CentraleSupélec

université PARIS-SACLAY

Decision Support and Business Intelligence

Master Thesis

Maria Gabriela Martinez Lopera

Machine learning methods for neuronal identification and prediction from electrophysiological recordings in the cerebellum

---

prepared at the Wolfson Institute for Biomedical Research
University College London
Defended on September 2020

Advisor: Dimitar Kostadinov - University College London d.kostadinov@ucl.ac.uk
Tutor: Nacéra Seghouani - CentraleSupelec nacera.bennacer@centralesupelec.fr

Acknowledgments

I would like to express my sincerest gratitude to the Big Data Management and Analytics Erasmus Mundus consortium for allowing me to challenge myself and become a better professional throughout this master's program journey. The outcome of these two years of academic commitment is something beyond this manuscript: it is a new version of myself shaped by wider perspectives.

My special thanks to Prof. Nacéra Seghouani for helping me achieve one of my most significant dreams. Her support was essential for me to be awarded an M2 scholarship at CentraleSupélec.

My heartfelt thankfulness to Prof. Michael Hausser for welcoming me to the Wolfson Institute for Biomedical Research at University College London and making this master's thesis possible. Also, to my advisor, Ph.D. Dimitar Kostadinov, and the cerebellum team members for their supportive collaboration, feedback, and for reminding me to always strive for excellence, the top of the qualities.

I am also profoundly grateful to István Papp, PhD student at the Statistics Department of University College London, for his theoretical and methodological guidance, which were fundamental for writing the lines that follow and that describe the most interesting project I have ever worked on.

Finally, and not least important, to my family and closest friends for their emotional support and encouragement when needed the most.

Abstract: the cerebellum is a fundamental component of the vertebrate brain that is mainly responsible for physical coordination and movement learning. Its internal working mechanisms are still beyond understanding mainly due to the inability of expert neuroscientists to confidently classify observed cells in the cerebellar cortex based on their electrophysiological behavior, this is, the way in which neurons conduct electricity. Thanks to the advances in microchip technology and computational power, it is now possible to use machine learning to translate this biological problem into a multi-label classification one. In this study, neuron samples previously taken from the mouse cerebellum were processed and analyzed to feed a set of unsupervised and supervised methodologies with state-of-the-art modeling features that allowed the identification and prediction of four of the five types of cells that constitute the cerebellar cortex: Purkinje, Golgi, Granule, and Mossy Fibers, with precision levels that outperform a baseline model set according to experts' assessment.

Keywords: Cerebellar cortical cells, Cerebellar cortex, Cerebellum, High-precision classifier, Machine Learning, Neuronal prediction, Unsupervised clustering, Supervised classification

Contents

Chapter 1

Introduction

1.1 Statement of the problem

The cerebellum is a fundamental component of the vertebrate brain that is mainly responsible for physical coordination and movement learning [Bagnall 2013]. Despite being a simple and anatomically-well defined structure located inside the base of the skull, the cerebellum internal functioning remains hypothesized but mostly unknown, as its underlying mappings to the rest of the brain, models, and computations involved in the generation of its outputs are still beyond understanding [Kawato 2003].

The lack of knowledge about the cerebellum internal model behavior is mostly caused by the inability to unambiguously distinguish which neurons belong to the different cell types that constitute the cerebellar cortex shown in Figure 1.1 [Chorev 2009]. As in a handwritten digit classification problem (e.g MNIST), the number and labels of the classes are known (i.e there are five main types of neurons in the cerebellar cortex), with the difference that it is hard for a human expert to confidently classify an observed neuron in the cerebellum as belonging to one of the available classes, mainly because neurons share overlapping features that hamper their classification based on their *electro*physiological behavior [Marr 1991, van Dijck 2013].

In general, cells (not exclusively neurons) are identified according to three aspects: their morphology (i.e. their physical shape), physiology (i.e. the mechanisms they employ to carry out their functions), and molecular properties (i.e. their intrinsic features at the sub-atomic level) [Zeng 2017]. In the case of neurons, their physiological characterization if often referred to as their *electro*physiological behavior because neurons work with electrical impulses. Thus, when the term electrophysiology is introduced, it references the electrical patterns emitted by a neuron or a category of neurons. For the remainder of this study, the terms *neuron* and *cell* will be used interchangeably.

While for a long time it has been laborious for expert neuroscientists to classify cerebellar cortical cells based on their electrophysiological behavior, nowadays machine learning seems to offer a promising set of tools to help in this direction. By abstracting the underlying patterns described by the overlapping features of these neurons, the possibility of separating their classes in alternative dimensionalities is more open than ever, and with this, their prediction.

Machine learning was not always feasible to answer the posed problem though. The lack of data has been a major obstacle to support it, as collecting samples of neurons from any brain tissue requires advanced microchip technology able to capture the electrical

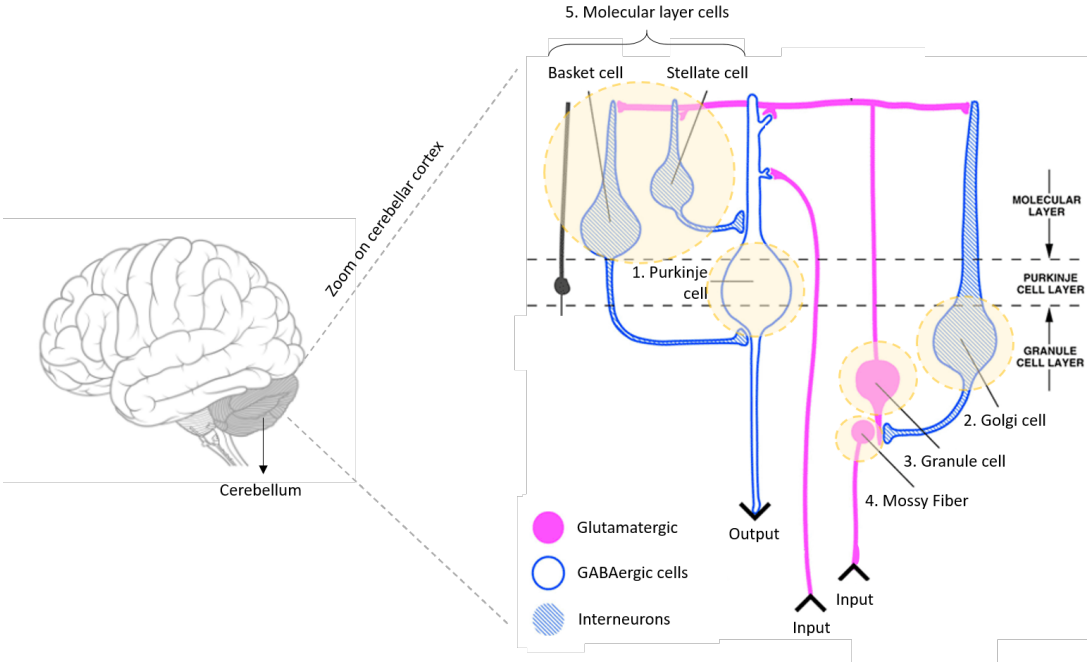

Figure 1.1: Neuron types in the cerebellar cortex. Note that the different neuron classes are found in distinct layers that are connected through a circuit. The cells of interest for this study are highlighted with yellow circles. Purkinje cells act as GABAergic, which means they diminish the effect of the input they receive from the molecular (Basket and Stellate cells) and the granular (Golgi, Granule and Mossy Fibers) layers. Purkinje's output goes directly to the deep cerebellar nuclei, which communicates an output to the rest of the brain. Granule cells and Mossy Fibers act as glutamatergic neurons, which means they stimulate the cells in the molecular layer. Basket, Stellate, and Golgi cells are considered as interneurons because they connect two different brain regions (in this case layers), enabling communication between sensory and motor neurons and the central nervous system [Purves 2001]. In this study, both Basket and Stellate interneurons were considered as a unique group of molecular layer cells. See appendix A for more theoretical details.

activity of microscopic entities [Kozareva 2020]. Fortunately, the recent availability of a high-density silicon probe technology has opened the possibility of collecting large-scale neuronal activity from thousands of neurons simultaneously [Steinmetz 2018] that can be used in machine learning classification and prediction pipelines.

In particular, this study focused on the classification and prediction of the five cell types in the cerebellar cortex, namely Purkinje cells, Golgi cells, Granule cells, Mossy Fibers, and Molecular Layer cells (MLI) (Figure 1.2), a matter that few studies have addressed up to now. To achieve this, samples from the mentioned neurons were previously obtained from the cerebellums of awake mice, whose brain structure is fundamentally the same as in humans[1].

[1] Almost all of the genes in mice share functions with the genes in humans, the reason why both species are biologically very similar. As a result, the behavior of cerebellar cortical cells in the mouse brain can be extrapolated to the same neurons in humans [Jackson Laboratory].

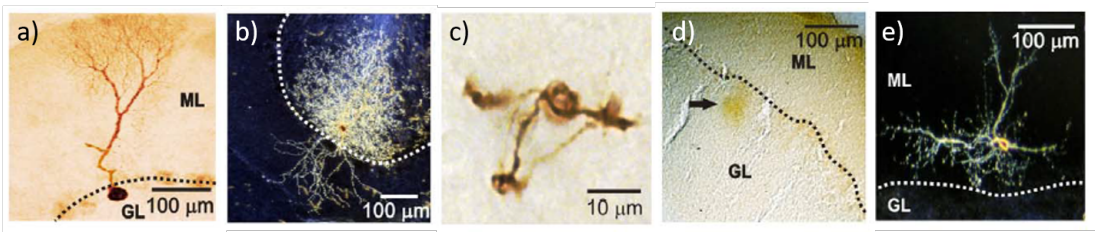

Figure 1.2: Microscopic view of the cerebellar cortical cells in the mouse brain. **a)** Purkinje cells, the most active type in terms of its electrical activity. **b)** Golgi cells, which can be understood as controllers that affect the time responses of the rest of the cerebellar neurons. **c)** Granule cells, not exclusive to the cerebellum, are the most abundant and one of the smallest types in the entire brain. **d)** Mossy Fibers, which conduct input messages between other cerebellar cortical cells. **e)** Basket/Stellate (Molecular layer) cells, which also contribute to the regulation of the effect that Purkinje cells create on the rest of the brain connections. Adapted from [van Dijck 2013]. See appendix A for more theoretical details.

1.2 Background and need

Understanding the mechanisms behind the operation of cortical neurons, not only in the cerebellum, is one of the central challenges for computational neuroscience [Malyshev 2015]. Cell-type classification and prediction projects have been key to progress in this direction, but have been developed with more intensity in other regions of the brain rather than the cerebellum [Zeng 2017]. This is the case for the retina and the cerebral cortex, two areas intensively studied in mice [Masland 2012, Sanes 2015, Shekhar 2016, Glasser 2016, Tasic 2016].

Commonly, these type of projects involve data collection *in vivo*, which means that neuronal activity is recorded directly from alive animals (either awake or anesthetized) [Zeng 2017]. *In vivo* recordings measure high-fidelity neuronal activity data that is challenging to obtain and analyze. This is mainly due to the number of variables that must be controlled during the experiment's setup and execution. Moreover, once the data is collected, it must be carefully processed, cleaned, and curated to avoid experimental biases to alter the results of any potential quantitative analysis.

In vivo recordings are performed using modern high-density electrodes (also known as probes) that can keep track of neuronal signals coming from several layers of the brain simultaneously [Steinmetz 2018]. The most recent advance in this regard was encapsulated in the new **Neuropixels** probes, which can record up to 1000 neurons in a parallel fashion setting (Figure 1.3) [Jun 2017]. For this purpose, the probes need to be properly fixed in the brain of the animals, usually mice. During the laboratory experiments, animals can be potentially stimulated to trigger the electrical response of the neurons of the area of interest[2]. These responses are encoded as signals usually known as **action potentials** or **spikes**, which must be recorded as the primary source of data to build potential cell-type classifiers.

[2]The mice used for this study were not stimulated. They stood still while their cerebellum activity was being recorded.

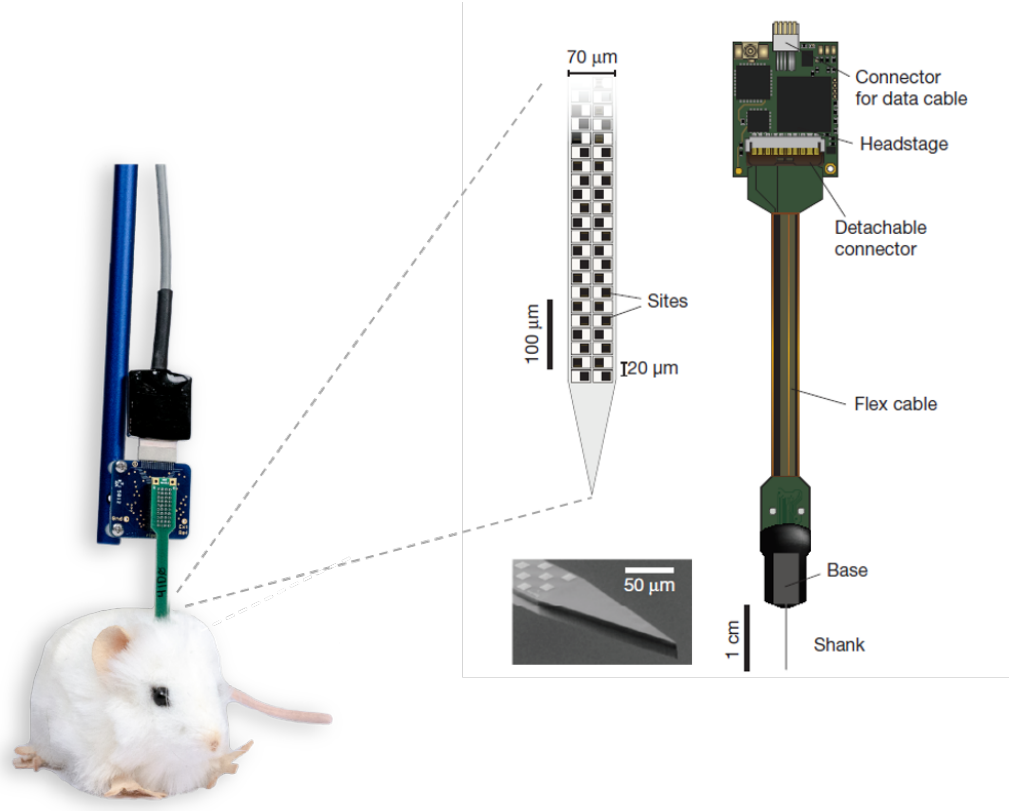

Figure 1.3: A mouse with a high-density silicon recording probe (Neuropixels) inserted. Adapted from [Steinmetz 2018].

Despite their ability to measure large densities of neurons, probes are blind to the neuron classes they record, which hinders the possibility of unambiguously classify cell samples into specific labels [van Dijck 2013]. On one hand, as *in vivo* recordings measure the real activity happening in the brain, samples of data will mix numerous electrical signals coming from neurons that are constantly and simultaneously triggering (or *firing*) electrical messages. Consequently, sorting out single neurons from the background noise is a hard task. On the other hand, when a neuron is finally isolated, it is impossible to know if it corresponds, for example, to the Purkinje, the Golgi, or any other class, which results in the absence of labeled data suitable to build a neuron classifier.

Even though there are techniques to label recorded neurons from probes, such as *optogenetics*[3], they are practically infeasible to label thousands of simultaneously recorded cells, mainly because these techniques manipulate the DNA of the target neurons, an expensive and time-consuming procedure[4] that cannot be massively applied to large and unobserved neuron populations [Deisseroth 2011]. Not limited to this, an expert neuroscientist can spend approximately 3 hours to assess the electrophysiological behavior of around 80 cerebellar cortical cells and despite his/her expertise, a conclusive decision on

[3] See section *Optogenetics: ground-truth data labeling* in Chapter 3 for more details.
[4] Takes a minimum of 5 weeks to allow the labeling of approximately 80 neurons.

the final label for all the examined neurons is not guaranteed.

1.3 Purpose of the study

The purpose of this study was to build a multi-label classifier to predict for an observed set of neurons in the cerebellar cortex, their corresponding class with higher precision than a baseline model based on experts' assessment. This classification task was performed using modeling features that explain neurons' electrophysiological behavior (also known as spontaneous electrical activity), this is, the way in which neurons conduct electrical impulses to communicate with each other. The following research questions were proposed to achieve the general goal of this study:

- Is it possible to find natural separation boundaries amongst cerebellar cortical cell classes from their spontaneous electrical activity?
- Is it possible to build a supervised classifier able to predict the five classes of cerebellar cortical cells based on their spontaneous electrical activity and with a higher precision than an expert neuroscientist?

To offer an answer to the previous concerns, more than 40 hours of previously collected and cleaned[5] *in vivo* recordings on 4 awake mice subjects were used. As part of the scope of this study, the available data was filtered to reduce laboratory-related biases and remove the effects of background noise frequencies inherent to the brain. Non-filtered out samples were considered as high-quality ones, which were then subject to a feature extraction process in which state-of-the-art modeling predictors were computed to serve as inputs for the supervised machine learning pipeline.

Initial unsupervised clustering techniques were applied to the data to confirm the existence of clusters of cells in the cerebellar cortex and to expand the number of ground-truth (labeled) samples available for supervised classification, which was very limited[6]. This approach to increase the labeled data via unsupervised learning has been reported to be effective in image classification [Peikari 2018] and was possible to implement thanks to the internal and external validation procedures performed on the obtained clusters.

The small proportions of labeled samples (already available at the beginning of the study), together with the validated discovered cells in the unsupervised clustering that were added as labeled data were still insufficient to perform robust classification on the cell classes of interest. The use of oversampling techniques, in particular, the *Synthetic Minority Over-sampling* (SMOTE) interpolation allowed the generation of artificial samples that complemented the existing labeled observations, introduced balance to the class proportions, and boosted classification performance, which was aligned with the results obtained by a variety of applications summarized by [Fernández 2018]. Subsequently, different supervised classifiers were trained to predict cerebellar cortical cells in the originally labeled data set (including the clustering additional observations) and the synthetically

[5] See section *Spike Sorting: data denoising* in Chapter 3 for more details.
[6] See section *Optogenetics: ground-truth data labeling* in Chapter 3 for the detailed numbers.

augmented one. In general, for both data sets the supervised classifiers reported cross-validated precision levels that outperformed the baseline model's, although this was not possible to confirm for the Molecular layer cells (MLI) given the limited number of labeled observations on this class. Figure 1.4 summarizes the methodological pipeline followed throughout this study.

The outline of this study is as follows: Chapter 1 presents the statement, background, and significance of the problem, as well as a summary of the purpose and contributions of this project. Chapter 2 delves into the relevant state-of-the-art and presents the theoretical framework on top of which this study relied upon. Chapter 3 goes in-depth into the details of the data at hand, filtering, and feature extraction processes. Chapter 4 presents the unsupervised and supervised methodologies used to achieve the purpose of this study, which concludes in Chapter 5 with an overall discussion on the main findings, limitations, and recommendations for future research.

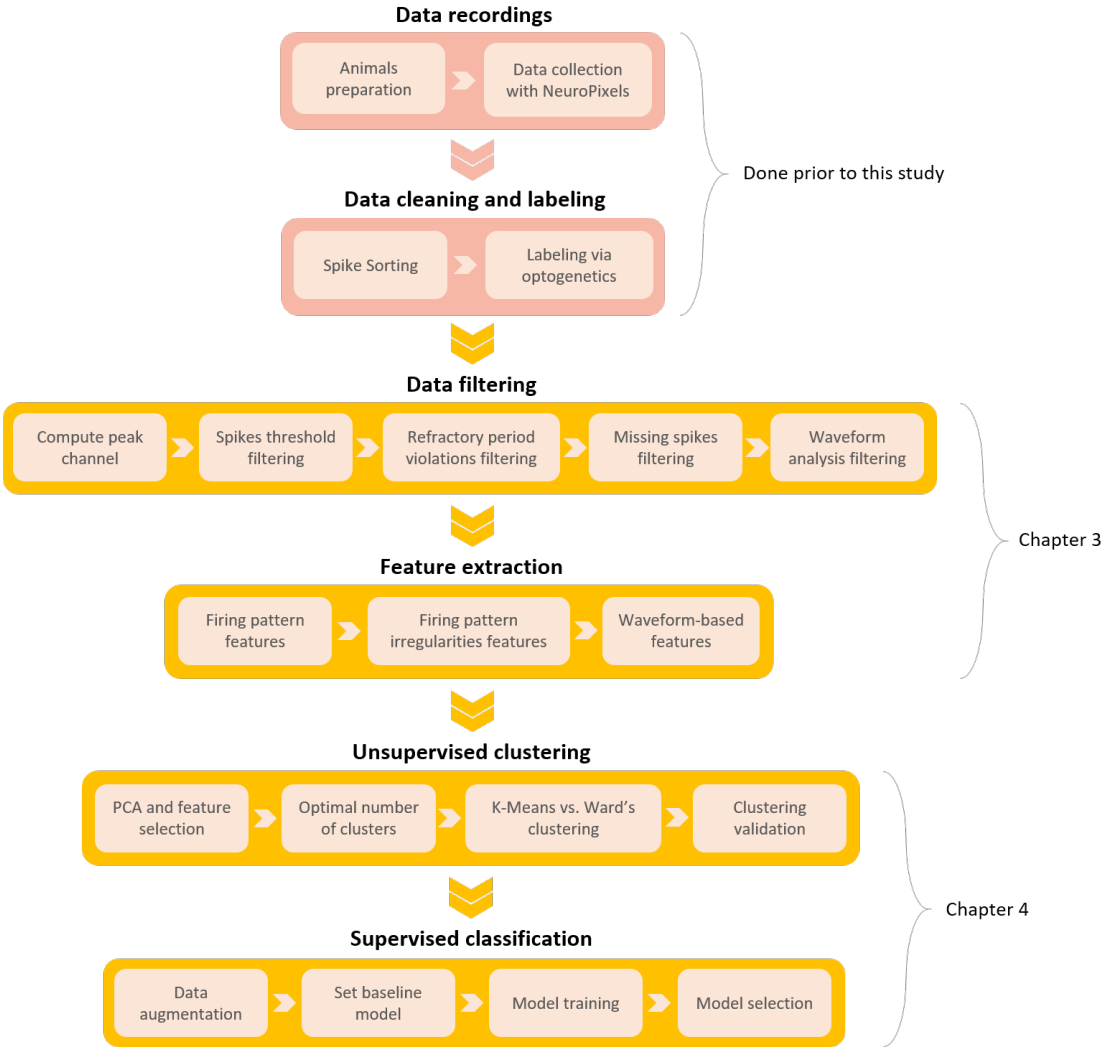

Figure 1.4: Study pipeline followed to answer the proposed research questions.

A high-precision classifier to predict cerebellar cortical cell classes opens invaluable possibilities to verify the hypothesized behavior of the internal circuit of the cerebellum, which is also a starting point to understand how this subsystem communicates with other brain regions. Moreover, this may have general applications to other brain regions whose cell types may also need to be classified to improve their understanding.

1.4 Contributions

The specific contributions of this study are summarized as follows:

- The design and implementation of a pre-processing and quality-check pipeline to select reliable samples out of simultaneous neuronal data recordings subject to inherent brain noise artifacts. A detailed explanation of this is presented in Chapter 3 under the section *Data preparation and processing*.
- The simultaneous use of two types of modeling features to explain the electrophysiological behavior of the five types of cerebellar cortical cells of interest of this study: those based on neuron spikes' electrical properties[7] and those based on their spikes' waveforms[8]. Details on the computed predictors are presented in Chapter 3 under the section *Feature extraction*.
- The application and validation of unsupervised clustering techniques to discover natural separation patterns amongst cerebellar cortical cells recorded from awake mice, an experiment without precedent in the related work. The entire pipeline is presented in Chapter 4 under the section *Unsupervised clustering*.
- The proposal and creation of a baseline model that served to assess the performance of supervised trained classifiers. Details are expanded in Chapter 4 under the section *Baseline model*.
- The proposal, application, and evaluation of a supervised learning methodology to approach a multi-label classification problem to predict cerebellar cortical cell classes. Details on the entire pipeline are presented in Chapter 4 under the section *Supervised classification*.

1.5 Significance to the field

Cell classification is not an end in itself for neuroscience, but a valuable mean to facilitate the understanding of how the brain works and how its internal mechanisms react to disease [Zeng 2017]. The identification and categorization of cells assist in the fulfilling of specific goals of the field, in particular:

- *Reproducibility*: learning to accurately classify cerebellar cortex neuron types enables their identification in a reproducible way regardless of the times, laboratories,

[7] See section *Modelling features based on spikes' electrical properties* in Chapter 2 for more details.
[8] See section *Modelling features based on spikes' waveforms* in Chapter 2 for more details.

and conditions [Zeng 2017]. This is also true for general cell classification in other parts of the brain and living organisms since neuroscientists should be able to study the same cell types under different settings and in repeated ways. As a result, science fundamentals around discovered cell patterns are robust and reliable, while contradictions are minimized [Crick 1999]. Reproducibility is key to find real cell classes that are not resulting as such due to laboratory or study-related biases.

- *Genetic access*: in general, when new cell types are discovered through classification projects, genetic access is enabled to manipulate their content via their DNA sequences, which can lead to the discovery of genes and potential processes in which they are involved (e.g. diseases) [Josh Huang 2013].

- *Discovery and evolution*: classification is the leader of discovery, which means while new cell types can be confidently identified with reproducible experiments, some others can emerge in the process. When cells have reached their plateau of discovery (i.e. they have been consistently and widely recognized by their respective research communities), a key benefit of classification is the possibility of understanding how cells are preserved during the evolutionary timeline, if they are unique to specific species and in what degree they hold the responsibility for critical differences between them and across times [Arendt 2016].

- *Understanding disease*: when the correct cell types can indistinguishably be identified, new possibilities to understand disease patterns appear [Zeng 2017]. This occurs on the premise that specific abnormalities affect only certain cell types. While these relationships are known for some illnesses (e.g. amyotrophic lateral sclerosis affects upper and lower motor neurons), many dysfunctions still have a mysterious cause or defects. Therefore, by finding valuable insights about cell types in classification disease models, groups of vulnerable neurons may be discovered [Taylor 2016, Yonehara 2016].

- *Getting a step closer to the brain's "inventory"*: as stated by Zeng & Zanes (2017), *"nowhere is the complexity of the brain more evident than in its enormous numbers of neurons and even greater numbers of synapses, both of which exhibit tremendous diversity"*. In this direction, a dimensionality reduction of the brain parts (i.e. the classification of its cells) seems to be the only way towards a better understanding of this sophisticated organ, as it is only conceivable that brain neurons can be parts of a whole rather than isolated individuals with unique function mechanisms.

1.6 Ethical considerations

This study was possible due to the collection of primary data recorded from awake male mice during the years 2018 and 2019. The corresponding animal procedures for such purpose were performed by neuroscientists at the Wolfson Institute for Biomedical Research (WIBR) subscribed to University College London (UCL) and according to the Animals Act 1986 under license from the UK Home Office and approval from the UCL Animal Welfare and Ethical Review Body.

Chapter 2

Related Work

The literature review of this study addresses a comprehensive research synthesis focused on the most relevant work done in the last decade with regard to cell classification of neurons in the cerebellum, not exclusively its cortex. The compilation of this state of the art serves as a valuable input to give shape and direction to future efforts that successfully unlock the knowledge behind cerebellar mechanisms by means of identifying the neurons that give them life.

In particular, the number of studies centered around the classification of cerebellar cortical cells is not vast. This is due to two main factors. On one side, most of the neuroscientists' attention has been focused on other brain areas such as the retina and cerebral cortex. On the other hand, many cell classification efforts in the past have been hindered by the lack of ability to collect data and laboratory-related biases [Zeng 2017].

Nowadays, best practices from electrophysiological techniques to record neuronal activity have enabled the existence of high-throughput methods to overcome most of these limitations. Proof of this is the recent availability of high-density silicon probes known as Neuropixels, that allow the recording of hundreds of living neurons simultaneously [Jun 2017], from which it is now possible to extract features that may explain their behavior, and therefore allow their classification.

Nevertheless, most of the studies presented in the following synthesis were performed with more limiting technologies (i.e. prior to Neuropixels) that severely restricted the data collection from awake animals. As a result, these research proposals had the incredible challenge of building machine learning algorithms capable of learning key relationships from small data samples that did not reflect the nature of fully living neurons (most of them used anesthetized observations). Nevertheless, a collection of useful features for cerebellar cortical cell classification emerged, which are discussed in this section and that serves as a starting point for future practitioners. Still, a key challenge remains: the extension of these pipelines to validate their results with bigger data sets recorded *in vivo* to allow more robust results.

2.1 Research synthesis

The last decade of advances in microchip technology and computational power has seen more data science techniques being used in all disciplines, and neuroscience was not the exception. The first results of this knowledge partnership appeared in the form of a new state of the art publications focused on using unsupervised and supervised learning (with

both *frequentist* and *Bayesian* approaches) to predict the classification of cell types within brain tissues, including the cerebellum and its surroundings.

This section aims to explore the work led by [Ruigrok 2011], [van Dijck 2013], [Hensbroek 2014] and [Özcan 2020], which constitute a fundamental background for the proposal of this master's thesis.

In their study *Spontaneous activity signatures of morphologically identified interneurons in the vestibulocerebellum* published in 2011, Ruigrok et al. presented one of the first machine learning applications for the classification of the cells living in one of the internal areas of the cerebellum, known as the vestibulocerebellum, which is a subsystem in charge of the balance and control of the eye movements. The cells of interest for Ruigrok et al. included three groups of neurons also at the target of this research, namely: Golgi, Granule, and Molecular layer cells. Additionally, the authors also considered the classification of Unipolar brush neurons, which are similar to the Granule ones but have been considered separately by some authors [Mugnaini 2011].

The purpose of the study presented by [Ruigrok 2011] was to build a classifier to identify neuron types in the vestibulocerebellum based on their firing properties, which tell how each kind of cell responds to an external stimulus by sending a specific electrical signature that is accompanied by the liberation of neurotransmitters to other connected neurons.

To achieve their goal, researchers collected data from 27 rats and 5 rabbits, both species **anesthetized**. From the neuronal recordings, 86 samples of correctly identified cells belonging to the types Golgi, UB, Basket, Stellate and Granule were used to build a neuron classifier for which data was approximately balanced across the different cell types ($\approx$ 17 neurons per class).

Ruigrok et al. trained a C4.5 decision tree classifier with the totality of the available data, mixing up cell samples coming from both anesthetized rats and rabbits. The curve-fitting process was made considering 48 features of interest related to the firing patterns of the cells, from which only 5 of them were used, as they were reported to maximize the overall accuracy of the classifier (Figure 2.1).

Results presented by Ruigrok et al. reported an overall accuracy (measured as positive classification rate) of 75% over the training set. Also, 2% of the cells were incorrectly classified in a different category, while a broader percentage of 23% were placed in the border category after mismatching all the decision rules imposed by the tree.

The impact of the classifier results in the field is explained by the authors of this study as a reliable and enhanced decision process towards the identification of these particular types of cells, as it provides certainty about the fact that some neurons rely on border zones and therefore cannot be effectively classified only by their firing activity-related features. Nevertheless, this is a weakly supported conclusion due to the several limitations spotted

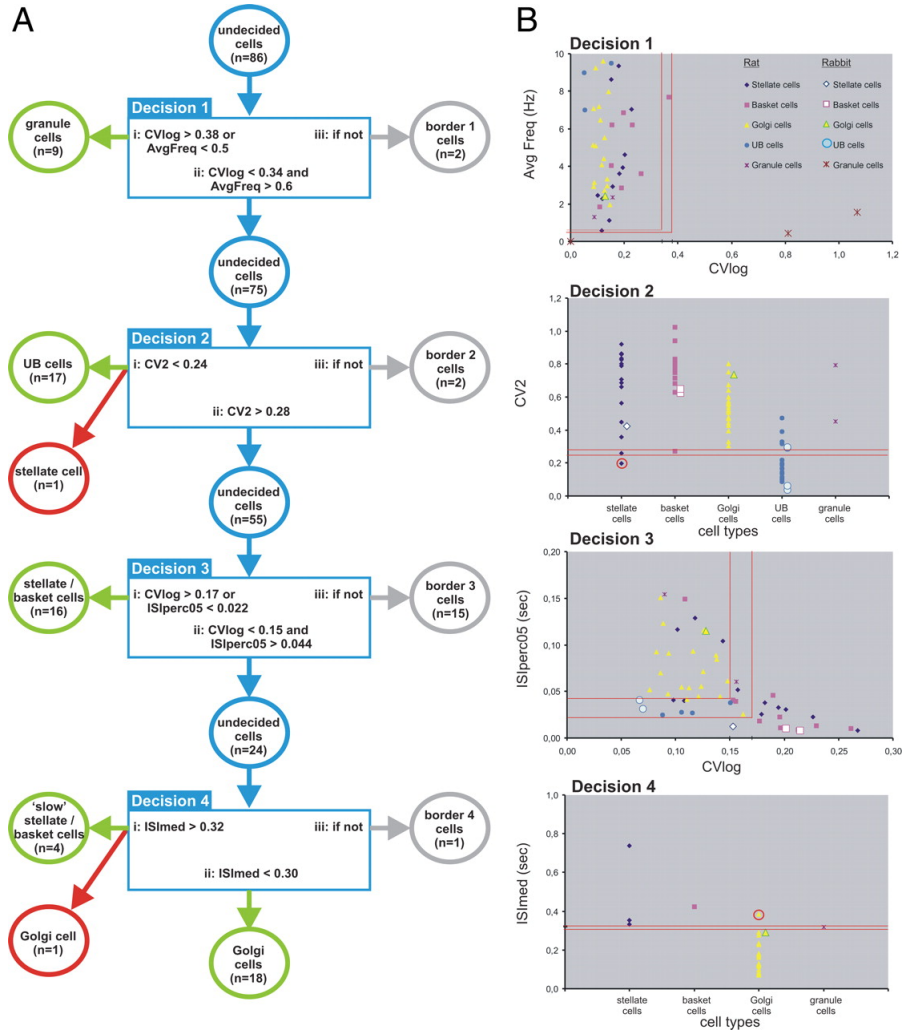

Figure 2.1: C4.5 decision tree provided by [Ruigrok 2011]. It consists of 4 decision steps that conclude individually either with the identification of the candidate cells or their allocation into a *border zone* if they cannot be classified according to existing criteria. Border cells are not carried out to the next decision steps in the tree.

in the study that hampers the generalization of its findings, specifically:

- There is no clarity on the fact that the cell samples recorded from rats and rabbits are generated from the same distribution, which could introduce additional bias and error in the classification process.
- The decision tree rules generated from this application are unlikely to be generalizable for neuron recordings in awake animals [Hartmann 2001, Santamaria 2007, Hevers 2008, Haider 2013].
- There is no evidence of the use of a methodological machine learning pipeline to support predictive robustness and avoid overfitting. Clear signs of this include the absence of holdout or cross-validation strategies to support the training phase, no hyper-parameter tuning was performed, classification metrics were not considered to

assess performance and no benchmark was proposed amongst alternative classifiers or a baseline model.

- In general, it is incorrect to only measure the performance of a classifier with the overall accuracy reported on the training set, as this leads to overfitting and poor generalization. A validation strategy needs to be put in place so that algorithms are evaluated on unseen data.
- For classification problems, alternative metrics to the accuracy are often more valuable to assess the performance of a machine learning technique. Indicators derived from the confusion matrix computation, such as precision and recall, could be considered.
- The use of small data and less robust techniques (such as decision trees) often results in highly overfitted models that are barely useful. The orthogonal decision boundary found by the fourth step of the algorithm is a proof of this (Figure 2.1). Decision trees are simple white-box techniques often preferred given their interpretability. However, they are more suitable to support the exploratory phases of a data analysis project than for prediction purposes [Fratello 2018].

A second study of relevance motivated by the previous work was published by [Hensbroek 2014] under the title *Identifying Purkinje cells using only their spontaneous simple spike activity.* The corresponding paper presented an extension of the work done by [Ruigrok 2011] to include into the classification pipeline the most important type of cells in the cerebellum: the Purkinje cells. These neurons release a neurotransmitter called GABA (gamma-aminobutyric acid) that produces an inhibitory effect on certain neurons, thereby reducing the transmission of nerve impulses. These inhibitory functions enable Purkinje cells to regulate and coordinate motor movements [Konnerth 1990]. The loss or damage to Purkinje cells has been associated to neurological pathologies such as the Huntington disease and the autism spectrum disorder [Cook 2020]. However, much of the influence of the work done by Purkinje cells is still beyond understanding, one of the reasons why the cerebellar cortex's way of working is still hypothesized.

[Ruigrok 2011] formulated a study with the purpose of training a cell classifier able to predict the neuron types in the vestibulocerebellum based on their firing properties. The settings of this research included the same data set consisting of 86 balanced samples corresponding to the types Golgi, UB, Basket, Stellate and Granule neurons, plus 110 additional Purkinje labeled cells recorded under anesthesia.

Hensbroek et al. adapted the C4.5 decision tree proposed by [Ruigrok 2011] by extending it to classify the Purkinje cells. This procedure added a decision step based on one extra feature and kept the previous rules and thresholds (Figure 2.2). In general, this is not a robust strategy to proceed when training a machine learning classifier, but an important improvement was introduced with the use of a holdout strategy to evaluate the model performance on training, testing, and validation data sets, each of approximate size ($\approx 33\%$).

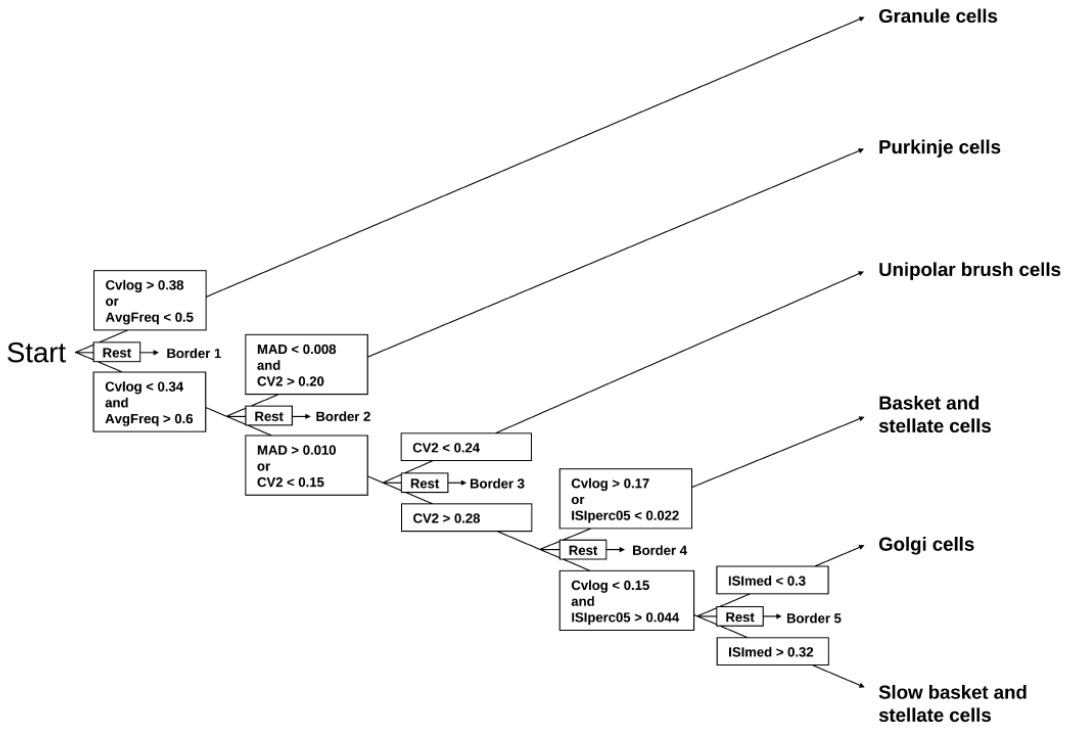

Figure 2.2: Extended C4.5 decision tree provided by [Hensbroek 2014]. The new decision step uses a combination of the MAD and CV2 statistics which are representative of the firing behavior of cells within the cerebellum.

Results presented reported overall accuracies of 78% for the training set, 81% for the testing set, and 86% for the validation set, which consisted of samples collected from awake animals (i.e. a different data distribution than the one used for training). These results suggest that a classifier trained on anesthetized neurons can be potentially generalized to predict *in vivo* observations recorded from fully awake animals. Moreover, the superior accuracy on the validation set is an indication that there was no overfitting, which is the ultimate goal of any machine learning implementation. Nevertheless, the performance of this model still needs to be tested under the light of different holdout validation thresholds (for instance 50-25-25) and alternative data sets, even coming from other laboratories' settings. Additionally, authors should not discard the future use of cross-validation, hyperparameter tuning techniques, feature selection, and model benchmarking to make their methods and conclusions more robust [Yadav 2016].

It is difficult to establish a direct comparison between the results presented by [Ruigrok 2011] and [Hensbroek 2014] due to the following reasons: i) Ruigrok et al. did not use a holdout validation strategy to report performance metrics, and as such, any figures deducted from the training set are not indicative of the real performance of a machine learning algorithm. ii) Even when the training performances could be compared for the two studies (75% vs 78%), this would not be a fair comparison, as the decision tree settings were completely different both in the number of samples (86 vs. 196) and the number of classes (Purkinje cells were only introduced later). As a result, it is not possible to tell

if the work presented by [Hensbroek 2014] does, in fact, improve the classification efforts made by [Ruigrok 2011], even though the methodology followed by Hensbroek et al was, in general, more robust, which does not mean there is no opportunity for improvement.

A third study of great relevance was presented by van Dijck et al. in their paper *Probabilistic Identification of Cerebellar Cortical Neurones across Species* [van Dijck 2013]. Its purpose was to develop a classification algorithm to unambiguously identify the main five types of cerebellar cortical cells based on their firing properties (i.e. electrical responses). The electrical signatures of these neurons were used not in a frequentist machine learning approach as in the studies presented by Ruigrok et al. and Hensbroek et al., but to build a Bayesian classifier based on Gaussian processes, which led to the estimation of probabilistic outputs over the class membership of every neuron.

To achieve the previous, the authors collected data from anesthetized rats and mice, and decerebrate cats, but the Gaussian Process Classifier was only fitted with the rats data set consisting of 120 observations and using a Radial Basis Function (RBF) kernel, following the ideas presented by [Girolami 2006] that suggest the use of a variational treatment for multi-class classification through the use of Gibbs sampling from the parameter posterior.

To fit the probabilistic model, van Dijck et al. used leave-one-out cross-validation and the overall reported accuracy was 99.2%. The authors also reported the need for replication of this experiment on awake animals, in which cerebellar cortical cells are likely to present more active firing rates [Chadderton 2004, Barmack 2008, Holtzman 2011].

van Dijck et al. provided a comparison of their accuracy levels with the ones achieved by a C4.5 decision tree such as the one proposed by [Ruigrok 2011] and [Hensbroek 2014], for a subset of samples. Figure 2.3 summarizes the results. In general, the probabilistic model, both un-thresholded (i.e. with a cutoff of 50% to consider a cell belonging to a particular class) and thresholded at 70% performed better than the tree-based method to correctly classify the neurons of interest. Note that the use of the Gaussian Process Classifier allowed a probabilistic comparison that is more informative than simple discrete predictions. For instance, it was possible to tell from Figure 2.3 that 4 Golgi cells were predicted to be as such type with a probability comprehended between 50% and 70%. Eventually, those borderline cells could be examined to understand what potential differences they might have with confidently identified cells. Findings could lead to useful insights into the membership properties of particular cell types.

Results presented by [van Dijck 2013] do not necessarily suggest that probabilistic models automatically lead to better performance for classification, as the overfitting effect of these models on their data cannot be discarded with the available results. It is true, however, that those probabilistic models allow a more careful assessment of the outputs than black-and-white predictions, as membership probabilities of 50% can significantly differ from those around 70% or over. Also, naturally probabilistic techniques tolerate outliers better, missing values, and unnormalized data, while non-probabilistic approaches usually

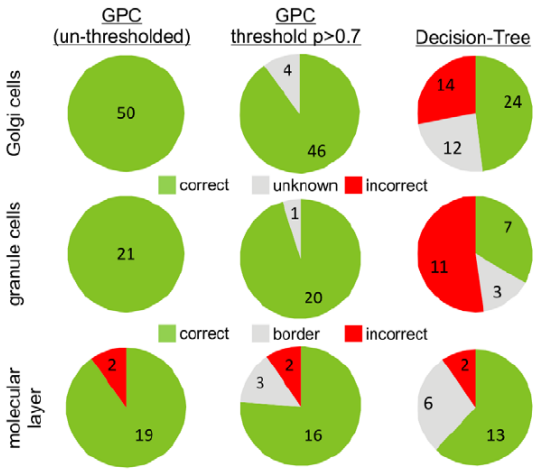

Figure 2.3: Accuracy comparison between the probabilistic model proposed by van Dijck et al. and a C4.5 decision tree as the one proposed by [Ruigrok 2011] and [Hensbroek 2014]. Gaussian Process Classifiers performed better for this multi-label classification problem. Taken from [van Dijck 2013].

report more efficient processing times. With the recent advancements in deep learning and computational processing, these model methodologies are more likely to collaborate rather than compete [Bzdok 2017].

