UCLouvain

École polytechnique de Louvain

Decoding sound categories in time

An electrophysiological study in sighted humans

Author: **Corentin Denis**
Supervisors: **Olivier Collignon, Michel Verleysen**
Readers: **Marco Barilari, Stefania Mattioni**
Academic year 2022–2023
Master [120] in Computer Science and Engineering

Abstract

The goal of this master thesis is to investigate different machine learning procedures for discriminating electroencephalography (EEG) responses to sounds stimuli belonging to different living and non-living categories. Due to the high temporal resolution of EEG, machine learning algorithms can be employed to localise the exact time (in milliseconds) at which the sounds are processed similarly or differently in the brain. Thus, it is possible to explore how representations of different sound categories emerge and resolve in time, which is of great importance in the field of auditory neuroscience. In this thesis, we explored different machine learning algorithms: first, for decoding: classifications algorithms tried were Support Vector Machines (SVM) with linear and radial basis function kernels, linear discriminant analysis; and a little exploration of the SVM C hyperparameter. Additionally, multiclass classification (in contrast to binary classification) was tried. Second, for processing dataset to modify some of its properties (dimension, size, quality of data) and to observe the impact on results: feature extraction and selection, averaging features of different time points, use of pseudo-trials and sliding time-window, and third, for evaluating the results with different cross-validation strategies: stratified KFold, leave exemplar out and leave one trial per exemplar out. It is complemented with metrics measurement that affect the results, which consequently affect the inferences made by neuroscientists about auditory categorisation. Measured metrics are accuracy and Area Under the Curve (AUC).

The results we have obtained suggested that SVM classifier with appropriate hyperparameters is well suited for this type of study. About cross-validation, the standard stratified KFold appears to be good and reliable. Leave examplar out could be useful to evaluate performances of a classifier with yet unseen sound. In several scenarios, results show statistically significant above chance level metrics following the onset of the sound and sometimes, just after the offset of the sound. Other outcomes of this work show that the use of pseudo-trials (trials averaging) leads to a certain improvement in the results obtained for both metrics. However, it has to be employed cautiously as the dataset size is reduced, it can be subject to underfitting. Reducing the dimensions with feature extraction and selection do not show any significant change in the outcomes but allowed for reduced execution times. Multiclass classification has shown interesting results with more prolonged statistical significance after onset of the stimulus than with binary classification. Concerning metrics, we have also made the observation that accuracy and AUC often had dissimilarities in their values.
This little review is left with the hope that a strategic combination of the tools and techniques discussed here can be designed to significantly improve the metrics obtained in future similar EEG decoding studies.

Acknowledgements

I would like to express my gratitude to some people without whom the achievement of this master thesis would have been much more complicated. First, I would like to thank my supervisors: professors Olivier Collignon and Michel Verleysen who gave me the opportunity to discover this beautiful crossover between neuroscience and machine learning and put their trust in me to work on this subject. Also, I would like to greatly thank Siddharth Talwar, for its patience, its great support to the ignorant person that I was (and that I still am, but maybe in a more moderated way), its availability and its huge help overall. Finally, I thank my friend Arthur Gengler and my father Pierre Denis for their re-lectures and advises.

Contents

1 Introduction **4**

Introduction **4**
1.1 Master thesis objectives . . . 4
1.2 Master thesis outline . . . 5

2 Background **6**
2.1 Electroencephalography . . . 6
2.1.1 Applications . . . 6
2.1.2 Advantages and drawbacks . . . 7
2.1.3 The referential montage . . . 7
2.1.4 Artifacts . . . 7
2.1.5 Evoked Related potential . . . 7
2.2 Machine Learning . . . 8
2.2.1 Generalization, overfitting and underfitting . . . 8
2.2.2 Support Vector Machines . . . 9
2.2.3 Linear Discriminant Analysis . . . 12
2.2.4 Evaluation . . . 13
2.2.5 Feature reduction . . . 17
2.2.6 Model selection . . . 19
2.3 Machine learning and EEG . . . 19

3 Pipeline **22**
3.1 Preprocessing . . . 22
3.2 Processing . . . 23
3.2.1 Time-window . . . 24
3.2.2 Time-window averaging . . . 24
3.2.3 Pseudo-trials . . . 24
3.2.4 Feature extraction . . . 25
3.3 Decoding . . . 25
3.3.1 SVM . . . 25
3.3.2 LDA . . . 27
3.3.3 Multiclass classification . . . 27
3.4 Evaluation . . . 29
3.4.1 Cross-Validation . . . 29
3.4.2 Metrics . . . 29
3.5 Statistical assessment . . . 30

4 Results and analysis **31**
4.1 Data . . . 31
4.1.1 Collection . . . 31
4.1.2 Preprocessing . . . 32
4.2 Algorithm for metrics computation . . . 33
4.3 Results . . . 34
4.3.1 Methodology . . . 34
4.3.2 Classifier and cross-validation . . . 35
4.3.3 Time-window . . . 40
4.3.4 Feature extraction . . . 40
4.3.5 Univariate feature selection . . . 41
4.3.6 Hyperparameter . . . 42
4.3.7 Pseudo-trials . . . 42
4.3.8 Time-window averaging . . . 44
4.3.9 Multiclass classification . . . 44

5 Conclusion and perspectives **46**
5.1 Improvements of the results . . . 47
5.2 Future works . . . 48

Chapter 1

Introduction

Since 1924, EEG allowed for the first time to record electrical brain activity. This activity is highly random by nature but contains also relevant information about brain state that can reflect potential neurological disease or brain disorders. Nowadays, decoding EEG signals has vast domains of applications: neuroscientific researches, medical diagnosis, brain-computer interface, etc. Knowing that EEG signals are multidimensional, visual inspection to decode them is a tedious, error-prone and complex labour. Recent advances in machine learning allowed for an efficient, more precise and automatic decoding. But effectively and properly decode EEG data with machine learning is a challenging task that requires specific knowledge and experience. Several reviews about the utilization of machine learning techniques with EEG have been published in the recent years: [1], [2] and for artifacts removal: [3]. Following them, we aimed to provide a work with a concrete application on sound stimuli decoding, with strategic combination of techniques cautiously thought and previously showed to be relevant in literature for this type of study case. In a larger scope, we hope to allow further works to increase global understanding about the brain's processing of sounds across time. Moreover, this study is one of the first to decode auditory stimuli that are not speech decoding [4] with EEG. Therefore, this opportunity can be taken to see if the electric fields produced by the brain contain relevant insights to discriminate sounds between categories in a general manner. Several studies have successfully managed to decode semantic differences from EEG data: [5], [6] with different kind of stimuli, but none have tried to decode from auditory stimuli that represent different living and non-living categories.

1.1 Master thesis objectives

The research questions of the present study are the following: how machine learning tools and techniques used on EEG data affect the final results and interpretations? Which ones are recommendable or not? What are the optimal combinations of techniques to obtain optimum results?

To answer the above-mentioned questions, multiple EEG binary decoders (a decoder, or classifier, is a framework that categorizes data into a category, or class) were trained to subsequently predict from EEG signals what is the category of a listened sound between two categories. This was performed for each recorded time point, for every possible category comparison and for each subject. Using the available data and those decoders, we have determined estimated averaged metrics that reflect the performance of the decoder. Finally, we obtained a time-varying pattern of metrics that could be analyzed to draw interesting conclusions regarding our understanding of the brain. This pattern was plotted for various

machine learning techniques and compared to other.

First, the choice of a classifier was studied. Three of them known to be suited for decoding EEG are tried: support vector machines with linear kernel and with radial basis function kernel and linear discriminant analysis. A first objective of this work is to analyze and compare the latter accompanied with an hyperparameter investigation to find the best combination in terms of metrics values, computational speed, interpretation and generalization. An other objective is to look over different cross-validation strategies and to identify one able to reliably evaluate the performances of the model: stratified KFold with 15 folds or 30 folds, leave exemplar out and designed in the context of this work, leave one trial per exemplar out.

A challenge in decoding EEG signals is the inherent presence of noise in EEG single-trials. It can be from various sources: eye-blinks, involuntary muscles activation, environmental noise, ... It can imply a negative impact on the stability and quality of the results. A third objective is thus to approach this noise problem by using data processing steps (just before feeding the data into the cross-validation process) and see their impact on the decodability. Two of them added temporal context to the data: the first one average the trials of each time point with neighbouring time points (future and past). Second one is a sliding time-window as introduced in [7]. The goal is to bring temporal context by concatenating the features contained in the sliding time-window to create a feature-augmented dataset that is fed to the classifier. An other strategy to approach noise problem was the use of pseudo-trials built from the average of time respective trials. We also wanted to observe the effect of a dimensionality reduction of the data with different feature extraction: principal component analysis with its kernel extension, as well as univariate features selection. Additionally, multiclass classification was experimented to contrast with binary classification. Two strategies were attempted: one vs. one classifier and one vs. rest classifier.

The metrics used to evaluate the performances were accuracy and area under the curve of the receiver operating curve. Those two metrics were analyzed and compared. There were also evaluated for statistical significance beyond chance-level (i.e. metrics obtained with a completely random classifier), and the evolution of "discrimination power" was tracked. Afterwards, it was possible to observe the time points that exhibited the highest discrimination between the experimental conditions and how the "discrimination power" evolved.

The global aim of this work is limited to the machine learning point of view: aforementioned machine learning tools and designs were tried and explored. Their impact on the metrics were observed to finally obtain and advise a model that provides the best results in terms of discriminability, biological validity and control of false positives.

1.2 Master thesis outline

The master thesis is structured as follows: Chapter 2 introduces needed machine learning and EEG concepts to correctly understand the report; chapter 3 details the whole pipeline followed in this master thesis, with specific techniques used in this work as well as justification from literature; chapter 4 details the process of EEG data's acquisition for this study and presents the obtained results; finally, chapter 5 holds an overview of all the techniques used with conclusions, recommendations and suggestions for future works.

