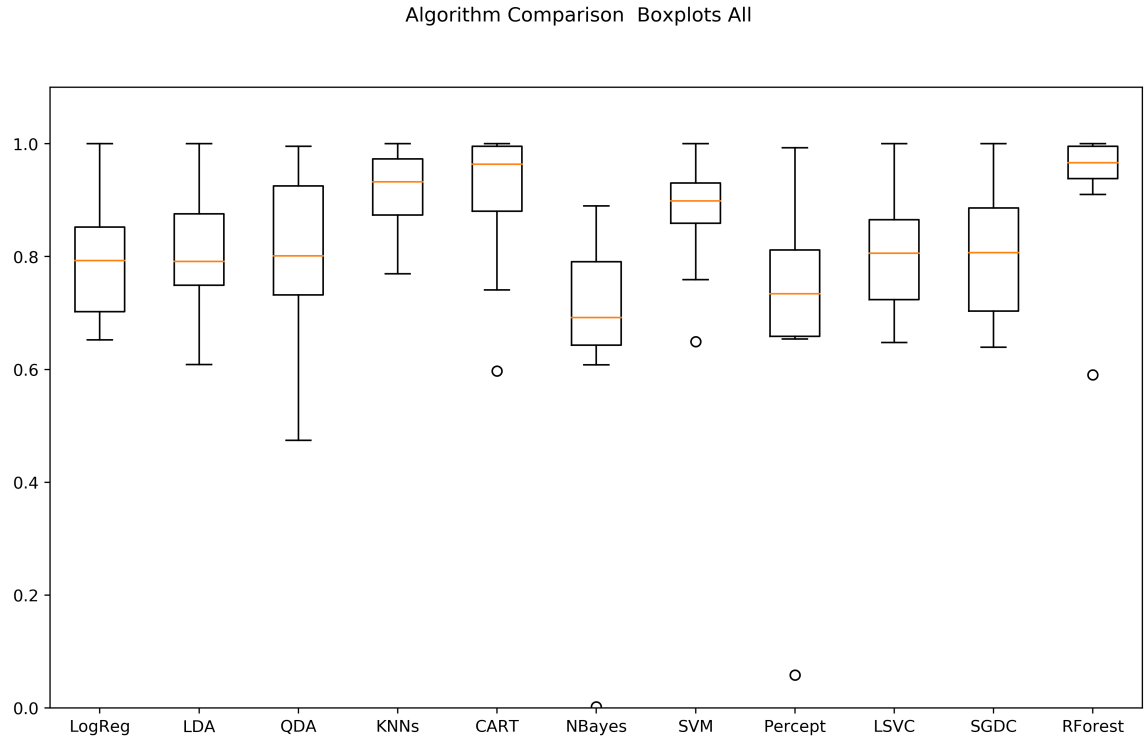


Figure 23: Comparison of the accuracy of different algorithms

5.10 Cross correlation between the features

Among the extracted features, some of them might be correlated, which means that they influence one another. We use the function corr() of the Pandas library to compute it. Several methods exist as Pearson, Kendall Tau and Spearman. The most common one is Pearson?s. A correlation of -1 or 1 shows a full negative or positive correlation respectively. Whereas a value of 0 shows no correlation at all and independence between the features. Pearson?s correlation coefficient assumes normal distribution of the attributes involved (at what moment are we standardizing the features?). Pearson's correlation measures the linear correlation between two sets of variables. It can be used to look for features x_i highly correlated with a class y of the data, which will subsequently be good at separating the different classes.

$$PearsonCorrelation(x_i, y) = \frac{cov(x_i, y)}{\sqrt{var(x_i)var(y)}} \tag{15}$$

Assessing the correlation between the features is essential as some features suffer poor performance if they are highly correlated features in the dataset. As the matrix is symmetrical, just half of it is plotted on the figure. The diagonal line is not plotted, as each attributes is in perfect correlation with itself, it is not a useful information. Correlation and covariance are highly related. As covariance of two groups of numbers describes how those numbers change together, it is a generalization of correlation which focuses on the linear relationship of these groups.