As in the previous studies synthesized in this section, the main limitation of [van Dijck 2013] has to do with the extension of this model to data collected from awake animals. The authors trained a model, on anesthetized samples, and tested it on 147 samples from awake monkeys. Figure 2.4 presents the separation boundaries found for each cell class, which shows a lack of generalization of the model. Only 2 out of 4 cell types got to be identified and they do not seem not to been highly isolated from each other.

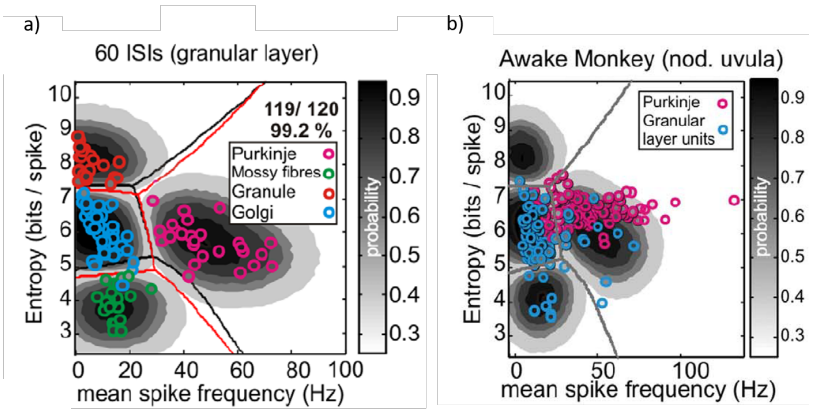

Figure 2.4: Clusters of cerebellar cortical cells identified by the Gaussian Process Classifier proposed by van Dijck et al. **a)** four types of cortical cells identified in the data set collected from anesthetized rats. **b)** target cells identified in the awake population of monkeys. Taken from [van Dijck 2013].

2.2 Relevant features for neuron classification

The works presented by [Ruigrok 2011, Hensbroek 2014, van Dijck 2013] were highly contributing to define an exhaustive set of features to extract as much information as possible from the electrophysiological activity (i.e. the electrical or firing pattern) of neurons. Those variables are described in detail in Chapter 3, as they were also considered and computed by this study. This section introduces the necessary intuition to understand how these and other additional modeling predictors proposed in the related work are derived.

2.2.1 What is a neuron spike or action potential?

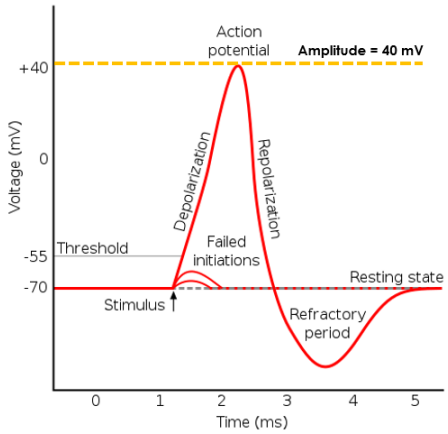

Figure 2.5: The action potential process in a neuron. External stimuli generate a depolarization of the membrane potential, which opens to allow positively charged ions inside the neuron and negatively charged ions out, once an electrical threshold is surpassed. This process generates a rapid increase in the positive charge of the entire cell (depolarization), followed by a repolarization phase, whose purpose is to restore the resting membrane potential. Before reaching the resting state, the neuron experiments a refractory period in which it is impossible that new spikes or action potentials are generated again (i.e. the neuron cannot be stimulated during this interval) [Bean 2007].

To understand how the firing pattern of a neuron works, it is essential to comprehend the concept of *neuron spiking* [Dorval 2011]. Neurons communicate with each other through a process known as *synaptic transmission* by releasing neurotransmitters, a chemical response provoked by an electrical gradient that occurs within the synaptic terminals[1]. Neurotransmitters serve as a stimulus to the cells that neighbor the emissary neuron, which, as a result, experiments a sudden change of electrical voltage often called *spike* or *action potential*. At the same time, a spike generates a new chemical release that continues its course through various chains of neurons (known as neural pathways) in the brain. A schema of an action potential is presented in Figure 2.5, and every time one is generated, it is said that the neuron "spiked" or "fired" and action potential, which at the same

[1]See section *Synaptic transmission* in appendix A for more details.

time has an associated *amplitude* that corresponds to the electrical voltage that the spike registered.

2.2.2 Modelling features based on spikes' waveforms

The most recent relevant study for the current research synthesis was presented by [Özcan 2020] in the paper *Differential Coding Strategies in Glutamatergic and GABAergic Neurons in the Medial Cerebellar Nucleus*. Even when it was not centered in the cerebellar cortex but rather in its nucleus, the study proposed a set of features to perform cell classification based on the waveforms described by their spikes or action potentials. The features used by the study are presented in Figure 2.6.

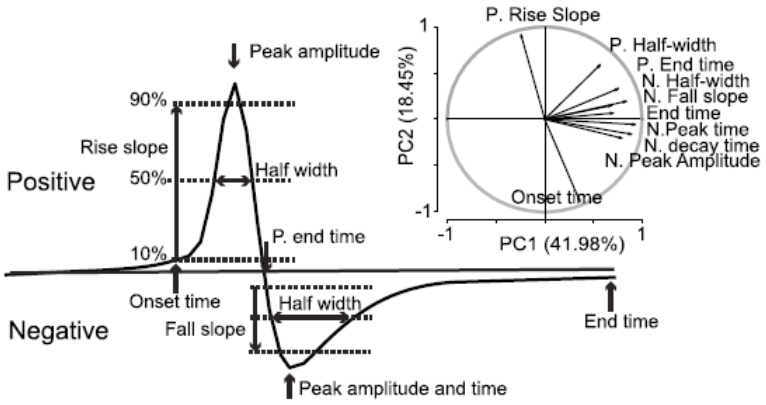

Figure 2.6: Waveform features proposed by [Özcan 2020]. These features characterize the peak and trough amplitudes registered by all the action potentials generated by a neuron, as well as other related metrics that are measured directly from the waveform described by the spikes.

The purpose of the study proposed by [Özcan 2020] was to identify potential neuron types in the nucleus of the cerebellum. To achieve this, the authors recorded 39 neurons from 13 male mice *in vivo*. The features from the waveforms of the action potentials described by the neurons of interest were computed and reduced via Principal Component Analysis (PCA), on top of which hierarchical unsupervised clustering was applied.

Results showed that two clusters were formed with approximately 50% of samples each, which is a highly trivial solution, yet validated by the authors. To confirm how sensible and realistic were the found clusters, the authors explored statistically significant differences between the groups' features using a Wilcoxon signed-rank hypothesis test. Figure 2.7 summarizes the process followed by the study.

Waveform-based predictors are of interest for future work in neuron classification, as they were reported to allow a very refined separation of cell classes, not only in the results achieved by [Özcan 2020] in the cerebellar nucleus, but also by [Jia 2019] in their recent study *High-density extracellular probes reveal dendritic backpropagation and facilitate neuron classification*, in which two types of neurons (regular-spiking and fast-spiking)

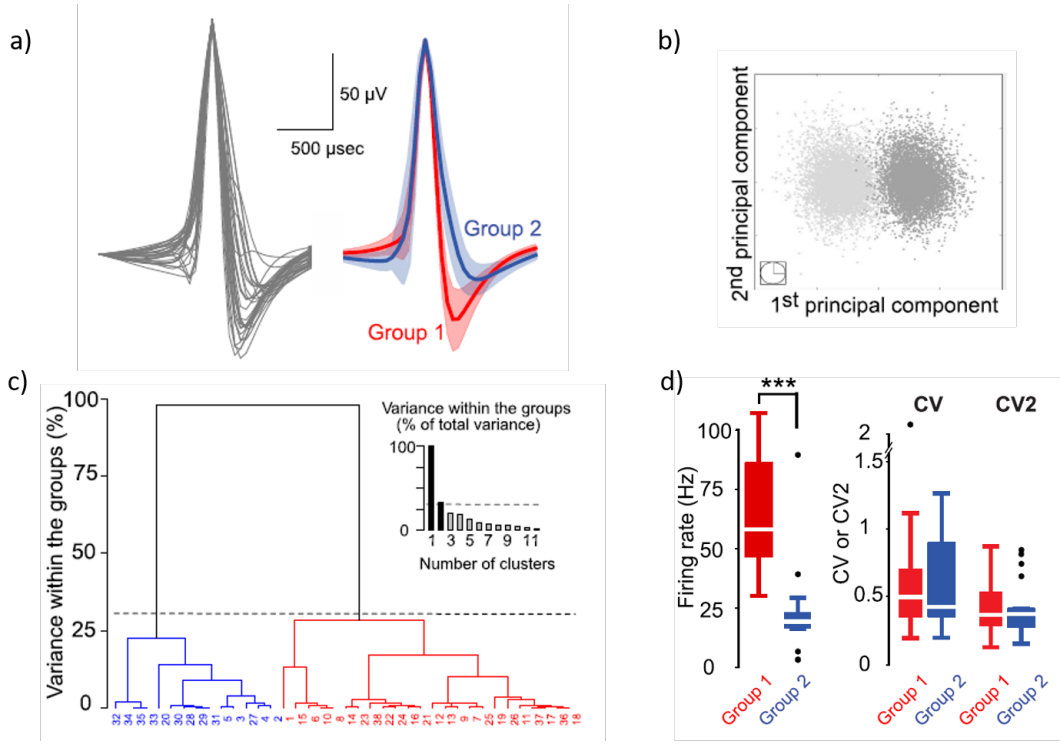

Figure 2.7: Evaluation of clustering results presented by [Özcan 2020]. **a)** Averaged spike waveforms for the two cell types found in the cerebellar nucleus. **b)** Projection of the spike waveforms of the recorded cells against the first two principal components. **c)** Ascending hierarchical tree showing the two clusters of cells identified via the elbow method. **d)** Box-plot of firing rates, CV and CV2 (all these firing properties of a neuron) for the two cell types. Wilcoxon rank test allowed the acceptance of the alternative hypothesis on the variable Firing Rate, indicating that the two groups of neurons had a statistically significant difference in their spiking behavior ($p = 1.4 \times 10^{-7}$. However, this was not the case for the rest of the features, for which p-values were 0.37 (CV) and 0.36 (CV2) [Özcan 2020].

were identified in the mammalian visual cortex via unsupervised clustering on top of the orthogonalization of the waveform features presented in Figure 2.8.

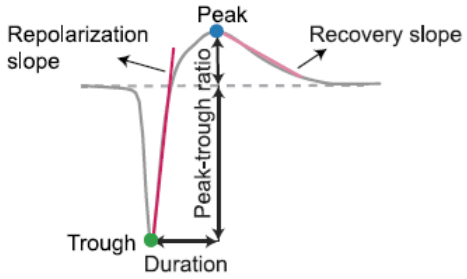

Figure 2.8: Waveform features proposed by [Jia 2019] to identify cell classes in the visual cortex of the mouse brain. Note that the authors from this study are not only interested in measuring the peak and the trough of the spikes, but also their duration and the slopes that characterize their firing and stabilization over time.

2.2.3 Modelling features based on spikes' electrical properties

Each neuron type has a unique signature of electrical (also firing or spiking) properties that they manifest in every spike or action potential generated. It is known, for example, that Purkinje cells in the cerebellar cortex are highly *excitable* neurons, which means they *spike* or *fire* very often, approximately at 60 Hz (i.e spikes per second), as a consequence of the high-throughput of messages received from other cells and their biophysical properties [Holtzman 2006]. This is not the case for the majority of the neurons in the brain, which fire at rates between 0.1-2Hz on average. This firing frequency is a special electrical property that has allowed the identification of cell types in the brain, such as the Purkinje one, and as such, a broader group of electrical variables is often considered to find more about the firing behavior of clusters of neurons.

For the purpose of this study, spikes' electrical properties were considered as in the works presented by [Ruigrok 2011, Hensbroek 2014, van Dijck 2013] because machine learning neuron classifiers need to be able to learn how these electrical features actually differentiate cell classes in the brain. Note that these electrical properties are unique to the neurons and any task to identify and classify cells from other body tissues would not consider them.

2.2.4 Inter-spike interval histograms

As stated, the electrical properties of neurons can represent a multidimensional collection of modeling features that can eventually differentiate cell types. Those predictors are usually extracted from the *inter-spike intervals* of a neuron, which represent **the times between consecutive spikes or action potentials** (Figure 2.9), usually measured in seconds (s) or milliseconds (ms).

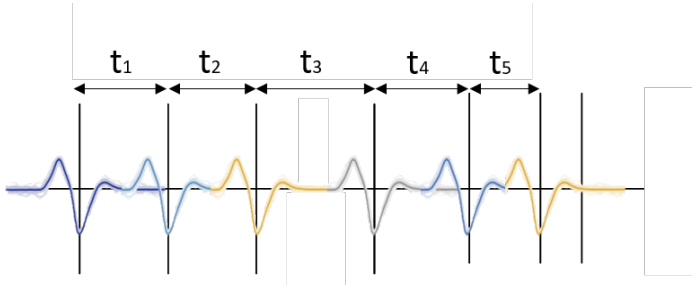

Figure 2.9: Schema of consecutive spikes fired by a neuron and the inter-spike time intervals between them (t_1, t_2, t_3, t_4, t_5).

These intervals are represented in the form of a histogram known as the *inter-spike interval histogram* (ISIh) or the *first-order interval histogram* as illustrated in Figure 2.10 [Stein 1965, Dorval 2011, Jeffery 2018]. Note that the term *first-order* makes reference to the fact that each spike is compared only with its adjacent (consecutive) spikes, which

reveals simple temporal relationships that may be of interest[2]. For example, the first bar in the histograms from Figure 2.10 represents the time difference between the second and the first spikes recorded. If this differencing chain is continued, an overall of $TotalSpikes - 1$ inter-spike intervals are obtained and can be consecutively visualized.

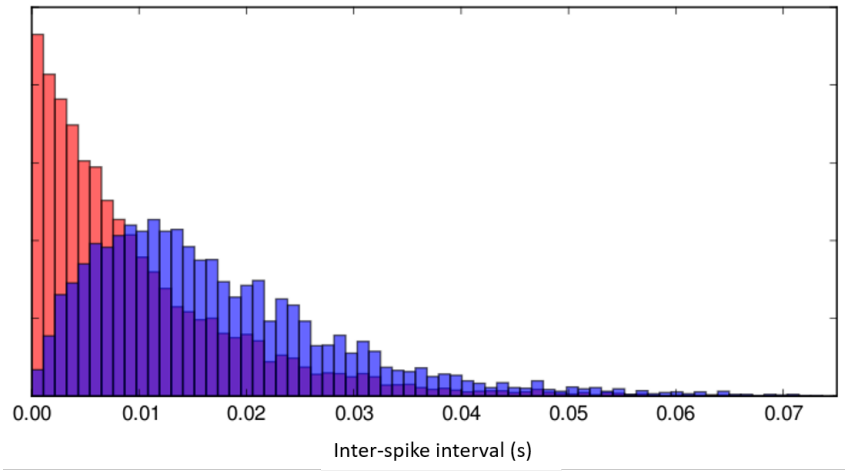

Figure 2.10: Histograms of the inter-spike intervals, in seconds, for two different neurons. Note that the red histogram represents differenced inter-spike intervals that follow an exponential probability distribution, while the blue ones describe a log-normal behavior. The underlying probability distribution attached to these time intervals is the source of many features of interest that explain the electrical properties of a neuron through the modeling of its spikes.

Spikes or action potentials can be seen as discrete events that happen as often as modeled by a random variable supposed to follow a Poisson process [Heeger 2000], which indicates that the time intervals between them should be modeled by an exponential distribution. Even though this is what the theoretical formulation suggests [Song 2018], there seems to be no consensus on what distribution the inter-spike histograms should follow, and they are often modeled as to originate from a variety of unimodal positively skewed distributions such as the log-normal [Kish 2015], the exponential [Song 2018], and the beta [Tsubo 2012, Toth 2018]. Per expert consensus and visualization of the inter-spike interval histograms described by the cerebellar cortical neurons at the hand of this study, the log-normal distribution was considered as the best one to approximate.

Inter-spike intervals are useful to compute a variety of features widely reported to characterize neuron firing patterns and their natural irregularities. These features mostly come from statistical measures that result from the modeling of the distribution described by the inter-spike interval histograms. For example, when examining Figure 2.10 it is possible to observe that the two ISIh's presented actually correspond to different probability distributions, which means these two processes are describing neurons with different

[2] An alternative full view on the multiple temporal periodicities between spikes is captured by an autocorrelation histogram, which shows patterns of interest beyond time intervals, for instance, how oscillatory the spikes of a neuron are [Jeffery 2018]. Details on this are presented in section *Autocorrelograms and refractory period violations* in Chapter 3.

behaviors, a reality that is likely to be revealed by some of their statistical measures of central tendency (i.e. the mode, the median, and the mean).

As part of its contributions, this study proposed the simultaneous consideration of both electrical and waveform-related modeling features to analyze the separability between the cerebellar cortical cell classes and provide an answer to the research questions posed, as precedent studies have not aimed to perform this analysis and rather have opted for using these two types of predictors independently.

In general, waveform-related predictors are worth considering because they are less prone to suffer from measurement irregularities during the data recording process (e.g electrical properties such as the average firing frequency can result severely distorted if for example mice move suddenly during the data collection). Additionally, it is encouraging to find whether these features can lead to a better separation of the cerebellar cortical cell classes, as there is no precedent study that has attempted cell classification in this tissue considering these variables.

2.3 Research synthesis main highlights

The main highlights that emerge from a comprehensive revision of the related work of this study are:

- The few studies that have addressed the cell classification problem of the neurons in the cerebellar cortex have done it mostly considering anesthetized data. To date, there is no register of any study in the state of the art that uses data samples from awake animals to classify and predict cerebellar cortical cells.

- Moreover, none of the relevant studies presented in this research synthesis has considered the use of waveform-based features to perform classification tasks in the cerebellar cortex, but rather in other brain structures such as the cerebellar nucleus and the visual cortex.

- One of the contributions of this study aims to conclude if the use of waveform-based predictors complement the traditional electrical/firing-pattern-based variables to improve the separation of cell classes within the cerebellar cortex.

- The studies presented in this synthesis have an opportunity for improving the robustness and generalization of their implemented machine learning pipelines.

- A common gap in the presented studies reveals that their machine learning implementations are never compared against a baseline model in terms of any classification quality metric. Experienced neuroscientists are capable of classifying neuron types based on simple decision rules, and any algorithm built for this very same purpose must offer multi-label classification capabilities that surpass in quality and/or speed the criteria of human expertise.

The following section of the related work offers a comprehensive view of the theoretical fundamental aspects of Component Analysis and both unsupervised and supervised learning methodologies and their main aspects.

2.4 Fundamentals of Component Analysis

2.4.1 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique originally formulated by Karl Pearson in 1901[3]. It consists of the application of an orthogonal transformation to a data set in order to project it into a lower-dimension linear space. As a result, a new coordinate system is generated such that the highest variance of the data is explained by the first coordinate (also known as the first principal component), the second-highest variance on the second coordinate, and so on [Pearson 1901, Jolliffe 2002].

The intuition behind PCA is the fitting of an ellipsoid of m dimensions on the data. Each axis of the ellipsoid represents a principal component such that if it is small, then the variance along that axis is also small, and therefore that component is not very useful to explain an important fraction of the information contained in the data. From the dimensionality reduction perspective, PCA exploits correlations between dimensions with the implicit hope that the variance along a small number of principal components provides a reasonable characterization of the complete data set, which is highly recommended to avoid the curse of dimensionality in most machine learning applications [Shlens 2014].

2.4.1.1 PCA via eigenvector decomposition

Consider a data set X, an initial matrix mxn, such that m is the number of dimensions and n is the number of samples. The goal of PCA is to find an orthonormal matrix P such that $Y = PX$, where C_Y, the covariance matrix of Y is a diagonal matrix, and the rows of the matrix P are the principal components of X.

The covariance matrix C_Y can be rewritten in terms of the unknown variable to compute P:

$$\begin{aligned}\mathbf{C_Y} &= \frac{1}{n}\mathbf{Y}\mathbf{Y}^T \\ &= \frac{1}{n}(\mathbf{PX})(\mathbf{PX})^T \\ &= \frac{1}{n}\mathbf{P}\mathbf{X}\mathbf{X}^T\mathbf{P}^T\end{aligned}$$

[3]PCA is also known under alternative names. For example, the discrete Karhunen–Loève transform (KLT) in signal processing, proper orthogonal decomposition (POD) in mechanical engineering, the Hotelling transform in multivariate quality control, singular value decomposition (SVD) [Van Loan 1983], eigenvalue decomposition (EVD) in linear algebra, the Eckart–Young theorem [Harman 1960], empirical orthogonal functions (EOF) in meteorological science, empirical eigenfunction decomposition [Sirovich 1987], empirical component analysis [Lorenz 1956], and empirical modal analysis in structural dynamics.

$$= \mathbf{P}\left(\frac{1}{n}\mathbf{X}\mathbf{X}^T\right)\mathbf{P}^T$$
$$\mathbf{C_Y} = \mathbf{P}\mathbf{C_X}\mathbf{P}^T$$

Which results indirectly in the estimation of the covariance matrix of the data set X, namely, C_X.

The bidirectional *spectral theorem* suggests that **a matrix A is symmetric if and only if it is orthogonally diagonalizable**. This means if there exists an orthogonal matrix S such that S^TAS is diagonal, it is also true that **a matrix is orthogonally diagonalizable if and only if it is symmetric** (due to the bidirectional nature of the theorem)[Halmos 1963]. As such:

$$\mathbf{A}^T = \left(\mathbf{E}\mathbf{D}\mathbf{E}^T\right)^T = \mathbf{E}^{TT}\mathbf{D}^T\mathbf{E}^T = \mathbf{E}\mathbf{D}\mathbf{E}^T = \mathbf{A}$$

Moreover, it is also true that **a symmetric matrix can be diagonalized by a matrix of its orthonormal eigenvectors**, namely, the matrix E, whose columns correspond to the eigenvectors of A, while the matrix D is the diagonal matrix introduced [Shlens 2014].

The wanted matrix P can be chosen such that $P \equiv E^T$. By means of the linear algebra theorem that suggests that **the inverse of an orthogonal matrix is its transpose** ($P^{-1} = P^T$), it becomes evident the chosen matrix P diagonalizes C_Y:

$$\begin{aligned}
\mathbf{C_Y} &= \mathbf{P}\mathbf{C_X}\mathbf{P}^T \\
&= \mathbf{P}\left(\mathbf{E}^T\mathbf{D}\mathbf{E}\right)\mathbf{P}^T \\
&= \mathbf{P}\left(\mathbf{P}^T\mathbf{D}\mathbf{P}\right)\mathbf{P}^T \\
&= \left(\mathbf{P}\mathbf{P}^T\right)\mathbf{D}\left(\mathbf{P}\mathbf{P}^T\right) \\
&= \left(\mathbf{P}\mathbf{P}^{-1}\right)\mathbf{D}\left(\mathbf{P}\mathbf{P}^{-1}\right) \\
\mathbf{C_Y} &= \mathbf{D}
\end{aligned}$$

As a result, the principal components of X can be written in terms of its eigenvectors and its covariance matrix:

- The principal components of the matrix X are the eigenvectors of its covariance matrix $C_X = \frac{1}{n}XX^T$.
- The i^th diagonal value of the covariance matrix C_Y is the variance of X along p_i.

In practice, to compute PCA via the eigenvector decomposition the following procedure is performed:

1. *Mean subtraction*: for every dimension m, the data must be centered around the origin by subtracting the vector mean from every observation n.
2. *Compute covariance matrix*: a square matrix containing the covariances between every pair of the m dimensions shall be constructed. The covariance is a measure of

the joint variability between two random vectors, such that if it is positive, higher values in one of the vectors correspond to higher values in the other [Rice 2006]. The diagonal of the covariance matrix contains the variances (the covariance of each element with itself).

3. *Compute the corresponding eigenvalues and eigenvectors of the covariance matrix*: the eigenvectors represent a simplified version of the original set of m dimensions. This uncorrelated linear combination of features has some associated eigenvalues, which happen to indicate the amount of variance explained by each dimension vector.

4. *Eigenvector normalization*: orthogonal eigenvectors need to be converted into unit vectors by means of their normalization. At this point, they are representatives of the axes of a fitted ellipsoid on the data.

5. *Compute explained variance*: for every component (orthogonal eigenvector), the proportion of variance it explains is given by the relation of its eigenvalue and the sum of all the eigenvalues corresponding to all the eigenvectors.

PCA can also be solved via Singular Value Decomposition (SVD). In fact, those two names are often used interchangeably due to their relatedness to explain the concept of *change of basis*. In summary, PCA entails mean subtraction for each dimension and the computation of either their SVD or the eigenvectors of the covariance matrix C_X.

2.4.2 Limitations

PCA is a procedure highly sensitive to data scaling/normalization, as the computation of the covariance matrix expects zero-centered dimensions. This is a limitation for its use in some applications in which mean subtraction is harmful. Moreover, PCA will fail to produce a good representation of the data if:

- Its underlying distribution is non-Gaussian.
- Its principal components are not orthogonal or linear.
- Large variances are not predominant (as low variances will be interpreted as noise).

To overcome many of these limitations, some methods do exist. For instance, the parametric application of PCA, often referred to as *kernel PCA*, can be used to include prior knowledge about any non-linearities known beforehand in the data (such in the case of Figure 2.11). Also, to remove high-order dependencies in the data, a common approach is the Independent Component Analysis (ICA), which aims to find not only reduced but also statistically independent dimensions in the data. ICA has demonstrated high success in the fields of image and signal processing [Shlens 2014].

2.4.3 Zero-Phase Component Analysis

Data whitening is an extended version of the Principal Component Analysis, and highly related to Independent Component Analysis as well. While it also focuses on finding uncorrelated linear eigenvectors to explain the dimensionality of a data set, it also ensures

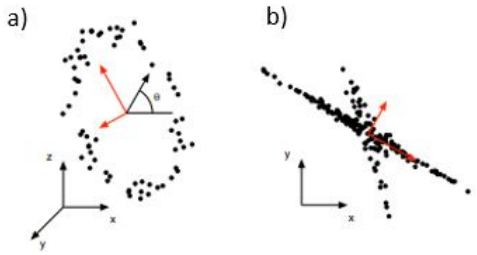

Figure 2.11: Examples of data distributions for which PCA fails to capture the correct components. **a)** A non-linear kernel describes the data. **b)** PCA on this data set extracts the first two principal components in the wrong direction. Taken from [Shlens 2014].

that the variance of such zero-centered vectors is equal to the identity matrix, case in which white noise is obtained.

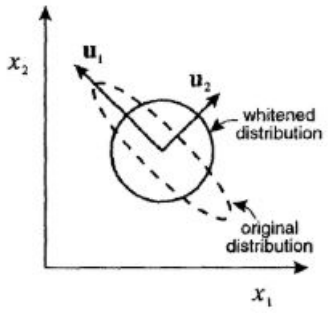

Figure 2.12: Whitening transforms a distribution using its eigenvectors in such a way that its covariance matrix becomes the unit matrix. Taken from [Bishop 1995].

A potential downside of this technique is that it reduces the importance of larger variances in the data and may lead to the overestimation of irrelevant features, since all the dimensions are being stretched to fit the unit variance schema. This is one of the reasons why data whitening is not often used as a dimensionality reduction technique but as a way to reduce noise in the data, such as the one generated by electrical stimulation artifacts commonly observed in electrophysiological recordings.

A common technique to performing data whitening is called Zero-Phase Component Analysis (ZCA) [Bell 1997]. While PCA focuses on dimensionality reduction, ZCA rotates the data in a different way so that the transformed vectors are still linear and uncorrelated but as close as possible to the original ones, which is highly desirable for noise elimination.

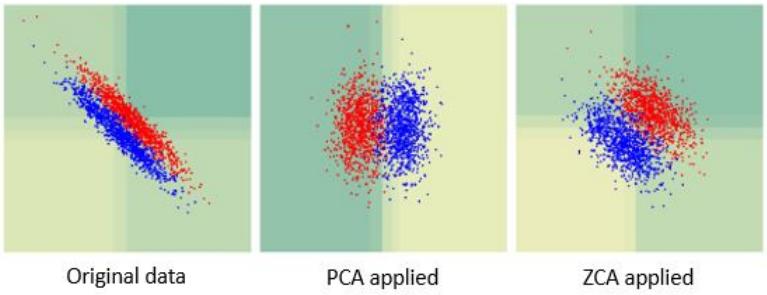

Figure 2.13: Illustration of how PCA and ZCA transform the data with different coordinate systems, such that PCA aims to maximize the captured variability via orthogonal components while ZCA aims to keep the natural inclination of the original space dimensions. Adapted from [Nam 2014].

2.5 Fundamentals of unsupervised clustering

2.5.1 Brief history of cluster analysis

The term *clustering* often referred to as *cluster analysis*, has a long trajectory that can be traced back to the 1850s, when Dr. John Snow made an effort to understand the behavior of the cholera outbreak that affected the city of London during those years. In 1854, the English physician plotted in a dot-map the location of the deceased, which turned out to conform clusters around specific street intersections that happened to surround polluted wells. By demonstrating that contaminated water rather than air was causing cholera, Snow managed to use clustering principles to expose both the problem and the solution of a deadly disease [McLeod 2000].

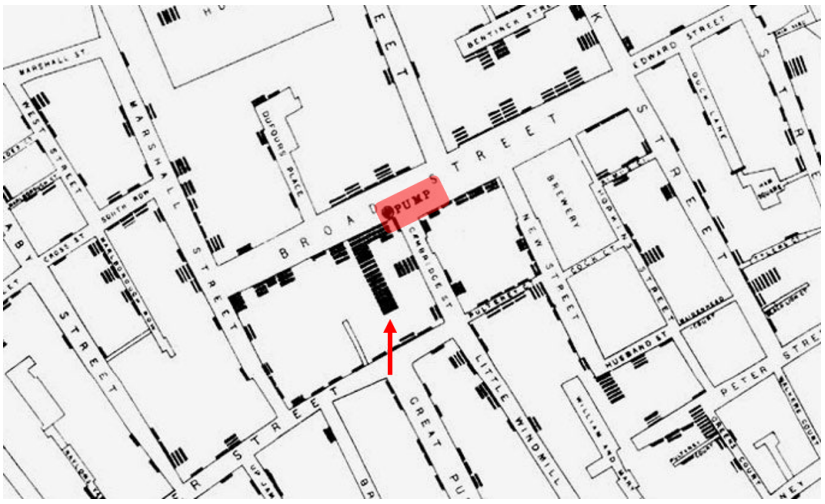

Figure 2.14: A section of the original map plotted by the English physician John Snow in 1854 during the cholera outbreak in the St. James, Westminster, area of London. Adapted from National Geographic archive files.

In particular, the theoretical foundations of clustering were of special importance in psychology, a field that spread its applications to other disciplines. In 1939, the behavioral

psychologist Robert Tryon was considered a pioneer in the use of clustering techniques to group a set of intellectual abilities measured in humans [Tryon 1939]. His contribution was considered as the first formal description of detailed clustering methods, reported in his monogram *Cluster Analysis* [Tryon 1958]. Additionally, other psychologists such as [Zubin 1938, Cattell 1943] were greatly recognized by the use of clustering to describe the spectrum of human personality traits.

From the 1950s, clustering literature grew exponentially. Not only it remained central to the spheres of disease and psychological studies, but it also has a rich history of applications in the fields of biology, geography, archaeology, psychiatry, marketing, and many more [Jain 1999]. Early examples of studies that popularized the use of clustering principles were lead by [Driver 1932].

During the last 60 years clustering has become more interesting, concretely for pattern recognition, information retrieval, and image processing, for which a vast number of publications and books can be found, inspired by authors such as [Anderberg 1973, Duran 1974, Hartigan 1975, Everitt 1979, Späth 1980, Jain 1988, Salton 1991, Rasmussen 1992, Backer 1995, Jain 1996].

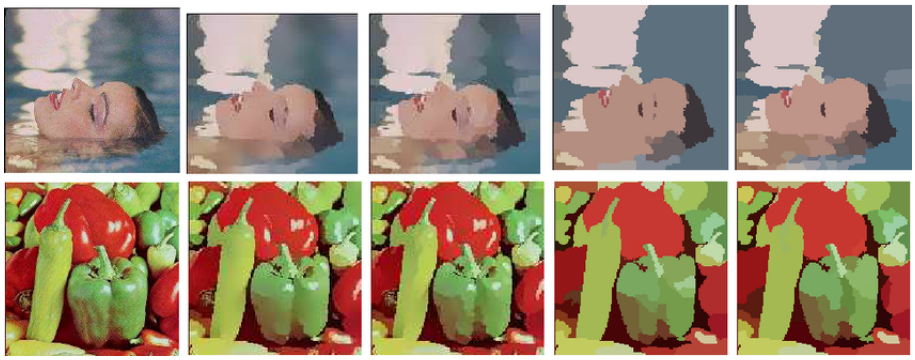

Figure 2.15: Image segmentation using K-Means with different inputs of clusters. Taken from [Backhouse 2007].

Due to the exponential growth in computational processing power, clustering is nowadays a more powerful tool than ever before. Cluster analysis represents a set of techniques mature enough in terms of its theoretical foundations, which has proven vast adaptability and applicability to different knowledge domains. Modern challenges of clustering approaches involve computational processing capabilities, so that large data sets such as the ones generated by DNA-sequencing in biology, or the Internet of Things (IoT) protocol can be analyzed in the search for groups of similar observations. In this sense, multi-core parallelized versions of classical partitional and hierarchical algorithms are the hype for modern researchers [Kanungo 2002, Kerdprasop 2010, Rao 2010, Fahim 2014, Bousbaci 2014, Böhm 2017].

2.5.2 Principles

Cluster analysis is the task of organizing a set of objects (usually in the form of unlabeled observations) into similarity groups called *clusters* [Rani[1] 2013]. The boundaries between the latter should be formed so that objects in the same cluster tend to be highly similar among them when compared to the objects that belong to the remaining class(es). In this sense, clustering can be understood as a multi-objective optimization problem that aims to minimize inter-cluster dissimilarity while maximizing the intra-cluster distances, as exemplified in Figure 2.16.

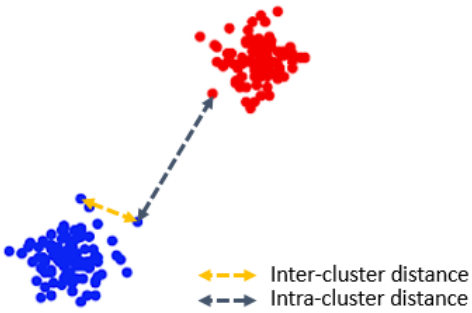

Figure 2.16: Illustration of inter and intra cluster distances.

In general, clustering applications need to be defined by the following variables:

1. **A proximity measure**, which can be either
   - **A similarity measure** $s(x_i, x_k)$**:** expected to be large if x_i, x_k are similar to each other.
   - **A dissimilarity (distance) measure** $d(x_i, x_k)$**:** expected to be small if x_i, x_k are similar to each other.

Figure 2.17: Illustration of proximity measures.

Mathematical formulations useful to measure dissimilarity include two special variations of the Minkowsky distance:

$$d_p(x_i, x_j) = (\sum_{k=1}^{m} \mid x_{ik} - x_{jk} \mid^p)^{\frac{1}{p}} \tag{2.1}$$

Where p is a positive integer.

- **Euclidean distance:** defined as an ordinary straight line between two points in an Euclidean space:

$$d_p(x_i, x_j) = \sqrt{(\sum_{k=1}^{m} \mid x_{ik} - x_{jk} \mid^2)} \tag{2.2}$$

- **Manhattan distance:** often more suitable for high-dimensional spaces, as it is more robust in the presence of outliers. The distance between two points is the sum of the absolute differences of their Cartesian coordinates:

$$d_p(x_i, x_j) = \sum_{k=1}^{m} \mid x_{ik} - x_{jk} \mid \tag{2.3}$$

- **Mahalanobis distance:** is a measure of the distance between a point P and the mean of a distribution D, originally introduced in 1936 by the Indian scientist P. C. Mahalanobis [Mahalanobis 1936]. This distance gives an idea of how many standard deviations lie between the point and the mean estimator of the distribution, which means that when the variance is re-scaled to one, the Mahalanobis distance corresponds to the standard Euclidean distance in a transformed space:

$$d_p(x_i, x_j) = \sqrt{(x_{ik} - x_{jk})^T S^{-1} (x_{ik} - x_{jk})} \tag{2.4}$$

  With S being the *covariance matrix* of the data.

2. **A criterion function:** to assess and evaluate the quality of the clusters formed.

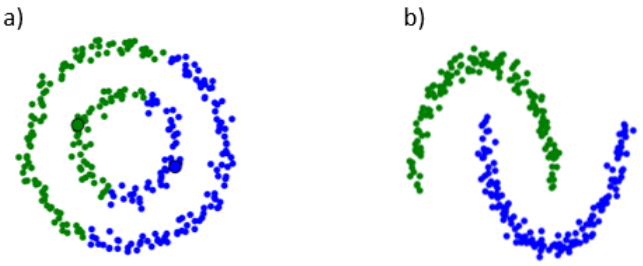

Figure 2.18: Examples of clusters formed in a toy data set by two different clustering approaches. **a)** Two clusters obtained via K-Means, which are bad-quality ones. **b)** The correct clusters for the data set were extracted using a DBSCAN algorithm. Taken from [Pedregosa 2011].

Cluster evaluation is a hard problem for which to find an optimal solution. In theory, two measures are computed to drive a better decision about this:

- **Intra-cluster cohesion (compactness):** to assess how near the data points in a cluster are to the centroid. The sum of the squared errors (SSE) is commonly used for this purpose.
- **Inter-cluster separation (isolation):** the best possible clustering shall separate the centroids of the clusters as much as possible.

Nevertheless, in practical applications, expert judgment is still the main contributor to judge the quality of any clustering task [Ullman 2014].

3. **An algorithm to compute clustering:** any technique that for example, works by optimizing a criterion function.

2.5.3 Techniques

In general, cluster analysis techniques can be categorized in two subgroups:

1. **Partitional:** centroid, model, graph, and spectral-based.
2. **Hierarchical:** agglomerative and divisive.

2.5.3.1 Partitional clustering

Partitional algorithms usually estimate the number of clusters in a data set all at once [Ullman 2014]. In general, these techniques aim to iteratively optimize an objective function with a global maximum that reflects the optimal level of "agreement" between the data and the partitions found for the cluster boundaries [Grira 2004].

Many of the techniques classified as partitional assume that the nature of the patterns that separate clusters can be explained by some kind of parametrical distribution (e.g. a Gaussian or mixture of Gaussians), a representative element(s) (e.g. a centroid(s)), or simple geometrical shapes (e.g. planes, circles, ellipses) in a given dimensionality [Fred 2003]. As part of this category, parametric density approaches such as *mixture decomposition techniques* [McLachlan 1988, Banfield 1993, Roberts 1998, Figueiredo 1999], *K-Means* [MacQueen 1967], and *K-Medoids* (in its variants PAM or CLARANS, represent a more robust option to K-Means in the presence of outliers) [Velmurugan 2010], can be found, which emphasize compactness in the form of spherical clusters fitted to the data.

The classical centroid-based K-Means is probably the most common partition algorithm, driven by the minimization of the squared error objective function given by:

$$J = \sum_{i=1}^{k} \sum_{i=1}^{n} \mid x_i^{(j)} - c_i \|^2$$

With $\mid x_i^{(j)} - c_i \|^2$ being a defined distance measure between a data point $x_i^{(j)}$ and the centroid of a cluster c_i. K-Means works by following the steps presented in Algorithm 1.

Algorithm 1 K-Means procedure

input : A set of elements $x_i, ..., x_j$
A k number of desired clusters

output : K set of clusters with assigned elements

Repeat:
Assign each element x_i to the closest mean centroid (minimal Euclidean distance)
Compute new mean (updated centroid) for each cluster

Until convergence criterion is met (usually when the assignment of members does not change)

When an initial guess about the number of clusters can be made, and the presence of outliers/noise is not predominant, K-Means is a computationally efficient clustering technique that identifies hyper-spherical clusters in a Euclidean space. Other variations of the original method include fuzzy approaches (also known as soft clustering or the Bayesian version of K-Means) [Pal 1995], the use of Mahalanobis distance to deal with hyperellipsoideal cluster shapes [Jain 1988], a K-modes formulation for categorical data [Huang 1997], the adaptations to straight-line fitting [Yin 1998], and modern implementations to deal with high-volume data sets [Kraus 2010].

In general, the intuition behind partition clustering algorithms is easy to understand but difficult to replicate with success in arbitrary data sets. This is due to their dependency on prior knowledge about the number of potential clusters, their sensitivity to random initialization of centroids [Fred 2003], entrapments into local optima[4], and poor cluster descriptors. Moreover, the square-error computation leads to the inability to deal with clusters of random shape, size, and density [Sisodia 2012], which represents an important obstacle for many practical applications in high dimensionalities. Figure 2.19 exemplifies a data distribution for which K-Means fails to find the expected clusters in a bi-dimensional plane.

2.5.3.2 Hierarchical clustering

Hierarchical algorithms find clusters using available information about previous clusters [Grira 2004]. These techniques deeply relate to the concept of distance, as they not only offer a cluster assignment for every observation at hand, but they also provide an overview of how distantly related the clusters are, information that is captured and presented via a **tree** or **dendrogram** (Figure 2.20). For instance, if patient medical records are being clustered, one might be interested to find that afflictions such as *pneumonia*, *influenza*, or *SARS* are part of a super-cluster called *respiratory diseases*. Note, however, that it would not be possible to know to what extent these clusters are related if partition-based clustering techniques such as K-Means are employed instead [Shalizi 2009], which is the reason why hierarchical clustering is often used in biological contexts.

[4]For K-Means to converge to the global minimum, it would need to iterate over all possible clustering solutions, but the number of iterations is exponential with respect to the size of the data set. Also, if the underlying distribution of the data is not Gaussian, then K-Means can be stuck to a local minima.

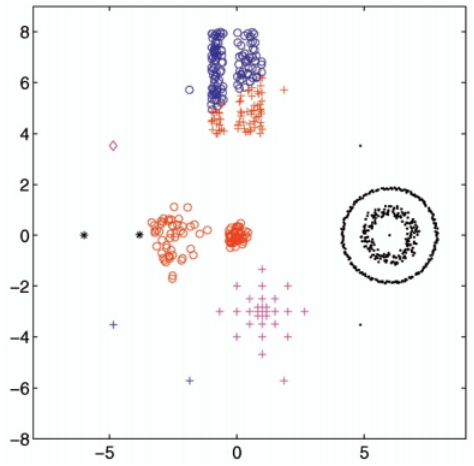

Figure 2.19: Imposed K-Means partition with 8 clusters over non-spherical data distributions for which expected groups of points are not correctly recognized. Taken from [Fred 2003].

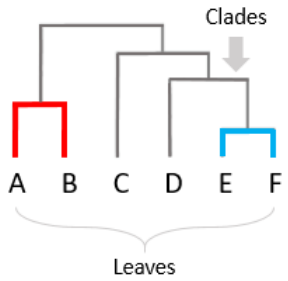

Figure 2.20: Example of a dendrogram generated by hierarchical clustering. The greater the difference in height between clades, the more dissimilarity that exists between them.

Agglomerative hierarchical clustering is a "bottom-up" approach that starts with each individual observation as a cluster, which is then successively merged into larger clusters by finding the closest pairs of clusters according to the distance metric of interest [Ullman 2014].

Such a procedure is *greedy* and deterministic, which means no initial random conditions must be provided (as opposed to partition-based methods such as K-Means). This agglomeration returns a sequence of **nested increasingly fined partitions** in which each level merges up two observations of the lower partition [Grira 2004]. Thus, in order to make this algorithm finite, it is mandatory to define at a given point, how close two *clusters* are and decide where to do the merges[5].