Chapter 2

Background

This chapter aims to provide the background knowledge about EEG and machine learning needed to fully understand the content of this work.

2.1 Electroencephalography

Electroencephalography (EEG) is a non-invasive measurement of the brain's electric fields invented by Hans Berger in 1924. Electrodes are placed on the scalp and record the electric fields induced by the current flowing in and around neurons. The brain is divided in two main parts: the inside white matter and the peripheral grey matter. Arising from its non-invasiveness property, the signal recorded by EEG comes exclusively from the grey matter, and captures the neurons' activation in this area. Activation refers to the generation of ionic currents in neurons through biochemical processes at the cellular level. When neurons located in the grey matter of the brain activate together in a specific location, the resulting current sums and induces an electromagnetic field, which is strong enough to be measured. Its electrical and magnetic components are measurable respectively with EEG electrodes and *magnetoencephalography* (MEG) sensors placed on scalp. The rough EEG device is simply made of two electrodes (usually more: e.g. 128 electrodes for this study) that measure a difference of potential (in micro-volts) between two specific location of the scalp. Accordingly, we need a clever zero-reference (e.g. near the mastoids for their electrical neutrality). EEG allows for a high sampling frequency (e.g. 1024 Hz for this study). Brain signals can thus be analyzed with a high temporal resolution. One must keep in mind that this captured metric represents the superficial electrical activity of the brain and does not give insights about the inner electrical manifestations i.e. it is difficult to find precise sources for the observed activity. Additionally, it is important to note that EEG oscillations vary according to the synchronized or desynchronized activity of a whole underlying neuronal population, electrodes cover large portions of the brain. Thus, measured voltage reflects big area activity of the brain [8], [9].

2.1.1 Applications

EEG has wide-spread domains of applications. It has many advantages in clinical diagnostics: range of epileptic seizure type, brain tumor, brain damage from head injury, brain dysfunction, inflammation of the brain, stroke, sleep disorders, Creutzfeldt-Jakob disease [10]. It is also used in building *Brain Computer Interfaces (BCI)*: communication channel between brain and a computer interface or a robotic device. Moreover, EEG is used in many

research realms: neuroscience, cognitive science, cognitive psychology, neurolinguistics and neuropsychological research. [8].

2.1.2 Advantages and drawbacks

There are a lot of alternative methods to study brain function: functional Magnetic Resonance Imaging (fMRI), positron emission tomography, magnetoencephalography (MEG), etc. EEG has unique characteristics that differentiate it from its peers. Regarding the pros: hardware prices are lower than its peers and the needed equipment is minimalist; it has a high temporal resolution (e.g. compared to fMRI); it is silent, which is indispensable to analyze brain responses to auditory stimuli; lastly, it is a non-invasive technique, contrary to electrocorticography. Concerning the drawbacks: EEG compensates its high temporal resolution with low spatial resolution (contrary to fMRI); EEG signals depend on the orientation and location of the reference dipole. Indeed, the neural activities' evoked and spontaneous potentials in EEG recordings are significantly affected by the choice of reference. This influence arises because each EEG electrode captures information about the electrical activity difference between two specific positions on the head. Next, as stated in the section above, EEG poorly measures electrical activity of the brain's lower layer; and finally, the process of connecting a subject to electrodes is long and signals obtained have a poor *Signal to Noise Ratio* (SNR): ratio of desirable signal to the level of background noise [8], [9].

2.1.3 The referential montage

Montage refers to the representation of the EEG channels. There exists a lot of different montages, each having specific use cases. In the referential montage, a channel represents the difference of potential between the channel and a reference point. Often, this reference point is an electrically silent point (e.g. the auricle of the ear) [11].

2.1.4 Artifacts

An *artifact* is an EEG signal that does not originate within the brain. It can have many different sources: eye blink, involuntary muscles, heart beat, environmental noise, ... Artifacts are inherent in EEG and we usually want to get rid of them as they are irrelevant for decoding. They can be removed directly by visual inspection of the raw EEG signal [12] or there are some ways to delete them automatically. For instance, the regression methods, making the assumption that raw EEG data are the sum of relevant EEG data and artifact, subtract to each channel the estimated artifacts built from exogenous reference channels (e.g. the auricle of the ear). A review on methods for automatic artifact removal was conducted in [3]

2.1.5 Evoked Related potential

An *epoch* (or trial) is a chunk, a slice of the multivariate time series produced by the EEG signal across all electrodes that is time-locked to the onset or offset of the stimulus with the EEG evoked response across all the electrodes. It may also include pre-stimulus activity as a measure to compare post-stimulus activity and preprocessing steps such as baseline subtraction.

Hans Berger discovered that the electrical signals obtained with EEG could be influenced by an external event (or stimulus). However, a single EEG epoch struggles to put in evidence the brain reaction to the stimulus. Indeed, single epochs are too noisy. By doing many trials to get multiple epochs and averaging them, a part of the noise cancels-out and evoked potential (electrical activity in areas of the brain and spinal cord in response to certain stimuli) becomes clearly visible. This averaged-epoch is called the *Evoked Related Potential* (ERP). First ERPs were observed by Pauline and Hallowell Davis in 1935-1936 [13]. These event-related responses can be described in terms of their amplitude, latency, duration, etc. ERPs can be analyzed by researchers or in a clinical context to draw conclusions about brain function or medical diagnosis. If we believe that a particular event-related response is reliably modulated by a particular type of psychological event, then we refer to it as an *event-related component* [14].

2.2 Machine Learning

Machine Learning (ML) is a branch of artificial intelligence in which the machine (i.e. the computer) "learns" from experience. It is fed with a collection of n inputs-outputs pairs $D := ((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n))$ with $\mathbf{x} \in \mathbb{R}^{n \times m}, y \in \mathbb{R}^{n \times 1}$. Each input has generally multiple features (m in this general case). A *feature* is an individual measure of a phenomenon or a characteristic. When the set of outputs is continuous it is called *regression*. When the possible output values are contained in a final set, it is called *classification* and binary classification if the set is of size two. Consider that each output has been generated by an unknown function $y = f(\mathbf{x})$. The goal of regression is to learn a model, that is actually an approximation of the unknown function $h = \hat{f}$, that predicts outputs from new inputs. For classification, the function $h(\mathbf{x})$ to find should associate the correct class to a data point $\mathbf{x}$. All these procedures are in the realm of so-called *supervised learning*[1]. Binary classification will be our main focus in this study. To achieve faithful predictions, machine learning uses algorithms that are made through a combination of mathematical principles, statistical techniques, and computational methods. As the computations are usually heavy, the high computational power provided by a computer is essential. There are a bunch of different ML techniques used for building classifiers (or decoders) in supervised learning classification, namely: K-Nearest-Neighbors, support vector machines, random forest, decision tree, linear discriminant analysis, etc. Those ML models generally require a quantitative amount of qualitative data to build accurate models.

2.2.1 Generalization, overfitting and underfitting

This section and section 2.2.2 content were inspired by the ML section of the AIMA book [15]. *Generalization* is the capacity of a model to correctly predict the output value y for novel data points. *Overfitting* is a phenomenon appearing when the function h is too complex for the training data and do not generalize well for novel data points (example on figure 2.1). *Underfitting* is the inverse problem: the model is too simple for the training data and leads also to a bad generalization. For example, in a binary classification problem, it could occur when the separation between classes is a line whilst the data from both classes are non-linearly separable and mixed in a specific pattern. A more complex model could

[1] Unsupervised learning differs from supervised in the fact that the outputs y are unknown and thus not given.

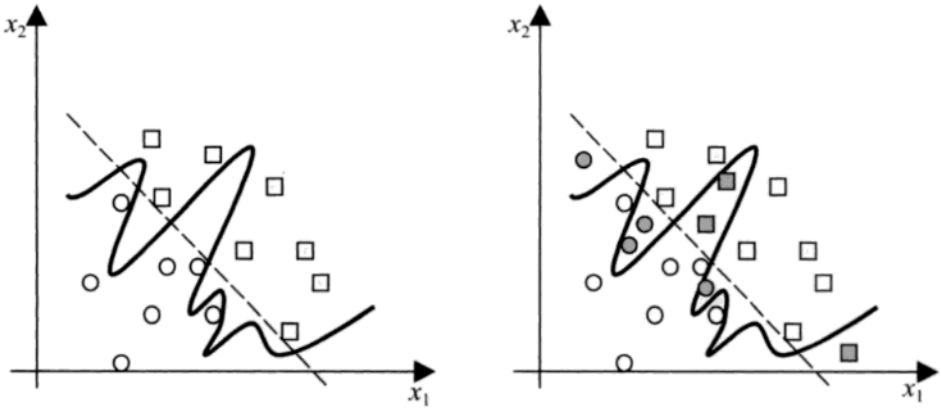


Figure 2.1: Overfitting illustrated for binary classification: in the left figure, we observe that the complex model (solid wiggly curve) classifies perfectly the training examples (empty circles and squares). However, on the right figure, we see that this same model totally fails to predict the classes of testing examples (filled circles and squares). While this complex model clearly overfits, the model corresponding to the dashed line is much more simple and generalize well to testing examples. Source: [16]

learn this pattern and generalize better. High generalization is a crucial criteria for an ML model and is obtained when a good balance between simplicity and complexity is found. Tools such as regularization (section 2.2.2) are introduced to manage the finding of this balance. Generalization and overfitting are illustrated on an example on figure 2.1.

2.2.2 Support Vector Machines

"*Support Vector Machines*" (SVM) is probably the most standard framework for supervised learning. If we were to project our data (a set of point in a certain space) on a 2D-plane, we can see SVM as a line (for binary classification) that separates our training data in their respective classes with the largest margin (i.e the distance between the set of data points from each class and the line is the largest possible). Then, to classify a new data point, we project it on the 2D-plane and we label it with the class corresponding to the half plane where it is located. We can generalize this idea to more dimensions, in which case the boundary line becomes an hyperplane. Compared to other linear classification methods, it can lead to a better generalization. To maximize the margin, we establish the finding that we only need to maximize the distance from the boundary to the nearest data points. Those points are called the *support vectors* (see figure 2.2).