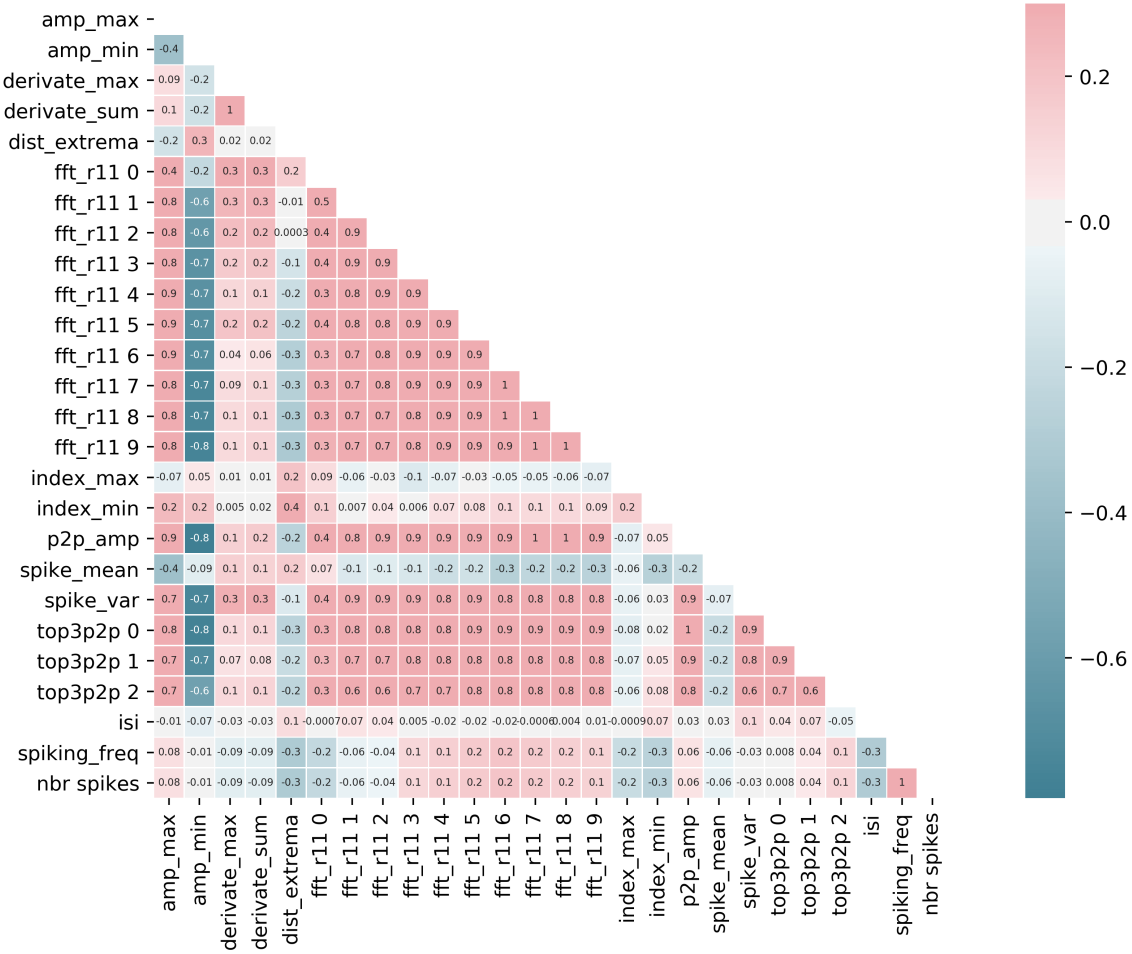


Figure 24: Crosscorrelation between different features

5.11 Measuring feature importance

Among the actual 26 features extracted from the signal, it would be interesting to know which are the most determinant ones in the separation of the different categories: healthy, amyloid B and memantine. In order to do so, we apply PCA, which is a method of dimensionality reduction which is unsupervised (does not consider the categories of the data samples), and aims at projecting the data into directions that optimize the variance of the data. The other technique is the Extra Tree Classifier, which is a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. [82]

The default values for the parameters controlling the size of the trees (e.g. max_depth, min_samples_leaf, etc.) lead to fully grown and unpruned trees which can potentially be very large on some data sets. To reduce memory consumption, the complexity and size of the trees should be controlled by setting those parameter values. + risk of overfitting

FeatureImp ExtraTree	FeatureImp PCA	featurename
0.0611683	0.265975	p2p_amp
0.0814273	0.25486	fft_r11 6
0.0651208	0.25156	top3p2p 0
0.0533615	0.249286	fft_r11 7
0.0468078	0.248982	fft_r11 5
0.0212595	0.245965	fft_r11 3
0.030622	0.244183	amp_max
0.0285323	0.241929	top3p2p 1
0.0524033	0.241708	fft_r11 4
0.0178631	0.239647	fft_r11 1
0.0303199	0.237467	fft_r11 2
0.0294083	0.233595	amp_min
0.0623839	0.233128	fft_r11 8
0.0414392	0.227468	top3p2p 2
0.100122	0.224457	fft_r11 9