There exist three main approaches to define the cluster distance threshold that leads to the termination of the hierarchical agglomeration:

[5]The distance between two **clusters** is the stopping criteria of hierarchical methods, not the distance between two data points or partitions.

1. **Ward's Method:** states that the distance between two clusters A and B is given by their *merging cost* of combining both clusters [Rasmussen 1992]. In particular, Ward finds out how much the sum of the squares increases with the merging, given than this sum starts at zero (because every data point represents a single cluster), and which the algorithm will try to keep as small as possible:

$$\Delta(A,B) = \sum_{i\in A\cup B} \|\vec{x}_i - \vec{m}_{A\cup B}\|^2 - \sum_{i\in A} \|\vec{x}_i - \vec{m}_A\|^2 - \sum_{i\in B} \|\vec{x}_i - \vec{m}_B\|^2$$
$$= \frac{n_A n_B}{n_A + n_B} \|\vec{m}_A - \vec{m}_B\|^2$$

Where $\vec{m}_j$ is the center of cluster j, n_j j is the number of points in it, and Δ is called the merging cost of combining the clusters A and B. In general, Ward's rule for clustering is conservative in the sense that if two pairs of clusters with equally far centers are present, Ward will prefer to merge the smaller ones [Shalizi 2009].

Figure 2.21 shows the dendrogram produced by Ward's method on images containing flowers, tigers, and oceans. Overall, the structure looks reasonable, as well as the classification of the images given their labels, except for the mistake that it makes with *flower 5*.

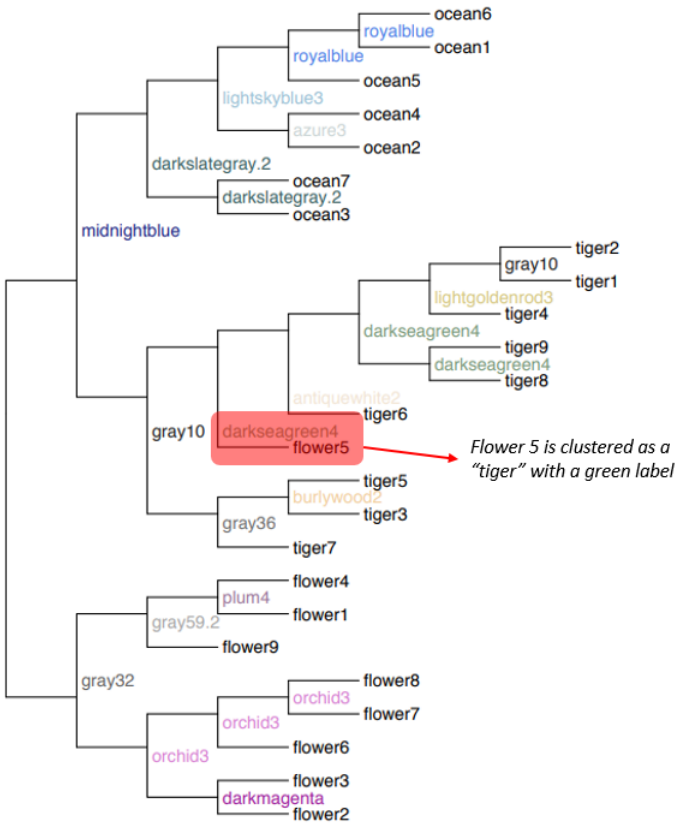

Figure 2.21: Ward's method used for the hierarchical agglomerative clustering of flower/tiger/ocean images. Each cluster is given a color label that is informative about the picture membership features. Adapted from [Shalizi 2009].

2. **Single-link:** also known as the **Nearest Neighbour Clustering**, this method defines the distance between two clusters as the minimum distance between their members [Rohlf 1980, Rasmussen 1992]:

$$d(A, B) \equiv \min_{\vec{x} \in A, \vec{y} \in B} \|\vec{x} - \vec{y}\|$$

The term "Single-link" implies that two clusters can be considered close to each other even if only a single pair of their observations is close enough. As a result, this method can handle complicated cluster shapes, at the expense of only caring about separation rather than compactness or cluster balance [Shalizi 2009]. This contrast makes single-link clustering one of the worst alternatives for spherical data agglomerations with the presence of a few outliers, as it is prone to generate long and thin clusters (Figure 2.22).

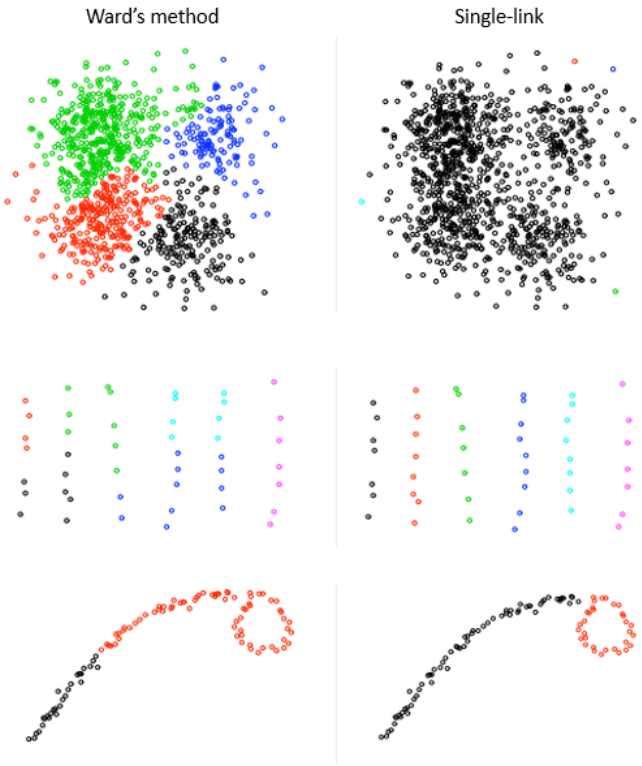

Figure 2.22: Ward's method is prone to better suit data set with spherical shapes, while single-link clustering is better at detecting extreme cluster distributions. Adapted from [Shalizi 2009].

3. **Complete-link:** another common approach of hierarchical clustering that defines the distance between two clusters A and B as the maximum distance among their members [Rohlf 1980, Rasmussen 1992]:

$$d(A, B) \equiv \max_{\vec{x} \in A, \vec{y} \in B} \|\vec{x} - \vec{y}\|$$

Similarly, alternative formulations of agglomerative clustering techniques have been formulated based on the idea of an **average-link** distance, some of which are mainly attributed to [Sokal 1958].

Divisive hierarchical clustering is a "top-down" approach that starts with the entire data set representing a cluster, which is subsequently divided into smaller and eventually singleton groups, again, based on a distance (dissimilarity) criterion [Ullman 2014]. This approach is much less used than the agglomerative one due to its computational complexity [Everitt 2001, Hexmoor 2015]. Consider a cluster that contains all the data patterns on which splitting will be made. For a cluster with M of these patterns, there are $2^{M-1} - 1$ potential subdivisions that must be considered by the algorithm [Igelnik 2013].

In summary, the main advantages associated with hierarchical clustering include:

- Its deterministic approach when compared to classical partition methods such as K-Means.
- Inspired in graph theory, it is useful to discover and recover underlying hierarchies in the data (common in biological applications).
- Its easy implementation in most statistical and analytical modern frameworks.
- The output of the clustering can be visualized via a dendrogram for an easier interpretation.

The main drawbacks of hierarchical clustering are:

- Its greedy approach may result in sub-optimal solutions. At each step, hierarchical clustering aims to find the best merging without optimizing a global goal.
- Its computational complexity, and therefore, cost. Even for medium-sized data sets ($>$ 0.25 GB) it will struggle to reach a faster solution when compared to partition algorithms such as K-Means. The standard hierarchical agglomerative clustering has a time complexity of $\mathcal{O}\left(n^3\right)$ and requires $\mathcal{O}\left(n^2\right)$.
- When it comes to large data sets, not only the computational expense becomes infeasible, but also the size of the dendrogram will hinder the process of visually recognize/interpret clusters.

2.5.4 The *optimal* number of clusters

Finding the optimal number of clusters in unsupervised learning is a crucial but blurry concern. There is not sufficient theory that support methods to find a convincing number of right clusters, and there is no consensus on what this "right" number of clusters means [Shalizi 2009]. Specific techniques to get clues on how many clusters are being explained by an arbitrary data set include:

1. **The elbow method:** aims to explain the percentage of variance described as a function of the number of clusters in a data set (Figure 2.23). In this sense, the *right* number of clusters is one such that adding one cluster more does not provide an improvement on the captured variance of the model. This method was initially proposed by Robert L. Thorndike in 1953 [Thorndike 1953]. Some authors such as [Ketchen 1996] has stated that the use of this criterion is highly unreliable because it cannot always be unambiguously identified. Variations of this criterion plot the **total or the within sum of squares** against the number of clusters so that this latter minimizes the error metric.

2. **The silhouette method:** the average silhouette is another useful criterion for assessing the natural number of clusters supported by the data. The silhouette of a data point explains how close that observation is to its cluster neighbors (inter-cluster similarity) and how loosely it is linked to observations in the remaining clusters (intra-cluster similarity) [Rousseeuw 1987]. Silhouette values range between -1 and 1, being a value close to 1 an indication that a specific data point is in the correct cluster. It is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

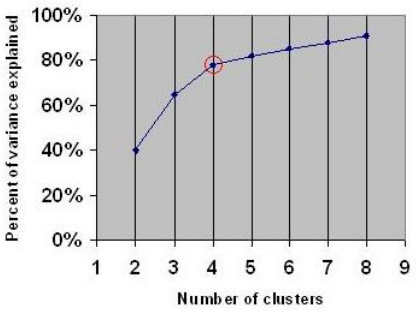

Figure 2.23: The *elbow* criterion represents the optimal number of clusters in a data set such that they explain most of the variance without overfitting.

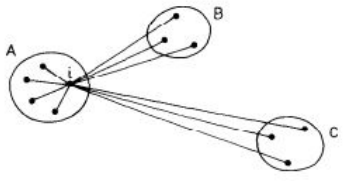

Figure 2.24: Elements involved in the computation of the *silhouette* values. Taken from [Rousseeuw 1987].

In practice, deciding on the optimal number of clusters often implies an iterative process of validating the clusters produced by an initial partition and rebuild the clustering until a consensus is reached.

2.5.5 Validation

Besides choosing the optimal number of clusters, a continuous challenge with cluster analysis is how seriously obtained clusters should be considered [Shalizi 2009]. There is not a fully compelling answer to this matter. In some applications, clusters shall be treated as mere indications of an unknown reality, or even as an advantageous way to summarize prior knowledge (e.g. a clustering on images of flowers, tigers, and oceans can conveniently show three clusters with no further cause or effect in reality. They were grouped only based on what is commonly known about them: their colors), while in others, they represent real divisions of the world (e.g. *pneumonia*, *influenza* or *SARS* are part of a super-cluster called *respiratory diseases*). In any case, the crucial question is up to what point clusters shall be reified and further *trusted* or considered as *valid*. In this regard, professor Shalizi well explained this concern: *"On the one hand, there are some theoretical constructs which it is absurd not to believe are real: germs and atoms, for instance. On the other hand, we do not think that constellations have any reality or meaning beyond giving convenient ways of dividing up the sky. How can we tell when our clusters are more like bacteria and when they are more like the signs of the zodiac?"* [Shalizi 2009].

Clustering techniques are to some extent subjective mainly because they are highly application dependant [Ullman 2014]. In practice, this means that even though clustering is useful to explore hidden patterns in data, its results are hard to evaluate. Nevertheless, there are some questions whose answers may indicate if a clustering strategy results are working well:

- *Is the data prone to contain clusters?* This is a concern that shall be approached in the early stages of the exploratory analysis. Clustering is a great tool to exploit the divisive nature of some data sets, but it is certainly not guaranteed to lead to substantial results in all cases where unlabeled data is available.

- *Are the identified clusters somehow in agreement with potential prior knowledge?* When there is no theoretical framework to support the clustering findings, all practitioners must proceed with caution before claiming that new clusters of observations were discovered. In those cases, reproducibility is key to reach a consensus. As opposed to this, when prior knowledge seems to match the cluster properties, the discovery capabilities of clustering act as confirmation techniques, which is highly desirable but not enough to prescribe absolute truths.

- *Do the identified clusters fit the data properly?* Good clusters should generalize well not only to new observations but also to new features. When clustering is based on the correct attributes, many other predictors allow the verification of the resulting clusters (e.g. a bird's species identification from the body shape enables the separation of other variables such as coloration and genome across bird clusters).

Considering the previous, there are two main approaches to validate clustering results:

1. **Internal validation:** gathers a set of techniques to assess the level of "agreement" between the data and the partitions, which means they consider whether the inter-cluster distances are maximized and the intra ones are minimized. Two of the most common metrics for this purpose are the Dunn and the Davies–Bouldin indexes.

   The *Dunn index* focus on the identification of compact clusters with small variances amongst their members, while measures how distant the means of the clusters are with regard to the within-cluster variance. A higher Dunn index usually represents better compactness and isolation, the reason why this ratio is also used to estimate the optimal number of clusters. Initially formulated by [Dunn 1973], this index is highly sensitive to outliers/noise and entails a high computational expense proportional to the dimensionality of the data set. Given m clusters, the Dunn index is defined as:

   $$\mathrm{DI_m} = \frac{\min_{1 \leqslant i < j \leqslant m} \delta\left(C_i, C_j\right)}{\max_{1 \leqslant k \leqslant m} \Delta_k}$$

   Where $\delta\left(C_i, C_j\right)$ represents the inter-cluster distance between observations C_i and C_j within the cluster k, and the maximum Δ_k is given by the maximum intra-cluster distance found between a pair of clusters.

   The Dunn index is considered a pessimistic estimator of the ratio between compactness and isolation. Consider the case of a set of clusters that are tightly packed but only one of them misbehaves and is sufficiently far away from the others. This particular scenario would result in a low Dunn index that is not necessarily representative of the characteristic nature of the clusters. This is the reason why several variations of this method have been formulated, many of them presented by [Bezdek 1984].

   The *Davies–Bouldin index*, introduced by David L. Davies and Donald W. Bouldin in 1979, relates the inter and intracluster distances in a different way: it can be interpreted as the average similarity between each cluster and its most similar one, averaged over all the clusters. This ratio behaves as an inverse function of the within-cluster compactness, which implies a lower value of the index means a better clustering [Davies 1979]. For a set of m clusters, its formal definition is given by:

   $$\mathrm{DB_m} = 1/\mathrm{m} \sum_{\mathrm{i}=1}^{\mathrm{m}} \max_{\mathrm{is}} \left\{ \frac{\Delta\left(\mathrm{C_i}\right) + \Delta\left(\mathrm{C_j}\right)}{\delta\left(\mathrm{C_i}, \mathrm{C_j}\right)} \right\}$$

   For m clusters, where $\delta\left(\mathrm{C_i}, \mathrm{C_j}\right)$ is given by the inter-cluster distance between clusters $\mathrm{C_i}$ and $\mathrm{C_j}$, and $\Delta\left(\mathrm{C_k}\right)$ is the intra-cluster distance of all the observations within cluster $\mathrm{C_k}$.

2. **External validation:** often the most valuable in practice, external validation gathers specific metrics and qualitative decision rules that come available only with the prior knowledge of the clustering problem at hand. They usually originate from experts in the field, whose knowledge is often essential to assess the quality of any

resulting clusters, at the expense, that they can sometimes result in a confirmation bias that is not prescriptive [Grira 2004].

2.6 Fundamentals of supervised classification

According to a widely spread definition born at the core of the Stanford University, Machine Learning (ML) *is the science of getting computers to act without being explicitly programmed.* However, more important than this is the fact that ML models can truly learn from data beyond fixed rule-based programming. When these models are fed with adequate data, they have the ability to generalize their knowledge for wider domain applications, the reason why these methods are highly valued.

ML applications are often categorized in one of these types: *1) Supervised learning*, in which an algorithm makes use of a function able to map inputs to desired outputs based on training examples, *2) Unsupervised learning*, in which labeled data is not available for mapping but rather the discovery of hidden patterns is performed, *3) Semi-supervised learning* as a combination of the previous approaches, *4) Reinforcement learning*, in which algorithms are expected to act and provide feedback about an external event for which they were trained to react to, or, more recently, *5) Transfer learning*, which reuses models trained on a particular type of data to be used in other applications [Ayodele 2010]. The latter two subgroups of applications are often related to the deep learning domain. Figure 2.25 summarizes three of the above categories that are most widely used.

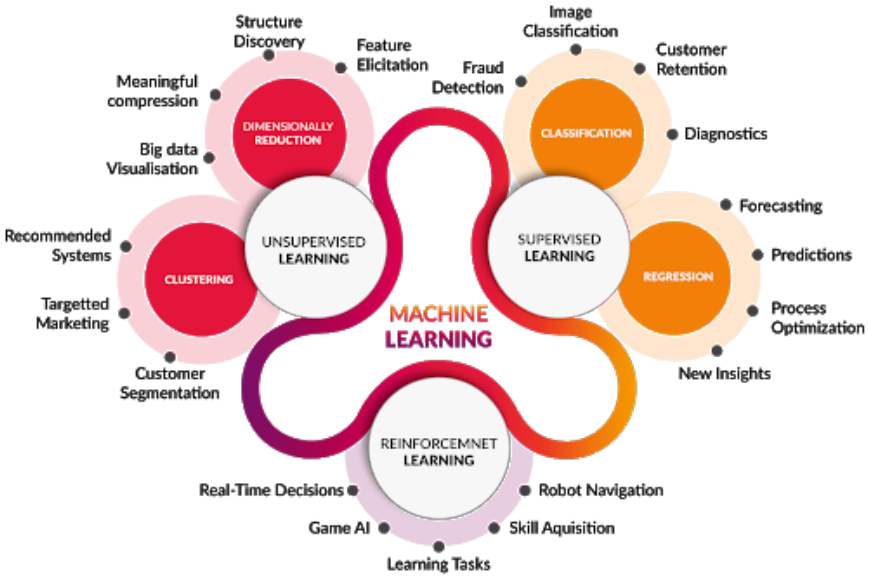

Figure 2.25: Machine learning categories and their main practical applications. Taken from cognub platform.

As a subcategory of supervised learning, classification has the goal of predicting categorical class labels for unseen observations, based on past ground-truth examples. In general, there are two main classification problems:

- *Binary classification:* in which observations can only take one out of two possible labels. Classical examples include spam detection, or disease diagnosis, cases in which observations will be assigned a 1 in the case of positive response to the predicted outcome, and 0 otherwise.
- *Multi-class classification:* case in which more than two valid labels are available to assign to unseen observations. This study tackles a multi-label classification problem in which 5 types of possible cell classes aim to be predicted.

One of the critical questions that supervised classification must tackle is whether the classes at hand are actually separable based on the predictors used and as such, how separable they are. Figure 2.26 summarizes a visual representation of this concern.

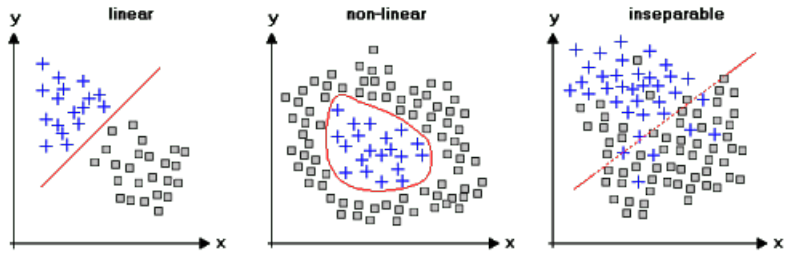

Figure 2.26: The intuition behind separability in supervised classification. Taken from [Lohninger 2010].

In this direction, algorithms that tackle supervised classification must be able to sort out how separation works amongst classes. Either probabilistic or non-probabilistic approaches can be considered for this purpose. The former methods normally involve the use of random variables and yield a probability distribution over a set of classes available for being chosen for every observation, whereas probabilistic approaches do not model probability distributions but instead divide the feature space and return the class label on which every individual sample relied on that dimensionality [Murphy 2012].

As such, the first question that must be answered when a classifier has been already decided to be used, is whether a discrete class or a probability distribution estimation is required by the application. Figure 2.27 presents suitable classifiers for both scenarios. It is worthy to note that methods such as Support Vector Machines and Naive-Bayes classifiers have been adapted to offer probabilistic estimations, but they do not offer so in their original formulations [Niculescu-Mizil 2005].

A second question that must be asked is whether a *white-box* technique is required or not to interpret the outputs of a given model. When using *white-box* models, the outputs are highly interpretable and reproducible because the algorithm being used is clear and transparent to the final user. Moreover, influencing variables can be recognized as well as their influence on the output. On the opposite side, *black-box* techniques are untraceable, and it is no possible to establish the mechanism that they used to produce a specific output for a particular observation [Loyola-Gonzalez 2019]. Figure 2.27 shows two examples of white-box classifiers, which means the remaining listed methods are considered as

black-box ones.

Once the previous concerns have been addressed, the law of parsimony should be considered when a model is going to be selected to solve a particular problem [Guyon 2010]. This law, also known as the Occam's razor principle, states that *"entities should not be multiplied without necessity"*, meaning that the simplest explanation is often the most appropriate one. In this sense, the models with the fewest assumptions or complexities shall be the preferred ones over the highly performant but complicated. In general, *linear classifiers* should be tried first, as their complexity is low and they can offer a high performance if the data is linearly separable. Within this group of models, *Logistic Regression* and *Passive-Aggressive Classifiers* are found [Hosmer Jr 2013]. From the point in which a *logit* model is insufficient and the size of the data is moderated, other simple yet effective techniques can be tried, such as K-Nearest Neighbors, or even Bayesian models such as *Naive-Bayes Classifiers*[6] [Bishop 2006]. Beyond these options, the no-free-lunch theorem invites to try a white-box technique such as decision-trees to get an intuition of potential solutions for the problem at hand, while more robust and *obscure* options include gradient boosting methods from the tree-based family [Natekin 2013, Chen 2016]. Moreover, when suspecting data non-linearities that may be hindering the encounter of class boundaries, kernel transformations can be applied to get separability. The principal actors here will be the Support Vector Machines [Schlkopf 2018]. Nevertheless, it is important to keep in mind that the hierarchy of models to be considered depends primarily on the size of the data and on the need for a probabilistic model in the first place.

For this study, the need for a naturally probabilistic model was not initially required, as only discrete labels wanted to be generated to predict cerebellar cortical cell classes[7]. Moreover, while it is preferable to deploy high-performance prediction models using white-box techniques, this was not a requisite for this application.

[6] Even when Naive-Bayes classifiers have a strong assumption of independence amongst the features, they have proven to be very effective in real-life applications [Zhang 2004].

[7] Nevertheless, this is something that may be worth trying once it can be confirmed if the features at hand can correctly separate the classes of interest.

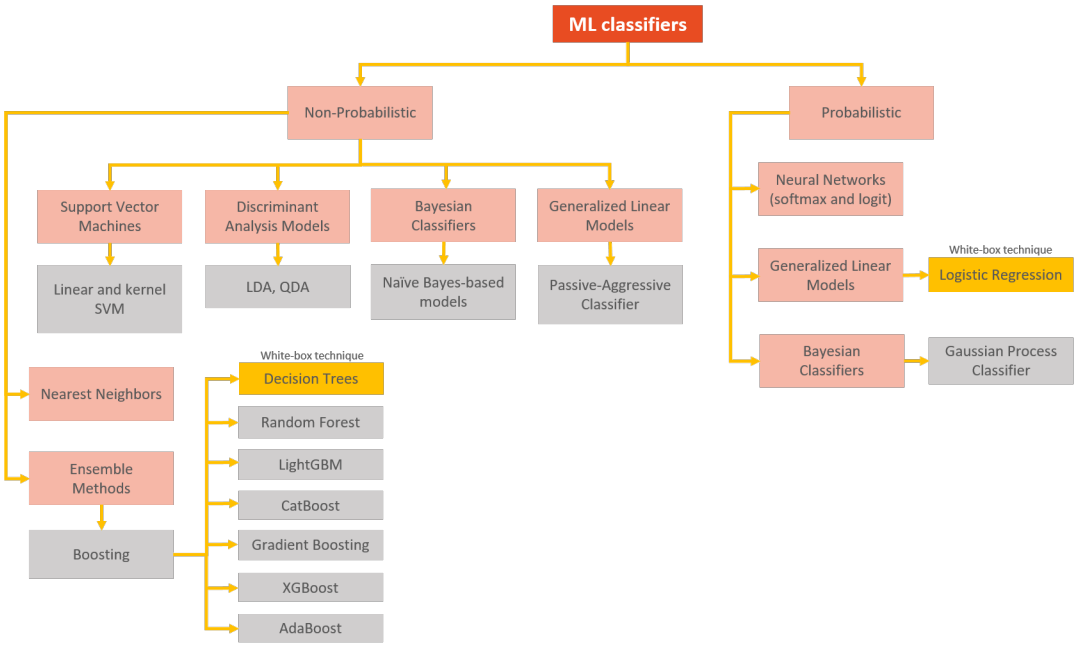

Figure 2.27: Comprehensive schema of supervised classifiers. More on probabilistic models by [Niculescu-Mizil 2005].

2.6.1 Logistic Regression

Logistic Regression (or logit regression) is a probabilistic Generalized Linear Model (GLM) that in its basic form uses a *sigmoid (or standard logistic) function* to model the behavior of a *binary dependant variable*, measured as the probability of a certain class or event happening (e.g. pass/fail). Despite its name, logistic regression is often used for classification purposes [Hosmer Jr 2013]. To do so, the sigmoid function output, which is a value in the range $(0, 1)$, is approximated to one of the non-inclusive extremes depending on a cutoff generally set up to 50%, which allows converting probabilities of occurrence into discrete labels. The previous implies that under this modified use of the model, any probabilistic interpretation of the predictions is lost. If some observations are more confidently closer to a specific class label than others, there is no way to tell from this classification perspective.

The general definition of the standard logistic function is given by:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The multinomial logistic regression (or mlogit regression) is an extended version of the original model that uses the **softmax function** instead of the sigmoid function to predict the most probable event out of more than two possibilities or classes associated with a random variable. This study was interested in the practical application of the multinomial logistic regression to predict five cerebellar cortical cell classes.

Similar to the binary case, in the multinomial case the softmax function is used to

convert arbitrary vectors of real numbers (continuous features) into discrete probability distributions squashed to the range $(0, 1)$. For a vector $x \in \mathbb{R}^n$, the softmax function is defined as:

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

The idea behind logistic regression is to express a problem (an event associated to a random variable) in the form of a generalized linear regression model, such that $\hat{y}$ represents a predicted event, $x_i, ..., x_n$ are the features used for prediction, and $\beta_1, ..., \beta_n$ are the coefficients to be learned by the regression:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

For the multinomial case, there are two ways to do so. The first one consists in the training of separate binary logistic regression models for every class (one vs. all strategy). This way, every observation x with a given label y goes through a score computation phase to obtain z such that $z = W^T x + b$, where matrix $W \in \mathbb{R}^{C \times M}$ and vector $b \in \mathbb{R}^C$ are parameters to be learned from the data. Subsequently, the probabilities for each class are obtained via the application of the sigmoid function. For example, the probability associated to the class label 1 would be estimated by: $p\left(y_c = 1\right) = \sigma\left(z_c\right) = \sigma\left(W_c^T x + b\right)$, a score then subject to discrete approximation to generate a class prediction. On the other hand, the second method predict classes straightforwardly after the computation of $z = W^T x + b$, such that the highest score $\hat{y} = \arg\max_i z_i$ is the predicted class (equivalent to the probabilistic estimation $\hat{y} = \text{softmax}(z)$).

For the application of logistic regression to the multi-class setting of this study, both approaches to predict classes were considered and cross-validated (the one vs. all binary estimations and the softmax-based method), as one disadvantage of estimating individual models on the classes is that the logit regression assumes the probabilities for each class are independent, which may not occur in reality.

When the softmax approach is used, the mlogit regression recurs to the **multi-class cross-entropy** function (or negative log-likelihood function) to measure how different the obtained predictions are from the real labeled values associated with the data. This comparison allows the estimation and minimization of the error associated with the learning process. Conversely, the cross-entropy function is used in the binary case. For two probability distributions $p(x)$ and $q(x)$, the cross-entropy function is given by:

$$H(p, q) = -\mathbb{E}_{x \sim p}[\log q(x)]$$

$$H(p, q) = -\sum_{x} p(x) \log q(x)$$

if p and q are discrete.

Model parameters W and b can be estimated by performing gradient descent on the loss function with respect to them. Fortunately, the logistic loss function is convex and it

is guaranteed to converge to a globally optimal value. Moreover, if a regularization term $\lambda\|w\|_2^2$ is added to the loss function ($\lambda > 0$), the latter becomes strictly convex and has a unique global minimum.

2.6.2 XGBoost

Extreme Gradient Boosting (XGBoost) is an *Optimized Gradient Boosting* non-probabilistic technique widely used in machine learning due to its scalable nature, speed, and higher performance when compared against single-algorithm methods [Morde 2019]. As summarized in Figure 2.28, XGBoost is not only an ensemble tree-based method but a technique that takes into consideration system optimization as well as advanced machine learning principles to increase performance via the use of a greedy additive strategy: *fix the errors and boost the correct knowledge one new tree at a time.*

The XGBoost objective function to be optimized is composed of several CART learners. As a result, this function of functions is not analytically tractable with traditional optimization methods, the reason why it is approximated using Taylor's theorem and Gradient Boosted Trees. To make overall predictions, both binary or multi-class, XGBoost uses either the logistic or the softmax functions.

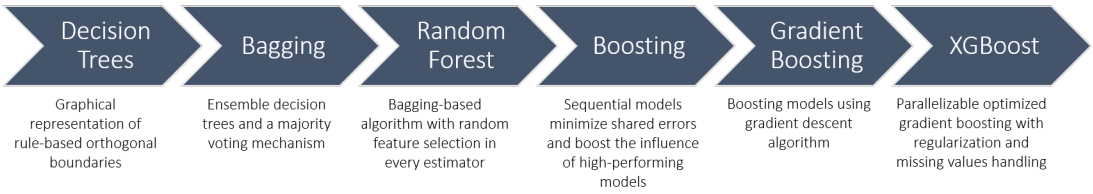

Figure 2.28: XGBoost essence has its background in the tree-based algorithms family and gradient-based optimization techniques. Adapted from [Morde 2019].

As a scalable machine learning system for tree boosting, XGBoost is available as an open-source portable package supported by all the major operative systems and programming languages (C++, Python, R, Java, Scala, and Julia) and most of the cloud integration frameworks (AWS, Azure, Spark). Since its original publication by [Chen 2016], XGBoost has been widely recognized as a method to generate state-of-the-art accurate predictions in a variety of domain application that include store sales prediction, high energy physics event classification, web text classification, customer analytics prediction, motion detection, ad click-through rate prediction, malware classification, product categorization, and hazard risk prediction. Moreover, XGBoost has been the driving force behind the top performant and winner models in several Kaggle competitions, in which deep learning methods were comparably good.

The higher performances associated with the use of XGBoost in machine learning naturally occur from the unified approach that this method implements. In fact, its novelty comes from the union of theoretical and practical principles borrowed from older techniques and applications, namely:

- *The use of gradient tree boosting:* had already been proven to be successful to make predictions more robust [Friedman 2000, Burges 2010, Chen 2015b].

- *The use of regularization to prevent overfitting:* the idea was already considered by regularized greedy forests [Johnson 2013], but the objective function to be optimized was simplified to be implemented in a parallel-fashion by XGBoost. Both L1 and L2 regularization methods are supported.

- *The use of a weighted quantile sketch for approximate tree learning:* XGBoost implements a novel proposal to determine candidate split points for its estimators. Finding those splits is especially non-trivial for large data sets, for which the typical quantile sketch algorithm is inappropriate [Greenwald 2001, Zhang 2007]. The general idea is to find weighted quantiles to split the data that is hard to learn from, as the regions of the data already well predicted do not need to be partitioned. Once these key splits are determined, the learning accuracy will be higher.

- *The use of column sampling:* typically implemented by Random Forest, is a simple and effective technique embedded in XGBoost [Breiman 2001].

- *The use of in-built cross-validation capabilities:* XGBoost internally estimates the number of boosting iterations to run at every step to maximize confidence in predictions.

- *The introduction of sparsity-aware learning:* an idea associated with generalized linear models that needed to be adapted to *wide* data [Fan 2008] (when the number of predictors exceeds greatly the number of samples) without overfitting (e.g. Lasso regression), was implemented to tree-based structured methods by XGBoost. This sparsity-awareness allows the efficient handling of naturally missing values and sparse patterns in the data.

- *The use of out-of-core computation and cache-aware learning:* while tree-based methods had already been parallelized by [Tyree 2011], XGBoost takes this idea further by working as an external memory algorithm with cache awareness to access all relevant data in one sequence and minimize the movement of memory pages.

Several studies have benchmarked XGBoost against other machine learning techniques to conclude on its efficiency. Some of them have reported XGBoost implementations up to 20 times faster than Gradient Boosting models [Chen 2015a], typical Decision Trees and Logistic Regressions, as well as more robust and stable than Support Vector Machines [Dhaliwal 2018, Fan 2018, Ji 2019]. Figure 2.29 shows graphically one of these studies comparisons.

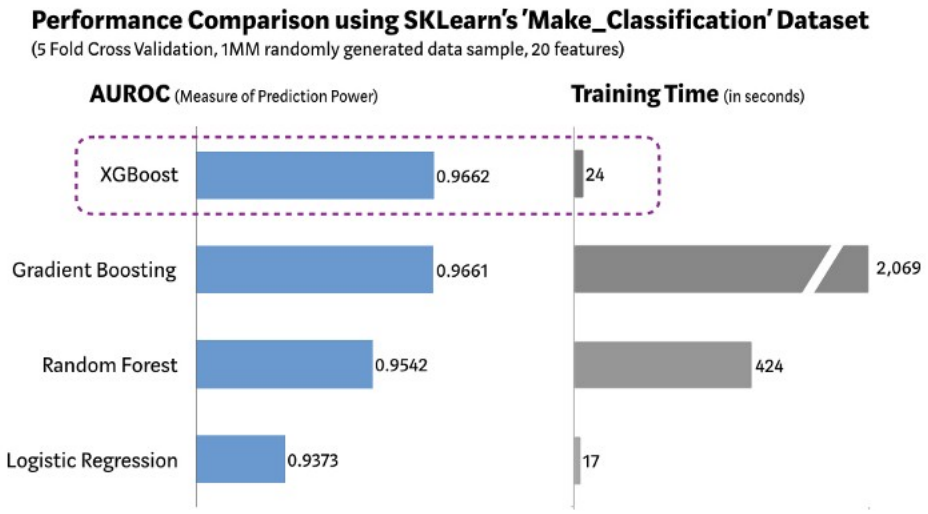

Figure 2.29: XGBoost vs. Alternative machine learning algorithms implemented on the sklearn's *Make_ Classification* data set. Taken from [Morde 2019].

2.6.3 Oversampling techniques

Several machine learning applications often suffer from the problem of class imbalance. When classes are not approximately equally represented in the data set, most predictive algorithms experience poorer performance.

Traditionally, the class imbalance problem has been tackled either via under-sampling the majority class(es) or over-sampling with replacement the minority class(es). While the first strategy has demonstrated to work well in practice, it is not always viable, especially in applications in which rare and abnormal observations want to be predicted. Moreover, the main issue with the latter strategy is that it can cause overfitting, as minority replicated examples drive the learning of small specific decision regions that do not necessarily generalize as desired.

2.6.4 Synthetic Minority Over-sampling

In particular, SMOTE, which stands for *Synthetic Minority Over-sampling Technique*, is a widely used over-sampling technique originally introduced by [Chawla 2002]. The successful adoption of this method is related to its avoidance of replacement replication to over-sample the minority classes. Instead, SMOTE creates synthetic examples from the interpolation of the already known observations of the classes to be extended. Depending on the amount of over-sampling required (an input parameter of the method), SMOTE uses the k-nearest neighbors randomly chosen to generate artificial observations in their directions. For instance, if 5 nearest neighbors are considered and the desired level of over-sampling is 200%, only two neighbors from the five nearest neighbors are chosen and one sample is generated in the direction of each.

The procedure to compute synthetic examples, as explained by the authors, is as fol-

lows: *"Take the difference between the feature vector (...) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general".*

SMOTE not only helps machine learning techniques to avoid overfitting by expanding the data instead of replicating it. The method has proven to be more effective to increase predictive performance than introducing under-sampling alone, and even imbalance class prior distributions in Bayesian models such as the Naive-Bayes classifier. Similar over-sampling techniques include a probabilistic adaptation of SMOTE known as ADAptive SYNthetic (ADASYN), DBSMOTE, and Borderline-SMOTE. In most practical applications, the use of SMOTE is enough to balance the underlying learning space [Han 2005, He 2008, Bunkhumpornpat 2012]. Therefore, SMOTE was of great utility for the construction of the supervised classification pipeline in this study, as the scarcity of labeled data for some cerebellar cortical cell classes was a threat to optimally train a model to predict them.

2.6.5 Evaluating model performance

Correct implementations of machine learning always consider different strategies to avoid the presence of the worst enemy of generalizable prediction models: **overfitting** [Dietterich 1995]. The most common starting point to avoid it is to use a holdout partition of the data, such that the algorithms can be trained and tested on separate chunks of observations. The risk of overfitting in the test-set, however, is still present because the hyper-parameter tuning process is not completely blind to this data, which can result in evaluation metrics that no longer report on generalization performance. As a consequence, a third validation set is required, on which the final evaluation is done after the end of the fitting experiment. Figure 2.30 summarizes this process, for which the finding of the optimal parameters can be performed using cross-validated strategies such as random grid search or Bayesian optimization.

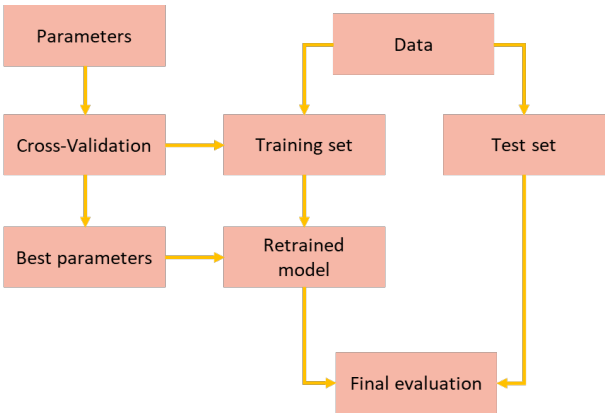

Figure 2.30: Typical holdout workflow in model training. Adapted from [Pedregosa 2011].

To overcome the limitation imposed by small sample sizes, in which a triple holdout may restrict the number of observations per set of training, test, and validation, the use of cross-validation is even more important, not only to estimate the better model parameters but to help to strengthen the training process [Moore 2001]. In such a procedure, a test set is still held out for the evaluation of the performance, but a validation set is no longer needed. In its basic approach, *k-fold cross-validation*, the available data set is split into k sets, from which $k-1$ are used for training purposes and the remaining chunk is used for model evaluation [Pedregosa 2011]. The performance metric reported by this approach is then the average of the values computed in the loop, assuring the data is not as wasted as if an arbitrary validation set was defined. Of course, this happens with the additional cost of a more computationally expensive process, and still, a generalized performance metric cannot be guaranteed. Figure 2.31 summarizes this process.

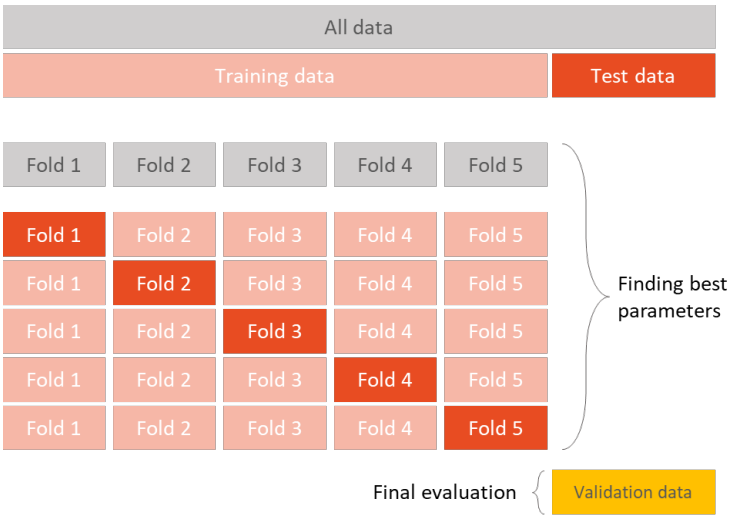

Figure 2.31: Cross-validation workflow for model training and testing. Adapted from [Pedregosa 2011].

2.6.6 Classification performance metrics

The greatest advantage of supervised classifiers is the possibility of measuring classification performance, the basis on which any machine learning algorithm can be assessed and compared to another, precisely the main obstacle of unsupervised clustering. Such performance is usually defined from the input of the confusion matrix associated with a particular classifier, which allows the computation of metrics such as *precision*, *recall*, *specificity*, and many others [Tharwat 2018].

For *non-scoring classifiers*, also known as *hard* classifiers, the previous metrics are often summarized in the *balanced accuracy* between sensitivity and specificity, or the *F1-Score*, which is the geometric mean between precision and recall. Hard classifiers are called as such because they output a discrete label z for every observation x, such that

$z(x) \in 1, 2, ..., k$, being k the number of available labels[8] [Richards 1999]. In Chapter 4 of this study, a machine learning pipeline is presented to build a high-precision non-scoring classifier to identify cerebellar cortical cells.

In general, the usual metrics to assess the performance of any supervised hard classifier are summarized in Figure 2.32. For unbalanced data sets in a multi-label setting, it is more appropriate to compute the macro average on the F1-Score to measure the classifier performance on the individual classes [Van Asch 2013]. To do so, precision and recall metrics shall be computed as usual, but they are weighted by the proportion of observations corresponding to every particular class before being added up.

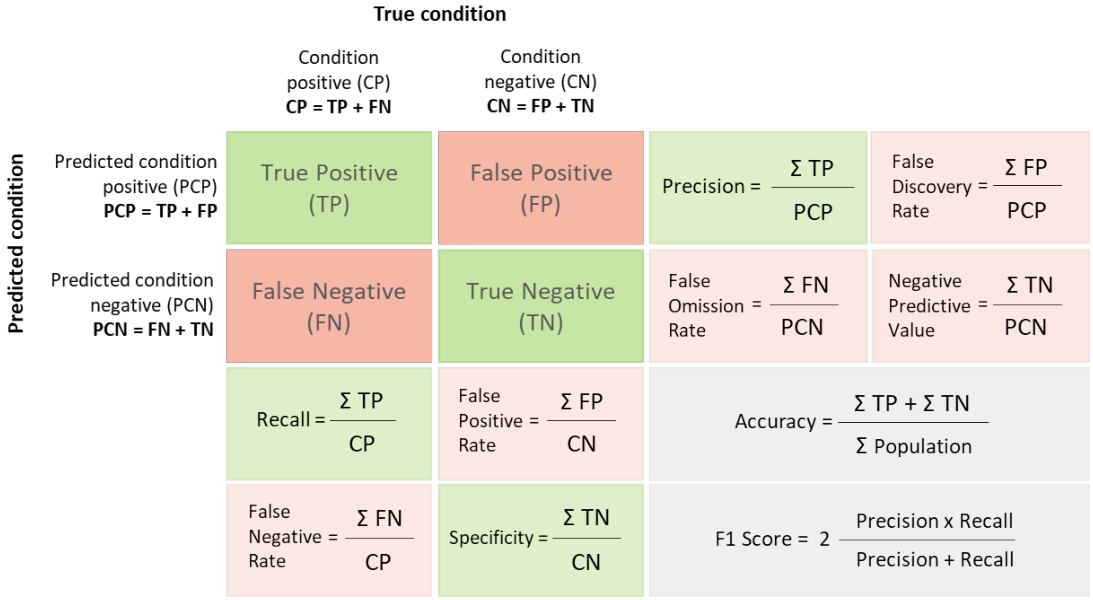

Figure 2.32: Metrics to measure supervised classification performance. Adapted from [Tharwat 2018].

In the next chapters, the data used by this study is introduced, as well as the procedures that were considered for its preprocessing. Later, both unsupervised and supervised learning are presented as complementary techniques that allowed not only the discovery of cell clusters in the cerebellar cortex but also their use for prediction purposes.