Formulation

Given that $(\mathbf{x}_i, y_i)$ is our set of (data points, target) pairs. The separator is described by the following set of points $\{\mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = 0\}$ (mathematical notations are illustrated on figure 2.3). If class 1 and class 2 corresponds to 1 and -1 respectively, the constraints for the separator are the following:

$$\mathbf{w}^\top \mathbf{x}_i + \boldsymbol{b} \geq 1, \forall \mathbf{x}_i \text{ with } y_i = 1 \text{ and } \mathbf{w}^\top \mathbf{x}_i + \boldsymbol{b} \leq -1, \forall \mathbf{x}_i \text{ with } y_i = -1 \tag{2.1}$$

And can be reduced to:

$$y_i \left(\mathbf{w}^\top \mathbf{x}_i + \boldsymbol{b} \right) \geq 1, \forall \mathbf{x}_i \tag{2.2}$$

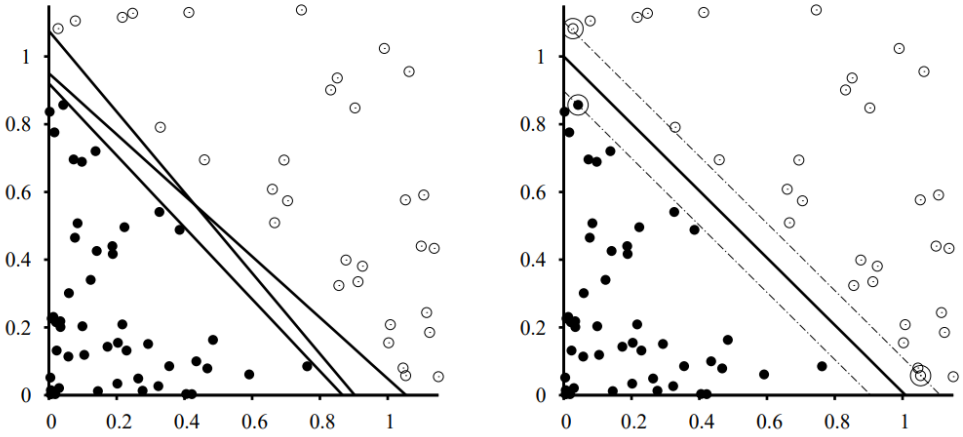


Figure 2.2: Maximal margin separator: we observe that among all the proposed separators on (a), the best one according to SVM is the one maximising the margin size as shown on (b). Support vectors are circled. Source: [15]

The goal is thus to find the weight vector $\mathbf{w}$ such that it maximizes the margin distance to support vectors and such that data points are in the correct half-plane (see figure 2.3):

$$\begin{aligned} &\max_{\mathbf{w},\boldsymbol{b}} \frac{2}{\|\mathbf{w}\|} \\ &\text{s.t. } y_i \left(\mathbf{w}^\top \mathbf{x}_i + \boldsymbol{b}\right) \geq 1, \forall \mathbf{x}_i \end{aligned} \tag{2.3}$$

NB: the 2 at the numerator is the scalar width of the margin (data can be scaled as such without loss of generality).

Which can be reformulated as a minimization problem:

$$\begin{aligned} &\min_{\mathbf{w},\boldsymbol{b}} \|\mathbf{w}\|^2 \\ &\text{s.t. } y_i \left(\mathbf{w}^\top \mathbf{x}_i + \boldsymbol{b}\right) \geq 1, \forall \mathbf{x}_i \end{aligned} \tag{2.4}$$

We can obtain the dual of the original (primal) problem, which highlights a scalar product:

$$\arg\max_{\boldsymbol{\alpha}} \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k \left(\mathbf{x}_j \cdot \mathbf{x}_k\right) \tag{2.5}$$

Subject to:

1. $\alpha_j \geq 0$
2. $\sum_j \alpha_j y_j = 0$

for all j This is a quadratic programming optimization problem, easily solvable by current technologies in optimization [17]. Once $\boldsymbol{\alpha}$ found, the weight vector is: $\mathbf{w} = \sum_j \alpha_j \mathbf{x}_j$

A drawback is that the problem becomes infeasible for data non-linearly separable. To address this issue, SVM relaxes the constraints 2.2.2 and introduces a regularization term (see section 2.2.2).

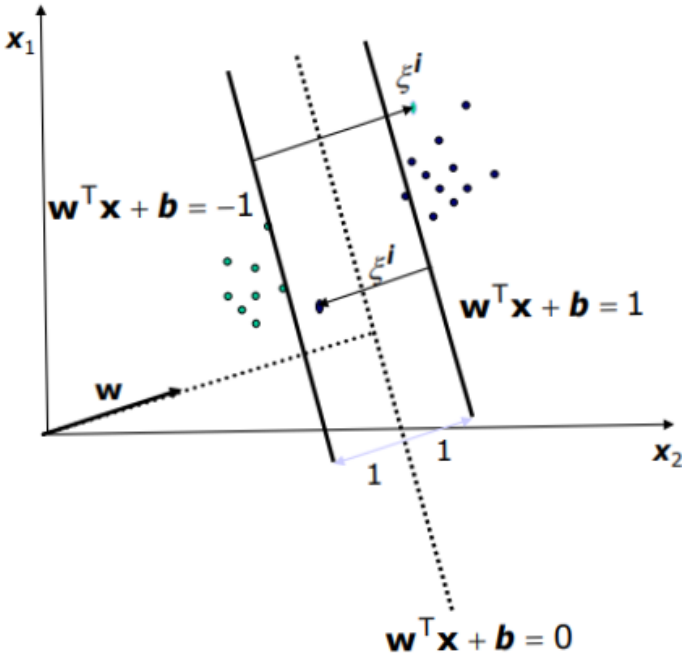


Figure 2.3: SVM illustration with maximal margin separator and mathematical notation. Source: [17]

However, linear models are often limited for classifying data that present non-linearity. To face this problem, SVM changes the representation of data: it maps the data, initially in the input space $\mathbf{x} \in \mathbb{R}^m$, into a higher-dimensional space, called the feature space $F(\mathbf{x}) \in \mathbb{R}^M$: $M > m$ where it is often linearly separable (see figure 2.4 for illustration). The optimization problem is almost the same except that the equation 2.5 becomes:

$$\arg\max_{\boldsymbol{\alpha}} \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k \left(F(\mathbf{x}_j) \cdot F(\mathbf{x}_k)\right) \tag{2.6}$$

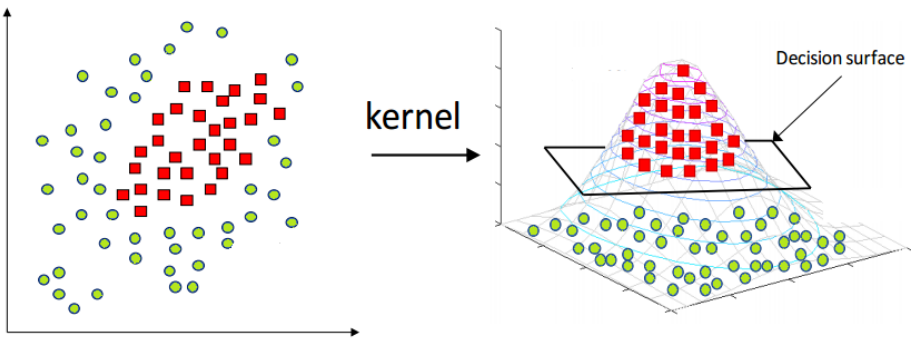


Figure 2.4: Changing of space can enhance linear separation: on the left, we have data non-linearly separable in $\mathbb{R}^2$. On the right, we map this data in $\mathbb{R}^3$ where it is linearly separable by an hyperplane. Source: [18]

Kernel trick

Knowing that feature space is often very high dimensional, calculating explicitly the scalar product $F(\mathbf{x}_j) \cdot F(\mathbf{x}_k)$ can be computationally intensive. However, it turns out that we can compute this scalar product while staying in the input space and thus reduce a lot the

computations. This is done with the help of a kernel function: $k(\mathbf{x_i}, \mathbf{x_j})$[2] that is much more simple. Indeed, with a bit of algebra, we have: $k(\mathbf{x_i}, \mathbf{x_j}) = F(\mathbf{x}_i) \cdot F(\mathbf{x}_j)$. This is known as the *kernel trick*. This allows SVM to effectively operate and find a linear separator in high-dimensional feature spaces without the need for explicit computations in that space [18]. The linear separator computed in feature space will be non-linear in the input space (except for the linear kernel). For more information about the construction of a kernel or about SVM in general refers to [16].

Kernels

There are a bunch of different kernels used for SVMs. For example, the polynomial kernel: $k\left(\mathbf{x_i}, \mathbf{x_j}\right) = \left(\mathbf{x_i} \cdot \mathbf{x_j} + 1\right)^d$ (where d is the degree of the polynomial), the Radial Basis Function (RBF) Kernel: $k\left(\mathbf{x_i}, \mathbf{x_j}\right) = \exp\left(-\gamma \left\|\mathbf{x_i} - \mathbf{x_j}\right\|^2\right)$, the sigmoid kernel: $k(x, y) = \tanh\left(\alpha x^T y + c\right)$ or the linear kernel: $k(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i} \cdot \mathbf{x_j}$ [19]. The choice of a kernel is discussed in section 3.3.

Regularization

A term can be added in the objective function 2.4 to penalize too complex models. For SVM, this term is the following: $C \sum_i \xi_i$, it relaxes the constraint of strict separation of each class as they are usually not linearly separable, even in the feature space. This term is called the *regularization term*. ξ_i is equal to the distance from the data point $\mathbf{x}_i$ to its correct margin boundary if it is misclassified and 0 otherwise (see figure 2.3). The C parameter allows us to decide the importance we want to address to the misclassified examples: for high C values, the margin will be smaller, trying to correctly classify as most training examples as possible, potentially leading to an overfitting; whereas low C values will lead to a wider margin with probably more misclassifications in the training examples but possibly a better generalization, although it is more susceptible to underfit [20] (see figure 2.5 and 2.6).