(a) Features Importance with PCA

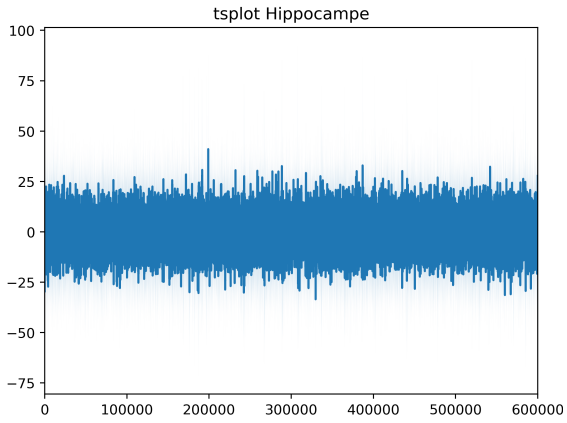
FeatureImp ExtraTree	FeatureImp PCA	featurename
0.100122	0.224457	fft_r11 9
0.0814273	0.25486	fft_r11 6
0.0719772	0.0802989	spiking_freq
0.0651208	0.25156	top3p2p 0
0.063984	0.0802989	nbr spikes
0.0623839	0.233128	fft_r11 8
0.0611683	0.265975	p2p_amp
0.0533615	0.249286	fft_r11 7
0.0524033	0.241708	fft_r11 4
0.0468078	0.248982	fft_r11 5
0.04466	0.0329304	index_min
0.0414392	0.227468	top3p2p 2
0.030622	0.244183	amp_max
0.0303199	0.237467	fft_r11 2
0.0294083	0.233595	amp_min

(b) Features Importance with the ExtraTree Classifier

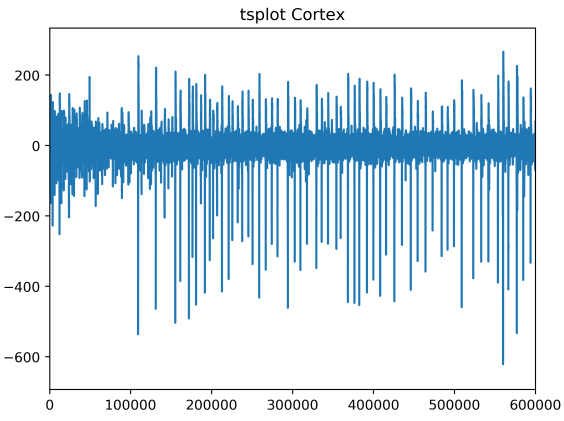
Figure 25: Features Importance

5.12 Comparing results for Hippocampus and Cortex neurons

Half of the channels record hippocampus neurons and half on the channels record cortex neurons. Comparing the signals of both types of neurons for all the files could provide an interesting insight on the data.

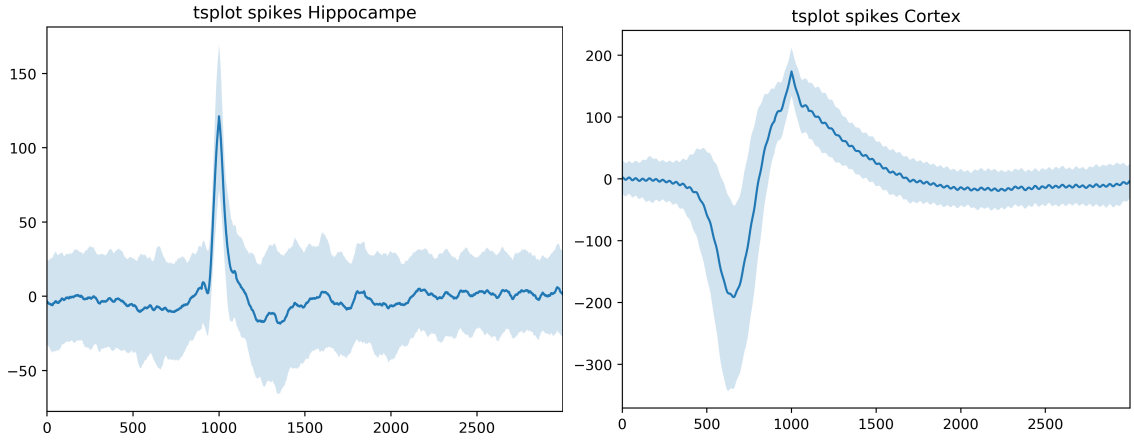


(a) 2017-10-16 amyloid Tsplot Hippocampus



(b) 2017-10-16 amyloid Tsplot Cortex

Figure 26: 2017-10-16 Amyloid Tsplots

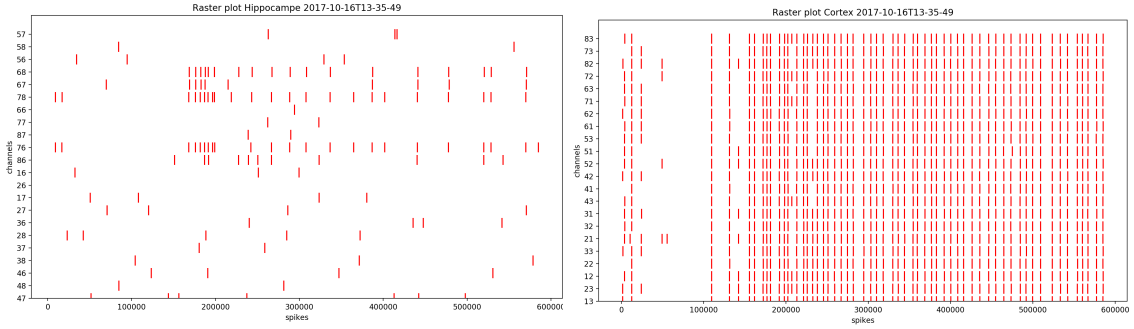


(a) 2017-10-16 amyloid Tsplot Spikes Hippocampus

(b) 2017-10-16 amyloid Tsplot Spikes Cortex

Figure 27: 2017-10-16 Amyloid Spikes Tsplots

Tsplots represent the average signal in dark blue with translucent bootstrapped confidence bands that represent the uncertainty of the signal. The translucent bars parameter could also be switched with standard deviation. Tsplots are relaized with the Seaborn library, which was created for drawing statistical graphic in the Python framework. Tsplots were created for the whole signal, as well as for the extracted spikes (Fig. 26 and 27)