[8] *Soft* classifiers output an associated score, usually in the form of a probability for every observation being predicted.

Chapter 3

Data

Before diving into the machine learning methodologies that were put in place to achieve the purpose of this study, it is essential to present the nature of the available data that allowed so, as well as the preprocessing and filtering procedures that were necessary for its exploitation into useful information. This chapter delves into these data-related matters, and also develops a formal introduction of the feature extraction process followed to compute meaningful modeling predictors able to explain the electrophysiological behavior of cerebellar cortical neurons.

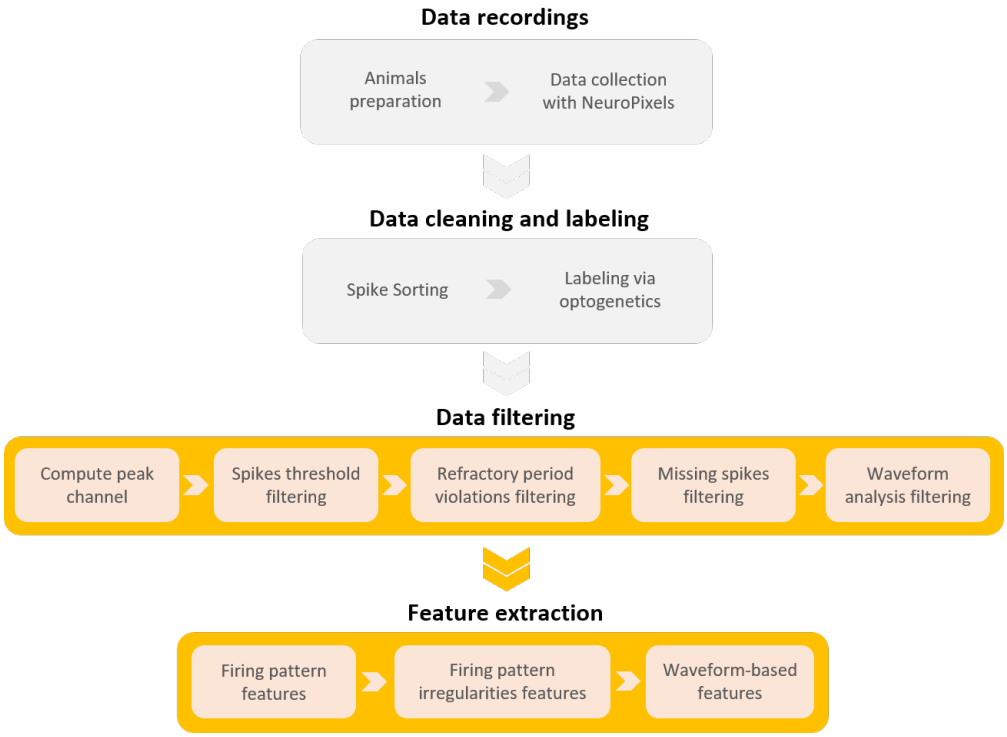

Figure 3.1: Pipeline phases covered in Chapter 3 are *Data filtering* and *Feature extraction*. Data recording, cleaning, and labeling procedures, although performed prior to this study are briefly introduced.

3.1 Experimental settings

This study used data previously recorded *in vivo* from awake mice, which means it was directly measured from living and fully functioning cerebellums. This constitutes an important precedent because there is no study in the state of the art that has performed classification tasks on the cerebellar cortical cells with awake animals. In general, neurons operate both chemically and electrically, and the measurement of their behavior can substantially change when the brain is either sliced or under anesthesia [Hevers 2008, Haider 2013], the approaches used by the precedent studies synthesized in Chapter 2.

3.1.1 Source

Four male mice subjects constitute the direct data sources for this study. They are considered to generate observations from the same distribution, as these mice are genetically engineered for research purposes. Therefore, experimental bias associated with the nature of the data is not a relevant risk for this study.

3.1.2 Instruments

In the field of neuroscience, electrophysiological methods play an important role. Electrophysiology is the field that studies the electrical properties of living cells and tissues, and specifically to this case, the way in which neurons communicate with each other by sending electrochemical messages inside the brain.

Best modern electrophysiological methods that record neuronal activity comply with some standards. They measure electrical gradients with a high temporal resolution in vivo, in arbitrary brain locations, and in multiple neurons at the same time while maintaining an affordable cost. For the last 50 years, the most critical challenge in the design of physical instruments to perform this task has been to encapsulate the largest number of recording sites in the thinnest probe or shank, so that many neurons can be recorded simultaneously while diminishing brain tissue damage at the moment of the insertion in the brain[1].

Previous considerations are well captured by Neuropixels, a novel multi-site recording silicon probe (available since 2016) whose design is based on complementary metal-oxide-semiconductor (CMOS) technology (Figure 3.2). From the 1950s, when only 1 neuron at a time could be recorded, a Neuropixels probe allows the collection of the electrical activity of almost 1000 neurons synchronously [Steinmetz 2018], which is a considerable increase given that the average mouse has nearly 70 million neurons in its entire brain [Williams 2000].

The recording of the cerebellar neuronal activity is carried out by inserting these novel

[1] In the case of mice and their diminutive brains, a probe that is half centimeter wider than another can create a significant distortion in the animal's brain tissue, thus increasing the risk of death and the loss of a source for data collection.

probes into the cerebellum of the awake animals. These instruments contain a silicon protected electrode, also called shank, of approximately 1 cm long, which is partially inserted into a mouse brain (4 mm) by means of a surgery that replaces part of their scalp bones and muscles with a head plate attached to their skull, which acts as a support for the probe.

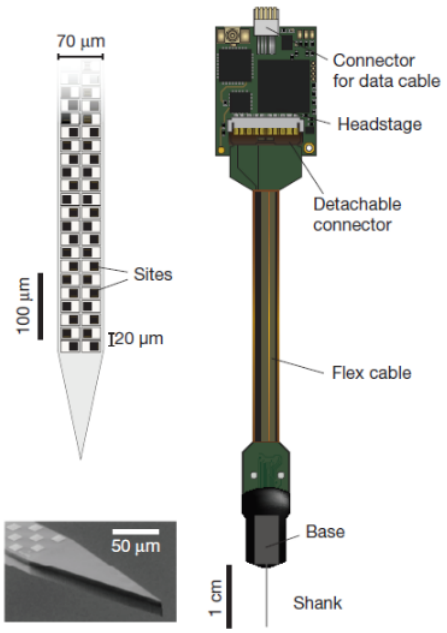

Figure 3.2: Schema of a Neuropixel probe. Each side of the probe has a total of 384 channels (recording sites) to record the electrical activity of the surrounding neurons. An additional digital channel (1 mm width) is included in the probe to amplify and digitize the data, thus, a total of 769 recording sites are available for data collection [Steinmetz 2018].

With a total weight of 0.3 grams and a width of 70 μm, each recording site of a Neuropixel probe is 14 x 14 μm and produces amplified data directly coming from the electrical neuronal signals. The electrical potential difference of a neuron over time is almost imperceptible (in the scale of microvolt units), the reason why an amplification degree is needed so that the detection of neuronal spikes or "firings" is actually possible. Once these analog signals are captured, they are amplified, digitized, and saved as binary files (Figure 3.3) [Steinmetz 2018].

Raw data files collected from the silicon probes record the internal voltage of many neurons over time (Figure 3.4 (b)). Depending on their location and proximity to the probe recording sites, different neurons will be captured by distinct particular channels (Figure 3.4 (c)). These analog signals over time are measured at intervals regularly lasting 1 hour and with a high-resolution frequency of approximately 30 kHz (30000 measurements are taken from each channel per second).

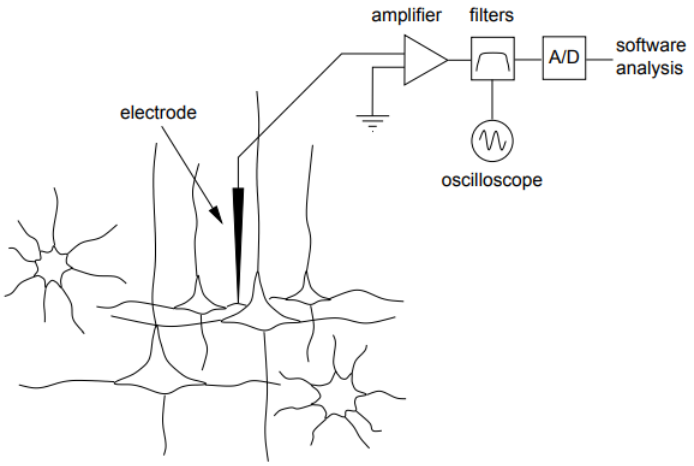

Figure 3.3: Neuronal signals can be measured and analyzed through the amplification of electrical responses captured by an electrode (probe).

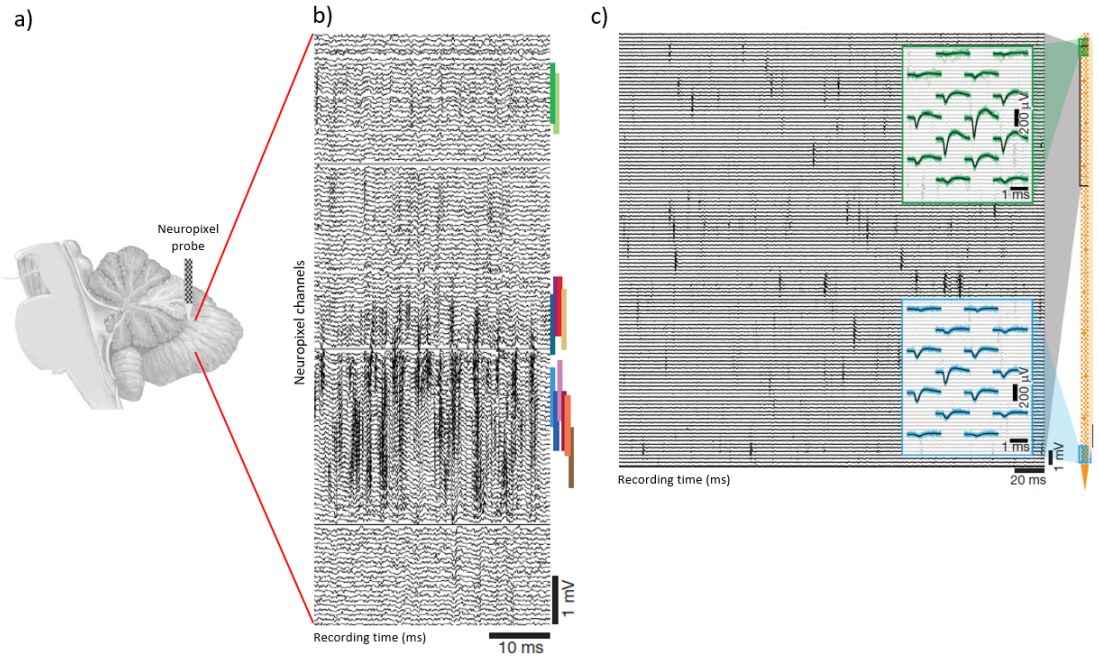

Figure 3.4: **a)** An example of raw data recorded from a Neuropixel probe inserted into the cerebellum. **b)** Electrical traces of the neurons being recorded by the sites of the probe in 130 of its channels. Overlapping traces represent spiking neurons, this is cells that are being stimulated and are emitting electrical signals in the form of action potentials. Vertical color lines represent the location of identified neurons along the probe channels. The green lines correspond to neurons closer to the cortex of the cerebellum, while the remaining ones correspond to neurons that are located deeper inside this brain tissue. **c)** Zoom on the raw data recordings allows the visualization of the traces of specifically identified neurons that were measured in different channels of the probe. Adapted from [Steinmetz 2018].

3.2 Data collection

3.2.1 Recordings

This study was carried out on data recordings previously made with Neuropixel probes on the cerebellar cortical neurons of mice subjects during a period of 6 months between 2018 and 2019. A total of 43 hours of recorded samples were collected during this period, which correspond to approximately 7 terabytes of raw observations ready to be preprocessed for posterior analysis. These data samples were stored in solid-state drives right after being recorded and amplified into digital binary files. Expert neuroscientists at the WIBR were in charge of this data collection process.

3.2.2 Spike Sorting: data denoising

Once the recordings are taken, data needs to be examined with care. The greatest advantage of Neuropixel electrodes is their cost-benefit relationship: affordable probes allow the recording of large-scale neuronal activity in the mammalian brain. However, as these probes are inserted in the brain of awake animals, they are subject to potential drift due to sudden movements of the animal, thus distorting the recordings of the neurons being measured at that time. This undesired effect is known as *electrode/probe drift* and needs to be removed before proceeding with any further analysis.

Additionally, as *in vivo* recordings measure the real activity happening in the brain, an inherent noise is generated and printed into the recordings, coming from the continuous and mixed interaction of neurons. This is the reason why cell types cannot be immediately identified with Neuropixel probes, and sorting them out is a hard task that needs to be performed before attempting any machine learning task.

In the data collection pipeline required for this study, neurons' electrical behavior was initially recorded (what is known as **traces**) and subsequently amplified and digitized (Figure 3.5). Up to this point, the whole process can be summarized as a translation of a biological response to machine-readable information.

The process of cleaning the raw binary files collected to diminish the bias introduced by natural noise (due to electrode drift and neurons' interaction) is called **Spike Sorting**, a pipeline originally proposed and implemented by [Pachitariu 2016] as a free software MATLAB package called **Kilosort** (github.com/MouseLand/Kilosort2). Appendix B expands on how this process was previously carried out by neuroscientists at the WIBR so that this study was actually feasible.

The idea behind Spike Sorting is to identify which electrical traces in the recording are truly generated by independent neurons, some of which potentially belong to the five types of cerebellar cortical cells that this study aims to explore (Purkinje cells, Golgi cells, Granule cells, Mossy Fibers, Molecular Layer cells (MLI) in appendix A). This allows the neurons characterization by extracting the main features from their real behavior, which subsequently opens the possibility of developing machine learning implementations. As a

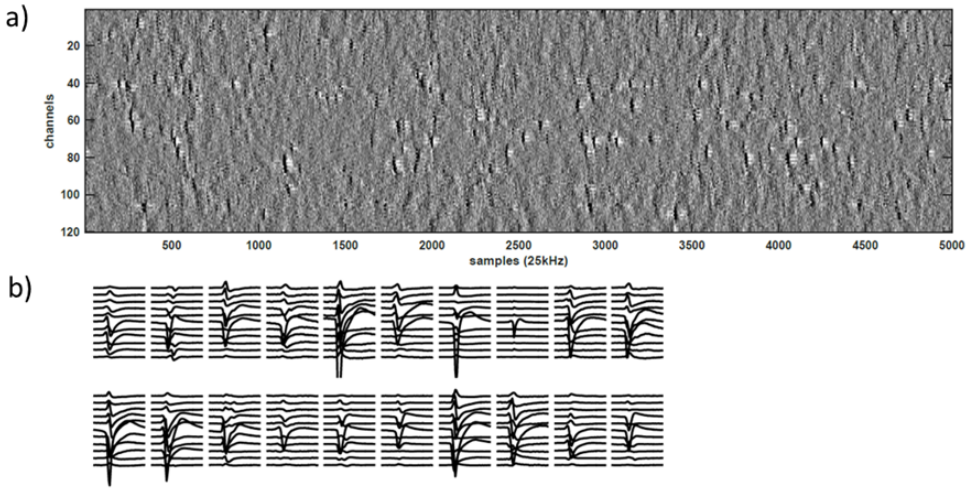

Figure 3.5: **a)** Traces of a neuron recorded on a Neuropixel probe. **b)** Amplified and digitized traces. Prominent signals are known as *spikes* after the Spike Sorting process. Adapted from [Pachitariu 2016].

result of the application of the Spike Sorting process, sorted neurons can be visualized as in Figure 3.6.

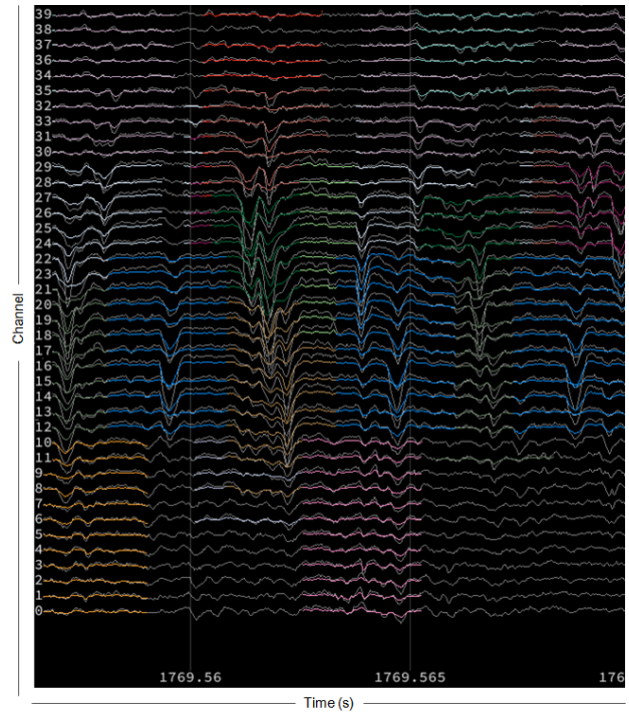

Figure 3.6: Visual inspection of the raw data collected: neurons' electrical spikes recorded with Neuropixel probes in mice. Colors represent different neurons measured by different channels according to their proximity. A neuron can be recorded by many channels, but its most pronounced spikes are registered only by the closest recording sites at the moment of firing.

3.2.3 Optogenetics: ground-truth data labeling

An important step towards the major goal of this study, build a high-precision classifier able to identify cell types in the cerebellar cortex of the mammalian brain, is to get labeled observations so that the training of a machine learning algorithm is feasible.

In general, processes for data acquisition (electrophysiological recordings) of neuronal activity are blind to the identification of cell types. While Spike Sorting is useful to identify neurons out of the brain noise, their type still remains unknown. Thus, a biological technique known as **optogenetics** was used to find ground-truth (i.e labeled) data, this is, recorded neurons that confidently corresponded to each of the main five cell classes of interest for this study, and which were intended to feed a multi-label classifier.

Optogenetics is a relatively young and novel technique that is widely used in biological and neuroscience experiments. It takes advantage of a light-sensitive protein found naturally in algae, which is inserted into the DNA of specific neurons. As a result, these cells act as photo-responsive entities in the presence of LED blue light [Deisseroth 2011], as illustrated in Figure 3.7.

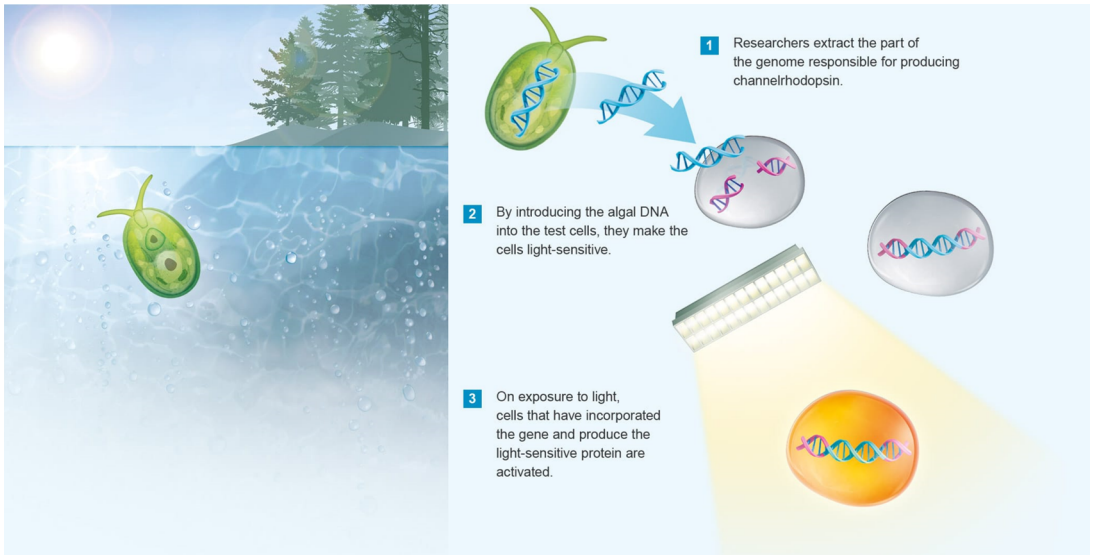

Figure 3.7: The optogenetics process. From [Bayer 2016].

A proportion of neurons corresponding to each of the five classes of cerebellar cortical cells that were recorded were genetically engineered to be photo-responsive via optogenetics. Neuroscientists at the WIBR applied LED stimulation to those cells and annotated the ID's of the responsive neurons, which will be considered as labeled data observations for the supervised classification pipeline aimed to be developed.

Neuronal data recordings were collected over continuous intervals of 60 minutes, in which only the first 20 of them are of interest for the feature extraction process that was part of this study. The remaining 40 minutes were discarded, as they corresponded to the

interval in which cells were stimulated with LED light with the only purpose of identifying responsive or labeled data points via optogenetics (Figure 3.8).

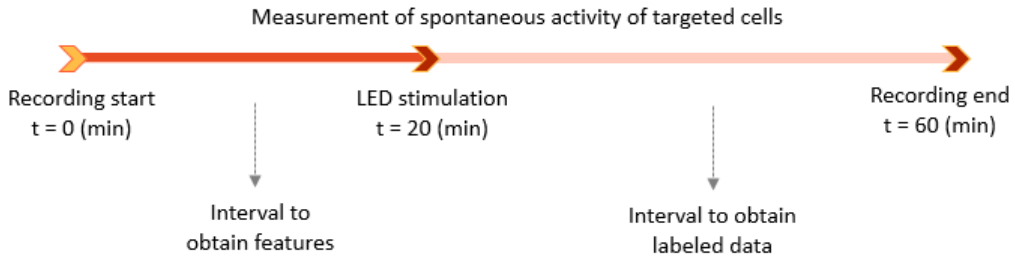

Figure 3.8: Schema of the measurement of neuronal activity over a 1-hour interval via Neuropixel recordings. The first 20 minutes of the recordings are used to extract useful information coded as features on the neurons' behavior, while the remaining 40 minutes aim to identify cells that belong to the classes of interest for this study.

The optogenetics process is time-consuming and must be carried out entirely in a biology laboratory. Due to the setback imposed by the global pandemic of Covid-19 and the subsequent closure of the facilities used for this purpose during the 2020 first semester, the number of labeled cells available for this study was considerably below the expected[2] (Table 3.1). Chapter 4 expands on the methods to overcome this insufficiency of data to feed a multi-label classifier.

Cell class	*Expected No. Responsive cells*	*Actual No. Responsive cells*
Purkinje	30	37
Purkinje (Complex Spikes)	30	15
Golgi	30	11
Granule	30	3
Molecular layer	30	4
Mossy Fibers	30	10
	180	80

Table 3.1: Expected vs. the actual number of responsive ground-truth labeled cells that were recorded and identified via optogenetics for each cerebellar cortical class. The classes Granule, Mossy Fibers, and Molecular layer cells are the most deficient ones. The Purkinje (Complex Spikes) correspond to a different electrical expression of the Purkinje cells in the brain, but they fundamentally refer to the same cell type.

[2]Following the central limit theorem [Kwak 2017], sample sizes equal to or greater than 30 are considered sufficient for the theorem to hold, which means collected data would approximately follow a Gaussian distribution and the parameter estimation for the entire population is possible.

3.3 Data preparation and processing

While the Spike Sorting process aimed to clean the data recordings so that real neurons could be identified amidst the mixed noise of frequencies inside the cerebellum, the procedure cannot identify to which cell types the neurons actually correspond. Moreover, it is unfeasible to use optogenetics to label every single neuron recorded in a single mouse, and this is why machine learning and statistical inference are useful tools for this research problem. Ultimately, the goal is to learn from labeled responsive neurons and use this knowledge to generalize and classify all recorded cells.

Even after the human intervention for sorting the neurons out, still, a proportion of them are contaminated in their frequencies due to abnormal noises introduced in them by other nearby neurons. As a result, further preparation and processing are required to finally select the neurons that are going to be analyzed by this study.

This section presents a detailed theoretical and technical description of the preparation and processing phases to which the collected recorded cells were subject. The entirety of these procedures was performed within the scope of this study and consumed more than 60% of the allocated time for this project.

3.3.1 Environment

To perform the data preparation and processing phases, the following software/hardware were used:

- Number of machines used: 1
- Operative system: x64 Windows 10 Enterprise
- Processor: Intel(R) Core(TM) i7-7600 CPU, 3.4GHz
- Available RAM: 64 GB
- Storage used: 10.2 TB (2 SSDs). The digitized *spike-sorted* raw data was 7 TB in size, and 3 additional TB were needed for intermediate processing files.
- Programming language: Python 3.7.6
- IDE: PyCharm 2019.3.4 (Professional Edition)
- Software for statistical analysis: Python 3.7.6 and RStudio 1.3.959

3.3.2 Quality check filtering process

The raw data used by this study was previously collected, cleaned, spike-sorted, curated and genetically tagged according to the reference processes mentioned earlier. Up to this point, raw binary files contained information about $\approx$ **4300** neurons, which were passed through a quality check filter to determine if they were suitable for further analysis. This filtering process was the first contribution of this study.

3.3.2.1 Peak channel computation

To carry out the quality check process, a first step consisted in the extraction of the channel (from the Neuropixel probes) in which the maximum peak (or electrical firing amplitude) was recorded for each neuron. As exemplified in Figure 3.9, for the neuron that is being recorded, channel 23 is the *peak channel* because it measured the most pronounced deflection of all the spikes coming from that specific neuron throughout the probe.

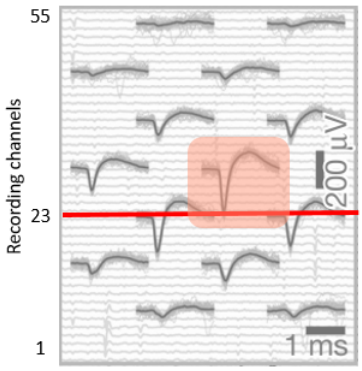

Figure 3.9: A neuron being recorded by a Neuropixel probe for which its peak channel is the 23$^{\text{rd}}$. All the channels are recording the same neuron, which *fires* several times this generating different spikes or action potentials. Adapted from [Steinmetz 2018].

Note from Figure 3.9 that a single neuron can behave differently (voltage deflections vary significantly) depending on the channel in which it is being recorded. Due do this, the computation of any quality metric or feature was always based on the neurons signature for their peak channel. The reason why a neuron deflection is maximum only in the peak channel is due to the fact that the body of that neuron was located in the proximity of the probe at that particular recording site.

To extract the peak channel of every neuron, a set of Python routines were adapted from the Python package *rtn*, developed by Maxime Beau, a PhD student at the WIBR (github.com/m-beau).

The remainder of this section will be focused on filtering out certain neurons recorded due to the observations being too noisy to use for predictive modeling, that is selecting $N \subset \mathcal{N}$. Most of these rules can be considered as a contribution of this study derived by a collaborative process with the neuroscience experts available at the WIBR.

3.3.2.2 Spikes threshold

As mentioned earlier, typical neuron recordings using Neuropixel probes last 1 hour. However, only the first 20 minutes of those recordings are considered for feature extraction purposes. These initial interval should be significant enough to measure the real behavior

of each neuron, but unfortunately, not all the recorded neurons show a sufficient level of activity during this interval, the reason why **a minimum of 300 spikes was required for a neuron to qualify for further analysis under this criterion**.

3.3.2.3 Autocorrelograms and refractory period violations

The second quality metric extracted for each neuron was based on its *autocorrelogram*. Recall from Figure 2.9 a series of spikes (action potentials) recorded for a neuron. As illustrated in Figure 2.10, the inter-spike interval histogram shows the times between those consecutive action potentials, which also happen to be correlated, the reason why after a maximum spike occurs (as in the peak channel from Figure 3.9), the electrical voltage is replicated in the form of a lag for future spikes.

The intuition behind autocorrelation aims to explain the similarity between observations as a function of the time lag that separates them. Autocorrelation, also known as serial correlation, is the correlation of a given signal (a neuron spike) with a delayed copy of itself (previous spikes).

In the context of this study, an autocorrelogram presents the autocorrelation of neuron spikes through time (usually measured in milliseconds). Thus, the highest peaks of an autocorrelogram represent the points in time at which more spikes per second occurred. Additionally, it is possible to tell from an autocorrelogram if the firing pattern of a neuron is sporadic or bursting.

Spike autocorrelograms also show an interesting property of neurons, known as the **refractory period** (Figure 3.10). During this interval, neurons are incapable of generating more spikes, which means that regardless of the stimulus they may be receiving from the external environment (neurotransmitters coming from nearby neurons), they cannot emit a response whatsoever. The refractory period can be understood as a *fatigue interval* that occurs between 1 and 4 milliseconds from the point zero of autocorrelation and it is useful to recognize if the spikes being analyzed actually come from a single neuron. If an autocorrelogram does not have a refractory period, it is likely that the spikes from two different neurons are being mixed (this can be avoided with a correct Spike Sorting process).

The refractory period can be used to estimate how contaminated the firing pattern of a neuron is by counting the number of spikes happening during this lapse. The number of violating spikes relative to the total number of spikes of the neuron is a metric known as **the *refractory period violation proportion*, which this study limited to 5% so that only the cleanest neurons passed this filter for further analysis.** Algorithm 2 summarizes the procedure to obtain this quality metric.

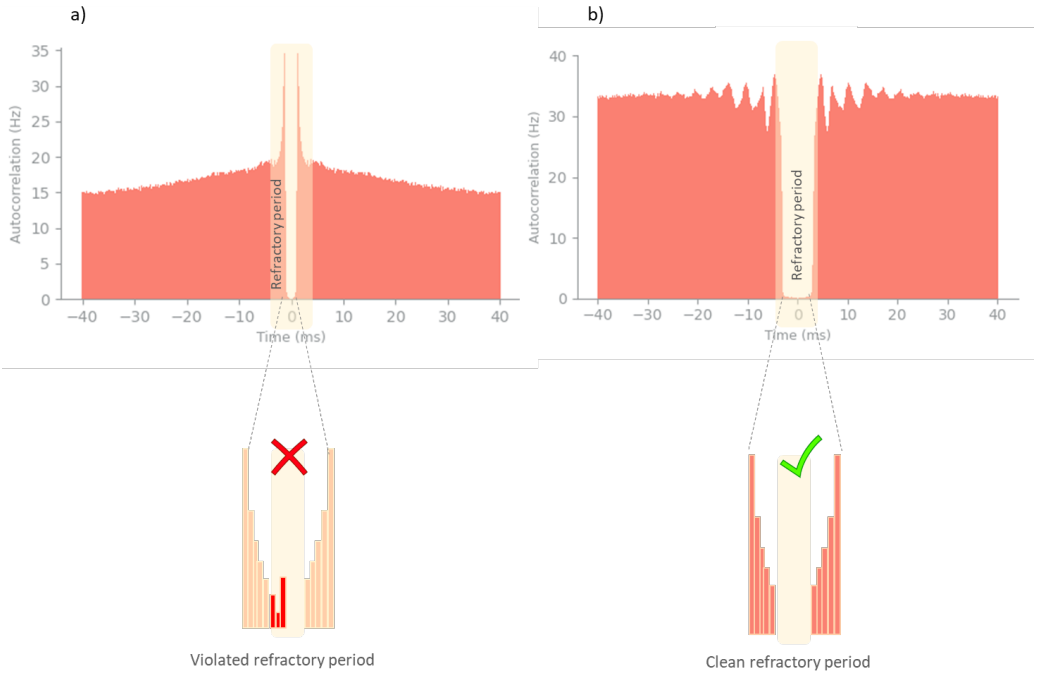

Figure 3.10: Autocorrelograms of two cerebellar cortical cells. **a)** The autocorrelated spike pattern suggests that this neuron fires sporadically (given the presence of only two high peaks). Note that zooming on the refractory period from the autocorrelogram shows the presence of some inter-spike intervals situated in this zone, which means some violating spikes (e.g generated from external noise or neighbor neurons) were incorrectly associated to this neuron. If the proportion of these violating intervals with respect to the total inter-spike intervals of the autocorrelogram was over 5%, the neuron was discarded for further analysis. **b)** As opposed to the first neuron, this cell presents a higher burstiness and tends to spike more often. A careful examination of the refractory period suggests no violations, which means all the spikes of this neuron are truly generated from its own activity.

Algorithm 2 Neurons filtering based on their proportion of refractory period violations

input : A list of n neurons

output: A list of n neurons and their corresponding refractory period violation proportion

Initialize interval to detect violations: 1 ms from the 0 point

for neuron i **to** n **do**

  $ACG \leftarrow$ compute ACG histogram

  $rpv \leftarrow$ number of spikes within the refractory period window

  $totalSpikes \leftarrow$ neuron total number of spikes measured along the entire ACG

  $rpv_{proportion} \leftarrow$ relationship between rpv and totalSpikes

  if $rpv_{proportion} \leqslant 0.05$ **then**

    $selectedNeuron \leftarrow 1$

  else

    $selectedNeuron \leftarrow 0$

  end

end

The refractory period violation proportions for the neurons of interest were computed based on the open-source Python package *rtn* developed by Maxime Beau.

3.3.2.4 Proportion of missing spikes

The previous two conditions were checked on the whole 20-minute interval for each neuron. From this point, the quality check pipeline became more granular and **the recording interval for every neuron was split into 20 chunks of 1-minute length that were analyzed separately**. The reason to do so lies in the effect of the *electrode drift* phenomenon mentioned in the section dedicated to Spike Sorting: data denoising. This study wanted to capture the time in which neurons behaved consistently and were not affected by any sort of external movement that could potentially capture their signals in a distorted way. Therefore, only chunks that proved to not contain traces of electrode drift were considered for further analysis. The quality metric below was used to assess this condition.

The metric *proportion of missing spikes* makes reference to the proportion of spikes in a 1-minute chunk that was lost (not measured) due to a sudden movement of the Neuropixel probe that caused it to go away from the neuron that was being recorded at a particular channel and time. As illustrated in Figure 3.11, the blue points represent different measures of spike amplitudes (voltage deflections) over time. A closer look at these observations, reveals that they were experiencing critical drift during the recording of the first 500 seconds, information that is as well reflected in the histograms of these data points, which account for the number of spikes that were missing considering their amplitudes follow a Gaussian distribution [Khonsary 2016].

The threshold to define the number or proportion of missing spikes is arbitrary. For this study, it was known as **the peak detection limit, set as the first percentile of the fitted amplitudes for a given batch. 1-minute batches with a proportion of missing spikes (relative to the total spikes) higher than 30% were rejected**

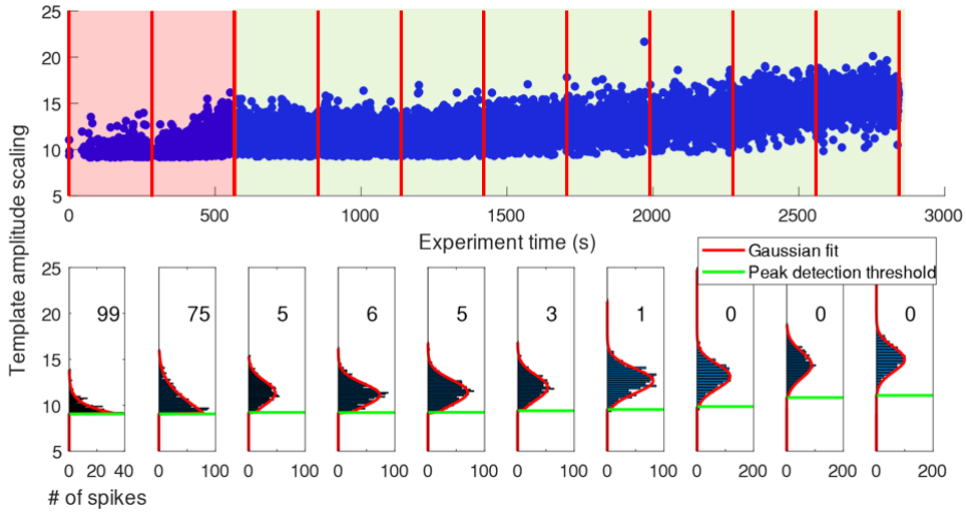

Figure 3.11: The undesired presence of electrode drift in electrophysiological recordings can be tracked by fitting a Gaussian distribution to the spike amplitudes in every batch of time. As a result, curves with more missing spikes than a given threshold are discarded for further analysis. In this example, the number of missing spikes was used as a filtering criterion, while in this study it was rather the proportion of missing spikes the metric of interest.

for further analysis.

The procedure to measure the proportion of missing spikes was based on the theoretical framework of [Hill 2011] and adapted from the MATLAB implementation developed by Julie Fabre (github.com/Julie-Fabre/), a PhD student in neuroscience at the Cortexlab (Universiy College London). The detailed steps to perform this quality metric are illustrated in algorithm 3.

In general, every chunk of 1-minute collected amplitudes was fitted a Gaussian curve of the form:

$$f(x) = A * e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{3.1}$$

With A being the amplitude of the Gaussian curve, which is initialized with the mode of the histogram of every chunk and then used in the fitting process to output a set of truly Gaussian parameters: A, μ, σ and the threshold cut initialized as the first percentile of the amplitudes used such that:

$$f(x) = \begin{cases} \text{f(x)} & \text{if x} \geqslant threshold_{cut} \\ 0 & \text{if x} < threshold_{cut} \end{cases}$$

Consider the amplitudes being analyzed as a random normal variable X such that $X \sim N(\mu, \sigma^2)$. Then, the Cumulative Distribution Function (CDF) of its standard normal distribution is denoted by:

$$F_X(x) = P(X \leqslant x) = \phi(\frac{x-\mu}{\sigma}) \tag{3.2}$$

Once the Gaussian is fitted to the batch amplitudes, the CDF is found at the limit imposed by the peak detection threshold. The resulting probability can be interpreted as the proportion of missing spikes.

Figure 3.12 shows the output of the process described to compute the proportion of missing spikes for two different time batches recorded for two neurons. Yellow curves represent the fitted Gaussian distributions, which explain the process described by the amplitudes plotted as gray histograms. The threshold cut was drawn as a red dotted line that represents the fitted first percentile of the original amplitudes. The blank area to the left of this line accounts for the missing spikes to complete the Gaussian distribution of amplitudes expected for that particular neuron.

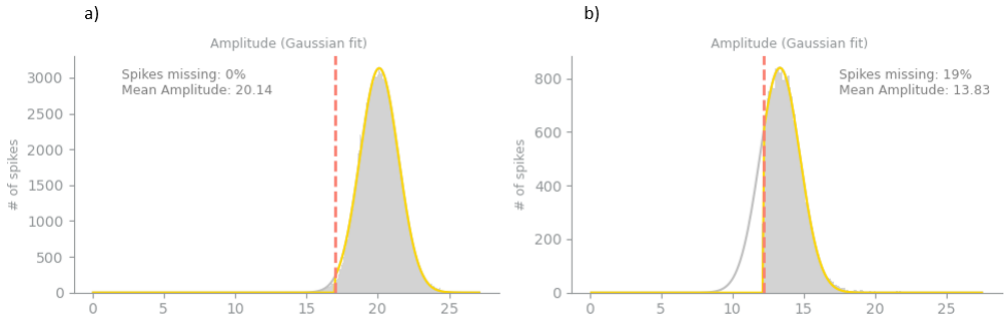

Figure 3.12: Visual representation of the proportion of missing spikes in two neurons. **a)** The ideal case in which electrode drift did not occur. All the measured spikes and their corresponding amplitudes drew a complete Gaussian and led to 0% missing spikes. **b)** Case in which 19% of the potential spikes of that particular recording were missed due to electrode drift. Still, these data observations are considered for further analysis because the missing threshold (30%) was not exceeded.

Algorithm 3 Neuron filtering based on their proportion of missing spikes

input : A list of n neurons

output: A list of m 1-minute chunks per neuron with their proportion of missing spikes

```
for neuron i to n do
    for chunk j to m = 20 do
        hist ← histogram of amplitudes
        mode ← hist mode amplitude
        binMode ← hist bin containing mode
        std ← hist standard deviation
        threshold ← hist first percentile
        params ← [mode, binMode, std, threshold]

        GaussianParams ← GaussianCurveFit(params)
        MissingSpikes ← GaussianCDF(GaussianParams[threshold])
        if MissingSpikes ⩽ 0.3 then
            selectedChunk ← 1
        else
            selectedChunk ← 0
        end
    end
end
```

3.3.2.5 Chunks pre-selection

Up to this point, the limits imposed by the refractory period violations, the minimum number of spikes per neuron, and the proportion of missing spikes per time chunk, allowed the filtering of data observations as represented by Figure 3.13.

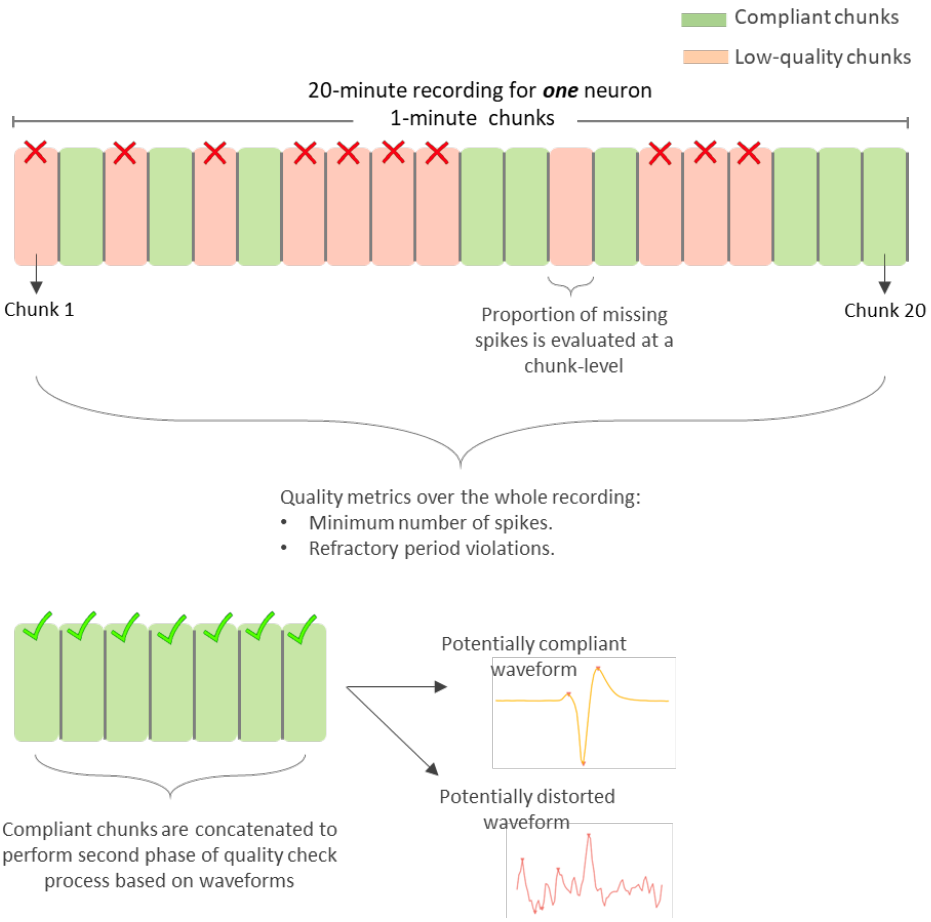

Figure 3.13: The quality check process that was performed on every neuron of interest filters both on a cell and a time chunk level.

The previous quality check processes that were performed filtered on a neuron and chunk basis, and focused mainly on attributes of the electrical spikes to assess an observation as either appropriate or not. Neurons that fulfilled the previous conditions were allowed to move into the next phase of the quality check pipeline, in which careful analysis of the waveforms of the action potentials was carried out.

3.3.2.6 Waveform analysis

As stated in Chapter 2, spikes (also known as action potentials) are electrical gradients measured inside the brain's activity and represent the way in which neurons communicate with each other. The voltage expressed by a neuron through its spikes can be represented as a signal, and as such, it can be drawn as a waveform in two dimensions (Figure 2.5).

In particular, this study was interested in analyzing quality waveforms that generate from the standard expected behavior of the neurons of interest. Figure 3.14 presents an overview of examples of acceptable waveforms as well as illustrations of signals that were severely affected and could not be considered for further analysis.