2.2.3 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a supervised learning algorithm that separates the two classes with a linear combination of the features. The main idea is to create a new axis with the help of other features such that it maximizes the separation between the projected data belonging to different classes. To build this axis, two criteria are used (illustrated on figure 2.7):

1. Maximize the distance between means of the two classes.
2. Minimize variance within each class.

This results in an optimization problem. We have to find the direction $\mathbf{v}$ such that it maximizes the ratio of distance between means of the two classes (named C1 and C2 here) on variance within each class:

$$\max_{\mathbf{v}} J(\mathbf{v}) = \frac{\widetilde{\mu_1} - \widetilde{\mu_2}}{\widetilde{s_1^2} + \widetilde{s_2^2}} \tag{2.7}$$

Where $\widetilde{\mu_1} = \frac{1}{n_1} \sum_{x_i \in c_1}^{n_1} \mathbf{v}^T x_i = \mathbf{v}^T \mu_1$ and $\widetilde{\mu_2} = \mathbf{v}^T \mu_2$ are respectively the means of data points projected on the axis of class C1 and class C2. $\widetilde{s_1^2} = \sum_{y_i \in c_1} (y_i - \mu_1)^2$ and $\widetilde{s_2^2} =$

[2]To be a valid kernel function, $k(\mathbf{x_i}, \mathbf{x_j})$ must be semi-definite positive and symmetric

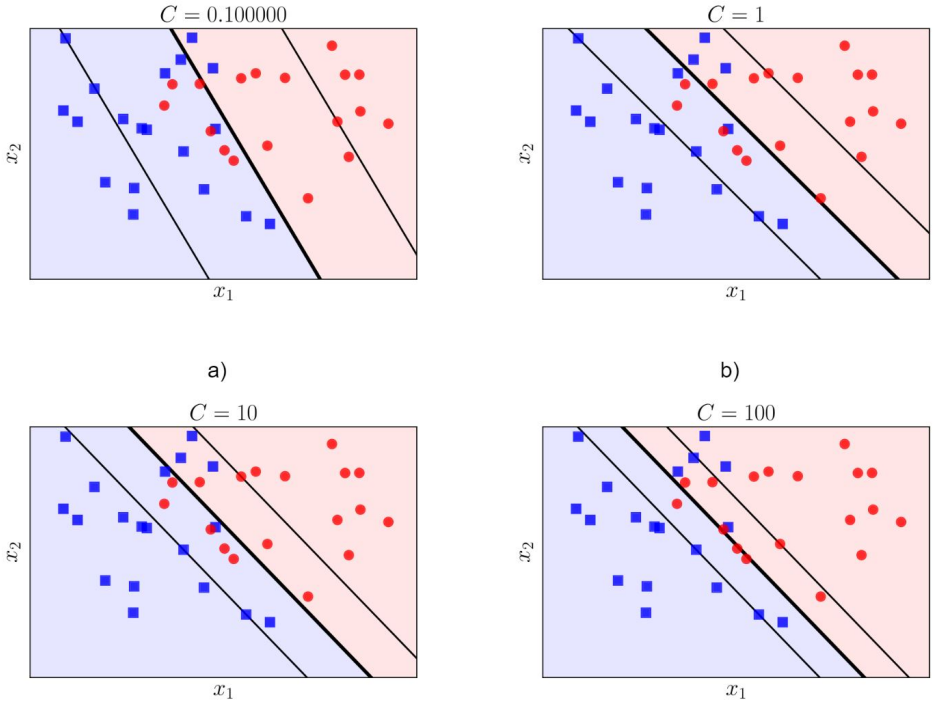


Figure 2.5: Different C values with their corresponding margins. Source: [20]

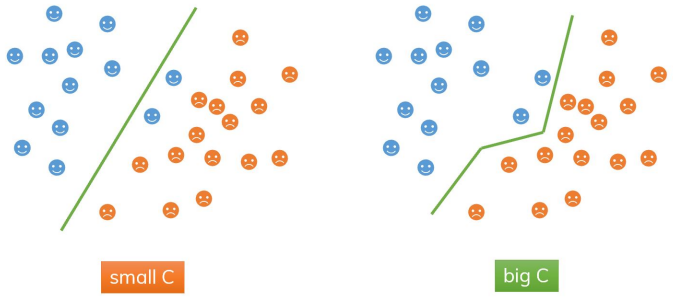


Figure 2.6: Small C VS Big C. Source: [20]

$\sum_{y_i \in c_1} (y_i - \mu_2)^2$ are respectively the variance of samples from class C1 and C2. The optimization problem is solved thanks to the following equation (we nullify the derivative):

$$\frac{dJ(\mathbf{v})}{d\mathbf{v}} = s_b\mathbf{v} - \frac{\mathbf{v}^t s_b \mathbf{v}\,(s_w\mathbf{v})}{\mathbf{v}^T s_w \mathbf{v}} = s_b\mathbf{v} - \lambda s_w \mathbf{v} = 0 \tag{2.8}$$

Where $s_b = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$ and $s_w = s_1 + s_2$ are respectively the scatter within classes and the scatter between classes. After solving the equations, the direction $\mathbf{v}$ is obtained. Details on [21] or [22].

2.2.4 Evaluation

A recurrent question in machine learning is how to evaluate the performances of our model? One could answer that we only have to train the model on our available data, then predict those data with our trained model and use those predictions to compute a metric like accuracy $acc = \frac{n_{right}}{n_{total}}$ (n_{right} is the number of data well-predicted, n_{total} is the total number of data). There is one pitfall with this approach: we have no clue about how the model

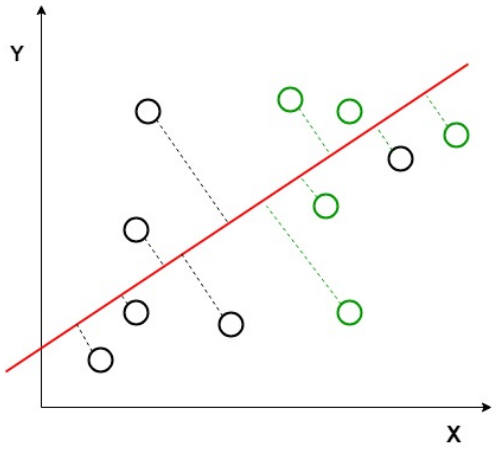


Figure 2.7: LDA example: the red axis is the one fulfilling the 2 criteria mentioned with the present data. Source: [21]

would perform with very new data and we could overfit. Predicting labels of data that have been used for the training itself is excessively simple for the classifier and this method has the tendency to overestimate the real performances of the model. One solution is to separate the dataset in two independent parts; one for the training and one for the testing. Their size can be for example 75% and 25% of the dataset size respectively. Accordingly, we don't have an overlap between the training and testing data anymore. The problem here is that we could be lucky and get a trained model that performs surprisingly well with the testing data. Moreover, when tuning an hyperparameter of the model (see section 2.2.6) based on performances on the test set, information of the test set could leak to the model and the tuned model could struggle to generalize. There are several solutions to prevent from this situation, one of them is called "cross-validation".

KFold cross-validation

Cross-validation is a crucial method to obtain a more reliable estimation of the metrics values. It helps to mitigate the risk of overfitting. First, it separates the data in a training set and a test set. The training set is further divided into folds (i.e subsets) that will be used for training and testing. Here is the KFold cross-validation algorithm (see figure 2.8 for illustration):

1. Split the data into training data and test data
2. Divide the training data in k equivalent folds
3. Train the classifier on $k-1$ folds
4. Predict the remaining fold labels with the trained classifier and get the metrics values for this subset
5. Repeat every steps until all folds have been used as test sets.
6. Average the k metrics values computed to obtain the estimation of the true metrics values

Cross-validation allows maximum utilization of the available data. Since each data point serves as part of the training and test sets, it helps in evaluating the model's performance even with limited data. The remaining test data can be used to obtain a final evaluation of our model, once all hyperparameters are set [23].

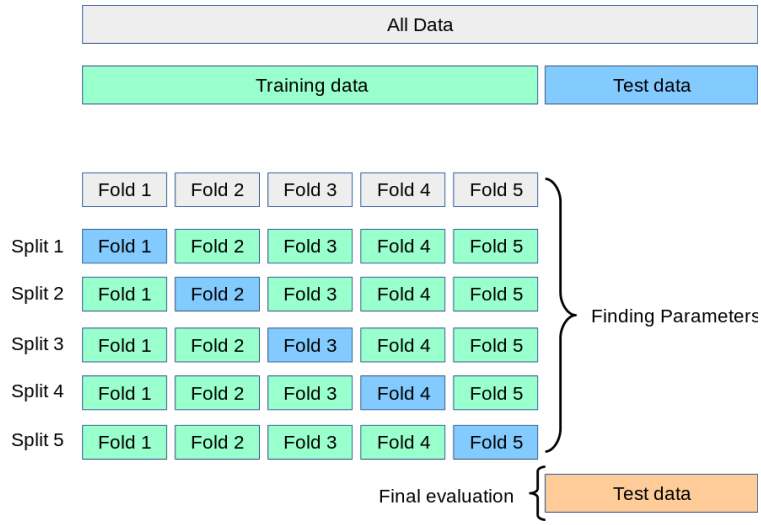


Figure 2.8: KFold cross-validation with $k = 5$. Source: [23]

Metrics

All the metrics used are based on these four values defined for a binary classification with two classes: positives and negatives.

1. True Positives (TP)
2. True Negatives (TN)
3. False Positives (FP)
4. False Negatives (FN)

Their meanings are explicit in the table 2.1 which corresponds to the confusion matrix. Index (i, j) of this matrix is the number of samples from class j classified as class i by the model with $(i, j) \in \{0, 1\}$ here.