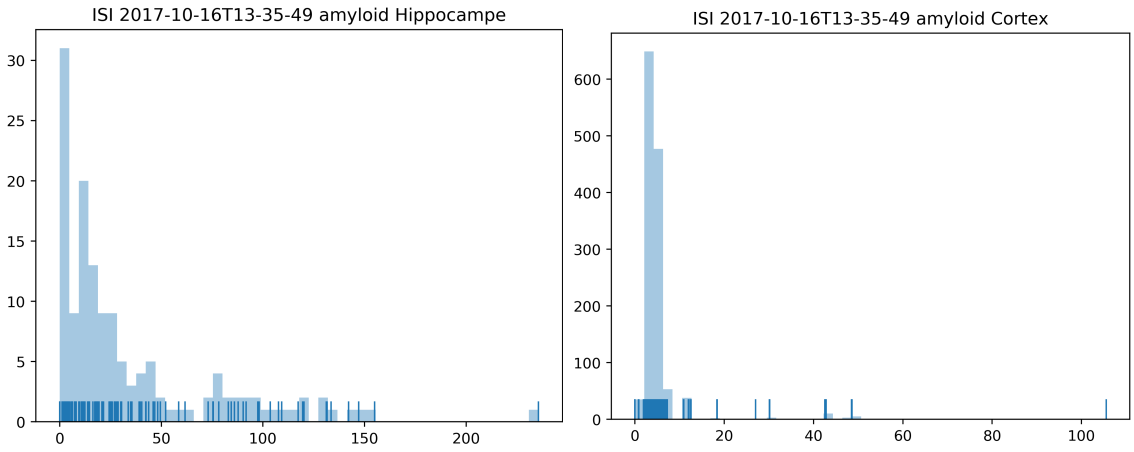


(a) Raster Plot Hippocampus

(b) Raster Plot Cortex

Figure 28: 2017-10-16 Amyloid Raster Plots

Raster plots extract spike events from the signal, and enable the visualization of the spike frequency of the different channels. In the provided example (Fig. 28), for the file 2017-10-16T13-35-49,we can picture the spiking frequency for all the channels, with the channels in the cortex and in the hippocampus separated. Interesting features to add to the model would be the spiking frequency of the Cortex and Hippocampe separately, or the relative difference between the two.



(a) 2017-10-16 amyloid ISI Hippocampus

(b) 2017-10-16 amyloid ISI Cortex

Figure 29: 2017-10-16 Amyloid ISI

Finally, the inter spike intervals (ISI) for Cortex and Hippocampal neurons are compared. ISI represents the distance between two spikes. Histograms are represented on figure 29