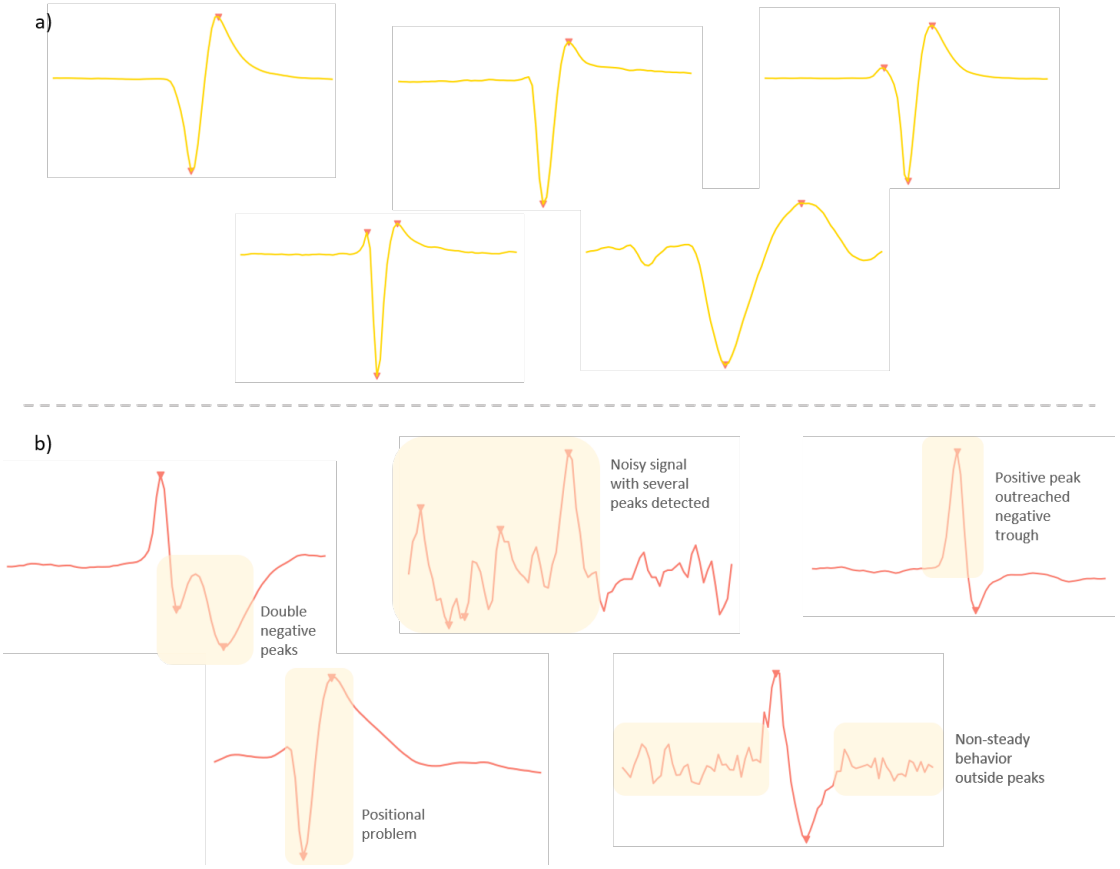

Figure 3.14: Examples of real spike waveform signals generated by different neurons in the cerebellar cortex, analyzed and filtered by this study. **a)** Genuine waveforms present a clearly defined negative voltage deflection that happens approximately half-way between the start and the end of the signals. Also, a quasi-steady behavior is observed before and after the spike formation, which also comes with the presence of small positive bumps around it. **b)** Distorted waveforms can show more than one negative deflection, a noisy series of peaks, a positive peak higher in amplitude than the negative trough, spikes happening in the wrong times, amongst others.

To determine the suitability of the waveforms of the gathered data, the first step consisted in the concatenation of the spikes generated by every neuron's compliant chunks, as showed in Figure 3.13. Concatenated spikes were represented by a NumPy array (in

Python), on top of which peak detection was performed [Oliphant 2006]. Identify the highest and lowest points of the signals generated by the set of all relevant spikes is necessary to detect the voltage deflections (spikes) that are supposed to characterize the behavior of the neurons. For this purpose, the technical implementation of this study was based on the peak detector developed by Eli Billauer and available as an open-source package in MATLAB (billauer.co.il/peakdet.html). Figure 3.14 shows the detected peaks of real neurons analyzed by this study, as small red triangles. A neuronal signal must present at least and at most one negative half-way deflection, and no more than two positive peaks surrounding the negative trough. Moreover, when positive peaks outreach the negative ones in amplitude, the neuron shall be discarded because this behavior is unlikely to be produced nearby the body of the cell (as can be observed in the top right corner waveform in Figure 3.14-**b.**

Additional to the number of peaks, which should not exceed three, for a signal to be considered for further analysis it must have its negative deflection approximately half-way between the start and the end of the waveform. This is, usually between milliseconds 1 and 1.67. Figure 3.15 shows an example of a distorted neuron according to this criterion.

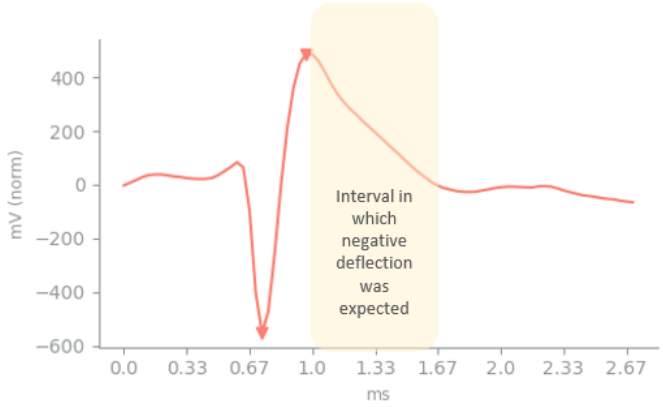

Figure 3.15: An example of a neuron with a negative deflection that occurred too soon.

Algorithm 4 Neurons filtering based on their spike waveforms

input : A list of n neurons, a list of m 1-minute compliant chunks per neuron
output: A list of n neurons and their waveform features

for neuron i **to** n **do**
  ConcatenatedChunks ← Concatenate m compliant chunks spikes
  peaks ← detect waveform peaks
  peaksCount ← count the number of total peaks

  if peaksCount $\leqslant 3$ **then**
    peaksCountCheck ← 1
  else
    peaksCountCheck ← 0
  end

  negPeak ← peaks[lowest deflection]
  posPeak ← peaks[highest peak]

  if abs(negPeak) >abs(posPeak) **then**
    negPeakHigher ← 1
  else
    negPeakHigher ← 0
  end

  if $1 \geqslant$ *negPeak[time(ms)]* $\leqslant 1.67$ **then**
    negPeakPos ← 1
  else
    negPeakPos ← 0
  end

  if negPeakHigher == 1 & peaksCountCheck == 1 & negPeakPos == 1 **then**
    goodWaveform ← 1
  else
    goodWaveform ← 0
  end
end

3.3.3 Summary statistics

The waveform analysis of neurons is the second and last phase of the quality check process that aimed to gather the correct data for further analysis. Algorithm 4 presents an overall view of this process. As a result of the application of the quality check pipeline, a summary chart and a *csv* file were generated for every neuron, as in the example presented in Figure 3.16.

The implementation of the quality check pipeline allowed filtering out 2848 neurons that did not comply with the minimal conditions for further analysis. As a result, **1477 neurons out of 4328 (34%)** were selected as good representatives for the next phase of the study, detailed in Chapter 4. The technical implementation of the quality check metrics was done using Python 3.7 and modules such as *numpy* and *scipy.stats* [Oliphant 2006,

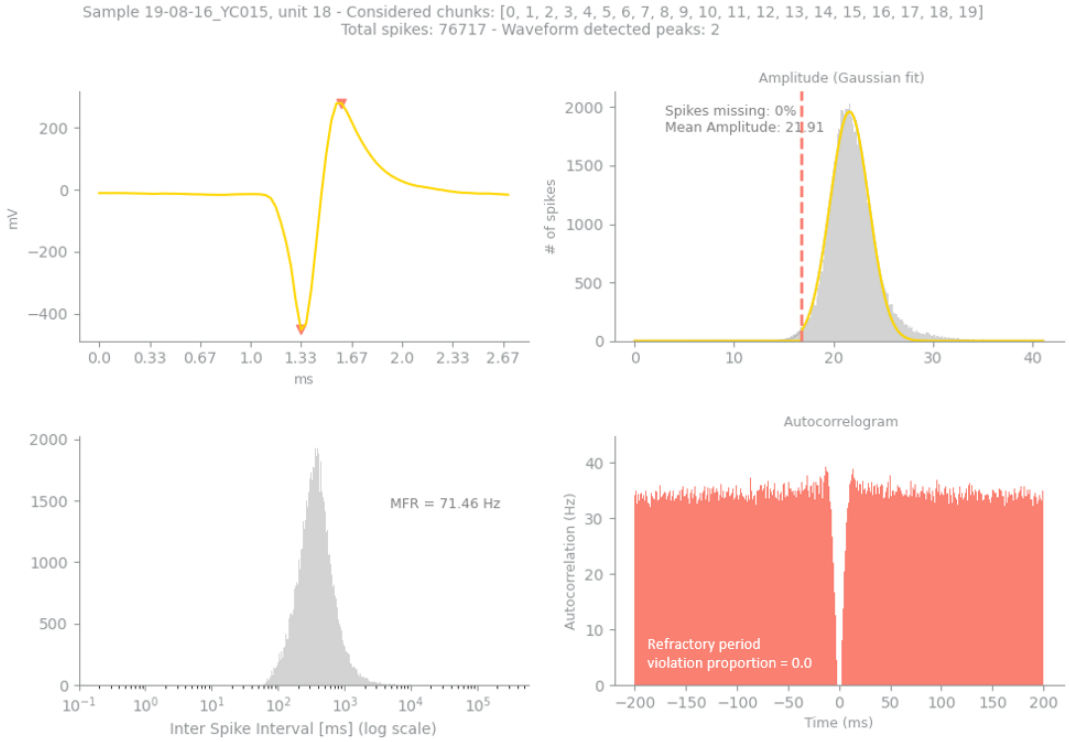

Figure 3.16: Illustration of the summary chart generated for a specific neuron in one of the recordings considered by this study. The report contains the essence of the quality metrics and some other interesting measures not used for filtering purposes, such as the Mean Firing Rate (MFR) computed from the inter-spike interval.

Virtanen 2020]. The codes are available in the [QualityCheck] repository link.

3.4 Feature extraction

Before diving into machine learning methods, the next step consisted in the computation of the modeling features that were used to separate and subsequently predict cerebellar cortical cell classes. A total of 28 features, previously introduced in the related work presented by [Ruigrok 2011, Hensbroek 2014, van Dijck 2013, Özcan 2020], were computed in this study to explain how the firing pattern of these neurons behave, as well as their irregularities.

3.4.1 Mathematical notation

For the remainder of this chapter there will be some notation introduced. Let $\mathcal{N}$ be the set of all recorded and spike-sorted neurons for which each neuron $n \in \mathcal{N}$ has two associated vectors of length k_n. These vectors will be called $T_n, A_n \in \mathbb{R}^{k_n}$ where T_n is a vector of timestamps and A_n a vector of electrical spikes' amplitudes at each timestamp. The elements of T_n and A_n will be denoted by $t_{n,i}$ and $a_{n,i}$ such that $T_n = [t_{n,1}, \ldots, t_{n,k}]$ and $a_{n,1}$ is the amplitude of the spike recorded at the time $t_{n,1}$ (measured in milliseconds) for neuron n.

3.4.2 Firing pattern modeling features

After applying the quality check pipeline the total number of recorded neurons in $\mathcal{N}$ (4328) was reduced to 1477 which will be denoted by the set $N \subset \mathcal{N}$. Recall that each neuron $n \in N$ has the associated vectors T_n and A_n of timestamps and amplitudes.

Recall also from Chapter 2 that a crucial vector for feature extraction is the one describing the inter-spike intervals, on top of which a probability distribution is fitted to their histogram, also referred to as ISIh. This vector will be denoted as $\Delta T_n = [\delta_{n,1}, \ldots, \delta_{n,k_n-1}]$ such that $\delta_{n,1} = t_{n,2} - t_{n,1}$ (the time difference between the spikes occurring at timestamps $t_{n,2}$ and $t_{n,1}$). For practical purposes, d_n will denote the cardinality of such vector of intervals ΔT_n, $\#\Delta T_n = k_n - 1 = d_n$.

The following features describe the nature of the firing of a neuron. They are known as electrical-based (or simply electrical) or firing pattern modeling features.

- *Mean Firing Rate* (MFR): defined as the average spiking rate of a neuron n, in Hz (spikes per second)[3]. It is computed as the number of inter-spike intervals d_n divided by the time difference between the first and the last spike's timestamp. The time difference between the first and the last spike, $t_{n,k-1} - t_{n,1}$, can be calculated as the sum of the d_n inter-spike intervals.

$$MFR_n = \frac{(k_n - 1)}{t_{n,k} - t_{n,1}} = \frac{d_n}{\sum_{i=1}^{d_n} \delta_{n,i}} \tag{3.3}$$

- *Mean Instantaneous Frequency* (MIFR): it is computed as the mean of the instantaneous frequencies described by the lengths of the inter-spike intervals:

$$MIFR_n = \frac{1}{d_n} \sum_{i=1}^{d_n} \frac{1}{\delta_{n,i}} \tag{3.4}$$

- *Median of the ISIh*: this is a relevant feature that helps to describe the shape of the positively skewed cumulative distribution function that determines the behavior of the inter-spike intervals of a neuron. Let an ordered set of the inter-spike intervals be defined as $G_n = \{g_{n,i} \in \Delta T_n : \delta_{n,i} \leqslant \delta_{n,i+1}\}$. The median of a series of d_n inter-spike intervals is defined as:

$$Median_n(\Delta T_n) = \begin{cases} g_{n,\frac{d_n}{2}+0.5}, & \text{, for odd } d_n \\ \frac{g_{n,\frac{d_n}{2}} + g_{n,\frac{d_n}{2}+1}}{2} & \text{, for even } d_n \end{cases}$$

- *Mode of the ISIh*: suppose there is a function that takes the values of a vector and returns a histogram with y bins $p : \mathbb{R}^x \times \mathbb{Z} \to \mathbb{R}^{y+1} \times \mathbb{Z}^y$.

[3] Prior to obtain this metric, ISI's are divided by the Neuropixel sampling frequency (30000 Hz) to obtain seconds from milliseconds.

If ΔT_n is inserted into function p for a given number of bins, which in this study are defined as $y = \sqrt{d_n}$ [Anderson 2020], then a vector of frequencies per bin of the histogram results, for example B. Thus, the mode of the ISIh is given by the lower limit of:

$$Mode_n(\Delta T_n) = argmax(B)_{LB}$$

- *Burstiness of firing*: computed as the 5th percentile of the inter-spike intervals histogram (ISIh), the burstiness refers to the tendency of neurons to fire action potentials in close temporal succession, so that the probability of a spike just after a spike has happened is relatively higher. A low percentile value for a given firing rate indicates more burstiness.

  Suppose that the probability distribution described by the vector of inter-spike intervals ΔT_n is given by $p(\Delta T_n)$. The 5th percentile of the ISIh is given by the $\delta_{n,i}$ inter-spike interval for which 5% of the inter-spike intervals are smaller: $p(\Delta T_n < \Delta T_{n,5\%}) = 0.05$

- *Skewness of the ISIh*: for normally distributed data, the skewness is approximately zero, while for unimodal continuous distributions its values will be greater than zero, indicating that more weight is in the right tail of the distribution[4].

3.4.2.1 Modeling features to describe firing pattern irregularities

It is equally important to assess the amount of irregularity that the firing behavior of a neuron presents, as electrical features may be impacted by external conditions that deviate their normal signature. For such purpose, the following predictors are well known:

- *Entropy of the log-ISIh*: a measure to estimate the variability of the inter-spike intervals. As the entropy captures the randomness of neuron spikes, low entropy means that neuron fires regularly, while a large entropy indicates a neuron fires more irregularly [Ramon 1911, Bhumbra 2005]. The entropy of the log of the ISIh is computed as:

$$Ent_n(\Delta T_n) = -\sum_{i=1}^{\delta_n} p(\delta_{n,i}) log_2 p(\delta_{n,i}) \tag{3.5}$$

- *Coefficient of variation* (CV): defined as the standard deviation of the ISI's divided by the mean ISI, it is an estimate of the variability of the firing pattern of a neuron when its ISI's come from different mean firing rates:

$$CV_n = \frac{std_n(\Delta T_n)}{\overline{\Delta T_n}} \tag{3.6}$$

[4] In this study, skewness was measured as the Fisher-Pearson coefficient of skewness implemented in the Python library *SciPy* [Virtanen 2020].

- *Average coefficient of variation for a sequence of 2 ISI's* (CV_2): the CV_2 value between 2 consecutive ISI's is the ratio of their standard deviation to their average, multiplied by a factor of $\sqrt{2}$:

$$CV_{2,n} = \sqrt{2}\frac{|\ \delta_{n,i+1} - \delta_{n,i}\ |/\sqrt{2}}{(\delta_{n,i+1} + \delta_{n,i})/\sqrt{2}} = \frac{2\ |\ \delta_{n,i+1} - \delta_{n,i}\ |}{\delta_{n,i+1} + \delta_{n,i}} \tag{3.7}$$

  The average CV_2 is obtained as:

$$\overline{CV_{2,n}} = \frac{1}{d_n + 1}\sum_{i=1}^{d_n}\frac{2\ |\ \delta_{n,i+1} - \delta_{n,i}\ |}{\delta_{n,i+1} + \delta_{n,i}} \tag{3.8}$$

  As explained by [Holt 1996], an average of CV_2 on all consecutive pairs of ISI's estimates the intrinsic variability of a spike in a neuron.

- *Local variation* (Lv): represents another measure of the variability of the spikes of a neuron. This metric is proportional to the average coefficient of variation for a sequence of 2 ISI's (CV_2):

$$L_{v,n} = \frac{3}{d_n - 1}\sum_{i=1}^{d_n}\left(\frac{\delta_{n,i} - \delta_{n,i+1}}{\delta_{n,i} + \delta_{n,i+1}}\right)^2 = \frac{3}{d_n - 1}\sum_{i=1}^{d_n}\left(1 - \frac{4\delta_{n,i}\delta_{n,i+1}}{(\delta_{n,i} + \delta_{n,i+1})^2}\right) \tag{3.9}$$

- *Geometric average of the re-scaled cross-correlation of ISI's* (S_V): to compute this metric, the ratio of the geometric mean and the arithmetic mean of the logarithm of the ISI's is first computed. Subsequently, these ratios are averaged:

$$S_{V,n} = -\frac{1}{d_n - 1}\sum_{i=1}^{d_n}\frac{1}{2}log\left(\frac{4\delta_{n,i}\delta_{n,i+1}}{(\delta_{n,i} + \delta_{n,i+1})^2}\right) \tag{3.10}$$

  The lower the S_V metric, the more homogeneous the ISI's, and therefore fewer irregularities are associated with the neuron spiking pattern.

- *The instantaneous irregularity* (IR): computes the average of the absolute difference between the natural logarithms of 2 consecutive ISI's:

$$IR_n = \frac{1}{d_n - 1}\sum_{i=1}^{d_n}|\ log_e\left(\frac{\delta_{n,i+1}}{\delta_{n,i}}\right)\ | \tag{3.11}$$

- *Revised local variation* (LvR): As Lv, S_I, CV_2, and IR are somewhat dependent on firing rate fluctuations, LvR is an adaptation of the Lv in which a natural random fluctuation α (usually set to 1 ms) is assumed for all the spikes of a neuron:

$$LvR_n = \frac{3}{d_n - 1}\sum_{i=1}^{d_n}\left(1 - \frac{4\delta_{n,i}\delta_{n,i+1}}{(\delta_{n,i} + \delta_{n,i+1})^2}\right)\left(1 + \frac{4\alpha}{\delta_{n,i} + \delta_{n,i}}\right) \tag{3.12}$$

3.4.3 Waveform modeling features

As introduced by the work presented by [Jia 2019, Özcan 2020], this study also considered the computation of features based on the waveforms described by neurons' spikes or action potentials in a bi-dimensional plane. Some additional variables that were formulated in this direction were inspired by the experience of neuroscientists at the WIBR. A comprehensive list of these waveform-based features follows:

- *Positive peak amplitude* (MaxAmp): this metric corresponds to the value of the maximum amplitude of a neuron spike measured in millivolts (mV) and normalized with respect to the minimum amplitude of the neuron waveform:

$$MaxAmp_n = max(A_n) = a_{n,m_{(+)}}$$

- *Positive peak amplitude time* (MaxAmpTime): measured as the time in milliseconds (ms) at which the positive peak amplitude occurred:

$$MaxAmpTime_n = t_{n,m_{(+)}}$$

- *Negative peak amplitude* (MinAmp): usually known as the trough, this metric corresponds to the normalized value of the minimum amplitude of a neuron spike measured in millivolts (mV):

$$MinAmp_n = min(A_n) = a_{n,m_{(-)}}$$

- *Negative peak amplitude time* (MinAmpTime): measured as the time in milliseconds (ms) at which the minimum peak amplitude occurred:

$$MinAmpTime_n = t_{n,m_{(-)}}$$

- *Positive 10%-90% rise time* (RiseTime): measured as the time interval in milliseconds (ms) that took the neuron waveform to reach 90% of the *Positive peak amplitude* from the point at which the 10% of that *Positive peak amplitude* started.

  To compute it, consider the amplitude which is at 10% of the *Positive peak amplitude* as $a_{n,p_{0.1}} = 0.1(a_{n,m_{(+)}})$ and the amplitude which is at the 90% as $a_{n,p_{0.9}} = 0.9(a_{n,m_{(+)}})$. The corresponding time points at which these amplitudes occurred were either directly observed, or were estimated by spline interpolation[5] and let them be denoted as $t_{n,p_{0.1}}$ and $t_{n,p_{0.9}}$ respectively. Then,

$$RiseTime_n = t_{n,p_{0.9}} - t_{n,p_{0.1}}$$

  Such that,

$$MinAmpTime_n < t_{n,p_{0.1}} < t_{n,p_{0.9}} < MaxAmpTime_n$$

[5] Implemented using the univariate spline functions available in the open-source package *scipy.interpolate* in Python 3.7 [Virtanen 2020].

In the data analysed in this study all amplitudes registered between the *Negative peak amplitude* and the *Positive peak amplitude* were monotonically increasing with time[6].

- *Positive half-width duration* (PosHwDuration): as stated by [Özcan 2020], this metric corresponds to the time interval in milliseconds (ms) that took to go from the point where the *Positive peak amplitude* reaches half its height at the left of the actual *Positive peak amplitude* (p_1) to the first observation of the same amplitude at the right of the *Positive peak amplitude* (p_2).

  Consider the set of timestamps $P_n = \{t_{n,i} : a_{n,i} = 0.5(MaxAmp_n)\}$ whose elements are either observed or imputed. Next, t_{n,p_1} is defined as the value where $MinAmpTime_n < t_{n,p_1} < MaxAmpTime_n$, which holds due to the monotonically increasing amplitudes registered between these points. Also, t_{n,p_2} is defined as $t_{n,p_2} = min(\{i \in P_n : i > MaxAmpTime_n\})$.

$$PosHwDuration_n = t_{n,p_2} - t_{n,p_1}$$

- *Onset amplitude* (OnsetAmp): normalized amplitude corresponding to the point at which the neuron spike waveform reached 5% of its first peak (either positive or negative):

$$OnsetAmp_n = a_{n,onset} = \begin{cases} 0.05(MaxAmp_n), & \text{if } t_{n,m_{(+)}} < t_{n,m_{(-)}} \\ 0.05(MinAmp_n), & \text{if } t_{n,m_{(-)}} < t_{n,m_{(+)}} \end{cases}$$

- *Onset time* (OnsetAmpTime): measured as the time in milliseconds (ms) at which the neuron spike waveform reached 5% of its first peak (either positive or negative). It is either observed or interpolated for $OnsetAmp_n$:

$$OnsetAmpTime_n = t_{n,onset}$$

- *Crossing time* (CrossTime): inflection time $a_{n,0*} = 0$, either observed or interpolated, at which the neuron spike waveform changes from the negative to the positive voltages (or vice-versa):

$$CrossTime_n = t_{n,0*}$$

  Such that

$$MinAmpTime_n < t_{n,0*} < MaxAmpTime_n$$

- *Positive decay time* (PosDecayTime): measured as the time interval in milliseconds (ms) between the time of occurrence of the *Positive peak amplitude* and the time at which at which the 10% of the *Positive peak amplitude* started:

[6] It should be noted that the above equations do not hold if this is not the case, as there might be several timestamps at which the amplitude is $a_{n,p_{0.1}}$ or $a_{n,p_{0.9}}$. In practice this would be a very strange behaviour described by a neuron and expert neuroscientists would need to decide how to best capture this.

$$PosDecayTime_n = MaxAmpTime_n - t_{n,p_{0.1}}$$

- *Negative 10%-90% rise time* (FallTime): measured as the time interval in milliseconds (ms) that took the neuron waveform to reach 90% of the *Negative peak amplitude* from the point at which the 10% of that *Negative peak amplitude* started.

  To compute it, consider the amplitude which is at 10% of the *Negative peak amplitude* as $a_{n,x_{0.1}} = 0.1(a_{n,m_{(-)}})$ and the amplitude which is at the 90% as $a_{n,x_{0.9}} = 0.9(a_{n,m_{(-)}})$. The corresponding time points at which these amplitudes occurred were either directly observed, or were estimated by spline interpolation, and let them be denoted as $t_{n,x_{0.1}}$ and $t_{n,x_{0.9}}$ respectively. Then,

  $$FallTime_n = t_{n,x_{0.9}} - t_{n,x_{0.1}}$$

  Such that,

  $$MinAmpTime_n < t_{n,x_{0.9}} < t_{n,x_{0.1}} < MaxAmpTime_n$$

  Note that the $MinAmpTime_n$ is always negative and $MaxAmpTime_n$ is always positive.

- *Negative half-width duration* (NegHwDuration): corresponds to the time interval in milliseconds (ms) that took to go from the point where the *Negative peak amplitude* reaches half its height at the left of the actual *Negative peak amplitude* (p_3) to the first observation of the same amplitude at the right of the *Negative peak amplitude* (p_4).

  Consider the set of timestamps $Q_n = \{t_{n,i} : a_{n,i} = 0.5(MinAmp_n)\}$ whose elements are either observed or imputed. Next, t_{n,p_3} is defined as the value where $t_{n,p_3} < MinAmpTime_n$. Also, t_{n,p_4} is defined as $t_{n,p_4} = min(\{i \in Q_n : i > MinAmpTime_n\})$.

  $$NegHwDuration_n = t_{n,p_4} - t_{n,p_3}$$

- *End amplitude* (EndAmp): normalized amplitude at which the neuron spike waveform reached 5% of its last peak (either positive or negative):

  $$EndAmp_n = a_{n,end} = \begin{cases} 0.05(MaxAmp_n), & \text{if } t_{n,m_{(+)}} > t_{n,m_{(-)}} \\ 0.05(MinAmp_n), & \text{if } t_{n,m_{(-)}} > t_{n,m_{(+)}} \end{cases}$$

- *End time* (EndAmpTime): measured as the time in milliseconds (ms) at which the neuron spike waveform reached 5% of its last peak (either positive or negative). It is either observed or interpolated for $EndAmp_n$:

  $$EndAmpTime_n = t_{n,end}$$

- *Negative decay time* (NegDecayTime): measured as the time interval in milliseconds (ms) between the time of occurrence of the *Negative peak amplitude* and the time at which at which the 10% of the *Negative peak amplitude* started:

$$NegDecayTime_n = MinAmpTime_n - t_{n,x_{0.1}}$$

Such that,

$$MinAmpTime_n < t_{n,x_{0.1}}$$

- *Waveform duration* (WvfDur) : measured as the time interval in milliseconds (ms) between the negative and positive peak amplitudes:

$$WvfDur_n = |MaxAmpTime_n - MinAmpTime_n|$$

- *Peak-trough ratio* (PkTrRatio): index that expressed the relation between the distances from the *Positive peak amplitude* and the *Negative peak amplitude* to the baseline of the waveform.

  In this study, the baseline was measured as the median of the first 40 data points of the waveform of a neuron spike (i.e. a number close to zero):

$$Baseline_n = median(\{a_{n,i} : i \in \mathbb{Z}_{[1,40]}\})$$

Therefore,

$$PkTrRatio_n = \frac{MaxAmp_n - Baseline_n}{MinAmp_n - Baseline_n}$$

- *Depolarization slope* (DepSlope): defined by fitting a regression line to the 30 observations registered after the point at which 10% of the amplitude of the first peak (either positive or negative) occurs. In practice, the depolarization slope measures the loss of electrical potential inside a neuron so that it is getting prepared to *fire*.

  Consider the amplitude:

$$a_{n,g} = \begin{cases} 0.1(MaxAmp_n), & \text{if } MaxAmpTime_n < MinAmpTime_n \\ 0.1(MinAmp_n), & \text{if } MinAmpTime_n < MaxAmpTime_n \end{cases}$$

Now, consider a set of possible timestamps (either observed or interpolated) such that:

$$S_n = \begin{cases} \{t_{n,i} : a_{n,i} = a_{n,g} \wedge t_{n,i} < MaxAmpTime_n\}, \\ \text{if } MaxAmpTime_n < MinAmpTime_n \\ \{t_{n,i} : a_{n,i} = a_{n,g} \wedge t_{n,i} < MinAmpTime_n\}, \\ \text{if } MinAmpTime_n < MaxAmpTime_n \end{cases}$$

$t_{n,g}$ is given by $max(S_n)$ and it is the timestamp at which 10% of the amplitude of the first peak (either positive or negative) occurs.

To fit the depolarization slope, the linear slope between the points $(t_{n,g}, a_{n,g})$ and $(t_{n,g+30}, a_{n,g+30}))$ is defined as:

$$DepSlope_n = \frac{a_{n,g+30} - a_{n,g}}{t_{n,g+30} - t_{n,g}}$$

- *Repolarization slope* (RepSlope): defined by fitting a regression line between the points at which the *Positive peak amplitude* and the *Negative peak amplitude* occur. Intuitively, the repolarization slope measures the how intensely the spike of a neuron is rising.

$$RepSlope_n = \frac{MaxAmp_n - MinAmp_n}{MaxAmpTime_n - MinAmpTime_n}$$

- *Recovery slope*: defined by fitting a regression line from the point at which the last peak (either positive or negative) occurred to the point a which the 30th observation occurs right after. In practice, the recovery slope measures how fast the neuron recovers its normal electrical state after firing.

  To compute, consider the amplitude:

$$a_{n,l} = \begin{cases} MinAmp_n, & \text{if } MaxAmpTime_n < MinAmpTime_n \\ MaxAmp_n, & \text{if } MinAmpTime_n < MaxAmpTime_n \end{cases}$$

  Now, $t_{n,l}$ is given by:

$$t_{n,l} = \begin{cases} MinAmpTime_n, & \text{if } MaxAmpTime_n < MinAmpTime_n \\ MaxAmpTime_n, & \text{if } MinAmpTime_n < MaxAmpTime_n \end{cases}$$

  To fit the recovery slope, the linear slope between the points $(t_{n,l}, a_{n,l})$ and $(t_{n,l+30}, a_{n,l+30}))$ is defined as:

$$RecSlope_n = \frac{a_{n,l+30} - a_{n,l}}{t_{n,l+30} - t_{n,l}}$$

- *Tau of end slope* (Tau): represented by the time constant of an exponential decay fitted to the tail of the waveform, i.e. the 30 observations that follow the last peak of the neuron spike (either positive or negative). The time constant τ is the amount of time that an exponentially decaying quantity takes to decay by a factor of $1/e$. In other words, the time consumed by the original quantity (e.g. a neuron spike) to decay approximately 36.8% (e.g. a registered maximum amplitude).

  Consider the function which fits an exponential decay to a set of points and returns the time constant parameter $\tau \in \mathbb{R}$ which comes from a function $expfunc : \mathbb{R}^{30} \to \mathbb{R}$

and the feature of interest is $\lambda = expfunc(\{a_{n,i} : i \in \mathbb{Z}_{[l,l+30]}\})$. Once the rate parameter λ is obtained[7], the time constant τ can be defined as:

$$\tau = \frac{1}{\lambda}$$

Figure 3.17 illustrates the modeling features used for waveform description.

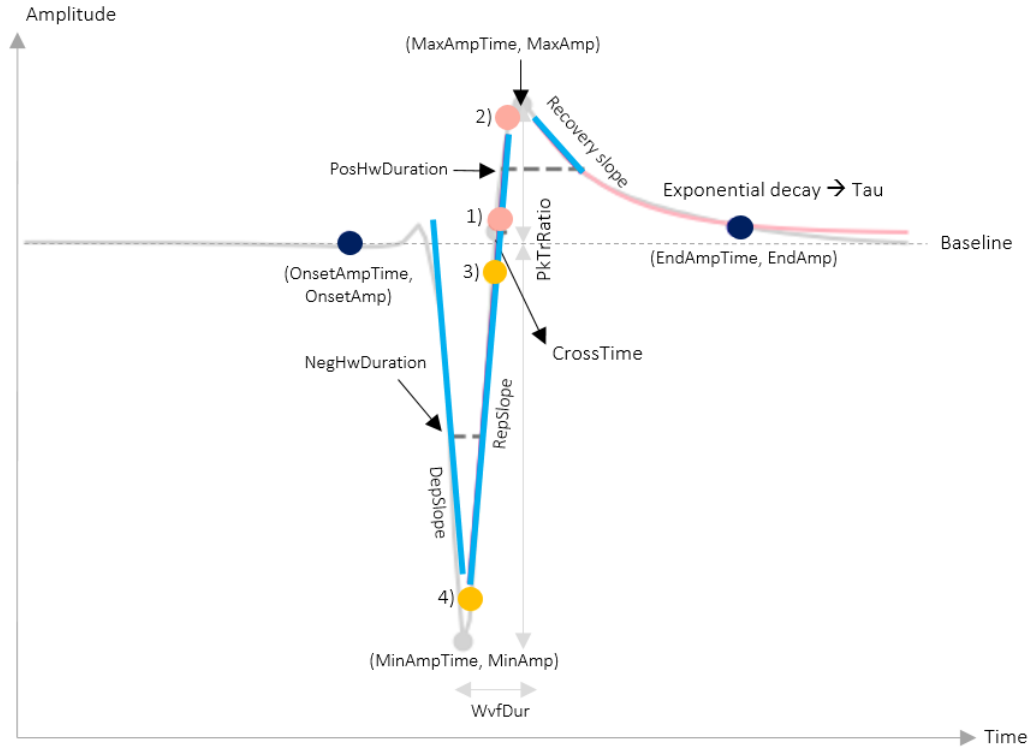

Figure 3.17: Visual representation of some waveform-based features computed by this study to characterize cerebellar cortical cells. The horizontal difference (in the time axis) between the points 1) and 2) represents the $RiseTime_n$. Similarly, the time difference between the points 3) and 4) stands for the $FallTime_n$.

The computation of the previous features was done using Python 3.7. The codes are available in the [FeatureExtraction] repository link. As a result of the feature extraction process, a summary chart and a *csv* file were generated for every good-quality neuron, as shown in Figure 3.18.

[7]Computed from an exponential fit implemented via the function *optimize.curve_fit* available in the open-source package *scipy.interpolate* in Python 3.7 [Virtanen 2020].

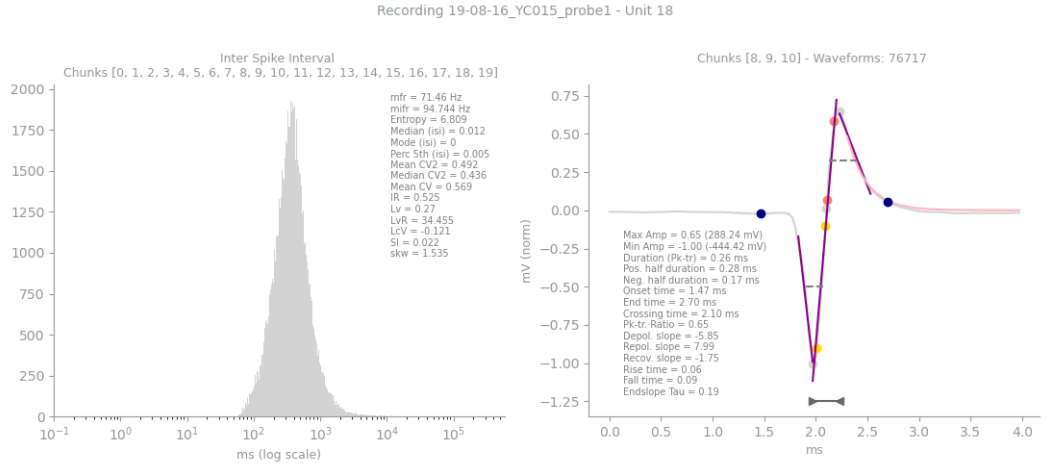

Figure 3.18: Plot of the summary chart generated for a specific neuron in one of the recordings considered by this study, in which the main computed features are presented. Figure computed via the package *matplotlib* available in Python 3.7 [Hunter 2007].

Chapter 4

Methods, Results and Findings

The ultimate goal of this study aims to contribute to a wider comprehension of the ways in which the mammalian cerebellum internal structure works. Several knowledge layers will have to be unveiled before attempting to conclude major gains in this understanding process, and a complete characterization and identification of the neurons that live in this brain tissue are fundamental tasks that need to be completed towards the attainment of that objective.

At first, when this study was proposed, it was unclear whether supervised or unsupervised learning was more appropriate to tackle the challenge of understanding how the electrical properties and the waveforms described by cerebellar cortical neurons actually represent boundaries between these cell classes. However, given the absence of sufficient labeled data, both approaches were used as part of a complementary methodology.

This chapter describes how unsupervised learning techniques allowed the identification of one of the most important cerebellar cortical neuron types: the Purkinje cells. Prior knowledge regarding these neurons could be confirmed by their spontaneous electrical activity. Also, some boundaries seemed to exist to presumably separate Granule cells, and some Mossy Fiber representatives. Nevertheless, unsupervised clustering also demonstrated that the features considered by this study were insufficient to confidently isolate Golgi and MLI cells, as well as most of the Mossy Fibers, at least in the traditional Euclidean space.

While unsupervised learning is a great tool for pattern discovery, the possibility of building a high precision classifier to predict future cerebellar cortical cells is out of the scope for this methodology and is a matter of great importance to answer one of the key research questions of this study. For this purpose, supervised learning is more suitable, yet infeasible due to the remarkable absence of sufficient labeled data. To overcome this limitation, this study proposes the use of a joint strategy between these two approaches, which is described in detail in this chapter. In general, it consisted of using the valuable input from unsupervised clustering, as well as interpolation techniques as a means to increase the labeled data on top of which supervised classification was possible. Preliminary results from the best classifiers allowed to report cross-validated levels of precision above the expected minimum as benchmarked against a baseline model. Hopefully, with the collection of more labeled observations in the future, a final model can be better calibrated for its extended use.

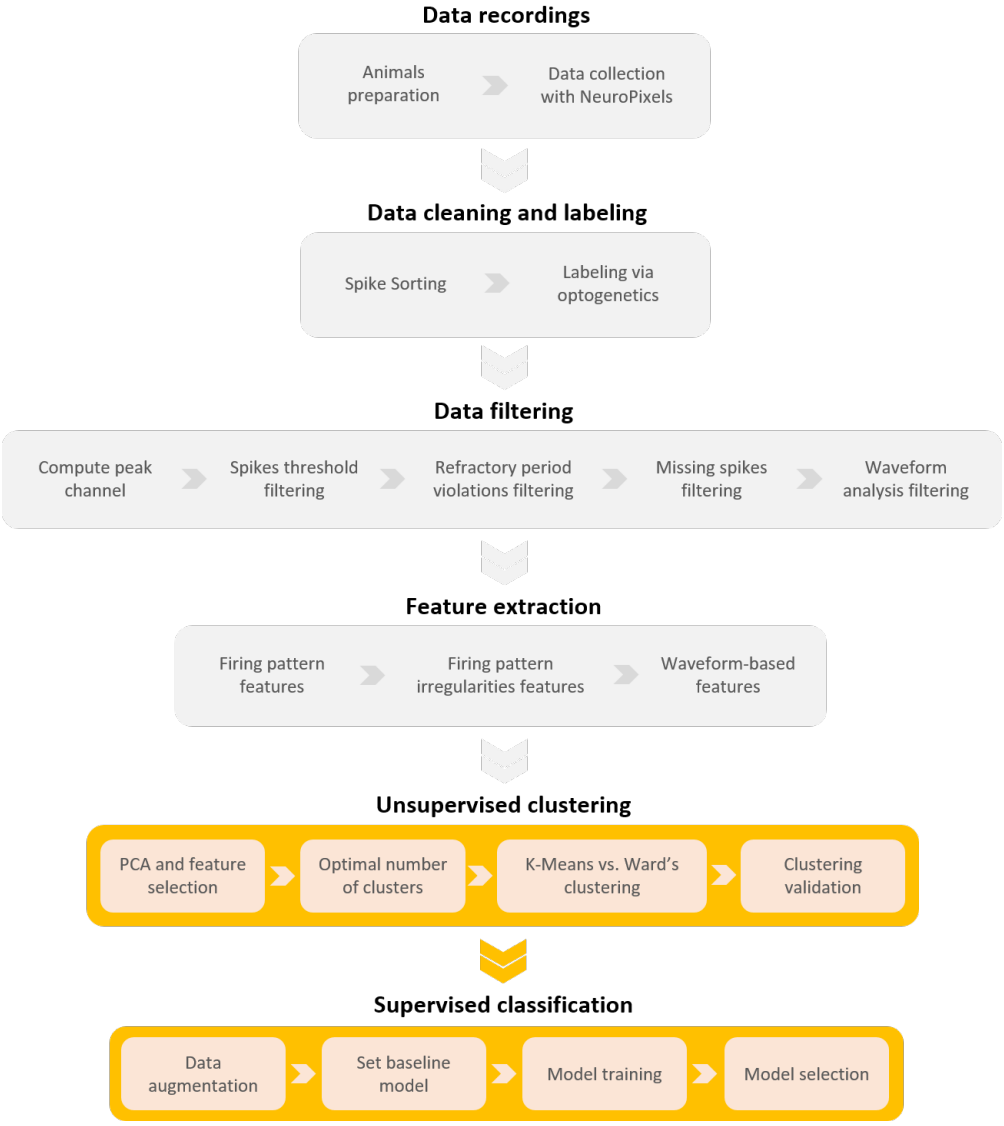

Figure 4.1: Pipeline phases covered in Chapter 4 are *Unsupervised clustering* and *Supervised classification*.

4.1 Exploratory analysis

This section aims to present exploratory analysis steps performed to identify potential features in low dimensional spaces in which natural separation occurs amongst cerebellar cortical cell types. In many practical applications, exploratory statistics serve as clustering tools themselves.

Recall from the section *Clustering Validation* in Chapter 2 that evaluating unsupervised clusters is a hard task in practice, mainly due to the subjectivity to which most of the applications are subject to. To reduce the risk of confirmation bias, one of the questions that are worth asking in the early exploratory analysis phase is: *is the data prone to contain clusters?* To answer such concern, Principal Component Analysis was used to reduce the dimensionality of the 28 computed features on 1477 neurons, as **simple scatter plots of pairs of variables did not allow a straightforward identification of any subgroups of cells.**

4.1.1 Finding potential clusters via PCA

The application of PCA[1] to the matrix of data available for this study revealed that the highest proportion of the variance contained in the samples was explained by only five features, which are presented in Figure 4.2(b). Moreover, the entire dimensional space in which the neurons lied could effectively be reduced to three principal components that contribute to the explanation of 86% of the total variance (Figure 4.3).