	True label positive	True label negative
Predicted positive	TP	FP
Predicted negative	FN	TN

Table 2.1: Confusion matrix

Accuracy

Accuracy is probably the most known metric. Here is the formula:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.9}$$

It is the number of well-classified samples divided by the total number of samples predicted.

Sensitivity

Sensitivity also known as True Positive Rate (TPR) :

$$TPR = \frac{TP}{TP + FN} \tag{2.10}$$

Is the fraction of well-classified positives samples among all positive samples predicted.

Specificity

Specificity also known as True Negative Rate (TNR) :

$$TNR = \frac{TN}{TN + FP} \tag{2.11}$$

Is the fraction of well-classified negatives samples among all negatives predicted.

False Positive Rate

The False Positive Rate (FPR) :

$$FPR = \frac{FP}{TN + FP} = 1 - specificity \tag{2.12}$$

Is the fraction of negative samples classified as positive among all negatives samples predicted. It is also equals to $1 - specificity$.

Area Under the Curve of Receiver Operating Characteristic

Area Under the Curve (AUC) of *Receiver Operating Characteristic* (ROC) is a commonly used evaluation metric in machine learning and statistics to assess the performance of binary classification models. It measures the ability of a model to distinguish between positive and negative classes. First, to obtain the ROC curve, we plot the true positive rate (sensitivity) against the false positive rate (1-specificity) for various *classification thresholds*: the probability value above which the classifier's probability output is considered as 1. AUC is the area below this curve. A straight line from $(0, 0)$ to $(1, 1)$ is a ROC curve corresponding to an aleatory classifier because it means that for every threshold, the true positive rate is equal to the false positive rate. The AUC corresponding to this case is 0.5 and is considered as the chance level. Figure 2.10c corresponds to this situation where the distribution of decoder's output probability of both classes are blended. An ideal classifier has a ROC curve that is a right angle in the top-left corner of the plot, it corresponds to an AUC of 1. The distributions are clearly separated (see figure 2.10a). AUC between 0.5 and 1 are the most common situations, where both distributions encroach each other (figure 2.10b). An AUC of 0 indicates that the model's predictions are completely reversed in terms of class labels. This means the model is treating positive classes as negative and negative classes as positive in its predictions (figure 2.10d). It should not happen in practice.

Even though AUC is usually used as a metric in response to class imbalance problem, which is not the case in the present study, this metric measures other characteristics of the model that are not reflected by accuracy. Firstly, the management of true positives and false positives. While accuracy reflects only the ratio of correctly classified examples, AUC considers also the TPR and FPR. Accordingly, if a model predicts better one class than

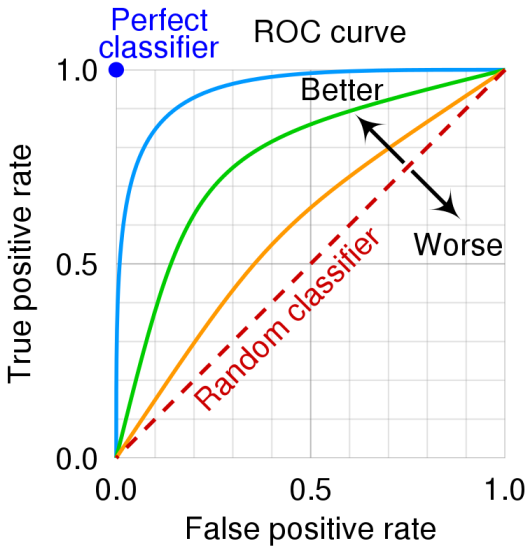


Figure 2.9: Different ROC curves. AUC is the area below the curve. Source: [24]

an another, it will negatively impact AUC. Further, AUC gives insights about the classifier performances across a whole range of different thresholds contrary to accuracy. Overall, it is interesting to complement accuracy measurements by an other metric in order to re-check results or to analyze them when differences are present.

2.2.5 Feature reduction

In machine learning, *feature reduction* is the process of smartly reducing the number of features while preserving as much relevant information as possible from the original dataset and to preserve—if not improve—performances. Feature reduction is a very frequent, if not crucial, step in a machine learning pipeline. It answers to several needs:

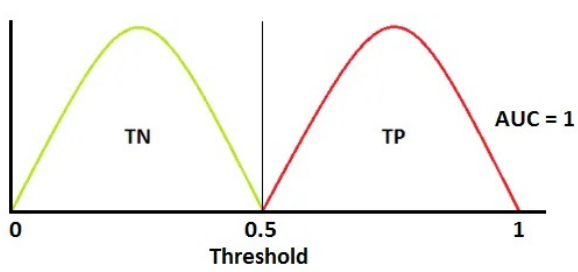
- Dimensionality reduction: a challenge in machine learning is to deal with high-dimensional data. Indeed, the data needed to fill a space grows exponentially with its dimension. This is sometimes referred as the *curse of dimensionality*. Models have generally better results when they use data of lower dimensions.
- Dataset visualization: some feature reduction methods can decrease the number of features down to two or three, allowing to visualize high-dimensional data.
- It can improve classification scores when wisely applied.

There are two main approaches for reducing the number of features: feature selection and feature extraction.

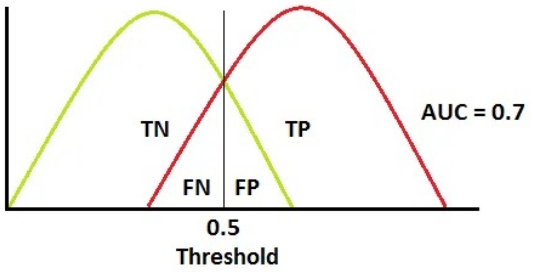
Feature selection

Feature selection aims to reduce the number of features by keeping a subset of important and relevant features from the original set. This importance and relevance are usually measured by two types of methods:

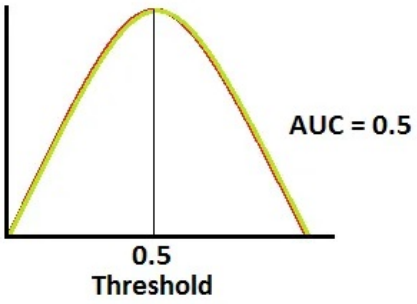
1. Wrapper methods: those methods are based on the interaction with a classifier i.e. different subsets of features are tested and the most promising are selected;



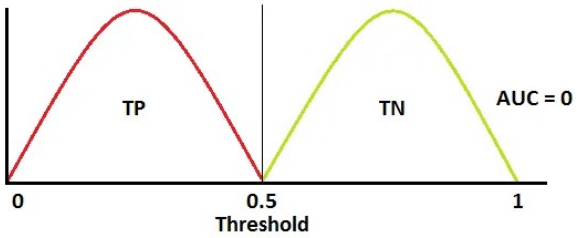
(a) AUC = 1



(b) AUC = 0.7



(c) AUC = 0.5



(d) AUC = 0

Figure 2.10: Different AUC values. The green and red curves are the empirical probability density function of the classifier's output probability for both classes. Samples on the left of the threshold (black vertical line) are considered as positive while those at the right are negative. Source: [25]

2. Filter method: compute univariate metrics giving information about the link between the feature and the target; those measured values are then used to select features with highest metrics scores, reflecting their importance to the target. Examples of such metrics are Mutual Information and F-test statistic. Contrary to wrapper methods, filter ones are model-independent.

Feature extraction

Feature extraction aims to reduce the dimensionality by crafting new features from original ones with the help of mathematical or statistical techniques. Those new features are built such that they express most of the information contained in the dataset. LDA can be used as such. The most famous and used technique is principal component analysis, which is detailed below.

Principal Component Analysis

Given a dataset where each data has n variables, *Principal Component Analysis* (PCA) reduces the dimension of the dataset from n to m where $m < n$ by crafting new uncorrelated variables derived from linear combinations of the previous ones called *Principal Components* (PCs). PCA compute those PCs such that they are orthogonal and they keep a maximum amount of variance from the original features. Those PCs are the covariance matrix's eigenvectors because they are the directions of the axes where there is the most variance. The data is then projected on the PCs to obtain the new dataset. PCA is mainly used for dimensionality reduction: it allows to project data on the first few PCs (i.e. those who explained the most variance). PCA with two or three PCs is frequently applied in order to visualize high-dimensional data [26].

Kernel PCA

Kernel PCA (KPCA) is an extension to PCA where the data points $\mathbf{x_i}$ are firstly "mapped" to a higher dimensional space. The objective is similar as in SVM: increase separability. PCs are then computed in this new space thanks to the kernel trick.

2.2.6 Model selection

Hyperparameters are model-specific parameters, e.g. kernel or C parameter for SVM. Model selection refers to the tuning of those hyperparameters. Tuning them based on cross-validation outcome would lead to overfit as we don't have any guarantee that it will perform as good on an other dataset. Therefore, we need a totally independent set which is the test set (see figure 2.8). This set should be set apart from the whole dataset before any computation. The test set offers a final and fair evaluation about how well the model would perform with a totally new and plausible dataset. Nevertheless, keeping a test set apart means that data is lost for the training. Therefore, it is important to balance correctly the training set and test sizes [23].

2.3 Machine learning and EEG

Until approximately the 2000's, the most popular methods to analyze neuroimaging data were univariate (i.e. taking into account only one electrode for EEG or one voxel for functional Magnetic Resonance Imagery (fMRI)). Before the prevalence of machine learning, analyzing techniques for neuroimaging data were longer, more tedious and less accurate process. Indeed, the main methods to excerpt information from EEG signals were statistical tests on Event-Related-Potential components' obtained from a univariate channel to assess a significant difference between conditions (e.g. [27]), or even a simple visual expert's analysis. In a general manner, they look for significant differences between conditions on a single site. Notwithstanding univariate methods are still used in specific domains (e.g. [28]), numerous analyses are now complemented, if not exclusively conducted, through *Multivariate Pattern Analysis* (MVPA): each time point has multiple values excerpted from the electrodes; in opposition with univariate analysis where each sample is associated to a unique value.