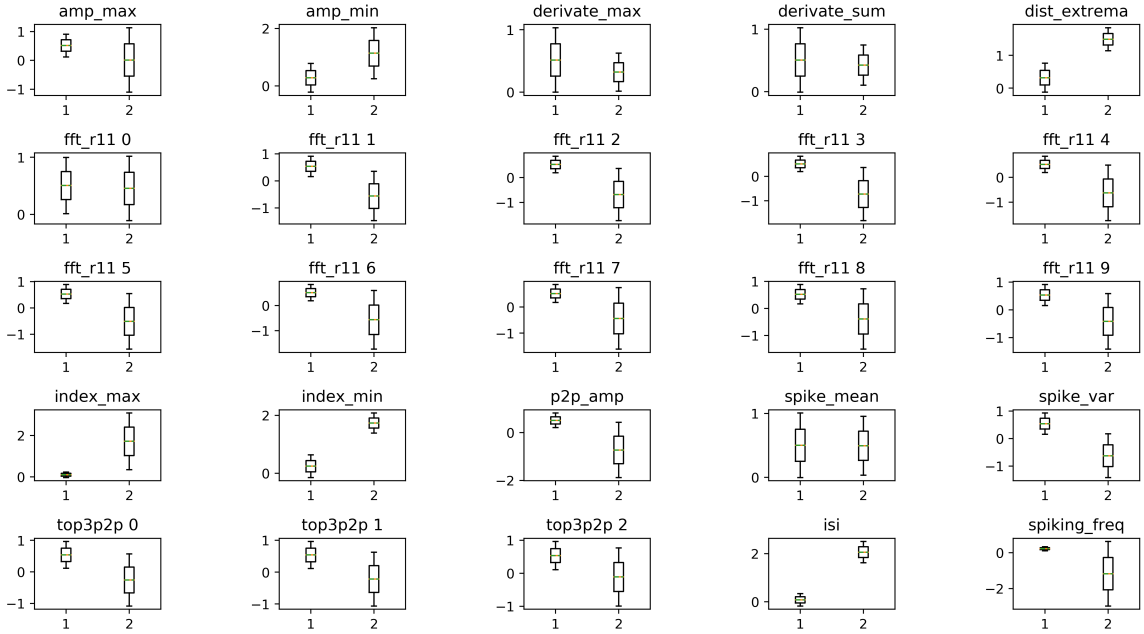
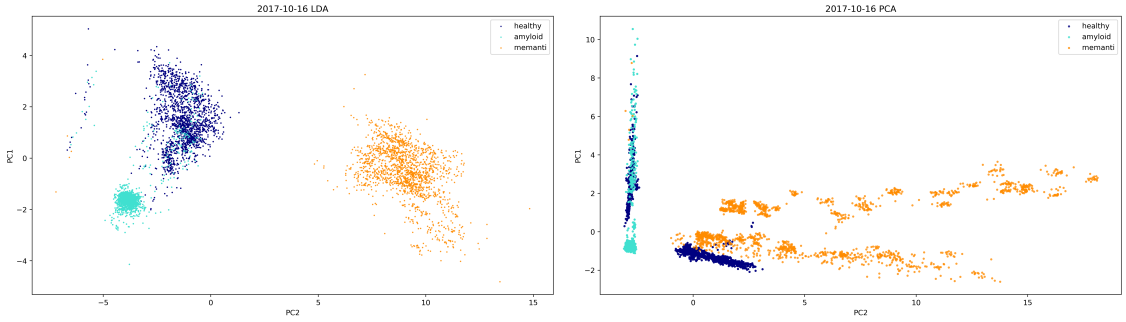


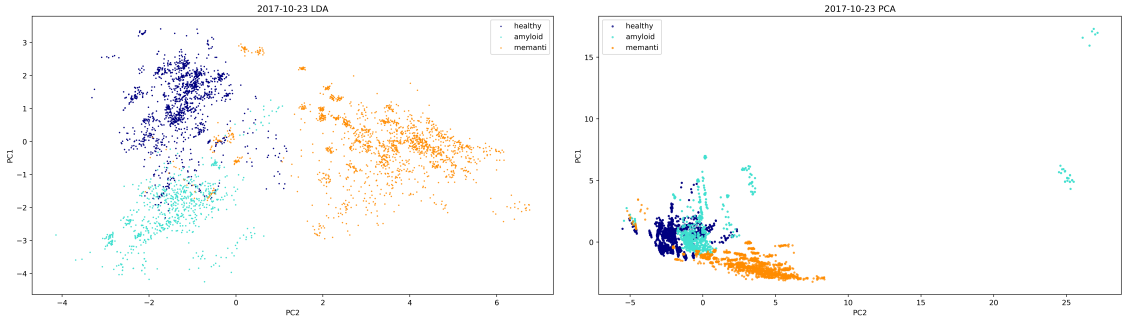
Figure 30: Boxplots 1: Cortex, 2: Hippocampus, 2017-10-16 Amyloid



(a) 2017-10-16 LDA (b) 2017-10-16 PCA

Figure 32: 2017-10-16: comparison of the efficiency of 2 methods for dimensionality reduction

5.13 Comparing PCA and LDA to separate the features



(a) 2017-10-23 LDA (b) 2017-10-23 PCA

Figure 31: 2017-10-23: comparison of the efficiency of 2 methods for dimensionality reduction

6 Supervision of the project of the 5 ECE students

I am in charge of supervising a project of 5 students from the ECE engineering school. They are working on the project half a day per week, and have 5 weeks of "rush- week" which means that they are fully dedicated for the project during these weeks.
Their role is: - Extract characteristic features in the neural signals -Use the Python libraries available for the analysis of neural signals and electrophysiological data Now they are in the phase of exploring the different python libraries relevant for their project and implementing the available functions. I am responsible of the supervising their project, answer their questions and help them through the challenges they encounter. I went to see them several times at their engineering school, exchanged through Skype, phone, e-mails. I enjoy the challenging part of managing a team, encourage these students to overcome problems and observe their progress. My approach is to be demanding with them, to be sure they make advances and learn from this experience, while being encouraging, positive and make sure they do not give up and appreciate the project with Microbrain Biotech.

7 Conclusion

This project sets a framework for a rigorous and complete analysis of MEA recordings of neurons. The set of parameters developed to characterize an electrophysiological signal constitute a powerful tool for an in depth characterization of a neural signal, as well as a valuable database to measure the synaptico-toxicological effects as well as the synapto-protective potentials of several pharmaceutical agents. For now, the analysis has been restricted to unilateral cortex-hippocampal neuronal networks, for the categories; spontaneous activity, amyloid β and memantine. Other neuronal networks and other pharmacological agents can be tested, with the same algorithm and a simple change of the input parameters. The analysis performed in this report covers few recordings, but as the project gets bigger and more data is generated, results will get statistically significant and relevant observation could be extracted thanks to this algorithm.