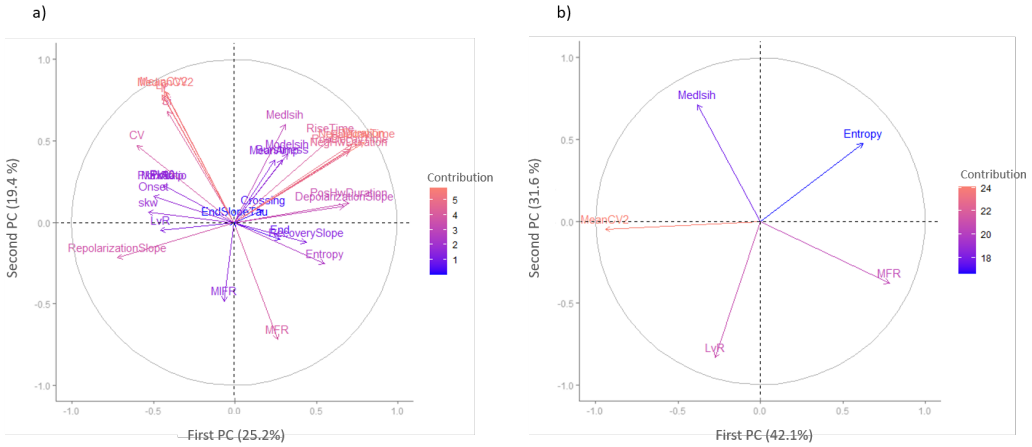

Figure 4.2: PCA performed on the 28-dimensional space for 1477 available neurons. **a)** The PCA correlation plot, which exposes all the relationships between variables, such that the positively correlated ones are group together and the negatively correlated ones are positioned in opposed quadrants. Moreover, the farther a variable vector is from the origin, the better it is represented in the vector map. **b)** The explainability of the variance within the data is maximized when PCA is applied on five electrical-based predictors: **the Mean Firing Rate (MFR), the Entropy, the Median of the Inter-Spike Interval Histogram, the Mean CV_2, and the Revised Local Variation LvR.**

[1] Via the use of the *prcomp* function available in the R open-source package *stats*, version 3.6.2 [R Core Team 2013].

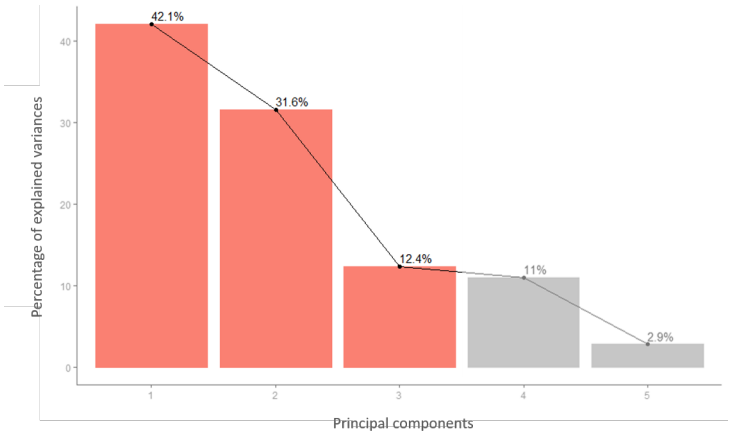

Figure 4.3: The first three principal components found via PCA on the top five most contributing features are enough to capture 86% of the variability of the whole data set, which effectively reduces the original dimensional space.

Figure 4.4 summarizes the location of the samples on the bidimensional space created by the first two principal components onto which the original data was projected, which explain over 70% of the total variance. Interestingly, this dimensional reduction allowed the clear visualization of a **small cluster of approximately 160 neurons** that seems to have its own signature based on the five electrical predictors chosen for the orthogonal decomposition.

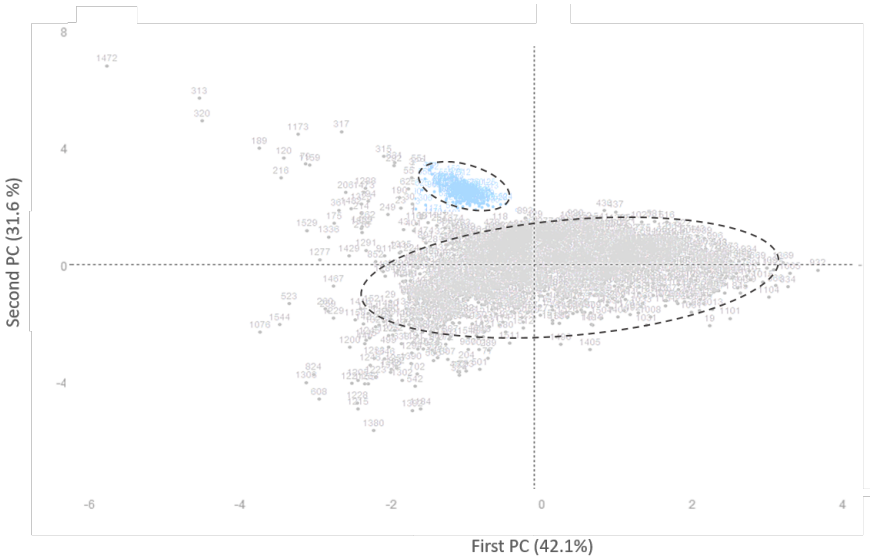

Figure 4.4: The plot of individual samples on the bidimensional space created by the projections of the first two principal components computed for the original data. A small cluster of neurons seems to be separated from the rest of the observations based on its electrical behavior.

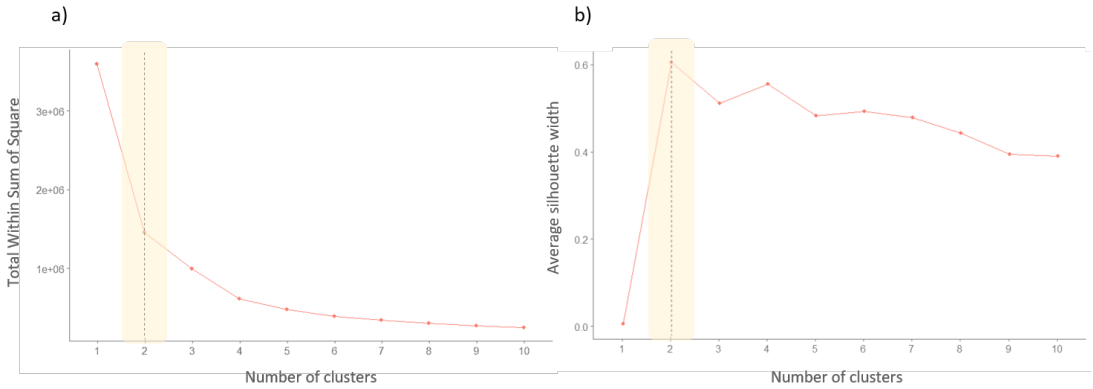

Figure 4.5: **a)** Total Sum of Squared Errors plotted against the number of optimal clusters. The elbow rule suggests the presence of two clusters. **b)** Silhouette metric plotted against the number of optimal clusters, for which the elbow rule also suggests a partition of 2 groups of cells.

To confirm the presence of the small identified group of cells, the optimal number of clusters was computed using the *total sum of squared errors* and the *silhouette* criterion[2], with both of them confirming the natural separation that exists between the isolated cluster and the remaining observations, therefore suggesting that 2 clusters were supported by the data (Figure 4.5).

To have a sense of the behavior of the neurons that were discovered as a cluster, their main waveform and autocorrelogram were plotted with the purpose of *externally* validating them with experts. After its revision by neuroscientists at the WIBR, **it was presumed that the neurons grouped in this cluster corresponded to an electrical expression of the Purkinje cells in the cerebellum, known as Complex Spikes, which usually fire at 1 Hz and follow wide loose waveforms** such as the ones identified (Figure 4.6). Moreover, **100% of the ground-truth observations labeled as Complex Spikes were clustered together in this subdivision** (15 samples). As a result and for practical purposes, Complex Spikes were considered as a separate class, but in theory, they are a subset of the Purkinje cells, which can fire two types of action potentials - simple spikes and complex spikes [Palmer 2010].

The exploratory analysis proposed in this section had the clear intention of extracting visual clues that could possibly indicate the presence of clusters in the data set of neurons available for this study. This effort was necessary before attempting any clustering technique. The application of the PCA procedure via eigenvectors was meaningful to separate Complex Spikes, clusters of Purkinje cells, from the rest of the cerebellar cortical neurons based on their electrical behavior coded in five main features of interest.

Further exploration of the remaining features did not offer any particular sign of natural separation between the remaining classes of neurons waiting to be identified: **Granule, Golgi, Mossy Fibers, and MLI cells.** Nevertheless, learning methods can still be

[2]Via the use of the functions *withinSS* and *fviz_nbclust*, available in the R open-source packages *DiscriMiner* version 0.1-29 and *factoextra* version 1.0.7.

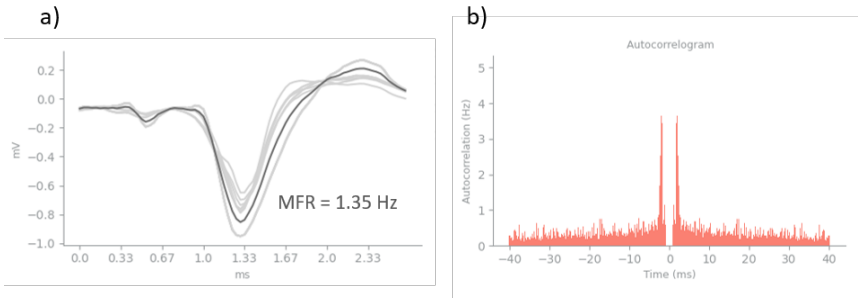

Figure 4.6: **a)** Waveforms of the apparent Complex Spikes as Purkinje representatives. Darker line is the average waveform. **b)** Average autocorrelogram of the discovered cluster, which is typical for a Complex Spike pattern.

applied to assess if the nature of the data set at hand can be separable in other meaningful ways that unveil the yet undiscovered boundaries amongst cerebellar cortical neuron classes.

4.1.2 Is PCA an innocuous tool for this study?

A question that may arise at this point should ask about the suitability of a technique such as PCA for the dimensionality reduction of the data considered by this study, as it is widely known that PCA has its own limitations and does not guarantee the finding of the correct projections in the absence of Gaussian distributions, when small variances are predominant across the data, or when non-linearities mainly describe the features. This study did not focus on the exhaustive proof of the last condition, but three main checks were made to conclude it was safer to proceed with PCA for this preliminary analysis:

- The amplitudes of the electrical voltages registered by neurons are considered to follow a Gaussian distribution [Khonsary 2016]. Moreover, this is also true for the inter-spike interval histograms, which follow a lognormal distribution but were transformed logarithmically[3] before the computation of the electrical predictors detailed in Chapter 2. As a result, it is expected that the entirety of these variables follow an approximately Gaussian distribution.

- Regardless of the original distributions of the features considered, the *central limit theorem* allows their consideration as approximately normally distributed dimensions, as the population of neurons is sufficiently large ($>> 30$). This premise is especially convenient for the waveform-based features, which are not presumably Gaussian from a theoretical perspective.

- Small variances are unlikely to affect the decomposition of the dimensions considered, as the electrical behavior of the cerebellar cortical cells is significantly different across neuron types [van Dijck 2013].

Given the previous, it was expected that the data used by this study could be linearly and orthogonally projected into a new basis computed via PCA.

[3] If the random variable X is log-normally distributed, then $Y = ln(X)$ has a normal distribution.

4.2 Unsupervised clustering

This section presents the set of unsupervised clustering tasks that were applied to the available data with the objective of finding clusters of neurons beyond the Purkinje (Complex Spikes) cell class.

4.2.1 Setting up the optimal number of clusters

During the exploratory analysis, the total sum of square errors and the silhouette metric were evaluated on the entire data set (not on the principal components) for a different number of potential clusters. As a result, these methods suggested the existence of two clusters already discussed. The prior knowledge on the neurons available for this study suggests that there are at least five cell classes mixed together in 1477 observations. However, imposing this number of clusters as *optimal* for any unsupervised clustering technique seems arbitrary and may lead to confirmation bias. For this reason, the Dunn and Davies-Bouldin indexes[4] were evaluated on the three most relevant principal components used to decompose the initial matrix of data, with the idea of getting to known the ideal number of clusters supported by those components, which is at least expected to be between 2 and 5 groups of cells. Table 4.1 shows that the optimal number of clusters for both a partitional (K-Means) and a hierarchical (Ward's) method is set to 6, considering that the higher the Dunn index the better the clustering, and the opposite happens with the Davies-Bouldin index. Moreover, both metrics seemed to reach a consensus with 6 clusters.

	K-Means		*Agglomerative Ward*	
Clusters	Dunn Index	DB Index	Dunn Index	DB Index
2	✗ 0.007	✗ 1.676	✗ 0.008	! 1.474
3	✗ 0.007	✗ 1.605	! 0.010	✗ 1.862
4	✗ 0.005	! 1.297	✓ 0.011	! 1.446
5	! 0.008	! 1.299	✓ 0.011	! 1.417
6	✓ 0.012	✓ 1.283	✓ 0.011	✓ 1.348
7	! 0.009	✓ 1.232	! 0.010	✓ 1.336
8	✓ 0.012	✓ 1.187	! 0.010	✓ 1.333

Table 4.1: The Dunn and the Davies-Bouldin indexes suggest for either a partition-based or a hierarchical clustering method that the compactness and isolation metrics are optimized when 6 clusters are extracted from the data, which indicates the potential existence of 6 cell types in the cerebellar cortex, for which 2 of them may correspond to the same neuron class (Purkinje and Complex Spikes). The values of the indices presented for the K-Means technique were averaged for 5 cross-validated repetitions of the algorithm.

[4]Via the use of the functions *dunn* and *index.DB*, available in the R open-source packages *clValid* version 0.6-9 and *clusterSim* version 0.49-1 [Brock 2008, Walesiak 2008].

4.2.2 Partition-based clustering

A first unsupervised clustering experiment consisted in the application of the basic partition algorithm K-Means[5] on the first three principal components that maximized the explainability of the variance of the data. The existence of three-dimensional eigenvectors that explain over 80% of the variability of the studied neurons allows the use of Euclidean distances to measure the compactness and isolation of potential clusters without experiencing the curse of dimensionality. Figure 4.7 summarizes the graphical clusters projected onto the first two principal components.

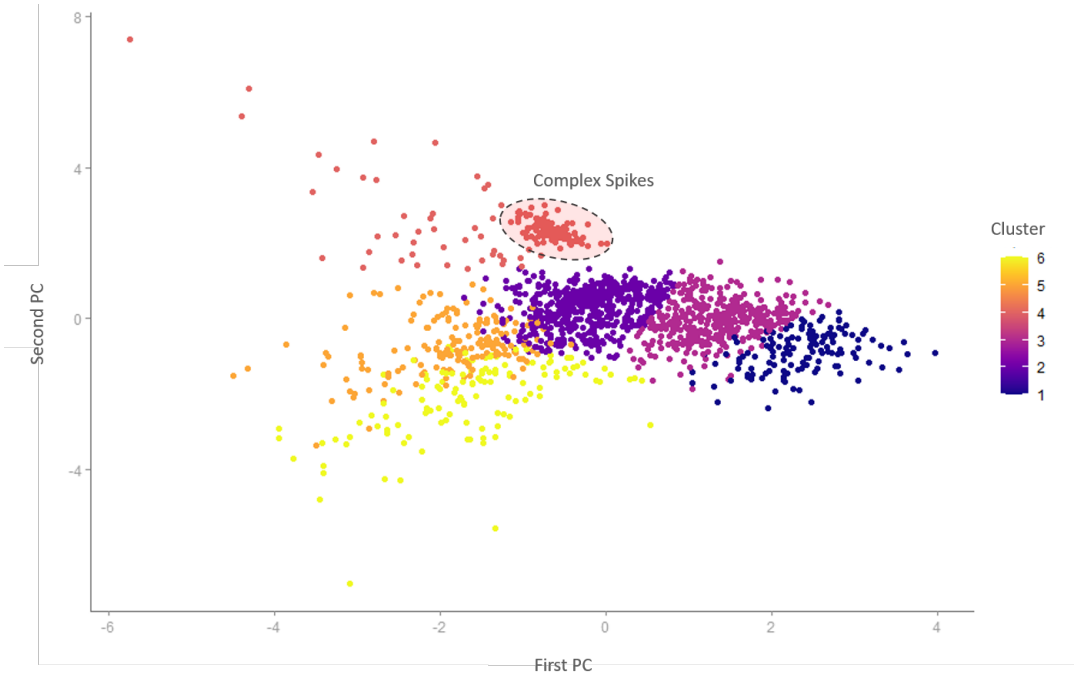

Figure 4.7: K-Means clusters of cerebellar cortical cells projected in the two first principal components of the initial data.

From Figure 4.7, it was evident that the separation of the Purkinje-Complex Spikes cells was correctly captured by the fourth cluster, in which 100% of the ground-truth labeled neurons for that class also lie. However, the separation between the remaining classes is still not evident.

4.2.3 Hierarchical-based clustering

Similarly, Ward's method for hierarchical agglomerative clustering was tried on the data PCA decomposition[6]. Figure 4.8 summarizes the resulting dendrogram and the boundaries for each cluster. Ward's method was also able to separate the Complex Spikes expressions

[5] Via the use of the function *kmeans* available in the R open-source package *stats* version 3.6.2 [R Core Team 2013].

[6] Via the use of the function *hclust* available in the R open-source package *stats* version 3.6.2 [R Core Team 2013].

of Purkinje cells that were visualized during the exploratory analysis and which follow the patterns presented in Figure 4.6. Assessment on the remaining clusters was still a pending task at this point.

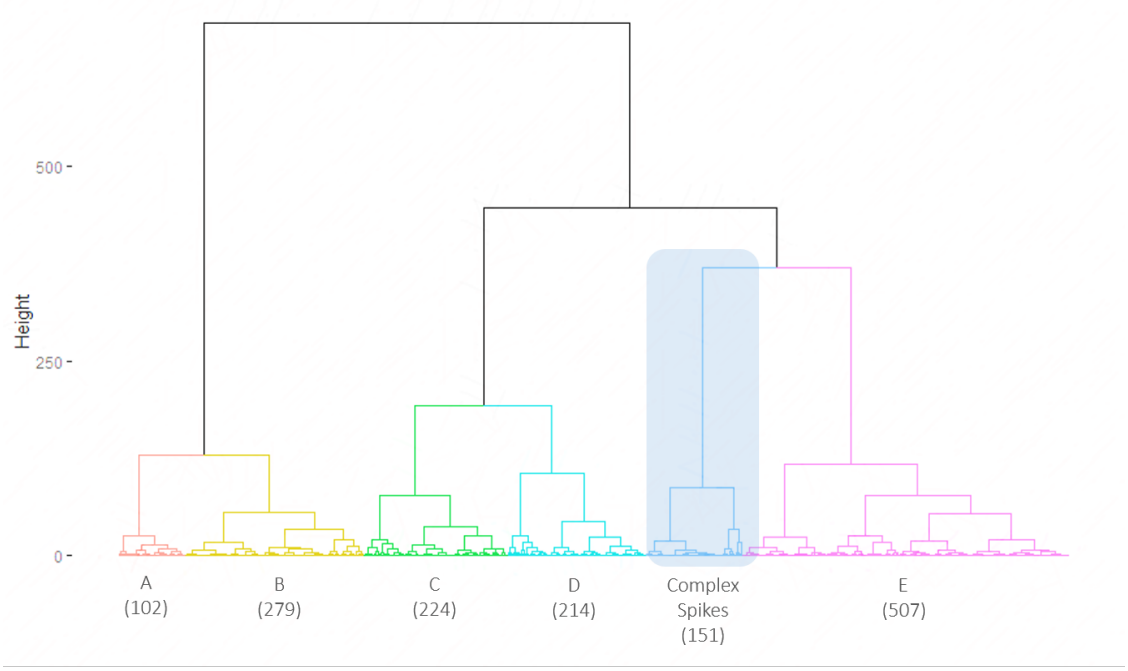

Figure 4.8: Ward's clusters of cerebellar cortical cells. As in the K-Means algorithm, hierarchical clustering also detected the existence of Complex Spikes in the data. Clusters A-E are presented and explained below.

4.2.4 Clustering strategy comparison

Establishing a fair comparison between partitional and hierarchical clustering techniques is only feasible when supervised labeled data exists to measure classification performance metrics. The data set at study only contained 5% of its samples labeled as one of the types of neurons found in the cerebellar cortex, which makes the decision of choosing partitional techniques over hierarchical ones or vice-versa particularly complex. Moreover, both K-Means and Ward's methods had almost the same level of compactness and isolation as measured by the Dunn and the Davies-Bouldin indexes presented in Table 4.1.

Table 4.2 shows a high proportion of overlapping observations between some of the clusters produced by the two clustering approaches. For example, in one particular run, cluster 1 in Ward's almost corresponds to cluster 6 in K-Means, which contains the Purkinje-Complex Spikes cells previously identified. This seems to indicate that both algorithms did not follow unreasonably different techniques to create clusters of cells, but this information is not enough to conclude that both techniques produce *valid* clusters, for instance, aligned with theoretical principles.

When the *No-Free-Lunch theorem* is considered, it becomes clear that *a general-*

purpose, universal optimization strategy is impossible, unless that strategy somehow specializes in handling a particular domain problem [Ho 2002]. This implies that in the long term, K-Means and Ward's would produce an equally good solution for this clustering application, when subject to the average of all the possibilities. However, as computing all possible cluster combinations is unfeasible, the choice of one method needs to be done for practical cluster validation. Ward's method was selected mainly due to two considerations:

- Thanks to the dendrogram visualizations, hierarchical clustering allows the tracking of potential neuron hierarchies that may be currently missed by expert neuroscientists. In a sense, hierarchical techniques act as a *second expert* in these discoveries and could potentially allow the identification of sub-classes of neurons contained within already known clusters.
- The visualization of the K-Means results in Figure 4.7 suggested that actual neuron classes follow hyper-spherical cluster shapes, which is unlikely to happen given their sparsity on the plane. The separation between classes different from Complex Spikes seemed unnaturally drawn and unlikely to represent the real boundaries defining neuron types. Figure 4.9 shows that, for instance, the ground-truth Purkinje neurons were spread across clusters 1, 2 and 3 in K-Means, which means their behavior was not accurately captured by the partition-based algorithm.

		Agglomerative Ward clusters							
		1	2	3	4	5	6		*Agreement*
K-Means clusters	1	0	3	0	4	118	1	126	94%
	2	0	108	1	0	15	243	367	66%
	3	1	10	0	139	24	0	174	80%
	4	0	376	0	78	57	0	511	74%
	5	0	0	101	0	0	35	136	74%
	6	150	10	0	3	0	0	163	92%
		151	507	102	224	214	279		avg 80%

Table 4.2: Comparison between Ward's method for hierarchical clustering and K-Means results on the initial data set. Most of the clusters are highly overlapping between the techniques. For instance, 94% of the observations from cluster 1 in K-Means were grouped together in cluster 5 created by Ward's approach. Similarly, 74% of the presumably Purkinje cells identified by K-Means were clustered by the hierarchical algorithm in cluster 2, and so on.

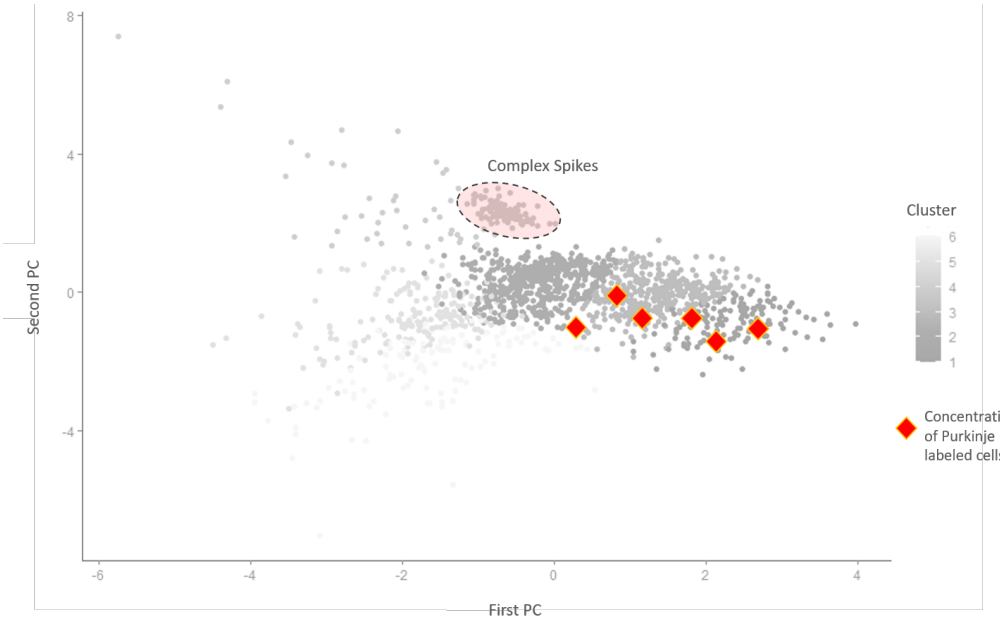

Figure 4.9: K-Means algorithm succeeded in the identification of Complex Spikes but clearly failed to cluster the original Purkinje cells in a single cluster. This indicates that the real underlying boundaries between cell classes different from Complex Spikes may not be well captured by a partitional strategy.

4.2.5 External validation

In practice, any unsupervised clustering application needs to be externally validated by experts or ground-truth data that can assert, to some extent, the validity of the clusters produced by a particular technique. Internal evaluation measures, such as the Dunn and the Davies-Bouldin indexes, even when mathematically grounded, do not provide a rationale to conclude if a set of compact and well-isolated clusters describe the reality they are supposed to represent.

In this study, a team of neuroscientists at the WIBR validated the clusters generated by Ward's method. To do so, they used their expertise and knowledge on three fundamental aspects:

- *The Mean Firing Rate:* correspond to one of the features computed to describe the behavior of cerebellar cortical cells. As presented in Chapter 2, this dimension has been mostly recorded on anesthetized animals but serves a starting point to differentiate specific cell classes such as Purkinje and Complex Spikes. Nevertheless, neuroscientists have not reached a consensus about the expected *in vivo* firing rate ranges for other cell types such as Golgi or MLIs. This lack of a unified vision on the firing pattern of all the cerebellar cortical cells is a motivator to find a potential classifier that can capture the underlying knowledge for all the classes at hand and predict their type correctly.
- *The autocorrelogram:* theoretically described in Chapter 2, represents the firing

pattern of neurons. Base knowledge in the field has allowed neuroscientists to learn how to identify particular cell types based on their autocorrelograms.

- *The signal waveforms:* as observed in Figure 4.6, neuroscientists have learned to identify specific cell types such as Complex Spikes based on their loose and wide waveforms. Similarly, Purkinje cells are prone to have a more acute negative deflection and Mossy Fibers sometimes accompany their signals with extra sags after their troughs. These clues are not completely confirmatory and cannot be taken as granted for every neuron, but they are useful to determine if the obtained clusters are likely to resemble these priors.

4.2.5.1 Ward's A cluster

Following expert validation of the cluster A from Ward's method (Figure 4.8), it was presumed to contain specimens from the Granule class. With a Mean Firing Rate of 2.28 Hz, Figure 4.10 shows the waveforms described by these clustered neurons as well as their average autocorrelogram. Regarding the ground-truth data available for this class, **100% of the labeled Complex Spikes samples were included in this cluster,** which supports the reification of this cluster as the *Granule* one.

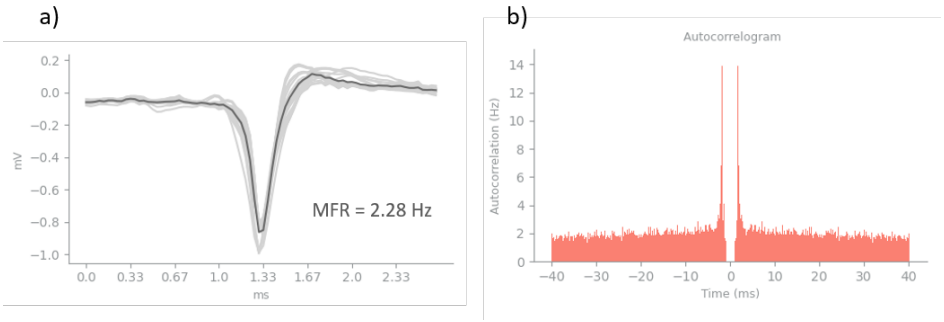

Figure 4.10: **a)** Waveforms of the apparent Granule cells. The darker line is the average waveform. **b)** Average autocorrelogram of the discovered cluster.

4.2.5.2 Ward's E cluster

Cluster E was the biggest one computed by Ward's. With over 500 cells, this cluster was split into 4 subgroups of neurons to validate their waveforms, autocorrelograms and mean firing rates. Figure 4.11 summarizes the evaluation metrics for each subcluster. This decomposition allowed detecting that this cluster was mainly formed by specimens from the Purkinje class, although 5 neurons were suspected not to belong in the group due to their autocorrelogram behavior. Regarding the ground-truth data available for this class, **91% of the labeled Purkinje samples were grouped together in this cluster,** which majorly supports the validation of this cluster as the *Purkinje* one.

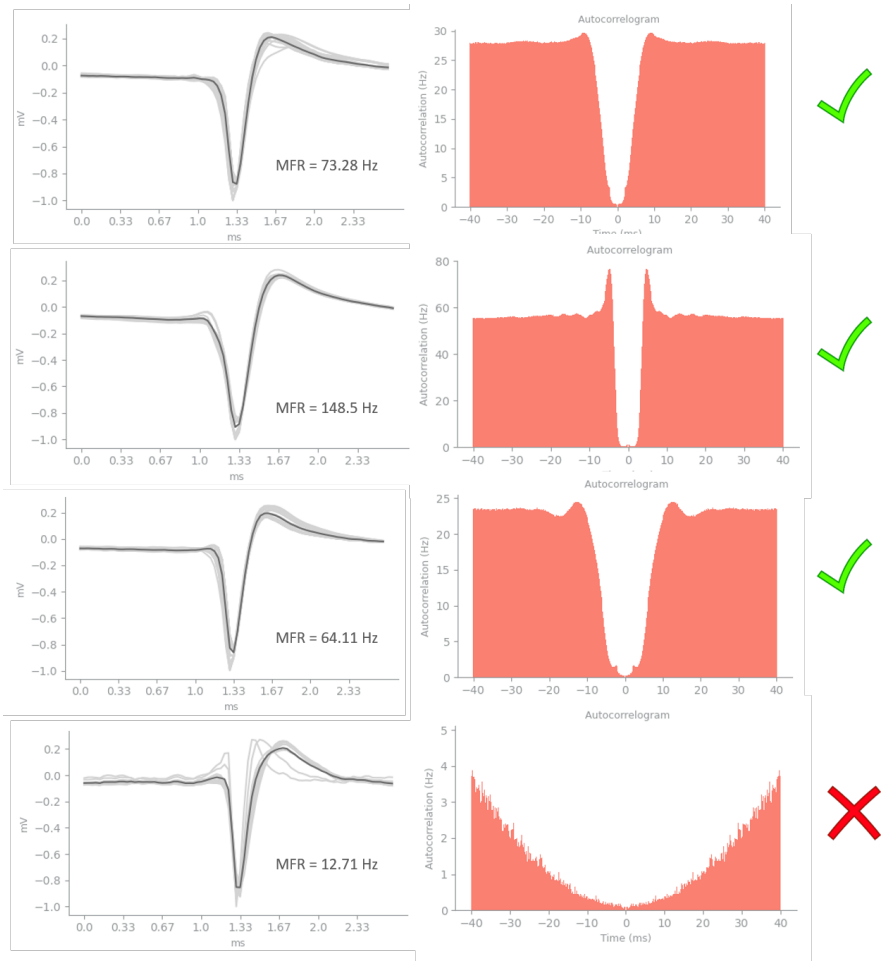

Figure 4.11: Waveforms and autocorrelograms of presumed Purkinje cells identified in the biggest cluster returned by Ward's method. Experts assessment coincided with this agglomeration, although the bottom group of neurons was not considered to belong to this class.

4.2.5.3 Ward's C cluster

To validate cluster C, the same procedure was followed as for evaluating cluster E. Subclusters found in this group were hard to catalog as belonging to one of the five classes of cerebellar cortical neurons. However, a group of neurons had a very specific pattern in their waveforms: an extra sag after their most negative electrical deflection, which is commonly observed in Mossy Fiber cells (Figure 4.12). Nevertheless, only **20% of the labeled Mossy Fiber samples were grouped in this subcluster**, indicating that most of the features that separate this cell class are beyond understanding.

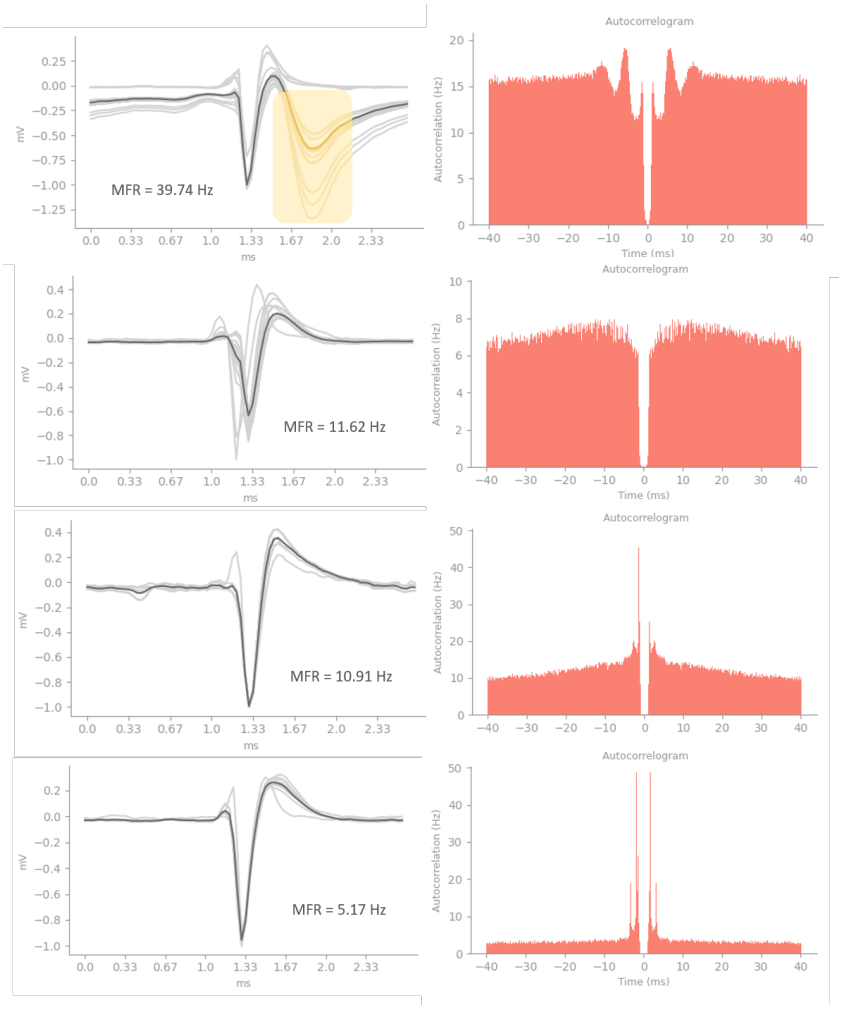

Figure 4.12: Waveforms and autocorrelograms of presumed Mossy Fiber cells. Only 62 of the neurons clustered here (top charts) were externally validated as belonging to this class mainly due to their waveform behavior. For the remaining ones, there is no clarity about their membership.

From the external validation perspective, clusters B and D could not be verified, as there is not sufficient theoretical background to understand if the cells contained in those clusters belong to either the classes Golgi, MLI, or Mossy Fibers. Moreover, for the validated clusters, it remained unclear to what extent they contained false positives. In order to clearly approach these concerns, the use of labeled data would be essential so that a correct cluster profiling can be performed. Final clusters and their presumed names are illustrated in Figure 4.13.

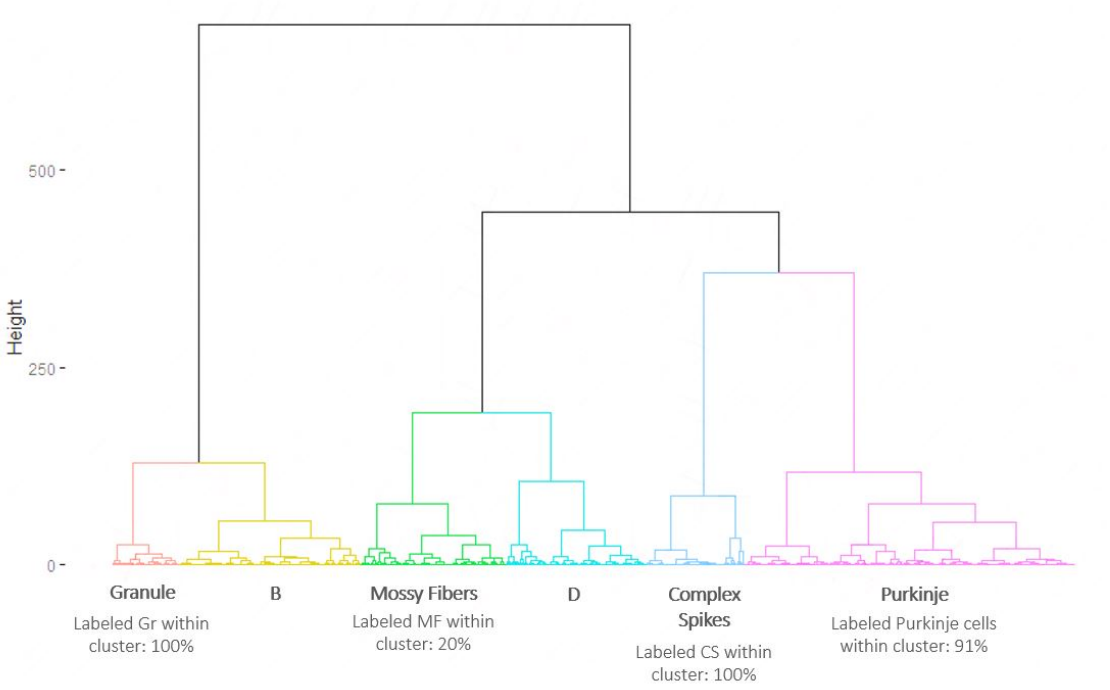

Figure 4.13: Ward's clusters of cerebellar cortical cells with their final descriptions assigned.

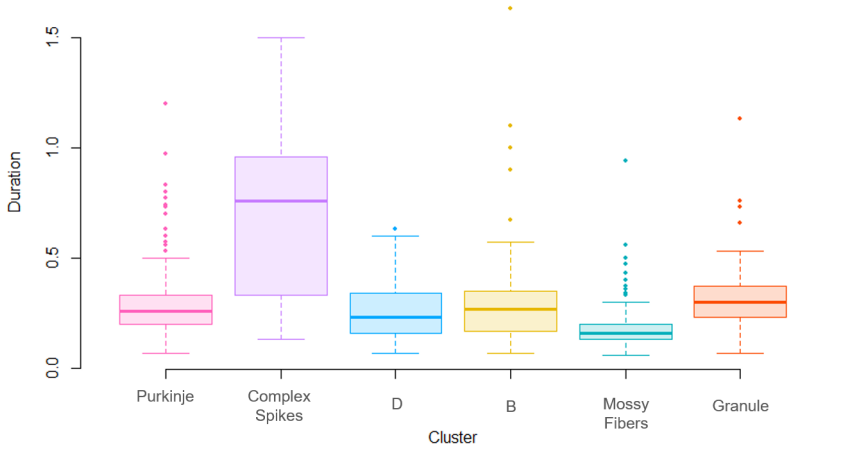

Figure 4.14: Box-plot charts for the waveform duration of the discovered clusters.

The waveform-based features considered by this study did not seem to have any major contributions to the class separability. In particular, the waveform duration (WvfDur in *Waveform modeling features*), which accounts for the time (in milliseconds) between the most negative and the most positive peak of a spike, was highly expected to be a key separator amongst cerebellar cortical cell types. Nevertheless, Figure 4.14 demonstrated

that the unsupervised discovered clusters did not expose substantial differences for this variable excepting for the Complex Spikes, whose waveform is already known to be wide and loose. Presumably, electrical predictors seem to better separate the cells of interest.

4.3 Supervised classification

Recall from the section *Optogenetics: ground-truth data labeling* in Chapter 3, that out of 1477 neurons analyzed, only 5% had a ground-truth label indicating their membership to one of the five classes of cerebellar cortical cells. Table 3.1 summarizes the numbers: Purkinje (37), Complex Spikes (15), Golgi (11), Granule (3), MLI/Molecular layer (4), and Mossy Fibers (10). With such low number of observations, the goal of building a high-precision classifier to predict cerebellar cortical cell classes is hindered.

The idea of having access to a supervised classifier is a major milestone for this study, mainly because unsupervised clustering does not offer generalizable decision rules that can be extended to classify future unseen observations. In the presence of new data, unsupervised techniques will compute a new dissimilarity matrix prone to end up in different associations of clusters that would need to be validated from the very beginning in a continuous loop as long as the clustering tasks are repeated. While unsupervised learning is a great tool for pattern discovery in the absence of labeled data, it is not so great for prediction purposes, for which supervised learning is a more appropriate methodology. This study proposes the use of a joint strategy between these two approaches so that a supervised classifier can be trained on a significantly larger labeled data set coming from the concatenation of the ground-truth labeled data, validated inputs from the unsupervised clusters, and artificially generated samples via interpolation. Hopefully, this classifier will predict neuron classes with a minimum expected level of precision established by a baseline model introduced later.

4.3.1 Data augmentation

Table 4.3 summarizes the number of additional cells that were added to each cell class, after going through a thorough external validation process with expert neuroscientists. Note that **adding these neurons and cataloging them as labeled observations entails a confirmation bias risk**, which wanted to be minimized and therefore only introduced for the types of Purkinje (Complex Spikes), Granule, and Mossy Fibers, for which more observations were required. Recall that for the Golgi class it was not possible to infer any additional cells. Moreover, the Purkinje cluster was decided not to be incremented, as the number of labeled samples in this group was already over 30, somehow covering the minimal requisites to make good use of the central limit theorem. MLI cells were excluded from further analysis at this point since the number of labeled samples was extremely low and the external validation process could not confirm the existence of more of these cells in the unsupervised exercise.

The labeled data at hand was subsequently augmented by using the SMOTE tech-

nique[7]. A total of $500 - n$ (with n being the number of labeled and validated samples per neuron type) observations were generated for each class. The idea behind setting up this oversampling process was to create a more balanced data set to obtain a more robust trained classifier.

Cell class	*Labeled ground-truth cells*	*Validated cells from clustering*	*Synthetic cells*	*Total*
Purkinje	37	0	463	500
Purkinje (Complex Spikes)	15	132	353	500
Golgi	11	0	489	500
Granule	3	59	438	500
Molecular layer	4	0	--	--
Mossy Fibers	10	51	439	500
	80	242	2182	2500

Table 4.3: Summary of labeled, validated, and synthetically generated samples per cell class. Additional cells were introduced after cluster external validation and the use of the SMOTE interpolation technique for data augmentation.

4.3.2 Baseline model

One of the contributions of this study aims to build a classifier able to maximize **precision** over other classification performance metrics such as recall, overall accuracy or F-Score. This is mainly because the users of this model will be neuroscientists at the WIBR and potentially other laboratories that want to maximize their confidence in their predictions. Final users do not need to worry about knowing how to correctly classify the entire universe of cerebellar cortical cells (i.e case in which a high recall would matter significantly) but those in their own data sets for experimental research.

The idea of building a supervised model to identify cerebellar cortical cells is only worthy if it surpasses the performance of a baseline model either in terms of accuracy or computational speed. In this study, a baseline model was proposed and set by measuring the performance of three expert neuroscientists at classifying the ground-truth data previously labeled via optotagging (refer to section *Optogenetics: ground-truth data labeling*, in Chapter 3 for more details). Experts were presented with a total of 80 labeled neurons that belonged to the cell classes of interest as shown in the *support* column of Table 4.4. The experts' task consisted of assigning a presumed label to every neuron based on the shape of its spike waveform and its mean firing rate (refer to section *Feature extraction*, in Chapter 3 for more details), and of course without knowing its ground-truth label beforehand.

Following the results of the labeling exercise submitted by the experts, their assessed labels were compared against the real ones. The precision levels reached by this baseline human-based model are presented in Table 4.4, which were computed considering

[7]Available via the Python library *imblearn* version 0.5.0 [Lemaître 2017].