In [29] it is stated that using Multivoxel (now, Multivariate) Pattern Analysis with fMRI enhanced the detection of cognitive states and that different semantic categories were associated in a reliable, distinct pattern of activity in the Ventral Temporal (VT) cortex. Previously, MVPA was widely employed in fMRI research, but its application to decoding techniques in M/EEG analysis has predominantly been confined to the realm of brain-computer interfaces [12]. In the 2010's, early researches started to demonstrate the potential of machine learning in M/EEG (e.g. [5]) and studies suggested that EEG decoding would benefit from a multivariate analysis [30], [31], [32]. For instance, [32] tried to characterize the timing at which brain EEG information occur in the brain in response to human face presence or absence in a natural environment. They first focused their analyses on a classical ERP univariate analysis on the P1 and N170 components[3] (classically associated

[3]P1 and N170 are both ERPs. The P1 (or P100) is the first positive-going component (when using a mastoid reference point) and its peak is normally observed in around 100 ms. It is related to visual stimuli. N170 is a face-sensitive ERP component characterized by a negative deflection in wave amplitude that occurs around 170 ms after the presentation of a face [33], [34].

with face processing) but found only modest modulations thereof. Whereas a multivariate analysis led to the more affirmative conclusion that face detection readout could come very early in the brain (approximately 100ms after the brain stimulus onset). Thus, combination of MVPA and EEG allows for a focus on the complex temporal dynamics of brain processing.

Machine learning (in our case, supervised learning) came with a significant automatization. It is able to deal with huge amount of multivariate labelled data (labelled data means data points with corresponding output) to learn complex interactions and dependencies in features that would be invisible with the naked eye or even with classical analysis. It can subsequently recognize their influence/importance on the target value. Then, it can use those features and in-between interactions to predict as accurately as possible a new given example's label by creating a model. Additionally, unlike univariate analysis, MVPA allows to take into account activities spanning different cortical areas (despite the low spatial resolution of EEG). Once the model is learned, it can predict labels from new data in the blink of an eye. Furthermore, nowadays the collection and digitalization of EEG data is wisely feasible and accessible. However, a drawback of EEG is the scarceness of data. As the acquisition process is long and tedious, the number of data at our disposal is often limited compared to what would be necessary to "fill" the feature space.

Classification tools can be used not only for predictions purposes in medical diagnosis but also it can help in gaining neuroscientific knowledge when used in context. For instance, looking at the performance of decoding metric between two conditions across time gives us insights about temporal evolution of decoding and allows us to understand when brain patterns manifest to discriminate categories.

In conclusion, due to its high-dimensional learning, and ease of dealing with numerous data, machine learning is now a major tool for brain imagery decoding and has made his marks in this domain.

Typical EEG signals processing

Typical steps for processing EEG signals in machine learning are addressed on figure 2.11. The grey part is the preprocessing, present in every EEG signals analysis, it is detailed in section 3.1. Light orange represents the steps for standard univariate analysis with the help of ERPs, machine learning is not required. Dark orange and pink are respectively the typical pathways for single-trial analysis and BCI.

A classification step is mandatory and requires machine learning to efficiently do it. In the single-trial analysis, averaging can be done to increase SNR but is not mandatory. The statistical analysis allows to assess the statistical significance of the difference between a measured metric against chance level (or against measured metric with other techniques).

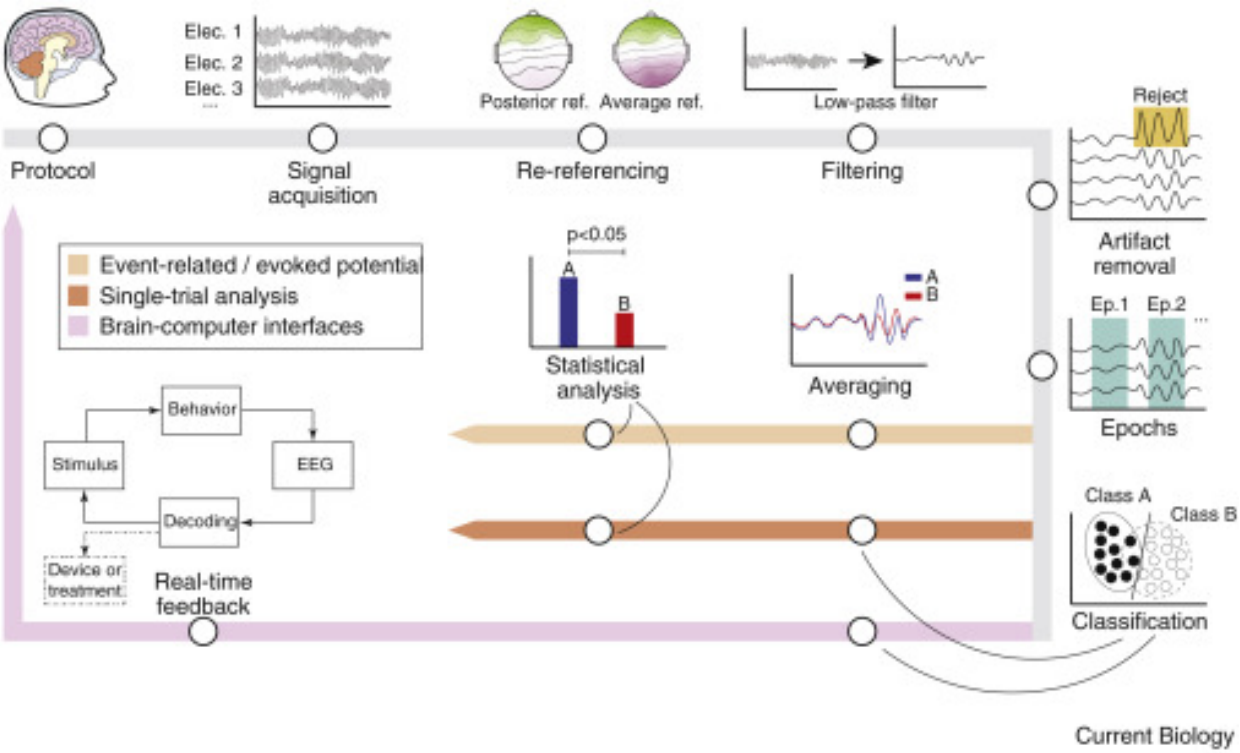


Figure 2.11: Typical EEG signals processing. Source: [8]

Chapter 3

Pipeline

This chapter presents the pipeline followed during this work, from the raw EEG signals to the final prediction scores with plots and statistical assessment.

There are many usual steps that are typical in EEG signals processing. The overall pipeline is presented on figure 3.1. It is divided in six steps:

1. **Data acquisition**: we had 15 subjects that listened to 24 sounds divided in 4 categories: animals, humans, environments and objects. Each individual sound is listened 45 times (45 trials) and EEG activity was recorded across 128 electrodes (section 4.1.1 for more details);

2. **Preprocessing** (section 3.1): all the preprocessing steps necessary to obtain a clean and well-dimension dataset;

3. **Processing** (section 3.2): techniques to modify the characteristics (dimensionality, size, quality) of the dataset fed to the decoder;

4. **Decoding** (section 3.3): choice of the binary (or multiclass, section 3.3.3) classsifier and its hyperparameters;

5. **Evaluation** (section 3.4): application of a cross-validation strategy and evaluation of the performance with metrics.

6. **Statistical assessment** (section 3.5): Assess the significance of the results, statistical comparison of the methods used.

Specific details about each technique tried and used are explained in depth in the sections below.

3.1 Preprocessing

Each step presented here is schematically represented in the grey line beside one of the dots of figure 2.11.

1. **Band-pass filtering** filters out signal to keep only a certain range of frequency with the help of a low-pass filter and a high-pass filter (or a bandpass filter). It is shown to exclude potential non-biological noise from the signal, since the signal of interest lies between 0.1 Hz to 100 Hz;

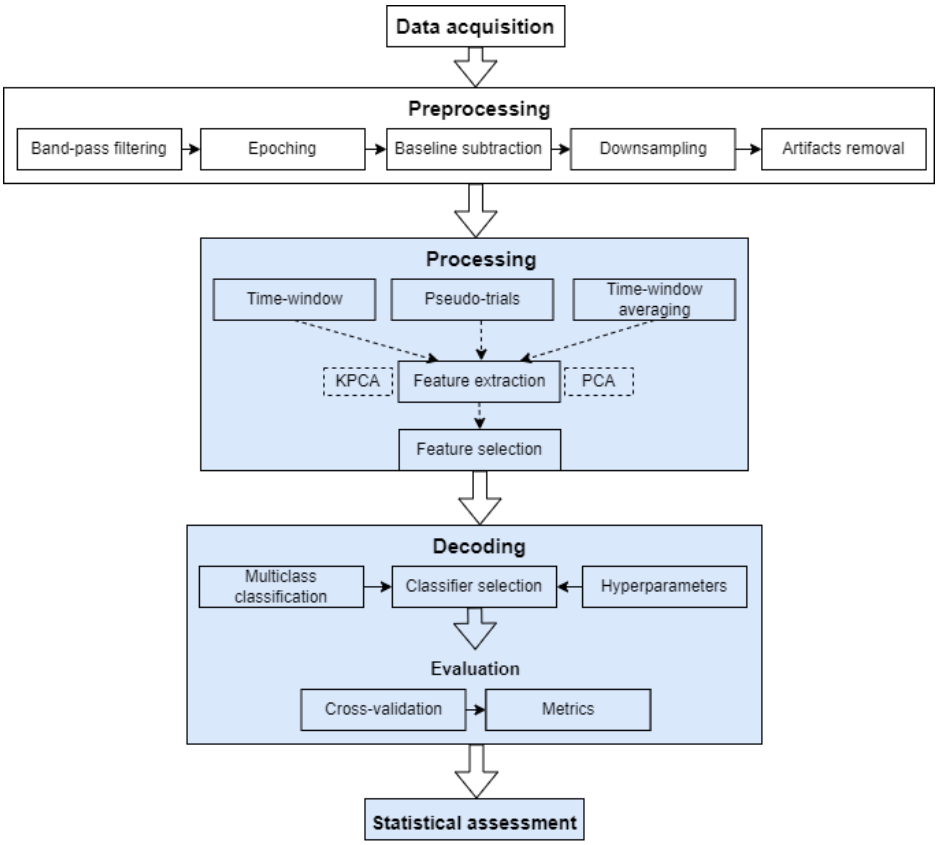


Figure 3.1: Pipeline followed. Blue frames are the steps that were studied in this work.