8 Perspectives of improvement

This report present the results of a 3 months project but several improvements could be added to it. This chapter summarize a couple of ideas that can be brought into this project. First, it would be useful to have a detailed and complete documentation on all the functions, for further use and improvements of future users. I discovered that too late and didn?t find time to insert it in the project, but I read that Sphinx is a good and widely use documentation generator of python projects and libraries. Sphinx Sphinx converts reStructuredText(sometimes abbreviated as RST, ReST, or reST, is a file format for technical documentation) files into pdf.
Furthermore, I would like to implement ensemble methods on the machine algorithm algorithms running on the data. The ensemble methods combine a set of several machine learning algorithms, and gives them different weights related to how they perform, in order to increase the performance of the classification.
I also would like to extend the use of the functions developed to other neuronal categories. This study focused on healthy neurons, neurons with Alzheimer and neurons with Alzheimer that received a Memantine treatment. Since the start up is also experimenting the use of other drugs as DMSO, BIC, BIA it would be interesting to analyze this data as well. BIC stand for Bicuculline, which is a pharmacological agent that has proven to increased neural network activity.This increase of activity has been noticed in previously activated electrodes, and also signals appeared on new electrodes resulting in an increased number of active electrodes.

9 Bibliography

1. Deleglise, Berangere, et al. *β-amyloid induces a dying-back process and remote trans-synaptic alterations in a microfluidic-based reconstructed neuronal network.* Acta neuropathologica communications 2.1 (2014): 145.
2. Seok J, Warren HS, Cuenca AG, Mindrinos MN, Baker HV, et al. 2013. Genomic responses in mouse models poorly mimic human inflammatory diseases. PNAS 110:3507-12
3. Suntharalingam, G. et al. (2006) Cytokine storm in a Phase 1 trial of the anti-CD28 monoclonal antibody TGN1412. N. Engl. J. Med. 355, 1018-1028
4. Hansen, S. and Leslie, R.G. (2006) TGN1412: scrutinizing preclinical trials of antibody-based medicines. Nature 441, 282
5. Dickson, M. and Gagnon, J.P. (2004) Key factors in the rising cost of new drug discovery and development. Nat. Rev. Drug Discov. 3, 417-429
6. Lamberti, M. J., &. Getz, K. (2015). Profiles of New Approaches to Improving the Efficiency and performance of Pharmaceutical Drug Development. Tufts Center for the Study of Drug Development.
7. http://www.lonza.com/products-services/bio-research/primary-cells/primary-cells-vs-cell-lines.aspx
8. https://en.wikipedia.org/wiki/Induced_pluripotent_stem_cell
9. Balijepalli, A., & Sivaramakrishan, V. (2017). Organs-on-chips: research and commercial perspectives. Drug discovery today, 22(2), 397-403.
10. Bhatia, S.N. and Ingber, D.E. (2014) Microfluidic organs-on-chips. Nat. Biotechnol. 32, 760-772
11. Haring, A. P., Sontheimer, H., & Johnson, B. N. (2017). Microphysiological Human Brain and Neural Systems-on-a-Chip: Potential Alternatives to Small Animal Models and Emerging Platforms for Drug Discovery and Personalized Medicine. Stem Cell Reviews and Reports, 1-26.
12. Huh, D. et al. (2011) From 3D cell culture to organs-on-chips. Trends Cell Biol. 21, 745-754
13. Huh, D. et al. (2013) Microfabrication of human organs-on-chips. Nat. Protoc. 8, 2135-2157
14. Benam, K. H., Dauth, S., Hassell, B., Herland, A., Jain, A., Jang, K. J., ... & Musah, S. (2015). Engineered in vitro disease models. Annual Review of Pathology: Mechanisms of Disease, 10, 195-262.
15. Park, J. W., Kim, H. J., Kang, M. W. & Jeon, N. L. Advances in micro uidics-based experimental methods for neuroscience research. Lab Chip 13, 509?521 (2013).
16. Huh, D., Hamilton, G. A. & Ingber, D. E. From 3D cell culture to organs-on-chips. Trends Cell Biol 21, 745-754 (2011).
17. Feinerman, O.,Rotem, A.&Moses, E. Reliable neuronal logic devices from patterned hippocampal cultures. NatPhys4,967-973(2008).
18. Downes, J. H. et al. Emergence of a small-world functional network in cultured neurons. PLoS Comput Biol 8, e1002522 (2012).
19. Bettencourt, L. M. A., Stephens, G. J., Ham, M. I. & Gross, G. W. Functional structure of cortical neuronal networks grown in vitro. P hys Rev E Stat Nonlin So Matter Phys 75, 21915 (2007).
20. Honegger, T., Thielen, M. I., Feizi, S., Sanjana, N. E., & Voldman, J. (2016). Microfluidic neurite guidance to study structure-function relationships in topologically-complex population-based neural networks. Scientific reports, 6.
21. http://microbrainbiotech.com/making-nerve-pathways-in-chips-for-brain-studies-royal-society-of-chemistry-read-more/
22. NeuroExplorer is a powerful data analysis program for neurophysiology.
23. http://neuralensemble.org/elephant/
24. http://neuralensemble.org/SpykeViewer/]

25. Prpper, R., & Obermayer, K. (2013). Spyke Viewer: a flexible and extensible platform for electrophysiological data analysis. Frontiers in neuroinformatics, 7.