Cell class	*Baseline model's precision*	*Baseline model's recall*	*Baseline model's F1-Score*	*Support (labeled cells)*
Purkinje	90%	70%	79%	37
Purkinje (Complex Spikes)	74%	100%	85%	15
Granule	10%	33%	14%	3
Golgi	22%	18%	20%	11
Molecular layer	10%	25%	13%	4
Mossy Fibers	0%	0%	0%	10

Table 4.4: Baseline model based on experts' assessment to identify cerebellar cortical cells.

the proportion of true and false positives obtained for each class. The hot spots for improvement correspond to the cell classes Golgi, Granule, MLI (Molecular Layer cells), and Mossy Fibers, as experts are already very precise at identifying Purkinje cells and Complex Spikes (which are also an expression of Purkinje neurons), the most prominent and active in the entire cerebellum. In particular, experts were unable to identify any MLI or Golgi cells in the labeled data set, the reason why the baseline levels of precision for these classes were set based on a uniform distribution $U(0, 1)$ such that if the random variable takes a value greater or equal to 0.5 a cell is considered to belong to the Golgi class, and to the MLI class otherwise.

Therefore, it is expected that the supervised classifier built on top of this data can offer substantially higher precision levels for the less easy-to-identify cell classes. Not limited to this, time is an additional factor to consider. Expert neuroscientists spent around 3 hours identifying the 80 cerebellar cortical cells they were presented with, a time window that could be significantly reduced by a supervised classifier capable of delivering high-precision predictions.

4.3.3 Method selection strategy

The first question to be addressed in supervised learning is whether a model is really necessary. For the purpose of this study, a supervised classifier would be justified if it can be generalized to unseen data while delivering higher precision and processing times than the baseline model, which has been built merely on the base of expert knowledge encoded as decision rules. To conclude on this, cross-validated precision levels were computed to decide out of a set of considered machine learning classifiers, which one could provide better results than simple rule-based decision making processes.

Secondly, it was considered that predicting cerebellar cortical cell classes did not require the specific use of a white-box technique. Precedent related works often used decision trees that resulted in non-generalizable conclusions and overfitting. Learning the lesson implied the potential sacrifice of the interpretability of decision trees in exchange for the robustness of a black-box model. An ideal classifier that combines both a simple, probabilistic, and

white-box formulation is the logistic regression, which was also considered as a potential classifier.

Subsequently, the need for a naturally probabilistic model was also discarded for this study. However, future developments in this direction include the implementation of such classifiers so that neuroscientists (final users) can be alerted about neurons with associated majority class probabilities under a specific threshold (or with low confidences). The overall purpose would be to use their expertise to calibrate a better model able to correctly identify these edge misleading observations.

As a consequence of the previous considerations and given the available techniques presented in Figure 2.27, a set of models was trained and cross-validated with the objective of finally choosing a top performant model in line with the Occam's razor principle and the promise of highly precise prediction capabilities when compared to the baseline. The trained models were:

1. *Generalized Linear Models*
    - Logistic Regression: simple, probabilistic white-box technique.
    - Passive Aggressive Classifier: simple, non-probabilistic method.
2. *Tree-based Classifiers*
    - Decision Trees: white-box technique.
    - Random Forest: boosting algorithm based on majority rule.
    - XGBoost: a parallelizable gradient boosting optimized algorithm.
3. *Support Vector Machines*: with linear, Gaussian, and polynomial kernels.
4. *K-Nearest Neighbors (KNN)*: as a supervised version of hierarchical clustering.
5. *Bayesian Classifiers*
    - Gaussian Naive Bayes Classifier: a probabilistic approach.
    - Gaussian Process Classifier: simple, non-probabilistic, with strong independence assumptions and yet very effective in practice.

4.3.4 Hyper-parameter tuning

Table 4.5 summarizes the list of hyper-parameters that were tuned via *grid search* with 3-fold cross-validation for every model considered to be trained for supervised classification. The list of tuned hyper-parameters is detailed in Appendix C.

4.3.5 Model training, testing and validation

Recall from table 4.3 the number of ground-truth data available for every cell class intended to be predicted. Most of them do not count on enough observations to serve a

Algorithm	*Tuned hyper-parameters*
Logistic Regression	Solver, penalty function, stopping tolerance, regularization strength, class weight
Passive-Aggressive Classifier	Stopping tolerance, loss function, regularization strength, class weight
Decision Trees	Maximum depth, optimization criterion, minimum samples to split, minimum samples per leaf, class weight
Random Forest	Maximum depth, optimization criterion, minimum samples to split, minimum samples per leaf, class weight
XGBoost	Learning rate, maximum depth, minimum child weight, subsample size, colsample size
Support Vector Machines	Kernel, degree, gamma, stopping tolerance, regularization strength, class weight
K-Nearest Neighbors	Number of neighbors, weights, algorithm, distance metric

Table 4.5: List of hyper-parameters tuned for every supervised classifier.

triple holdout partitioning strategy for training and correctly estimating a classifier generalizable performance. To overcome the limitation imposed by the sample sizes of the ground-truth data, a small number of additional cells were labeled via external validation of clustering, and the SMOTE oversampling technique was used to generate hundreds of artificial interpolated observations for every class in order to obtain a balanced data set.

Despite the previous efforts, the holdout testing and validation phases continued to be unfeasible. Artificially generated samples cannot be used for model evaluation, as their correct use is merely intended to improve the training process and the hyper-parameter tuning. In other words, testing and validation sets cannot contain artificially generated samples, but neither could be formed exclusively by ground-truth data, as these few observations were also needed for training.

To overcome the previous, a 5-fold cross-validation strategy rather than a traditional arbitrary split holdout was set up to generate prediction estimates for each input sample (neuron) in the augmented data set. During the cross-validation approach, each sample belongs to exactly one test set, and its prediction is computed with an estimator fitted on the corresponding training set [8]. Nevertheless, it is important to highlight that **passing these estimations into an evaluation metric may not be valid to measure the generalized performance of any classifier, which shall be computed with exclusively real unseen data, currently unavailable.**

The training of the considered classifiers, as well as their tuning and validation pipelines were implemented in Python 3.7 via the machine learning library scikit-learn version 0.23.1 *sklearn* [Buitinck 2013]. The codes are available in the [Classifier] repository link.

4.3.6 Model benchmarking

Cross-validated classification metrics for the models trained on the balanced augmented data set (including ground-truth, validated and artificially generated cells) are summarized in Table 4.6. These results suggest that, overall, the **current baseline model to predict cerebellar cortical neuron classes was outperformed by the trained classifiers.**

[8] This strategy was applied via the use of the function *cross_val_predict()* available in the Python library scikit-learn version 0.23.1 [Pedregosa 2011].

However, results must be examined at the class-level.

From Table 4.6 is evident that for a sufficiently big and balanced data set across the five classes of cells to be predicted, almost perfect precisions can be obtained to separate Purkinje, Complex Spikes, and Granule cells. In particular, **techniques such as Support Vector Machines and XGBoost reported the highest macro precision levels and perform especially well on predicting the cell types that are less evidently separable: Golgi cells and Mossy Fibers**. In fact, a high precision model to better identify those latter is much more valuable than a high precision pipeline to detect Purkinje cells or Complex Spikes, since this is absolutely one of the weak spots for expert neuroscientists to unambiguously identify.

Precision	Logistic Regression	Passive-Aggressive Classifier	Decision Trees	Random Forest	XGBoost	Support Vector Machines	K-Nearest Neighbors	Gaussian Naïve Bayes	Gaussian Process Classifier	Support	Best model	Baseline
Purkinje	100%	88%	97%	100%	100%	99%	98%	90%	38%	500	100%	90%
Complex Spikes	100%	98%	99%	99%	99%	100%	99%	94%	99%	500	100%	74%
Granule	100%	79%	97%	96%	99%	99%	99%	90%	96%	500	100%	22%
Golgi	92%	67%	86%	91%	97%	97%	94%	86%	91%	500	97%	10%
Mossy Fibers	95%	75%	99%	98%	99%	99%	99%	84%	60%	500	99%	0%
Macro precision	**97%**	**81%**	**95%**	**97%**	**99%**	**99%**	**98%**	**89%**	**72%**			
Macro Recall	97%	81%	96%	97%	99%	99%	97%	89%	61%			
Macro F1	97%	81%	96%	96%	99%	99%	98%	89%	61%			

Table 4.6: Cross-validated precision per class, macro recall, and macro F1-Scores for various classifiers trained on the synthetically augmented cerebellar cortical cells data set. The Gaussian Process Bayesian classifier used without known prior conjugates demonstrated the poorest performance. Most of the techniques were highly precise to predict Purkinje, Complex Spikes, Granule, and Mossy Fiber classes. XGBoost and Support Vector Machines offered the highest precision to classify Golgi cells. All the supervised classifiers resulted in considerably better precision levels when compared against the baseline model, while reaching a balance in the recall.

Excepting for the Passive-Aggressive and the Gaussian Process Classifiers, which had the worst performance on the augmented data set, all the models from Table 4.6 were separately trained and cross-validated on the ground-truth and validated data from clustering (without artificial observations) to have an idea of the potential of these techniques when applied on exclusively real observations. Results are presented in Table 4.7, which shows expected lower levels of precision to predict Golgi cells, being this class the most underrepresented one in terms of the number of observations. In general, under these settings still higher-than-baseline precision levels are the rule but note that this is barely true for the case of Purkinke cells, for which the baseline model seems to do the same work

with simple decision rules. Precision scores close to 100% are no longer being obtained for the Purkinje, Complex Spikes, and Granule classes, which is an indicator of the effect that the amount of training data and class balance have on a classifier prediction capabilities.

Interesting work for future reference would consist of the implementation of classifiers to only focus on the detection of the less identifiable neuron classes, namely: Granule, Golgi and Mossy Fibers.

	Precision								Support	Best model	Baseline
	Logistic Regression	Decision Trees	Random Forest	XGBoost	Support Vector Machines	K-Nearest Neighbors	Gaussian Naive Bayes	Voting Classifier			
Purkinje	94%	94%	94%	90%	91%	91%	89%	92%	37	94%	90%
Complex Spikes	95%	94%	97%	95%	95%	92%	93%	93%	147	97%	74%
Granule	82%	73%	82%	90%	70%	77%	78%	84%	61	90%	22%
Golgi	40%	25%	31%	43%	27%	10%	67%	100%	11	100%	10%
Mossy Fibers	84%	77%	81%	74%	91%	77%	81%	84%	27	91%	0%
Macro precision	**89%**	**85%**	**89%**	**89%**	**86%**	**84%**	**87%**	**90%**			
Macro Recall	77%	73%	77%	76%	73%	68%	74%	76%			
Macro F1	78%	72%	77%	77%	74%	69%	75%	78%			

Table 4.7: Cross-validated precision per class, macro recall, and macro F1-Scores for the top performant models trained on the ground-truth cerebellar cortical cells data set (originally labeled observations plus added ones from the unsupervised clustering).

Interestingly, for the ground-truth data set it became harder to assess which model performed better overall. While XGBoost continued to be one of the top classifiers, its ability to predict Mossy Fibers was highly exceeded by other algorithms such as Support Vector Machines or Logistic Regression. Nevertheless, the total macro precision of the Support Vector Machine decreased when compared against its performance on the augmented data set, which could be a hint of the instability of this method already reported by some studies [Fan 2018]. Moreover, despite the excellent performance of the Logistic Regression, it remains unclear if it can outperform an XGBoost in future predictions with unseen data, as this did not happen when both techniques were applied on the augmented data set. Additionally, it is important to highlight the performance of the Gaussian Naive-Bayes classifier, as it reached an overall macro precision of 87% and had the best precision to predict Golgi cells. Being such a simple technique with such strong assumption on the independence of the features used (which is clearly violated in this study), it performed almost as good as the XGBoost and Logistic Regression methods, and even better than the Support Vector Machine. This conclusion is in line with previous academic studies that demonstrated how Naive-Bayes is an unrealistic yet very practically effective classifier

[Zhang 2004].

4.3.7 Ensemble modeling

Selecting a model for a multi-class classification problem is often harder than for the binary case, as not all the classes are highly identifiable and predictable by the same algorithm due to their own nature or their imbalances. This was evident before with the Golgi class, as in the ground-truth data set it was more easily identified by a Gaussian Naive Bayes, not the case when the augmented (and therefore balanced) data set was considered, scenario in which XGBoost and Support Vector Machines did a better job recognizing this class.

Consequently, a joint model consisting of a majority voting classifier was trained on the ground-truth data set. This strategy combined the outputs from four different models, which were then subject to a hard voting majority system. Considered estimators were: XGBoost, Logistic Regression, Support Vector Machines, and Gaussian Naive-Bayes (especially useful to detect Golgi cells). Overall precision levels achieved by this joint classifier are reported in Table 4.7, showing that this model is almost as good as a Logistic Regression to predict most of the classes, with an important extra boost given to the Golgi class[9].

4.3.8 Feature importance

PCA decomposition on the predictors used in this study indicated that only five electrical-based features were the most essential ones to explain most of the variance within the data set. As a result, those features were used to perform hierarchical clustering on their three principal components.

Before applying any supervised classifier, a feature ranking was computed using *Recursive Feature Elimination* (RFE)[10], a procedure that estimates weights associated with the predictors according to their relevance for a given model. RFE applied on the XGBoost and the Logistic Regression classifications on the ground-truth data set revealed that in the ranking of importance associated with the 28 features computed in this study, the firing pattern modeling features were the leaders, which confirmed that waveform-based modeling features were not that useful to separate cerebellar cortical cell classes, as was seen in the unsupervised experiment. In fact, the two classifiers shared in their top 5 features 4 predictors also highlighted by PCA: **The Mean Firing Rate, the Median of the inter-spike interval histogram, the Mean $CV2$, and the Revised Local Variation LvR.**

Given the previous, all the trained classifiers were trained only with the 13 electrical-based modeling features introduced in section *Firing pattern modeling features* in Chapter

[9] Note that precision levels of 100% may be generated occasionally as a result of random fluctuations in the algorithms that do not work deterministically.

[10] Available in the Python library scikit-learn version 0.23.1 [Buitinck 2013].

3, instead of with the 28 initial variables considered, which improves the ratio between the number of features and available observations, and reduces unnecessary dimensionality.

4.3.9 Model selection

Upon the completion of the previous supervised classification tasks, the following conclusions shall be mentioned:

- The inherent distribution of the data is determinant for the success of any classifier. The models implemented above for two different data sets, yet related, confirmed the no-free-lunch theorem and the fact that no absolute truths exist about which classification technique is better, as this is highly dependent on the data.
- Even when it is impossible to conclude which algorithm will perform better before applying a set of them to a data set, models that encourage parsimony, avoid overfitting, implement built-in cross-validation, and are computationally efficient are often preferred because they lead to robust solutions. This is the reason why boosting models are generally chosen over simple decision trees, or logistic regression over support vector machines when their performances compete.
- In general, choosing a model needs to go beyond simple performance metrics. In this direction, validation is essential. Best classifiers need to be tested on different sets of real unseen observations before being considered for calibration or being launched in production.
- The class imbalance is an important player in model selection. Conclusions about the best models changed when the augmented (balanced) and the ground-truth (imbalanced) data sets were considered. Inherently, some algorithms are better at detecting imbalanced classes (e.g. Logistic Regression or some Bayesian classifiers), while others need to be correctly tuned to do so (e.g. Ensemble and boosting techniques).

For this particular application, there is a dual answer to what the best model is. On one hand, XGBoost is considered to be the most appropriate model (over Support Vector Machines) given its robustness. However, this decision is subject to the availability of an approximately balanced data set or a special tuning for it to better capture imbalance. On the other hand, when achieving class balance and sufficient observations seems infeasible, Logistic Regression may result in a comparable performance while offering interpretability, and a simpler solution to the separability problem. In case the logit-based classifier does not meet expectations, XGBoost could still be the choice to proceed, considering the possibility of complementing its output with the ones from simpler models in a majority voting or even a stack classifier. Table 4.8 summarizes the top models trained.

	Top models
Augmented data set	**Multiclass Softmax XGBoost**
Best parameters	`{'eta': 0.1, 'max_depth': 9, 'min_child_weight': 5, 'subsample': 1, 'colsample_bytree': 1}`
Augmented data set	**Polynomial Support Vector Machine**
Best parameters	`{kernel='poly', degree=3, tol=0.001, C=0.2, class_weight='balanced'}`
Only ground-truth data set	**Multiclass Softmax XGBoost**
Best parameters	`{'eta': 0.3, 'max_depth': 9, 'min_child_weight': 5, 'subsample': 1, 'colsample_bytree': 0.8}`
Only ground-truth data set	**Multinomial Logistic Regression**
Best parameters	`{C= 0.9, tol=1e-05, penalty= 'l1', solver= 'liblinear', class_weight=class_weights}`
Only ground-truth data set	**Voting Classifier**

Table 4.8: Best prediction models and hyper-parameters trained and cross-validated on both the augmented and the ground-truth data sets.

Chapter 5

Discussion

The cerebellum, a relatively simple folded brain structure involved in motor learning and coordination, is yet one of the most mysterious of the vertebrate brain, as its internal working mechanisms are still beyond understanding [Kawato 2003]. The previous is mainly due to the inability of expert neuroscientists to classify cerebellar cortical cells based on their electrophysiological behavior (also known as spontaneous electrical activity) [van Dijck 2013], this is, the way in which neurons conduct electricity.

The last decade of advances in microchip and computing power has seen more machine learning techniques being used in all disciplines, with neuroscience not being the exception. It is nowadays possible to simultaneously collect thousands of samples from neurons in any area of the brain, whose can be analyzed in search for underlying key patterns that allow the separation of clusters of neurons that behave similarly, enabling the further prediction of unseen data observations, which ultimately would unlock unprecedented knowledge about many brain structures poorly understood up to now.

Given the new possibilities driven by this technological revolution, the purpose of this study focused on building a supervised classifier that used electrophysiological features to predict for observed neurons in the mouse cerebellar cortex, their corresponding class out of five potential labels (i.e Purkinje, Golgi, Granule, Mossy Fibers, and Molecular layer cells). This classifier was expected to report higher precision levels when compared against a baseline model set by experts' assessment.

The goal of this study was shared by previous projects that did their research with more limiting technologies that restricted the collection of data in awake animals, which is the ideal scenario to study what neurons really do. Most of the machine learning methodologies presented by the related work had opportunities to improve their robustness but left an important legacy of modeling features that were used by this study to predict cerebellar cortical cell classes.

More than 4300 neurons were analyzed and quality-checked, from which approximately 34% constituted the data set on which deeper exploration was done to give an answer to the following research questions:

- Is it possible to find natural separation boundaries amongst cerebellar cortical cell classes from their spontaneous electrical activity?
- Is it possible to build a supervised classifier able to predict the five classes of cerebellar cortical cells based on their spontaneous electrical activity and with higher precision than an expert neuroscientist?

Both unsupervised and supervised learning methodologies were used to answer the previous concerns:

On one hand, unsupervised techniques (PCA and hierarchical clustering) allowed the identification of natural clusters in the data, some of which truly corresponded to the theorized cell types expected to be found in the cerebellar cortex. This supposed an affirmative initial answer to the first research question, although with a caveat: not all the cerebellar cortical cells are separable in a Euclidean space based on their electrophysiological behavior, especially Golgi cells, Mossy Fibers, and Molecular layer cells, a conclusion in line with the findings of recent studies such as the one presented by [Haar 2015].

On the other hand, the application of supervised learning techniques for classification seemed first infeasible due to the absence of sufficient labeled data previously obtained via optogenetics. To overcome this limitation, samples from the unsupervised clusters were subject to an external validation with neuroscience experts, who concluded on the definite labels for some of the cells they were presented with, which could be used as labeled observations to increase the size of the ground-truth data to feed a classifier. Unfortunately, this exercise could not be replicated for *Molecular layer cells*, which needed to be excluded from the classification pipeline.

Not limited to using unsupervised clustering to complement the labeled data previously obtained, synthetic neuron samples were generated via SMOTE interpolation as a means to increase the size of the data set used for supervised classification, promote balance amongst the cell classes, and boost performance.

Different supervised classifiers were tuned, trained, and cross-validated on the synthetically augmented data set and solely on the ground-truth data validated via optogenetics and clustering. In both cases, most techniques reported levels of precision over 90% at predicting almost all the classes at target, thus outperforming the baseline model mainly based on neuroscientists' decision rules. In particular, XGBoost and Logistic Regression were remarkable models capable of reaching precision levels close to 100% for specific classes.

5.1 Limitations

Although this study successfully translated a neuroscience problem into a classification machine learning pipeline with higher-than-baseline precision levels, there were several limitations to consider for future work, with most of them being related to the sample sizes of the available labeled data.

Ground-truth cells found via optogenetics were intrinsically insufficient to build a robust machine learning classifier. Part of the absence of labeled data was managed by adding validated observations from the unsupervised clustering, which introduced the risk of confirmation bias to the predicted classes with the supervised models. While expert neuroscientists were very confident in their estimations, their assessment could be poten-

tially predisposed.

The lack of sufficient labeled data severely restricted the classification and subsequent prediction of the molecular layer cells (MLI), which were not used to feed the classification models. As experts were not able to distinguish and validate any of these cells from the unsupervised clustering, future work to improve in this direction includes a deeper familiarization with the specific signature of these neurons, the prioritization of the data collection for this class, and the potential separation of molecular layer cells into two separate classes (Basket and Stellate) to analyze if there are significant differences between them that ease their separation from the remaining cerebellar cortical cell types.

Another consequence of the lack of sufficient labeled data meant that testing and validation data sets could not be extracted to report the real performance of the trained classifiers on unseen data. Validated observations coming from unsupervised clustering and synthetic oversampling were only useful to introduce balance to the data set and strengthen the training process of the supervised classifiers. As models were not tested on real, new, and independent observations, there is no way to tell how the models would have performed under these scenarios.

As a result of the absence of testing and validation holdout data sets, getting an idea of how much the trained models overfitted became much more challenging. While measures to avoid overfitting were put in place (i.e cross-validation, regularization, feature selection, and hyper-parameter tuning), models were not truly evaluated and their reported performances could be possibly inflated. Hopefully, with new upcoming labeled data, it will be possible to validate if the precision levels of the best models (i.e. Logistic Regression and XGBoost) remain the same or slightly drop, case in which overfitting could be confidently discarded and models would be ready for production release.

5.2 Recommendations for future research

Based on the results and findings of this study, several recommendations can be considered for future work. First, most of the limitations outlined previously may be minimized and even eliminated if more ground-truth data is collected. In the initial proposal of this study, 30 labeled cells per class were considered as a good number to proceed. However, it is likely that at least 60 labeled neurons per type would be required so that minimally 30 of them can be used for training purposes[1] while being able to separate a subset of the samples for testing and validation. Moreover, collecting approximately the same number of labeled neurons per class is important to preserve balance without undersampling the majority classes or having to recur to oversampling techniques to improve classification performance.

Secondly, in a revised version of this study, it would be worthy to reformulate the

[1] Following the central limit theorem [Kwak 2017]. Sample sizes equal to or greater than 30 are considered sufficient for the theorem to hold, which means collected data would approximately follow a Gaussian distribution and the parameter estimation for the entire population is possible.

machine learning classification problem posed such that the supervised models are only trained to predict the most indistinguishable cell types, which in the case of this study correspond to all of them with the exception of Purkinje cells, for which precision levels of approximately 90% can be reached by relying on simple decision rules in the mind of expert neuroscientists (i.e the baseline model). Unless thousands of observations are available for Purkinje cells, a machine learning model to predict this class is not adding much value, since the baseline precision level is barely outperformed with the current ground-truth observations from this class. Instead, efforts should be focused on predicting the remaining cell classes, which are not separable in the dimensionalities we can perceive by eye.

While this study focused on the application of non-probabilistic frequentist classifiers, it would be worthy to try probabilistic approaches that can estimate the underlying probability distributions that generate the neuron observations based on key electrophysiological features. With such methods, the possibility of performing error analysis opens even more [Ng 2011], such that incorrectly classified neurons and neurons that are not confidently assigned a probability of membership to a particular class can be studied in-depth with the ultimate goal of understanding why the probabilistic approaches are having problems recognizing the signatures of those observations.

Another important direction that might be useful to tackle the problem at hand of this study is to consider Bayesian-based methodologies to estimate the posterior probabilities of memberships of every neuron to particular cell classes. However, this is a more complex exercise that requires a profound knowledge of the electrophysiological behavior of the cerebellar cortical cell classes because appropriate priors must be built to boost the performance of these techniques.

5.3 Conclusion

Three main lessons can be derived from this study. The first one is that while some particular cerebellar cortical cell classes are more prone to be effectively isolated in a Euclidean decision space, as happens with Purkinje and Granule cells, not all the neuron classes of interest are separable in low dimensional spaces. Golgi cells, Mossy Fibers, and Molecular layer cells were particularly difficult to identify as part of a unified cluster, which does not imply that these cell classes are not separable in higher dimensionalities that can be exploited via machine learning. In fact, most of the supervised algorithms trained seemed to discover the underlying patterns behind these classes separability. In particular, five firing pattern modeling features were the most relevant for the supervised classifiers: the mean firing rate, the mode, the median, and the entropy of the inter-spike interval histograms. Amongst the features that describe firing pattern irregularities, the revised local variation (LvR), and the average coefficient of variation (CV_2) were also key to enhance classes separability. None of the waveform-based modeling features were as relevant as the electrical-based ones.

A second conclusion from this study suggests that if different dimensionalities are re-

quired to separate all the cerebellar cortical cell classes of interest, special care needs to be taken to avoid the curse of dimensionality. Supervised techniques to be applied for this classification problem will be successful as long as they can count on sufficient observations, especially for the classes that are most difficult to distinguish in Euclidean spaces. A proof of this was obtained via the synthetic observations generated by this study, which boosted the cross-validated precision levels of all the classifiers by adding more data from which to learn underlying patterns and key relationships in high dimensionalities mapped to the final labels.

A third conclusion comes from the *no-free-lunch* theorem and the fact that the nature and structure of the data have an unfathomable impact on any machine learning application. In this study, the best models to predict cerebellar cortical cell classes were an XGBoost and a Logistic Regression. Both models have proven robustness, stability, and computational efficiency, but the Logistic Regression was more easily tuned to perform better on the imbalanced ground-truth-only data set, while the former was the most precise when the synthetic observations were considered, as more balance was introduced to the classes. Therefore, the best model between these two shall be chosen by testing them both on future unseen observations and keeping track of their performance.

This study represented an effort to translate a biological problem into a machine learning one, and as such, it was highly dependent on the expertise of human knowledge (yet limited) to find automated answers to a concern that is still beyond the understanding of the neuroscience research community. This is an important step towards the attainment of unprecedented knowledge about the cerebellum, but it is certainly an early one. In the upcoming future, it is likely that larger data sets of labeled neurons can be gathered, and with these, the comprehension that both human experts and machines have on the way in which cerebellar cortical cells are classified.

Appendix A

Definitions

Cerebellar cortex

The cerebellar cortex is the outer layer of the cerebellum. It has a simple structure composed of three folded layers (leafs) that in the case of the human brain would be the size of a sheet of paper if unfolded. These leaves contain different cell types (neurons) that are in charge of transmitting electrochemical messages to the rest of the brain areas and their neurons in order to generate a sole output (for instance, a specific arm movement) [Squire 2012].

The cerebellum has a different anatomy and functioning with respect to other brain areas mainly because its neurons are specially tailored and unique. In particular, this study was interested in the identification and classification of the five types of neurons that are found in the cerebellar cortex, which are consequently named cortical cells: Purkinje cells, Granule cells, Golgi cells, Molecular layer cells (including both Basket and Stellate types), and Mossy fibers. These neurons' positioning and anatomy are presented in Figure A.1.

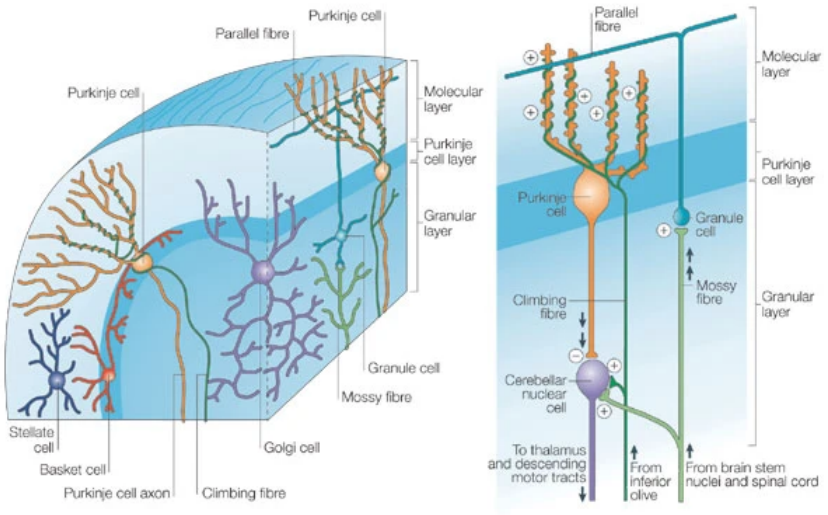

Figure A.1: Schematic view of the cerebellar cortex. Taken from [Apps 2005].

Golgi cells

Similar to what Purkinje cells do, Golgi cells are also a source of inhibition for the excitatory messages coming from the granule cells [Eccles 1973]. In particular, one of the functions associated with Golgi cells is that of providing control over the time at which

specific information is transmitted within the cerebellar network, due to a mechanism known as the feed-forward inhibition [D'Angelo 2008]. As a result, Golgi cells can be understood as controllers that affect the time responses of the rest of cerebellar neurons.

Granule cells

Granule cells are highly abundant in the entire human brain and represent approximately 60% of the available neurons. They are not exclusive to the cerebellum, and they are one of the smallest cell types. These neurons receive excitatory input from the Mossy fibres and communicate this message to the Purkinje cells [Squire 2012]. Traditionally, evidence suggested that granule cells encoded their input in a high-dimensional way in order to communicate an advanced sensory and motor message. However, recent studies have also associated complex cognitive functions to them, such as the expectation of reward [Wagner 2017].

Molecular Layer cells (MLI)

Molecular layer cells (MLI) are divided into Basket and Stellate. Both of them have an inhibitory effect on the response released by the Purkinje cells. However, their distinction obeys to the type of inhibition they create on the latter [Chu 2012].

Mossy Fibers

The main function of Mossy Fibres is to conduct excitatory input to the overall cerebellar cortex, as they originate from many locations in the brain and the spinal cord [Sillitoe 2012]. Following the principles of the neurotransmission, this excitatory input will be captured and transformed by other post-synaptic neurons to generate a final response.

Purkinje cells

Inside the cerebellar cortex circuit, Purkinje cells are the most prominent ones, with large branching dendrites (terminations) and the easiest to recognize due to their high electrical activity and their ability to integrate large amounts of information to learn [Paul 2019]. These cells create an inhibitory effect on the inputs generated and transmitted by the granule cells, which associates Purkinje neurons with the control and correction of motor movements [Squire 2012].

Synaptic transmission

The communication between neurons is achieved by means of a process called neurotransmission or synaptic transmission (Figure A.2). Through structures called synapses, neurons can communicate information by transforming electrical voltage differences (known as action potentials) into neurotransmitters (chemical substances) that are passed from a sender (presynaptic neuron) to a receptor (post-synaptic neuron). At the same time, neurotransmitters can generate action potentials that ultimately trigger a chemical response

[Holz 2012].

Synaptic transmission can be either excitatory or inhibitory. When excitatory, the input from the presynaptic neuron increases the chance of the postsynaptic neuron to provoke an action potential (firing) and the release of neurotransmitters. When inhibitory, the input signal reduces the probability of the receptor neuron to fire. The 5 types of cells found in the cerebellar cortex have either excitatory or inhibitory behaviors when they create synaptic transmissions12. For instance, a motor movement coordinated by the cerebellum is the result of the interaction between potentially millions of synaptic messages that are magnified and inhibited in unknown ways to create that physical output.

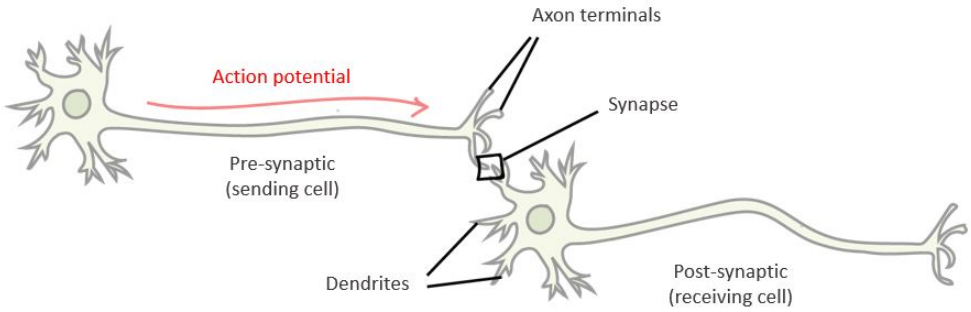

Figure A.2: The process of synaptic transmission between two neurons.

Appendix B

Fundamentals of Spike Sorting

Filtering and whitening

The first step in the Spike Sorting process consists in eliminating the noise from the data samples. Voltage recordings of neurons may be subject to reflect electrical differences due to other activating brain processes, happening at the moment of the recordings. The denoising of the raw data was developed as described:

- *Remove channels that were not used in the recording*: for this study, data was collected from the totality of the available channels (384 per shank side).
- *Filter channels with very few traces*: by default, channels with mean firing rates under 10 Hz (1 trace every 10 seconds) were not considered, as it was likely that these channels did not record any neuron in their surroundings. Usually, 150-300 Hz is defined as the appropriate threshold frequency. To perform this step, traces were normalized across all the channels, and then the mean firing rate was computed and evaluated.
- *Remove correlated noise through whitening*: channels record activity from many neurons that add stereotypical noise to all recording sites. This noise was eliminated by decomposing the data through zero-phase component analysis (ZCA). As a result, mean-centered spikes of neurons with a unit variance were identified on every channel.

Spikes detection

The second step of the Spike Sorting pipeline splits the already filtered and whitened 1-hour recordings into 2-second batches, on top of which the main units of measurement of neuronal activity (i.e spikes) are detected. A spike or action potential, as introduced in Chapter 2, is an electrical recorded trace that surpassed a voltage threshold, and as such, contains useful information about the neuron that *fired* it. The process to detect spikes and neurons is as follows:

- *Traces decomposition*: by using Principal Component Analysis (PCA), the first 2 components of every trace were computed.
- *Spike identification*: recorded traces above a given amplitude threshold were compared (in terms of their first two principal components) against predefined templates of what spike waveforms should look like for a particular neuron. Only those spikes that matched these templates (given a cosine similarity threshold) remained under

consideration and were used as relevant centroids to initialize a K-Means over all the 2-second batches. This process is also known as *template matching*.

- *Spike clustering via K-Means*: up to this moment, true spikes were identified at the batch level. Their relationship to neurons is many-to-one, which means several spikes could be generated by the same recorded cell. Thus, K-Means was applied over all 2-second batches to group similar spikes together.

Drift correction via time reordering

Electrode drift is a major source of noise and incorrectness of data during *in vivo* recordings. The third step of the Spike Sorting pipeline is known as *time reordering* and aims to recognize during which intervals of the data collection process the probe moved and started to alter the quality of the recorded neurons. To do so, the previously identified 2-second batches and their spikes are analyzed and the drift-subject ones are discarded for further analysis[1]. Time re-ordering procedure was as follows:

- *Batches comparison*: clusters of spikes retrieved for every 2-second batches are compared against each other by computing L2 norm (euclidean) distances between them.
- *Build a dissimilarity matrix*: with the input of the previous step. As a result, some batches will have more similar clusters (i.e. with approximately the same spike templates) and therefore will be closer to each other.
- *Dissimilarity matrix reordering*: so that similar batches of spikes are diagonal-centered, which means that the batches of time affected by electrode drift will remain as far as possible from the diagonal (as illustrated in Figure B.1). This is done by using a **manifold embedding algorithm** [Cayton 2005].

Neurons detection

Up to this point, true spikes have been identified out of the noise, and in drift-free recorded batches. Every 2-second batch contains spikes generated by a set of yet unidentified neurons. Moreover, it is also true that a neuron is potentially responsible for the generation of several spikes spread across different batches. The third step in the Spike Sorting process consists of the clustering of similar spikes across all the batches, which were then assumed to be generated by the same neuron. The idea is to get a set of many spikes associated only to one neuron. Note that those spikes were previously identified and clustered across batches via K-Means for drift correction purposes, so they need to be clustered again in the drift-free space resulting from the time re-ordering. As a result, spikes that are close to each other were generated by the same neuron that was recorded in a particular set of channels.

During this drift-free clustering, for every 2-second batch and each of its clusters of

[1] Batches are used because it is unlikely that the electrode moved from its original position during the entire hour of the recording.

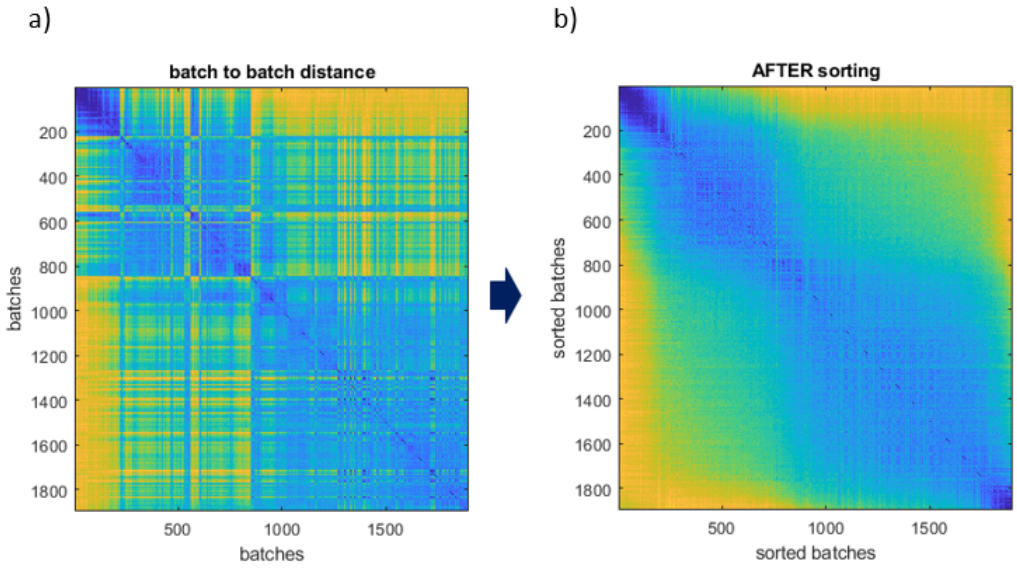

Figure B.1: Time re-ordering established the order in which data was traversed and allowed to identify batches subject to electrode drift, which need to be discarded for further analysis. Examples of dissimilarity matrices of batches of neuronal spikes recordings. **a)** Drift calculation before batch reordering. **b)** Drift identification after embedding reordering. Yellow zones refer to batches of spikes with highly different clusters of neuron spikes when compared to the initial batches of the recording session, which indicates that electrode drift was probably happening [Pachitariu 2016].

spikes a prototype waveform was extracted by using PCA decomposition, on top of which K-Means was run (i.e using the 1D waveforms coded as NumPy arrays in Python).

Final merges and splits

As a consequence of the previous clustering, a number of waveform prototypes are generated based on the information of the spikes contained in every cluster of the recording. These prototypes are supposed to be generated by unique detected neurons. However, in practice not all these prototypes are actually generated by isolated cells, as noise cannot be completely eliminated. To overcome this, an expert intervention is required to tune the results obtained by the automated Spike Sorting process. This manual curation and revision process is usually carried out by neuroscientists via a Graphical User Interface (GUI) called **Phy** (github.com/cortex-lab/phy), which is an open-source Python library developed at the Cortexlab (University College London) that allows the visualization of large-scale electrophysiological data and therefore the identification of false neuron clusters.

An overall summary of the Spike Sorting process is presented in Figure B.2. Note that the Spike Sorting process allowed to detect true spikes generated by actual neurons out of the cerebellar noise, but the entire pipeline was blind to the cell types to which those

neurons correspond. The optogenetics technique[2] was useful to complete the task of the Spike Sorting process by assisting with the identification and labeling of some cell types, which was necessary to perform a supervised machine learning task.

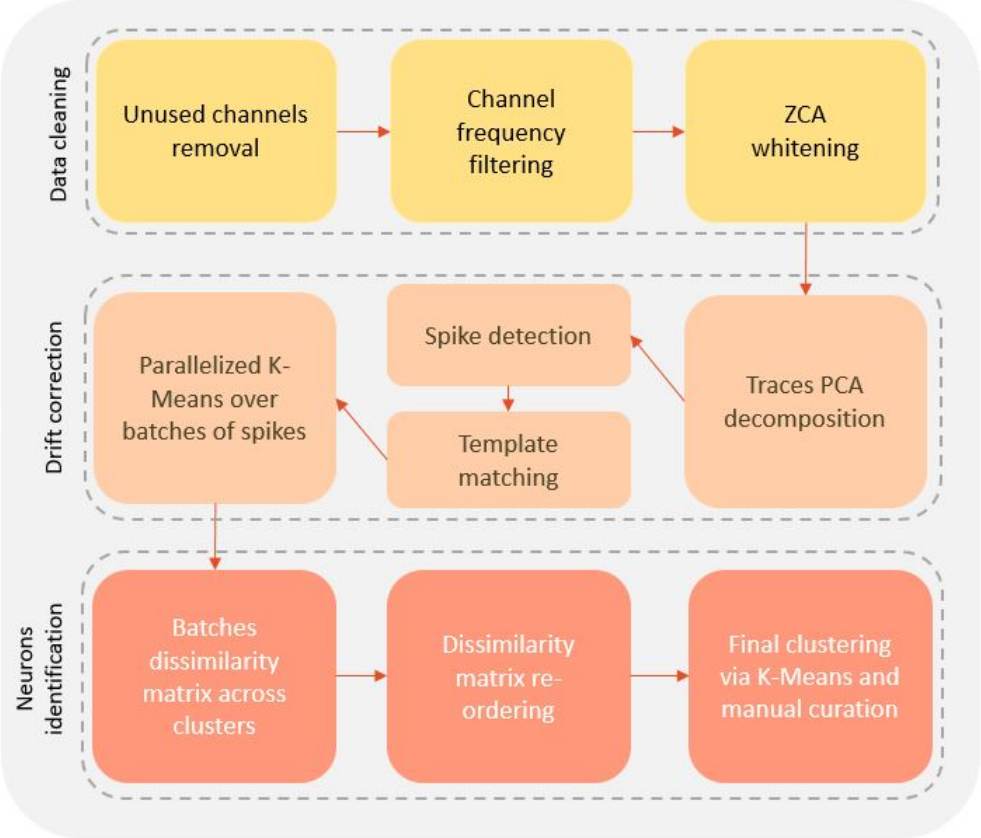

Figure B.2: Spike Sorting process to clean electrophysiological data recordings. Adapted from [Pachitariu 2016].

[2]See section *Optogenetics: ground-truth data labeling* in Chapter 3 for more details.

Appendix C

Tuned hyper-parameters for supervised classifiers

The following hyper-parameters were tuned for each of the following classifiers:

Logistic Regression

1. *Solver:* algorithm to use in the optimization problem. Both *linear* and *saga* solvers were considered. The saga solver is an stochastic gradient method within the family of the Stochastic Average Gradient solvers that supports the use of L1 regularization.

2. *Penalty function:* the norm used in the penalization. Both L1 (Lasso) and L2 (Ridge) penalties were considered to reduce overfitting, since it is unclear which one leads to better results. In particular, Lasso performs feature selection by possibly shrinking some variable coefficients to zero, whilst Ridge does not eliminate any dimension in a sparse feature space, being both options potentially beneficial.

3. *Stopping tolerance* (*tol*)*:* tolerance for stopping the optimization criteria. A set of values in the range [1e-6, 1e-5, 1e-4, 1e-3, 1e-2] was considered.

4. *Regularization strength* (*C*)*:* inverse of regularization strength, so that smaller values specify stronger regularization. A set of values in the range [0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.1, 10, 100, 150] was considered.

5. *Class weight:* parameter to train a multi-label classifier based on either a balanced data set or the specific weights associated with each class. Both options were considered.

Passive Aggresive Classifier

1. *Stopping tolerance* (*tol*)*:* tolerance for stopping the optimization criteria. A set of values in the range [1e-6, 1e-5, 1e-4, 1e-3, 1e-2] was considered.

2. *Loss function:* options *Hinge* and *Hinge Squared* were considered. Hinge is equivalent to PA-I in the reference paper, while Hinge Squared corresponds to PA-II in the reference paper [Crammer 2006].

3. *Regularization strength* (*C*)*:* inverse of regularization strength, so that smaller values specify stronger regularization. A set of values in the range [0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.1, 10, 100, 150] was considered.

4. *Class weight:* parameter to train a multi-label classifier based on either a balanced data set or the specific weights associated with each class. Both options were considered.