2. **Epoching** is the process of dividing the continuous EEG signals into multiple epochs. An epoch is generally time-locked to the onset of a stimulus with both pre-stimulus and post-stimulus activity;

3. **Baseline subtraction** to remove drifts due to DC components of the signal, where the epoch activity is subtracted by the mean pre-stimulus activity;

4. **Downsampling** is the process of reducing a dataset size while preserving its essential information. It is made by artificially decreasing the sampling frequency (e.g. to divide the sampling frequency by 2, we take only one sample out of two). The dilemma here is to reduce the time dimension to decrease computational time and increase SNR ([24]), while keeping as much as possible time points for faithful decodability;

5. **Artifacts removal** Signals are visually inspected to identify obvious artifacts due for example to muscular activity. Signals with such artifacts are deleted. Removing artifacts can also be performed automatically with a variety of algorithms: [3].

3.2 Processing

Few processing steps were added to try enhancing the performances, they are all performed just before the cross-validation:

1. **Time-window**: taking into account the temporal neighbours for each time point. See section 3.2.1;

2. **Time-window averaging**: averaging neighbouring time points together using a sliding window. See section 3.2.2;

3. **Pseudo-trials**: averaging trials together to increase SNR. See section 3.2.3.

4. **Feature reduction**: Feature extraction and selection to reduce dimensions. See section 2.2.5 for details on the method and 3.2.4 for EEG applications.

3.2.1 Time-window

In order to give more temporal information, it is possible to create a sliding window [7], [12]. This allows the classifier to have EEG informations about early past and near future data.

The main idea is to augment the number of features. For a time window of size n, each data point will have $n \times 128$ features instead of 128. Indeed for a time point i, its new set of features corresponds to the $n \times 128$ features retrieved from these time points: $\{i - \lfloor (n/2) \rfloor, ..., i, ..., i + \lfloor (n/2) \rfloor\}$. Leading to a new dataset of size $540 \times n \times 128$ (for one subject, one comparison and at one particular time point).

3.2.2 Time-window averaging

EEG recordings are very prone to noise at a single trial level that can come from a huge variety of non-brain activity sources: eye movement, muscle activity, environmental noise, ... One of the stake is to try reducing this noise and thus increasing the SNR. By averaging the features corresponding to a specific time point with features of time points around it, averaged features could contain temporal information while staying in a reasonable dimensional space. We used the same kind of time window as in section 3.2.1 except that here, the features within the time window are averaged to build the new features instead of just concatenated together.

3.2.3 Pseudo-trials

Each of the 24 exemplar sound was listened 45 times by each of the 15 subject, each listening is called a trial. Individually, those trials are very noisy and this noise can impact negatively the performances. It has been shown that averaging the time-locked trials enhance the SNR and the Event-Related potential (ERP) emerges [35]. However, a reasonable number of data is crucial for training and testing our model. Therefore, instead of using single-trial per individual sound (that would be the 45 trials averaged), we built several pseudo-trials. The adopted methodology to build n pseudo-trial among m trials is described down here:

1. Take a random sample of $\frac{m}{n}$ trials (without replacement)

2. Average them

3. Repeat the steps to build other pseudo-trials until there is no more trials (i.e. n times).

Following this procedure, we end with n pseudo-trials for each individual sound and we can use them to build the classifier instead of using the original trials. This indeed reduces the number of trials for training, however the classifier can benefit from higher SNR. If the number of single-trials per exemplar is scarce, the trials used to build pseudo-trials can be sampled with replacement (bootstrapping)[12].

3.2.4 Feature extraction

Feature extraction is an important step in the processing of EEG signals. The goal is to keep a maximum amount of embedded information in the crafted features while having features that can enhance the performance of the classification task. Indeed, multiple studies have empirically demonstrated that the presence of redundant or non-discriminatory features diminishes the capacity of classifiers to effectively learn decision boundaries that separate data belonging to distinct classes [36]. In a general way, reducing the number of features for learning is usually beneficial for machine learning models and can lead to lower training times. For EEG decoding, we often have lots of features compared to the number of data points (e.g. 570 data points and 128 features for this study), a dimensionality reduction could be beneficial.

NB: The preferred feature extraction methods in this work used mathematical techniques. Crafted features were linear (or non-linear) combinations of the original ones; accordingly, we had the drawback of losing the interpretability of the orignal features.

PCA

PCA can extract orthogonal components from the original data while keeping a maximum amount of variance. When choosing a number of components such that $n_{comp} < n_{feat}$, we can project our original data on the components and obtain a dataset of smaller dimension (i.e. less features) that represents quite well our original dataset. In [37] the EEG emotion recognition task performances were improved by the use of PCA with almost all the ML models tried. Best results were obtained with PCA and SVM. Moreover, « *PCA can improve signal similarity to improve accuracy for signal classification* » [38] and in [12] it is stated that: « *PCA can separate out noise and artifacts such as eye blinks into their own components* ».

KPCA

PCA can have difficulties with analysis of complex dataset [1]. Therefore, variations of PCA such has KPCA (see section 2.2.5) or sparse PCA have been proposed. « *KPCA applies a non-linear transformation to make the data points more separable in the transformed space* » [39].

3.3 Decoding

In this section, we describe and argue with literature background the different supervised ML algorithms considered in this EEG study with a discussion on their hyperparameters (when applicable). The strategies for multiclass classification are also presented.

3.3.1 SVM

In [1], they conducted a recent (2021) qualitative systematic review on ML and deep learning models for decoding EEG signals. Their pie chart about the classifier choice (figure 3.2) suggests that SVM is the most popular in each EEG domain of application and, asides with k-nearest neihbors (KNN), outperforms other supervised ML classifiers. SVM has the ability to handle high-dimensional data effectively. Moreover, the computational complexity

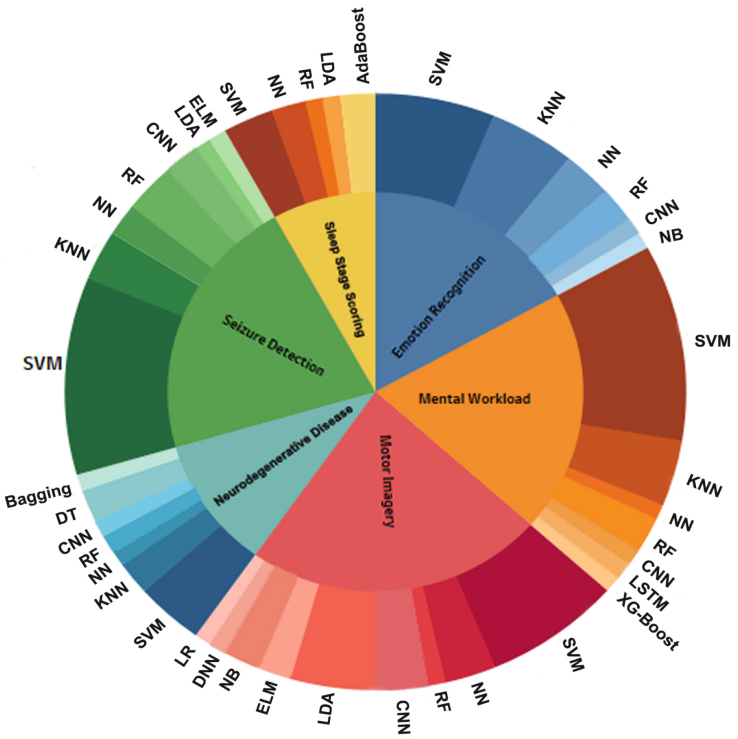


Figure 3.2: Classification algorithms used for each EEG domain. Inner circle represents the different EEG domains, outer one represents the utilization rate of the supervised machine learning and deep learning classification algorithms used for each task across all studies evaluated Source: [1]

of SVM is lower than other classification algorithms such as artificial neural networks and KNN [36].

Kernel

SVMs have a few different kernels: linear, polynomial, Gaussian Radial Basis Function (RBF), etc. [36]. The goal of classification in machine learning is usually to maximize accuracy. However, in neuroscience, there is a tendency to trade this high prediction accuracy with a more interpretable and more simple model but sometimes with lower predictive power [12].

Linear kernels have a rigid separator which limits the risk of overfitting. Moreover, we can easily retrieve the weights assigned to each feature and have an idea about their importance in the prediction (interpretability criterion). Nevertheless, having rigid boundaries is not a sufficient criteria to affirm that we are safe from overfitting. Indeed, an hyperplane in N dimensions has N degrees of freedom (e.g. it can be described by the intersections with each axis) meaning that we have a huge number of possible placements for this hyperplane. Especially in EEG where the hypothesis space is high-dimensional (equal to the number of features, i.e. 128 in our case). That's why a feature reduction is somehow important. Other non-linear kernels like RBF allow for more complex boundaries (see figure 3.3) and can be relevant for data that are not linearly separable or that present non-euclidean geometries but are less interpretable and also more susceptible to overfit training data due to their high flexibility. Both methods have their pros and cons.