26. http://neuralensemble.org/OpenElectrophy/

27. https://pypi.python.org/pypi/neo/

28. http://www.who.int/mediacentre/factsheets/fs362/en/]

29. https://www.alz.org/10-signs-symptoms-alzheimers-dementia.asp

30. Thal DR, Rb U, Orantes M, Braak H (2002) Phases of A beta-deposition in the human brain and its relevance for the development of AD. Neurology58(12):1791?1800

31. Glenner GG, Wong CW. Alzheimer's disease: initial report of the purification and characterization of a novel cerebrovascular amyloid protein. Biochem Biophys Res Commun. 1984;120:885?890.

32. Wong CW, Quaranta V, Glenner GG. Neuritic plaques and cerebrovascular amyloid in Alzheimer disease are antigenically related. Proc Natl Acad Sci U S A. 1985;82:8729?8732.

33. Pearson, Hugh A, and Chris Peers. ?Physiological Roles for Amyloid ? Peptides.? The Journal of Physiology 575.Pt 1 (2006): 5?10. PMC. Web. 14 Nov. 2017.

34. Serrano-Pozo, Alberto et al. ?Neuropathological Alterations in Alzheimer Disease.? Cold Spring Harbor Perspectives in Medicine: 1.1 (2011): a006189. PMC. Web. 14 Nov. 2017.

35. Terry RD (2000) Cell death or synaptic loss in Alzheimer disease.J Neuropathol Exp Neurol 59(12):1118?1119

36. Scheff SW, Price DA, Schmitt FA, DeKosky ST, Mufson EJ (2007) Synapticalterations in CA1 in mild Alzheimer disease and mild cognitive impairment.Neurology 68(18):1501?1508

37. Deleglise, Berangere, et al. "Synapto-protective drugs evaluation in reconstructed neuronal network." PloS one 8.8 (2013): e71103.

38. Yu, E. Y., et al. "Efficacy and tolerance of Memantine monotherapy and combination therapy with Reinhartdt And Sea Cucumber Capsule on agitation in moderate to severe Alzheimer disease." Zhonghua yi xue za zhi 97.27 (2017): 2091.

39. Riordan, Katherine C., et al. "Effectiveness of adding memantine to an Alzheimer dementia treatment regimen which already includes stable donepezil therapy: a critically appraised topic." The neurologist 17.2 (2011): 121-123.

40. Epperly, Ted, Megan A. Dunay, And Jack L. Boice. "Alzheimer Disease: Pharmacologic and Nonpharmacologic Therapies for Cognitive and Functional Symptoms." American Family Physician 65.12 (2017).]

41. Takahashi-Ito, Kaori, et al. "Memantine inhibits B-amyloid aggregation and disassembles preformed B-amyloid aggregates." Biochemical and Biophysical Research Communications 493.1 (2017): 158-163.

42. Gross, Guenter W., et al. "The use of neuronal networks on multielectrode arrays as biosensors." Biosensors and Bioelectronics 10.6-7 (1995): 553-567

43. Kapucu, Fikret Emre, et al. "Joint analysis of extracellular spike waveforms and neuronal network bursts." Journal of neuroscience methods 259 (2016): 143-155.

44. Defranchi E, Novellino A, Whelan M, Vogel S, Ramirez T, van Raven-zwaay B, et al. Feasibility assessment of micro-electrode chip assay asa method of detecting neurotoxicity in vitro. Front Neuroeng 2011;4:6, http://dx.doi.org/10.3389/fneng.2011.00006.

45. Bal-Price AK, Sunol C, Weiss DG, van Vliet E, Westerink RHS, Costa LG. Application of in vitro neurotoxicity testing for regulatory purposes: sympo-sium III summary and research needs. NeuroToxicology 2008;29(3):520?31,http://dx.doi.org/10.1016/j.neuro.2008.02.008.

46. Johnstone AFM, Gross GW, Weiss DG, Schroeder OH-U, Gramowski A, ShaferT. Microelectrode arrays: a physiologically based neurotoxicity testing platform for the 21st century. NeuroToxicology 2010;31(4):331?50,http://dx.doi.org/10.10

47. Novellino, Antonio, et al. "Development of micro-electrode array based tests for neurotoxicity: assessment of interlaboratory reproducibility with neuroactive chemicals." Frontiers in neuroengineering 4 (2011).