Decision Trees and Random Forest

1. *Maximum depth:* the maximum depth of the tree, tuned to avoid overfitting, case in which all the nodes are expanded until all leaves are pure. A set of values in the range [4, 5, 6, 7] was considered.

2. *Optimization criterion:* function to measure the quality of a split. Both *Gini* impurity and *entropy* (information gain) was considered.

3. *Minimum samples to split:* the minimum number of samples required to split an internal node. A set of values in the range [5, 6, 7, 8, 9, 10] was considered.

4. *Minimum samples per leaf:* the minimum number of samples required to be at a leaf node. A set of values in the range [5, 6, 7, 8, 9, 10] was considered.

5. *Class weight:* parameter to train a multi-label classifier based on either a balanced data set or the specific weights associated with each class. Both options were considered.

6. *Maximum features:* the number of features to consider when looking for the best split. By default, all dimensions were considered.

XGBoost

1. *Learning rate* (*eta*)*:* acts as a regularizator for the training process. Similar to a learning rate in stochastic optimization. A set of values in the range [0.3, 0.2, 0.1, 0.05, 0.01, 0.005] was considered.

2. *Maximum depth:* the maximum depth of the tree, tuned to avoid overfitting, case in which all the nodes are expanded until all leaves are pure. A set of values in the range [6, 7, 8, 9, 10, 11, 12] was considered.

3. *Minimum child weight:* the minimum sum of weights of all observations required in a child. Acts as a regularizator for the training process such that the adequate values prevent the model from learning highly specific relations for a subset of trees.

4. *Subsample size:* the fraction of observations to be randomly selected for each tree. A set of values in the range [0.7, 0.8, 0.9, 1, 1.1] was considered.

5. *Colsample size:* the fraction of columns to be randomly selected for each tree. A set of values in the range [0.7, 0.8, 0.9, 1, 1.1] was considered.

Support Vector Machines

1. *Kernel:* the kernel type to be used in the algorithm. *Linear, polynomial, rbf,* and *sigmoid* functions were considered.

2. *Gamma:* kernel coefficient for *polynomial, rbf* and *sigmoid* functions. The two options considered were $1/(n_features * X.var())$ and $1/n_features$.

3. *Stopping tolerance* (*tol*)*:* tolerance for stopping the optimization criteria. A set of values in the range [1e-6, 1e-5, 1e-4, 1e-3, 1e-2] was considered.

4. *Regularization strength* (*C*)*:* inverse of regularization strength, so that smaller values specify stronger regularization. A set of values in the range [0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.1, 10, 100, 150] was considered.

5. *Class weight:* parameter to train a multi-label classifier based on either a balanced data set or the specific weights associated with each class. Both options were considered.

K-Nearest Neighbors

1. *Number of neighbors:* a set of values in the range [2, 3, 4, 5, 6, 7] was considered.

2. *Weights:* weight function used in prediction. Two scenarios were considered. For option *uniform*, all points in each neighborhood are weighted equally. For option *distance*, closer neighbors will have a greater influence than farther ones.

3. *Algorithms:* nearest neighbors could be computed by one of these algorithms: *BallTree, KDTree* or brute-force search.

4. *Distance metric:* both *manhattan* and *euclidean* distances were cross-validated.

Bibliography

[Anderberg 1973] Michael R Anderberg. Cluster analysis for applications: probability and mathematical statistics: a series of monographs and textbooks, volume 19. Academic press, 1973. (Cited on page 27.)

[Anderson 2020] David R Anderson, Dennis J Sweeney, Thomas A Williams, Jeffrey D Camm and James J Cochran. Modern business statistics with microsoft excel. Cengage Learning, 2020. (Cited on page 73.)

[Apps 2005] Richard Apps and Martin Garwicz. Anatomical and physiological foundations of cerebellar information processing. Nature Reviews Neuroscience, vol. 6, no. 4, pages 297–311, 2005. (Cited on page 115.)

[Arendt 2016] Detlev Arendt, Jacob M Musser, Clare VH Baker, Aviv Bergman, Connie Cepko, Douglas H Erwin, Mihaela Pavlicev, Gerhard Schlosser, Stefanie Widder, Manfred D Laubichleret al. The origin and evolution of cell types. Nature Reviews Genetics, vol. 17, no. 12, page 744, 2016. (Cited on page 8.)

[Ayodele 2010] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. New advances in machine learning, vol. 3, pages 19–48, 2010. (Cited on page 39.)

[Backer 1995] Eric Backer. Computer-assisted reasoning in cluster analysis. Prentice Hall International (UK) Ltd., 1995. (Cited on page 27.)

[Backhouse 2007] Andy Backhouse, Irene YH Gu and Tiesheng Wang. ML Nonlinear Smoothing for Image Segmentation and its Relationship to the Mean Shift. In 2007 IEEE International Conference on Image Processing, volume 4, pages IV–337. IEEE, 2007. (Cited on page 27.)

[Bagnall 2013] Martha Bagnall, Sascha [du Lac] and Michael Mauk. Chapter 31 - Cerebellum. In Larry R. Squire, Darwin Berg, Floyd E. Bloom, Sascha [du Lac], Anirvan Ghosh and Nicholas C. Spitzer, editeurs, Fundamental Neuroscience (Fourth Edition), pages 677 – 696. Academic Press, San Diego, fourth edition édition, 2013. (Cited on page 1.)

[Banfield 1993] Jeffrey D Banfield and Adrian E Raftery. Model-based Gaussian and non-Gaussian clustering. Biometrics, pages 803–821, 1993. (Cited on page 30.)

[Barmack 2008] Neal H Barmack and Vadim Yakhnitsa. Functions of interneurons in mouse cerebellum. Journal of Neuroscience, vol. 28, no. 5, pages 1140–1152, 2008. (Cited on page 14.)

[Bayer 2016] Research Division Bayer. Controlling cells using optogenetics methods. pages 28–33, 2016. (Cited on page 57.)

[Bean 2007] Bruce P Bean. The action potential in mammalian central neurons. Nature Reviews Neuroscience, vol. 8, no. 6, pages 451–465, 2007. (Cited on page 16.)

[Bell 1997] Anthony J Bell and Terrence J Sejnowski. Edges are the'Independent Components' of Natural Scenes. In Advances in neural information processing systems, pages 831–837, 1997. (Cited on page 25.)

[Bezdek 1984] James C Bezdek, Robert Ehrlich and William Full. FCM: The fuzzy c-means clustering algorithm. Computers & Geosciences, vol. 10, no. 2-3, pages 191–203, 1984. (Cited on page 38.)

[Bhumbra 2005] GS Bhumbra and REJ Dyball. Spike coding from the perspective of a neurone. Cognitive processing, vol. 6, no. 3, pages 157–176, 2005. (Cited on page 73.)

[Bishop 1995] Christopher M Bishopet al. Neural networks for pattern recognition. Oxford university press, 1995. (Cited on page 25.)

[Bishop 2006] Christopher M Bishop. Pattern recognition and machine learning. springer, 2006. (Cited on page 41.)

[Böhm 2017] Christian Böhm, Martin Perdacher and Claudia Plant. Multi-core k-means. In Proceedings of the 2017 SIAM International Conference on Data Mining, pages 273–281. SIAM, 2017. (Cited on page 27.)

[Bousbaci 2014] Abdelhak Bousbaci and Nadjet Kamel. A parallel sampling-pso-multi-core-k-means algorithm using mapreduce. In 2014 14th International Conference on Hybrid Intelligent Systems, pages 129–134. IEEE, 2014. (Cited on page 27.)

[Breiman 2001] Leo Breiman. Random forests. Machine learning, vol. 45, no. 1, pages 5–32, 2001. (Cited on page 45.)

[Brock 2008] Guy Brock, Vasyl Pihur, Susmita Datta and Somnath Datta. clValid: An R Package for Cluster Validation. Journal of Statistical Software, vol. 25, no. 4, pages 1–22, 2008. (Cited on page 89.)

[Buitinck 2013] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pages 108–122, 2013. (Cited on pages 102 and 105.)

[Bunkhumpornpat 2012] Chumphol Bunkhumpornpat, Krung Sinapiromsaran and Chidchanok Lursinsap. DBSMOTE: density-based synthetic minority over-sampling technique. Applied Intelligence, vol. 36, no. 3, pages 664–684, 2012. (Cited on page 47.)

[Burges 2010] Christopher JC Burges. From Ranknet to Lambdarank to Lambdamart: An Overview.; 2010, 2010. (Cited on page 45.)

[Bzdok 2017] Danilo Bzdok and BT Thomas Yeo. Inference in the age of big data: Future perspectives on neuroscience. Neuroimage, vol. 155, pages 549–564, 2017. (Cited on page 15.)

[Cattell 1943] Raymond B Cattell. The description of personality: Basic traits resolved into clusters. The journal of abnormal and social psychology, vol. 38, no. 4, page 476, 1943. (Cited on page 27.)

[Cayton 2005] Lawrence Cayton. Algorithms for manifold learning. Univ. of California at San Diego Tech. Rep, vol. 12, no. 1-17, page 1, 2005. (Cited on page 120.)

[Chadderton 2004] Paul Chadderton, Troy W Margrie and Michael Häusser. Integration of quanta in cerebellar granule cells during sensory processing. Nature, vol. 428, no. 6985, pages 856–860, 2004. (Cited on page 14.)

[Chawla 2002] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall and W Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, vol. 16, pages 321–357, 2002. (Cited on page 46.)

[Chen 2015a] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich and Yuan Tang. Xgboost: extreme gradient boosting. R package version 0.4-2, pages 1–4, 2015. (Cited on page 45.)

[Chen 2015b] Tianqi Chen, Sameer Singh, Ben Taskar and Carlos Guestrin. Efficient second-order gradient boosting for conditional random fields. In Artificial Intelligence and Statistics, pages 147–155, 2015. (Cited on page 45.)

[Chen 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pages 785–794, 2016. (Cited on pages 41 and 44.)

[Chorev 2009] Edith Chorev, Jérôme Epsztein, Arthur R Houweling, Albert K Lee and Michael Brecht. Electrophysiological recordings from behaving animals—going beyond spikes. Current opinion in neurobiology, vol. 19, no. 5, pages 513–519, 2009. (Cited on page 1.)

[Chu 2012] Chun-Ping Chu, Yan-Hua Bing, Heng Liu and De-Lai Qiu. Roles of molecular layer interneurons in sensory information processing in mouse cerebellar cortex Crus II in vivo. PLoS One, vol. 7, no. 5, 2012. (Cited on page 116.)

[Cook 2020] Anna A Cook, Eviatar Fields and Alanna J Watt. Losing the beat: contribution of Purkinje cell firing dysfunction to disease, and its reversal. Neuroscience, 2020. (Cited on page 12.)

[Crammer 2006] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz and Yoram Singer. Online passive-aggressive algorithms. Journal of Machine Learning Research, vol. 7, no. Mar, pages 551–585, 2006. (Cited on page 123.)

[Crick 1999] Francis Crick. The impact of molecular biology on neuroscience. Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences, vol. 354, no. 1392, pages 2021–2025, 1999. (Cited on page 8.)

[Davies 1979] David L Davies and Donald W Bouldin. A cluster separation measure. IEEE transactions on pattern analysis and machine intelligence, no. 2, pages 224–227, 1979. (Cited on page 38.)

[Deisseroth 2011] Karl Deisseroth. Optogenetics. Nature methods, vol. 8, no. 1, pages 26–29, 2011. (Cited on pages 4 and 57.)

[Dhaliwal 2018] Sukhpreet Singh Dhaliwal, Abdullah-Al Nahid and Robert Abbas. Effective intrusion detection system using XGBoost. Information, vol. 9, no. 7, page 149, 2018. (Cited on page 45.)

[Dietterich 1995] Tom Dietterich. Overfitting and undercomputing in machine learning. ACM computing surveys (CSUR), vol. 27, no. 3, pages 326–327, 1995. (Cited on page 47.)

[Dorval 2011] Alan D Dorval. Estimating neuronal information: logarithmic binning of neuronal inter-spike intervals. Entropy, vol. 13, no. 2, pages 485–501, 2011. (Cited on pages 16 and 19.)

[Driver 1932] H.E. Driver and A.L. Kroeber. Quantitative expression of cultural relationships. Publications in American archaeology and ethnology. University of California Press, 1932. (Cited on page 27.)

[Dunn 1973] Joseph C Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. Journal of Cybernetics, 1973. (Cited on page 38.)

[Duran 1974] Benjamin S Duran and Patrick L Odell. Cluster analysis: a survey. Springer Science & Business Media, 1974. (Cited on page 27.)

[D'Angelo 2008] Egidio D'Angelo. The critical role of Golgi cells in regulating spatio-temporal integration and plasticity at the cerebellum input stage. Frontiers in neuroscience, vol. 2, page 8, 2008. (Cited on page 116.)

[Eccles 1973] JC Eccles. Review lecture. The cerebellum as a computer: Patterns in space and time. J. Physiol.(Lond.), vol. 229, pages 1–32, 1973. (Cited on page 115.)

[Everitt 1979] Brian S Everitt. Unresolved problems in cluster analysis. Biometrics, pages 169–181, 1979. (Cited on page 27.)

[Everitt 2001] Brian S Everitt, S Landau and M Leese. Cluster analysis arnold. A member of the Hodder Headline Group, London, pages 429–438, 2001. (Cited on page 35.)

[Fahim 2014] Ahmed M Fahim. Parallel implementation of K-means on multi-core processors. Computer Sciences and Telecommunications, no. 1, pages 53–61, 2014. (Cited on page 27.)

[Fan 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. Journal of machine learning research, vol. 9, no. Aug, pages 1871–1874, 2008. (Cited on page 45.)

[Fan 2018] Junliang Fan, Xiukang Wang, Lifeng Wu, Hanmi Zhou, Fucang Zhang, Xiang Yu, Xianghui Lu and Youzhen Xiang. Comparison of Support Vector Machine and Extreme Gradient Boosting for predicting daily global solar radiation using temperature and precipitation in humid subtropical climates: A case study in China. Energy Conversion and Management, vol. 164, pages 102–111, 2018. (Cited on pages 45 and 104.)

[Fernández 2018] Alberto Fernández, Salvador Garcia, Francisco Herrera and Nitesh V Chawla. SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. Journal of artificial intelligence research, vol. 61, pages 863–905, 2018. (Cited on page 5.)

[Figueiredo 1999] Málrio AT Figueiredo, José MN Leitão and Anil K Jain. On fitting mixture models. In International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition, pages 54–69. Springer, 1999. (Cited on page 30.)

[Fratello 2018] Michele Fratello and Roberto Tagliaferri. Decision trees and random forests. Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics, page 374, 2018. (Cited on page 12.)

[Fred 2003] Ana LN Fred and José MN Leitão. A new cluster isolation criterion based on dissimilarity increments. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 8, pages 944–958, 2003. (Cited on pages 30, 31 and 32.)

[Friedman 2000] Jerome Friedman, Trevor Hastie, Robert Tibshiraniet al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics, vol. 28, no. 2, pages 337–407, 2000. (Cited on page 45.)

[Girolami 2006] Mark Girolami and Simon Rogers. Variational Bayesian multinomial probit regression with Gaussian process priors. Neural Computation, vol. 18, no. 8, pages 1790–1817, 2006. (Cited on page 14.)

[Glasser 2016] Matthew F Glasser, Timothy S Coalson, Emma C Robinson, Carl D Hacker, John Harwell, Essa Yacoub, Kamil Ugurbil, Jesper Andersson, Christian F Beckmann, Mark Jenkinsonet al. A multi-modal parcellation of human cerebral cortex. Nature, vol. 536, no. 7615, pages 171–178, 2016. (Cited on page 3.)

[Greenwald 2001] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. ACM SIGMOD Record, vol. 30, no. 2, pages 58–66, 2001. (Cited on page 45.)

[Grira 2004] Nizar Grira, Michel Crucianu and Nozha Boujemaa. Unsupervised and semi-supervised clustering: a brief survey. A review of machine learning techniques for processing multimedia content, vol. 1, pages 9–16, 2004. (Cited on pages 30, 31, 32 and 39.)

[Guyon 2010] Isabelle Guyon, Amir Saffari, Gideon Dror and Gavin Cawley. Model Selection: Beyond the Bayesian/Frequentist Divide. Journal of Machine Learning Research, vol. 11, no. 1, 2010. (Cited on page 41.)

[Haar 2015] Shlomi Haar, Ronit Givon-Mayo, Neal H Barmack, Vadim Yakhnitsa and Opher Donchin. Spontaneous activity does not predict morphological type in cerebellar interneurons. Journal of Neuroscience, vol. 35, no. 4, pages 1432–1442, 2015. (Cited on page 110.)

[Haider 2013] Bilal Haider, Michael Häusser and Matteo Carandini. Inhibition dominates sensory responses in the awake cortex. Nature, vol. 493, no. 7430, pages 97–100, 2013. (Cited on pages 11 and 52.)

[Halmos 1963] Paul R Halmos. What does the spectral theorem say? The American Mathematical Monthly, vol. 70, no. 3, pages 241–247, 1963. (Cited on page 23.)

[Han 2005] Hui Han, Wen-Yuan Wang and Bing-Huan Mao. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In International conference on intelligent computing, pages 878–887. Springer, 2005. (Cited on page 47.)

[Harman 1960] HH Harman. Modern factor analysis. Chicago: Univer, 1960. (Cited on page 22.)

[Hartigan 1975] John A Hartigan. Clustering algorithms. John Wiley & Sons, Inc., 1975. (Cited on page 27.)

[Hartmann 2001] Mitra J. Hartmann and James M. Bower. Tactile responses in the granule cell layer of cerebellar folium crus IIa of freely behaving rats. Journal of Neuroscience, vol. 21, no. 10, pages 3549–3563, may 2001. (Cited on page 11.)

[He 2008] Haibo He, Yang Bai, Edwardo A Garcia and Shutao Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), pages 1322–1328. IEEE, 2008. (Cited on page 47.)

[Heeger 2000] David Heeger. Poisson model of spike generation. Handout, University of Standford, vol. 5, pages 1–13, 2000. (Cited on page 20.)

[Hensbroek 2014] Robert A Hensbroek, Tim Belton, Boeke J van Beugen, Jun Maruta, Tom JH Ruigrok and John I Simpson. Identifying Purkinje cells using only their spontaneous simple spike activity. Journal of neuroscience methods, vol. 232, pages 173–180, 2014. (Cited on pages 10, 12, 13, 14, 15, 16, 19 and 71.)

[Hevers 2008] Wulf Hevers, Stephen H Hadley, Hartmut Lüddens and Jahanshah Amin. Ketamine, but not phencyclidine, selectively modulates cerebellar GABAA receptors containing α6 and δ subunits. Journal of Neuroscience, vol. 28, no. 20, pages 5383–5393, 2008. (Cited on pages 11 and 52.)

[Hexmoor 2015] Henry Hexmoor. Chapter 6 - Diffusion and Contagion. In Henry Hexmoor, editeur, Computational Network Science, Emerging Trends in Computer

Science and Applied Computing, pages 45 – 64. Morgan Kaufmann, Boston, 2015. (Cited on page 35.)

[Hill 2011] Daniel N Hill, Samar B Mehta and David Kleinfeld. Quality metrics to accompany spike sorting of extracellular signals. Journal of Neuroscience, vol. 31, no. 24, pages 8699–8705, 2011. (Cited on page 64.)

[Ho 2002] Yu-Chi Ho and David L Pepyne. Simple explanation of the no-free-lunch theorem and its implications. Journal of optimization theory and applications, vol. 115, no. 3, pages 549–570, 2002. (Cited on page 92.)

[Holt 1996] Gary R Holt, William R Softky, Christof Koch and Rodney J Douglas. Comparison of discharge variability in vitro and in vivo in cat visual cortex neurons. Journal of neurophysiology, vol. 75, no. 5, pages 1806–1814, 1996. (Cited on page 74.)

[Holtzman 2006] Tahl Holtzman, Thimali Rajapaksa, Abteen Mostofi and Steve A Edgley. Different responses of rat cerebellar Purkinje cells and Golgi cells evoked by widespread convergent sensory inputs. The Journal of physiology, vol. 574, no. 2, pages 491–507, 2006. (Cited on page 19.)

[Holtzman 2011] Tahl Holtzman, Vanessa Sivam, Tian Zhao, Oivier Frey, Peter Dow van der Wal, Nico F de Rooij, Jeffrey W Dalley and Steve A Edgley. Multiple extra-synaptic spillover mechanisms regulate prolonged activity in cerebellar Golgi cell–granule cell loops. The Journal of physiology, vol. 589, no. 15, pages 3837–3854, 2011. (Cited on page 14.)

[Holz 2012] Ronald W Holz and Stephen K Fisher. Synaptic transmission and cellular signaling: an overview. In Basic Neurochemistry, pages 235–257. Elsevier, 2012. (Cited on page 117.)

[Hosmer Jr 2013] David W Hosmer Jr, Stanley Lemeshow and Rodney X Sturdivant. Applied logistic regression, volume 398. John Wiley & Sons, 2013. (Cited on pages 41 and 42.)

[Huang 1997] Zhexue Huang. A fast clustering algorithm to cluster very large categorical data sets in data mining. DMKD, vol. 3, no. 8, pages 34–39, 1997. (Cited on page 31.)

[Hunter 2007] J. D. Hunter. Matplotlib: A 2D graphics environment. Computing in Science & Engineering, vol. 9, no. 3, pages 90–95, 2007. (Cited on page 81.)

[Igelnik 2013] Boris Igelnik. Efficiency and scalability methods for computational intellect. IGI Global, 2013. (Cited on page 35.)

[Jain 1988] Anil K Jain and Richard C Dubes. Algorithms for clustering data. Prentice-Hall, Inc., 1988. (Cited on pages 27 and 31.)

[Jain 1996] Anil K Jain and Patrick J Flynn. Image segmentation using clustering. IEEE Press, Piscataway, NJ, 1996. (Cited on page 27.)

[Jain 1999] Anil K Jain, M Narasimha Murty and Patrick J Flynn. Data clustering: a review. ACM computing surveys (CSUR), vol. 31, no. 3, pages 264–323, 1999. (Cited on page 27.)

[Jeffery 2018] Kate J Jeffery, Roddy Grieves and Jim Donnett. Recording the Spatial Mapping Cells: Place, Head Direction, and Grid Cells. In Handbook of Behavioral Neuroscience, volume 28, pages 95–121. Elsevier, 2018. (Cited on pages 19 and 20.)

[Ji 2019] Cun Ji, Xiunan Zou, Yupeng Hu, Shijun Liu, Lei Lyu and Xiangwei Zheng. XG-SF: An XGBoost classifier based on shapelet features for time series classification. Procedia computer science, vol. 147, pages 24–28, 2019. (Cited on page 45.)

[Jia 2019] Xiaoxuan Jia, Joshua H Siegle, Corbett Bennett, Samuel D Gale, Daniel J Denman, Christof Koch and Shawn R Olsen. High-density extracellular probes reveal dendritic backpropagation and facilitate neuron classification. Journal of neurophysiology, vol. 121, no. 5, pages 1831–1847, 2019. (Cited on pages 17, 18 and 75.)

[Johnson 2013] Rie Johnson and Tong Zhang. Learning nonlinear functions using regularized greedy forest. IEEE transactions on pattern analysis and machine intelligence, vol. 36, no. 5, pages 942–954, 2013. (Cited on page 45.)

[Jolliffe 2002] Ian T Jolliffe. Springer series in statistics. Principal component analysis, vol. 29, 2002. (Cited on page 22.)

[Josh Huang 2013] Z Josh Huang and Hongkui Zeng. Genetic approaches to neural circuits in the mouse. Annual review of neuroscience, vol. 36, pages 183–215, 2013. (Cited on page 8.)

[Jun 2017] James J Jun, Nicholas A Steinmetz, Joshua H Siegle, Daniel J Denman, Marius Bauza, Brian Barbarits, Albert K Lee, Costas A Anastassiou, Alexandru Andrei, Çağatay Aydınet al. Fully integrated silicon probes for high-density recording of neural activity. Nature, vol. 551, no. 7679, pages 232–236, 2017. (Cited on pages 3 and 9.)

[Kanungo 2002] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pages 881–892, 2002. (Cited on page 27.)

[Kawato 2003] Mitsuo Kawato, Tomoe Kuroda, Hiroshi Imamizu, Eri Nakano, Satoru Miyauchi and Toshinori Yoshioka. Internal forward models in the cerebellum: fMRI study on grip force and load force coupling. In Neural Control of Space Coding and Action Production, volume 142 of Progress in Brain Research, pages 171 – 188. Elsevier, 2003. (Cited on pages 1 and 109.)

[Kerdprasop 2010] Kittisak Kerdprasop and Nittaya Kerdprasop. Parallelization of k-means clustering on multi-core processors. In Proceedings of the 10th WSEAS

international conference on Applied computer science, volume 10, pages 472–477. World Scientific and Engineering Academy and Society (WSEAS), 2010. (Cited on page 27.)

[Ketchen 1996] David J Ketchen and Christopher L Shook. The application of cluster analysis in strategic management research: an analysis and critique. Strategic management journal, vol. 17, no. 6, pages 441–458, 1996. (Cited on page 36.)

[Khonsary 2016] Seyed Ali Khonsary. Molecular and Cellular Physiology of Neurons. Surgical Neurology International, vol. 7, page 311, 2016. (Cited on pages 63 and 88.)

[Kish 2015] Eszter A Kish, Claes-Göran Granqvist, András Dér and Laszlo B Kish. Lognormal distribution of firing time and rate from a single neuron? Cognitive neurodynamics, vol. 9, no. 4, pages 459–462, 2015. (Cited on page 20.)

[Konnerth 1990] Arthur Konnerth, Isabel Llano and Clay M Armstrong. Synaptic currents in cerebellar Purkinje cells. Proceedings of the National Academy of Sciences, vol. 87, no. 7, pages 2662–2665, 1990. (Cited on page 12.)

[Kozareva 2020] Velina Kozareva, Caroline Martin, Tomas Osorno, Stephanie Rudolph, Chong Guo, Charles Vanderburg, Naeem M Nadaf, Aviv Regev, Wade Regehr and Evan Macosko. A transcriptomic atlas of the mouse cerebellum reveals regional specializations and novel cell types. bioRxiv, 2020. (Cited on page 2.)

[Kraus 2010] Johann M Kraus and Hans A Kestler. A highly efficient multi-core algorithm for clustering extremely large datasets. BMC bioinformatics, vol. 11, no. 1, page 169, 2010. (Cited on page 31.)

[Kwak 2017] Sang Gyu Kwak and Jong Hae Kim. Central limit theorem: the cornerstone of modern statistics. Korean journal of anesthesiology, vol. 70, no. 2, page 144, 2017. (Cited on pages 58 and 111.)

[Lemaître 2017] Guillaume Lemaître, Fernando Nogueira and Christos K. Aridas. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. Journal of Machine Learning Research, vol. 18, no. 17, pages 1–5, 2017. (Cited on page 99.)

[Lohninger 2010] H Lohninger. Fundamentals of statistics. Retrieved December, vol. 5, page 2010, 2010. (Cited on page 40.)

[Lorenz 1956] E.N. Lorenz and Statistical Forecasting Project (Massachusetts Institute of Technology). Empirical orthogonal functions and statistical weather prediction. Scientific report. Massachusetts Institute of Technology, Department of Meteorology, 1956. (Cited on page 22.)

[Loyola-Gonzalez 2019] Octavio Loyola-Gonzalez. Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. IEEE Access, vol. 7, pages 154096–154113, 2019. (Cited on page 40.)

[MacQueen 1967] James MacQueenet al. Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, no. 14, pages 281–297, 1967. (Cited on page 30.)

[Mahalanobis 1936] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. Proceedings of National Institute of Sciences (India), vol. 2, pages 49–55, 1936. (Cited on page 29.)

[Malyshev 2015] Aleksey Malyshev, Roman Goz, Joseph J LoTurco and Maxim Volgushev. Advantages and limitations of the use of optogenetic approach in studying fast-scale spike encoding. PloS one, vol. 10, no. 4, page e0122286, 2015. (Cited on page 3.)

[Marr 1991] David Marr and W Thomas Thach. A theory of cerebellar cortex. In From the Retina to the Neocortex, pages 11–50. Springer, 1991. (Cited on page 1.)

[Masland 2012] Richard H Masland. The neuronal organization of the retina. Neuron, vol. 76, no. 2, pages 266–280, 2012. (Cited on page 3.)

[McLachlan 1988] Geoffrey J McLachlan and Kaye E Basford. Mixture models. Inference and applications to clustering. mmia, 1988. (Cited on page 30.)

[McLeod 2000] Kari S McLeod. Our sense of Snow: the myth of John Snow in medical geography. Social science & medicine, vol. 50, no. 7-8, pages 923–935, 2000. (Cited on page 26.)

[Moore 2001] Andrew W Moore. Cross-validation for detecting and preventing overfitting. School of Computer Science Carneigie Mellon University, 2001. (Cited on page 48.)

[Morde 2019] Vishal Morde and Venkat Anurag. XGBoost Algorithm: Long May She Reign! 2019. (Cited on pages 44 and 46.)

[Mugnaini 2011] Enrico Mugnaini, Gabriella Sekerková and Marco Martina. The unipolar brush cell: a remarkable neuron finally receiving deserved attention. Brain research reviews, vol. 66, no. 1-2, pages 220–245, 2011. (Cited on page 10.)

[Murphy 2012] Kevin P Murphy. Machine learning: a probabilistic perspective. MIT press, 2012. (Cited on page 40.)

[Nam 2014] Woonhyun Nam, Piotr Dollár and Joon Hee Han. Local decorrelation for improved detection. arXiv preprint arXiv:1406.1134, 2014. (Cited on page 26.)

[Natekin 2013] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. Frontiers in neurorobotics, vol. 7, page 21, 2013. (Cited on page 41.)

[Ng 2011] Andrew Ng. Advice for applying machine learning. In Machine learning. 2011. (Cited on page 112.)

[Niculescu-Mizil 2005] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In Proceedings of the 22nd international conference on Machine learning, pages 625–632, 2005. (Cited on pages 40 and 42.)

[Oliphant 2006] Travis E Oliphant. A guide to numpy, volume 1. Trelgol Publishing USA, 2006. (Cited on pages 69 and 71.)

[Özcan 2020] Orçun Orkan Özcan, Xiaolu Wang, Francesca Binda, Kevin Dorgans, Chris I. De Zeeuw, Zhenyu Gao, Ad Aertsen, Arvind Kumar and Philippe Isope. Differential Coding Strategies in Glutamatergic and GABAergic Neurons in the Medial Cerebellar Nucleus. Journal of Neuroscience, vol. 40, no. 1, pages 159–170, 2020. (Cited on pages 10, 17, 18, 71, 75 and 76.)

[Pachitariu 2016] Marius Pachitariu, Nicholas Steinmetz, Shabnam Kadir, Matteo Carandini and Harris Kenneth D. Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels. bioRxiv, 2016. (Cited on pages 55, 56, 121 and 122.)

[Pal 1995] Nikhil R Pal and James C Bezdek. On cluster validity for the fuzzy c-means model. IEEE Transactions on Fuzzy systems, vol. 3, no. 3, pages 370–379, 1995. (Cited on page 31.)

[Palmer 2010] Lucy M Palmer, Beverley A Clark, Jan Gründemann, Arnd Roth, Greg J Stuart and Michael Häusser. RAPID REPORT: Initiation of simple and complex spikes in cerebellar Purkinje cells. The Journal of physiology, vol. 588, no. 10, pages 1709–1717, 2010. (Cited on page 87.)

[Paul 2019] Manika S Paul and Faten Limaiem. Histology, Purkinje Cells. In StatPearls [Internet]. StatPearls Publishing, 2019. (Cited on page 116.)

[Pearson 1901] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, vol. 2, no. 11, pages 559–572, 1901. (Cited on page 22.)

[Pedregosa 2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay. Scikit-learn: Machine Learning in Python . Journal of Machine Learning Research, vol. 12, pages 2825–2830, 2011. (Cited on pages 29, 47, 48 and 102.)

[Peikari 2018] Mohammad Peikari, Sherine Salama, Sharon Nofech-Mozes and Anne L Martel. A cluster-then-label semi-supervised learning approach for pathology image classification. Scientific reports, vol. 8, no. 1, pages 1–13, 2018. (Cited on page 5.)

[Purves 2001] D Purves, GJ Augustine, D Fitzpatrick, LC Katz, AS LaMantia, JO McNamara and S Mark Williams. Circuits within the Cerebellum.). Neuroscience, 2nd edition edn. Sunderland (MA): Sinauer Associates. pˆ pp, 2001. (Cited on page 2.)

[R Core Team 2013] R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0. (Cited on pages 85 and 90.)

[Ramon 1911] Y Ramon and S Cajal. Histologie du systeme nerveux de l'homme et des vertebres. Maloine, Paris, vol. 2, pages 153–173, 1911. (Cited on page 73.)

[Rani[1] 2013] Yogita Rani[1] and Harish Rohil. A study of hierarchical clustering algorithm. ter S & on Te SIT, vol. 2, page 113, 2013. (Cited on page 28.)

[Rao 2010] SN Tirumala Rao, EV Prasad and NB Venkateswarlu. A critical performance study of memory mapping on multi-core processors: An experiment with k-means algorithm with large data mining data sets. International Journal of Computer Applications, vol. 1, no. 9, pages 90–98, 2010. (Cited on page 27.)

[Rasmussen 1992] Edie M Rasmussen. Clustering algorithms. Information retrieval: data structures & algorithms, vol. 419, page 442, 1992. (Cited on pages 27, 33 and 34.)

[Rice 2006] John A Rice. Mathematical statistics and data analysis. Cengage Learning, 2006. (Cited on page 24.)

[Richards 1999] John A Richards and JA Richards. Remote sensing digital image analysis, volume 3. Springer, 1999. (Cited on page 49.)

[Roberts 1998] Stephen J Roberts, Dirk Husmeier, Iead Rezek and William Penny. Bayesian approaches to Gaussian mixture modeling. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 11, pages 1133–1142, 1998. (Cited on page 30.)

[Rohlf 1980] F James Rohlf. Single-link clustering algorithms. IBM Thomas J. Watson Research Division, 1980. (Cited on page 34.)

[Rousseeuw 1987] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, vol. 20, pages 53–65, 1987. (Cited on page 36.)

[Ruigrok 2011] Tom JH Ruigrok, Robert A Hensbroek and John I Simpson. Spontaneous activity signatures of morphologically identified interneurons in the vestibulocerebellum. Journal of Neuroscience, vol. 31, no. 2, pages 712–724, 2011. (Cited on pages 10, 11, 12, 13, 14, 15, 16, 19 and 71.)

[Salton 1991] Gerard Salton and Chris Buckley. Global text matching for information retrieval. Science, vol. 253, no. 5023, pages 1012–1015, 1991. (Cited on page 27.)

[Sanes 2015] Joshua R Sanes and Richard H Masland. The types of retinal ganglion cells: current status and implications for neuronal classification. Annual review of neuroscience, vol. 38, pages 221–246, 2015. (Cited on page 3.)

[Santamaria 2007] Fidel Santamaria, Patrick G Tripp and James M Bower. Feedforward inhibition controls the spread of granule cell–induced Purkinje cell activity in the

cerebellar cortex. Journal of neurophysiology, vol. 97, no. 1, pages 248–263, 2007. (Cited on page 11.)

[Schlkopf 2018] Bernhard Schlkopf, Alexander J Smola and Francis Bach. Learning with kernels: support vector machines, regularization, optimization, and beyond. the MIT Press, 2018. (Cited on page 41.)

[Shalizi 2009] Cosma Shalizi. 9.54 Class 13, Data Mining: Unsupervised learning, Clustering. page 8, 2009. (Cited on pages 31, 33, 34, 35 and 37.)

[Shekhar 2016] Karthik Shekhar, Sylvain W Lapan, Irene E Whitney, Nicholas M Tran, Evan Z Macosko, Monika Kowalczyk, Xian Adiconis, Joshua Z Levin, James Nemesh, Melissa Goldmanet al. Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics. Cell, vol. 166, no. 5, pages 1308–1323, 2016. (Cited on page 3.)

[Shlens 2014] Jonathon Shlens. A tutorial on principal component analysis. arXiv preprint arXiv:1404.1100, 2014. (Cited on pages 22, 23, 24 and 25.)

[Sillitoe 2012] Roy V Sillitoe, YuHong Fu and Charles Watson. Cerebellum. In The mouse nervous system, pages 360–397. Elsevier, 2012. (Cited on page 116.)

[Sirovich 1987] Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. Journal of the Optical Society of America, vol. 4, no. 3, pages 519–524, 1987. (Cited on page 22.)

[Sisodia 2012] Deepti Sisodia, Lokesh Singh, Sheetal Sisodia and Khushboo Saxena. Clustering techniques: a brief survey of different clustering algorithms. International Journal of Latest Trends in Engineering and Technology (IJLTET), vol. 1, no. 3, pages 82–87, 2012. (Cited on page 31.)

[Sokal 1958] Robert R Sokal. A statistical method for evaluating systematic relationships. Univ. Kansas, Sci. Bull., vol. 38, pages 1409–1438, 1958. (Cited on page 35.)

[Song 2018] Sangmin Song, Ji Ah Lee, Ilya Kiselev, Varun Iyengar, Josef G Trapani and Nessy Tania. Mathematical modeling and analyses of interspike-intervals of spontaneous activity in afferent neurons of the zebrafish lateral line. Scientific reports, vol. 8, no. 1, pages 1–14, 2018. (Cited on page 20.)

[Späth 1980] H. Späth. Cluster analysis algorithms for data reduction and classification of objects. Computers and their applications. E. Horwood, 1980. (Cited on page 27.)

[Squire 2012] Larry Squire, Darwin Berg, Floyd E Bloom, Sascha Du Lac, Anirvan Ghosh and Nicholas C Spitzer. Fundamental neuroscience. Academic Press, 2012. (Cited on pages 115 and 116.)

[Stein 1965] Richard B Stein. A theoretical analysis of neuronal variability. Biophysical Journal, vol. 5, no. 2, pages 173–194, 1965. (Cited on page 19.)

[Steinmetz 2018] Nicholas A Steinmetz, Christof Koch, Kenneth D Harris and Matteo Carandini. Challenges and opportunities for large-scale electrophysiology with Neuropixels probes. Current Opinion in Neurobiology, vol. 50, pages 92 – 100, 2018. Neurotechnologies. (Cited on pages 2, 3, 4, 52, 53, 54 and 60.)

[Tasic 2016] Bosiljka Tasic, Vilas Menon, Thuc Nghi Nguyen, Tae Kyung Kim, Tim Jarsky, Zizhen Yao, Boaz Levi, Lucas T Gray, Staci A Sorensen, Tim Dolbeareet al. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. Nature neuroscience, vol. 19, no. 2, page 335, 2016. (Cited on page 3.)

[Taylor 2016] J Paul Taylor, Robert H Brown and Don W Cleveland. Decoding ALS: from genes to mechanism. Nature, vol. 539, no. 7628, pages 197–206, 2016. (Cited on page 8.)

[Tharwat 2018] Alaa Tharwat. Classification assessment methods. Applied Computing and Informatics, 2018. (Cited on pages 48 and 49.)

[Thorndike 1953] Robert L Thorndike. Who belongs in the family? Psychometrika, vol. 18, no. 4, pages 267–276, 1953. (Cited on page 36.)

[Toth 2018] Peter G Toth, Petr Marsalek and Ondrej Pokora. Ergodicity and parameter estimates in auditory neural circuits. Biological cybernetics, vol. 112, no. 1-2, pages 41–55, 2018. (Cited on page 20.)

[Tryon 1939] Robert Choate Tryon. Cluster analysis: correlation profile and orthometric (factor) analysis for the isolation of unities in mind and personality. Edwards brother, Incorporated. Ann Arbor, 1939. (Cited on page 27.)

[Tryon 1958] Robert C Tryon. Cumulative communality cluster analysis. Educational and Psychological Measurement, vol. 18, no. 1, pages 3–35, 1958. (Cited on page 27.)

[Tsubo 2012] Yasuhiro Tsubo, Yoshikazu Isomura and Tomoki Fukai. Power-law inter-spike interval distributions infer a conditional maximization of entropy in cortical neurons. PLoS Comput Biol, vol. 8, no. 4, page e1002461, 2012. (Cited on page 20.)

[Tyree 2011] Stephen Tyree, Kilian Q Weinberger, Kunal Agrawal and Jennifer Paykin. Parallel boosted regression trees for web search ranking. In Proceedings of the 20th international conference on World wide web, pages 387–396, 2011. (Cited on page 45.)

[Ullman 2014] S. Ullman, T. Poggio, D. Harari, D. Zysman and D. Seibert. 36-350, Data Mining: Distances between Clustering, Hierarchical Clustering. Center for Brains, Minds & Machines, 2014. (Cited on pages 30, 32, 35 and 37.)

[Van Asch 2013] Vincent Van Asch. Macro-and micro-averaged evaluation measures [[basic draft]]. Belgium: CLiPS, vol. 49, 2013. (Cited on page 49.)

[van Dijck 2013] Gert van Dijck, Marc M. van Hulle, Shane A. Heiney, Pablo M. Blazquez, Hui Meng, Dora E. Angelaki, Alexander Arenz, Troy W. Margrie, Abteen Mostofi,

Steve Edgley, Fredrik Bengtsson, Carl Fredrik Ekerot, Henrik Jörntell, Jeffrey W. Dalley and Tahl Holtzman. Probabilistic Identification of Cerebellar Cortical Neurones across Species. PLoS One, vol. 8, no. 3, mar 2013. (Cited on pages 1, 3, 4, 10, 14, 15, 16, 19, 71, 88 and 109.)

[Van Loan 1983] Charles F Van Loan and Gene H Golub. Matrix computations. Johns Hopkins University Press Baltimore, 1983. (Cited on page 22.)

[Velmurugan 2010] T Velmurugan and T Santhanam. Computational complexity between K-means and K-medoids clustering algorithms for normal and uniform distributions of data points. Journal of computer science, vol. 6, no. 3, page 363, 2010. (Cited on page 30.)

[Virtanen 2020] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 2020. (Cited on pages 71, 73, 75 and 80.)

[Wagner 2017] Mark J Wagner, Tony Hyun Kim, Joan Savall, Mark J Schnitzer and Liqun Luo. Cerebellar granule cells encode the expectation of reward. Nature, vol. 544, no. 7648, pages 96–100, 2017. (Cited on page 116.)

[Walesiak 2008] Marek Walesiak, Andrzej Dudek and MA Dudek. clusterSim: Searching for optimal clustering procedure for a data set. R package version 0.36-1, 2008. (Cited on page 89.)

[Williams 2000] Robert W Williams. Mapping genes that modulate mouse brain development: a quantitative genetic approach. In Mouse brain development, pages 21–49. Springer, 2000. (Cited on page 52.)

[Yadav 2016] Sanjay Yadav and Sanyam Shukla. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In 2016 IEEE 6th International conference on advanced computing (IACC), pages 78–83. IEEE, 2016. (Cited on page 13.)

[Yin 1998] Peng-Yeng Yin. Algorithms for straight line fitting using k-means. Pattern recognition letters, vol. 19, no. 1, pages 31–41, 1998. (Cited on page 31.)

[Yonehara 2016] Keisuke Yonehara, Michele Fiscella, Antonia Drinnenberg, Federico Esposti, Stuart Trenholm, Jacek Krol, Felix Franke, Brigitte Gross Scherf, Akos Kusnyerik, Jan Müller et al. Congenital nystagmus gene FRMD7 is necessary for establishing a neuronal circuit asymmetry for direction selectivity. Neuron, vol. 89, no. 1, pages 177–193, 2016. (Cited on page 8.)

[Zeng 2017] Hongkui Zeng and Joshua R Sanes. Neuronal cell-type classification: challenges, opportunities and the path forward. Nature Reviews Neuroscience, vol. 18, no. 9, page 530, 2017. (Cited on pages 1, 3, 7, 8 and 9.)

[Zhang 2004] Harry Zhang. The Optimality of Naive Bayes, 2004. American Association for Artificial Intelligence (www. aaai. org), 2004. (Cited on pages 41 and 105.)

[Zhang 2007] Qi Zhang and Wei Wang. A fast algorithm for approximate quantiles in high speed data streams. In 19th International Conference on Scientific and Statistical Database Management (SSDBM 2007), pages 29–29. IEEE, 2007. (Cited on page 45.)

[Zubin 1938] Joseph Zubin. A technique for measuring like-mindedness. The Journal of Abnormal and Social Psychology, vol. 33, no. 4, page 508, 1938. (Cited on page 27.)