48. Uchida T, Suzuki S, Hirano Y, Ito D, Nagayama M, Gohara K. Xenon-inducedinhibition of synchronized bursts in a rat cortical neuronal network. Neu-roscience 2012;214:149?58, http://dx.doi.org/10.1016/j.neuroscience.2012.03.063.

49. Gerstner, Wulfram, et al. Neuronal dynamics : From single neurons to networks and models of cognition. Cambridge University Press, 2014.

Bibliography

50. Lisman JE. Bursts as a unit of neural information: making unreliable synapses reliable. Trends Neurosci 1997;20(1):38-43.

51. Izhikevich EM, Desai NS, Walcott EC, et al. Bursts as a unit of neural information: selective communication via resonance. Trends Neurosci 2003;26(3):161-7.

52. Van Pelt J, Wolters PS, Corner MA, et al. Long-term characteriza- tion of firing dynamics of spontaneous bursts in cultured neural networks. IEEE Trans Biol Med Eng 2004;51(11):2051-62.

53. Chiappalone M, Vato A, Tedesco M, et al. Network of neurons coupled to microelectrode arrays: a neuronal sensory system for pharmacological applications. Biosens Bioelectron 2003;18:627-34.

54. Li Y, Zhou W, Li X, et al. Dynamics of learning in cultured neuronal networks with antagonists of glutamate receptors. Biophys J 2007;93:4151-8.

55. Legendy C, Salcman M. Bursts recurrences of bursts in the spike trains of spontaneously active striate cortex neurons. J Neurophysiol 1985;53(4):926-39.

56. Cocatre-Zilgien JH, Delcomyn F. Identification of bursts in spike trains. J Neurosci Methods 1992;41(1):19-30.

57. Kaneoke Y, Vitek J. Burst oscillation as disparate neuronal proper- ties. J Neurosci Methods 1996;68:211-23.

58. TurnbullL,DianE,GrossG.Thestringmethodof burst identification in neuronal spike trains. J Neurosci Methods 2005;145:23-35.

59. Chiappalone M, Novellino A, Vajda I, et al. Burst detection algorithms for the analysis of spatio- temporal patterns in cortical networks of neurons. Neurocomputing 2005;65/66:653?62.

60. Chen, L., Deng, Y., Luo, W., Wang, Z., & Zeng, S. (2009). Detection of bursts in neuronal spike trains by the mean inter-spike interval method. Progress in Natural Science, 19(2), 229-235.

61. Tam, D. C. (2002). An alternate burst analysis for detecting intra-burst firings based on inter- burst periods. Neurocomputing, 44, 1155-1159.

62. Kapucu, F. E., Tanskanen, J. M., Mikkonen, J. E., Yla-Outinen, L., Narkilahti, S., & Hyttinen, J. A. (2012). Burst analysis tool for developing neuronal networks exhibiting highly varying action potential dynamics. Frontiers in computational neuroscience, 6.

63. Brownlee, J. (2011). Clever algorithms: nature-inspired programming recipes. Jason Brownlee.

64. Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. Computers & Electrical Engineering, 40(1), 16-28.

65. Saeys, Y., Inza, I., & Larraaga, P. (2007). A review of feature selection techniques in bioinformatics. bioinformatics, 23(19), 2507-2517.

66. Wikipedia:
https://en.wikipedia.org/wiki/Feature_selection
https://en.wikipedia.org/wiki/Dimensionality_reduction
https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction
https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
https://en.wikipedia.org/wiki/Mutual_information
https://en.wikipedia.org/wiki/Relief_(feature_selection)
http://scikit-learn.org/stable/modules/feature_selection.html

67. https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning/

68. https://en.wikipedia.org/wiki/Kernel_principal_component_analysis

69. http://www.cs.haifa.ac.il/ rita/uml_course/lectures/KPCA.pdf

70. https://en.wikipedia.org/wiki/Kernel_method

71. http://scikit-learn.org/stable/modules/lda_qda.html

72. https://en.wikipedia.org/wiki/Supervised_learning

73. http://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html

74. Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1. New York: Springer series in statistics, 2001.

75. https://en.wikibooks.org/wiki/Support_Vector_Machines

76. Wu, Xindong, et al. "Top 10 algorithms in data mining." Knowledge and information systems 14.1 (2008): 1-37.

77. Brownlee, Jason. "Machine learning mastery." URL: http://machinelearningmastery. com/discover-feature-engineering-how-toengineer-features-and-how-to-get-good-at-it (2014).

78. Naive Bayes:
https://en.wikipedia.org/wiki/Naive_Bayes_classifier
https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/
https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
http://scikit-learn.org/stable/modules/naive_bayes.html

79. https://machinelearningmastery.com/evaluate-performance-machine-learning-algorithms-python-using-resampling/

80. https://en.wikipedia.org/wiki/Cross-validation_(statistics)

81. Guinon, J. L., Ortega, E., Garca-Antn, J., & Prez-Herranz, V. (2007, September). Moving average and Savitzki-Golay smoothing filters using Mathcad. In International Conference on Engineering Education (Vol. 1, pp. 1-4).

82. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html