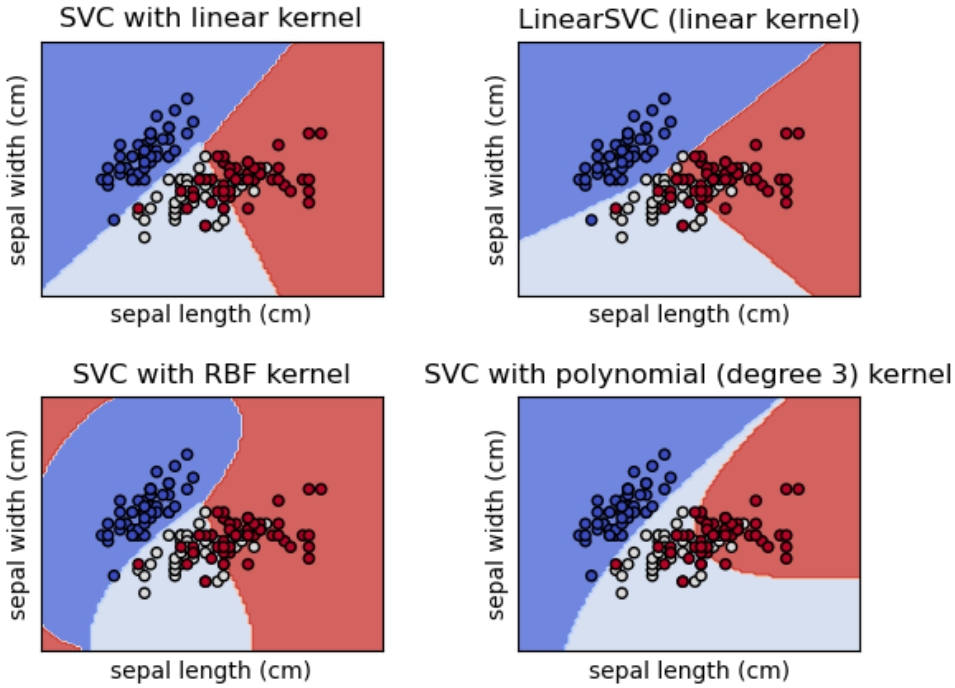


Figure 3.3: Different kernels illustrated. The classification problem here aims to predict the type of iris from its sepal length and width. We can see that linear kernel has rigid boundaries whereas RBF and polynomial kernels have non-linear ones. Source: sklearn website [40]

.

Hyperparameters

The C parameter controls the trade-off between achieving a low training error and a low testing error. A small value of C will allow more misclassifications in the training data, while a large value of C will penalize misclassifications more heavily, resulting in a narrower margin hyperplane and potentially overfitting the training data.
The optimal value of C depends on the particular dataset and the specific learning problem at hand.

3.3.2 LDA

LDA is based on the assumption that the density for the data is normally distributed, with equal covariance for both classes. It is a versatile classification algorithm, simple to setup as there are almost no hyperparameters. It has a lower computational complexity than SVM [36] and has successfully been applied to classification tasks in BCI [1]. They are linear, which has a potential of good generalization for the classification but can also lead to poor results on non-linear EEG data.

3.3.3 Multiclass classification

In multiclass classification, the classifier takes as input the data corresponding to the four categories and learn a model that can predict category of a new data point between more than two categories. This could be interesting to observe how our classifier deals with data from the four categories and if it is able to discriminate them better or worst than with

individual binary classifications. Typical supervised ML algorithms are designed to only work with binary classification and do not support classification with more than two possible outputs. But there are ways to build artificial classifiers that can support a bigger set of possible outputs. *One vs. Rest* and *One vs. One* are both multiclass classification strategies actually made of multiple binary classifiers [41]. Also, multiclass classification is not often done in other researches. To cite a few, [42] and [43] applied multiclass classification on the publicly available epileptic benchmark EEG dataset. We wanted to give it a chance in our study case and compare results between the two most used techniques: OvO and OvR. Others techniques such as minimum output codes (MOC) and error correcting output codes (ECOC) (used in [42]) exist but were left away for this work.

One vs. Rest (OvR)

It builds one binary classifier per classes. Each classifier is fitted with its corresponding data class against all other classes. For instance, in our problem:

- Animals vs (Humans, Environments, Objects)
- Humans vs (Animals, Environments, Objects)
- Environments vs (Animals, Humans, Objects)
- Objects vs (Animals, Humans, Environments)

The four classifiers output a score proportional to the probability of the given example of belonging to its class. The example's predicted class will be labelled by the class of the classifier with the highest score (i.e. the most confident). Therefore, this method requires that the machine learning method outputs a score instead of a simple binary output. This is not inherent in SVM classifiers but it is possible to retrieve a probability.

One vs. One (OvO)

We have one binary classifier per unique comparison:

- Animals vs Humans
- Animals vs Environments
- Animals vs Objects
- Humans vs Environments
- Humans vs Objects
- Environments Vs Objects

To predict the class of a given data point, we predict the class with each classifier and we do a majority vote: the category that has been the more predicted among the six comparisons is assigned to the tested data point.

3.4 Evaluation

Estimating the performances of our models is not always easy. Indeed, there are many different metrics. They all have their advantages but having a high specific metric does not specially mean that the model generalizes well. It could overfit by learning too much the training data or by taking advantage of the class-imbalance to obtain some good metrics scores (it is not the case here as the classes are balanced). Importantly, we have very few data compared to the number of features, it makes generalization complex.

The main discussions about evaluation concern the cross-validation algorithm and the metrics used, they are addressed in this section.

3.4.1 Cross-Validation

Different cross-validation schemes were tried.

KFold

KFold is the backbone cross-validation algorithm and other cross-validation strategies are based on the latter. You can find details about it in the section 2.2.4.

Stratified KFold

Stratified KFold is a variation of KFold cross validation where folds are made by preserving the proportion of each class in the original dataset. In our case, it allows both classes (one category is one class) to be 50% represented in each fold. Once folds built, the algorithm stays the same as KFold.

Leave 1 trial per examplar Out

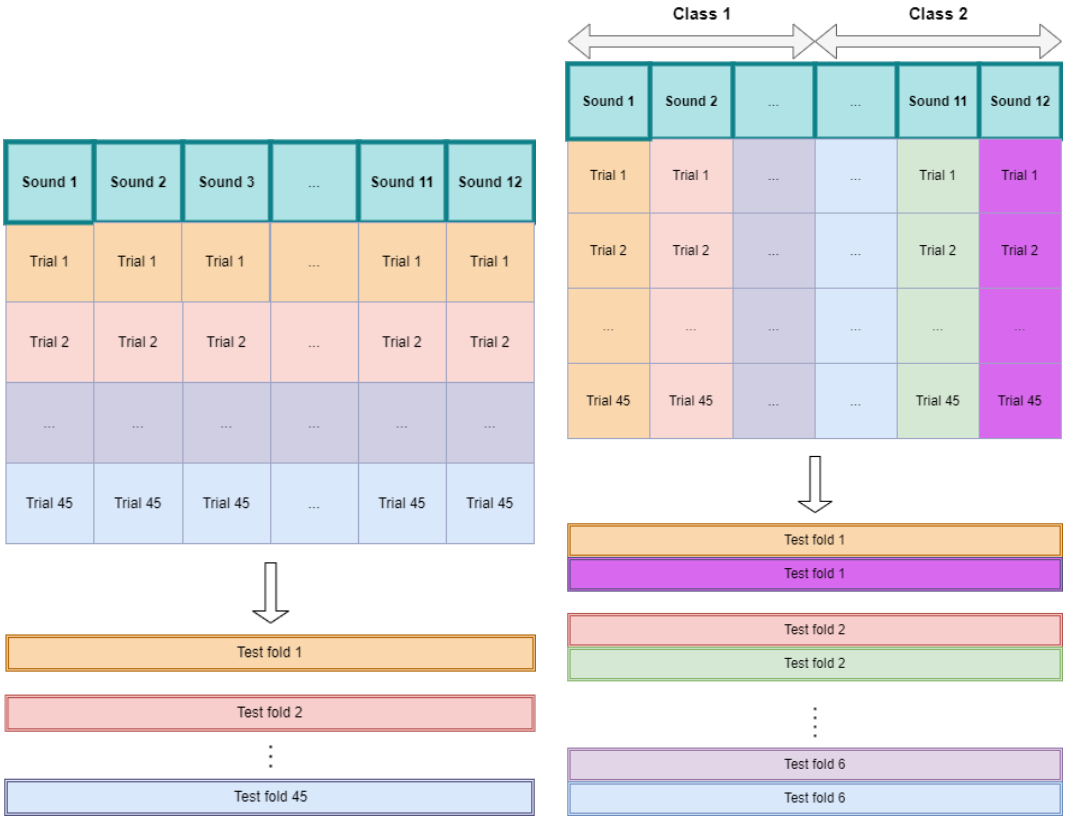
Leave 1 Trial per Examplar Out (L1TEO) is another variation of KFold. The idea is to keep one trial of each sound away for each test set of the cross-validation. In consequence we have, in our case, 45 splits (which corresponds to the number of trials per sound) and each test fold has 12 trials (6 from both categories). See figure 3.4a. Accordingly, the test folds are representative of the original dataset distribution i.e. one trial of each sound in the training set is present in the test set.

Leave Exemplar Out

In *Leave Exemplar Out* (LEO), there are 6 splits; one per two exemplar sounds. At each split, we leave out one exemplar per class (i.e. their 45 respective trials, which makes a total of 90 data points per split in a binary classification), for the testing of the model. This cross-validation strategy is relevant to evaluate the ability of the classifier to generalize its learning to yet unseen exemplars (new sounds) [12].

3.4.2 Metrics

AUC and accuracy were both used in our study. Accuracy is usually used as a unique performance measure [12], [44] or AUC [45] but we did not find any study that used both metrics together. We take this opportunity to bring a comparison between them.



(a) Leave 1 Trial per Exemplar Out (b) Leave examplar out

Figure 3.4: Cross-validation strategies: in 3.4b, to build one final test fold, we take all the trials for one particular sound from both classes.

3.5 Statistical assessment

To assess statistically the performances of our results, we have used the *sign permutation test*. A simple t-test or others tests were put away as they make assumptions about the distribution of the data (e.g. normal distribution for t-test) whereas we don't know it in this case. Sign permutation test makes less assumptions about it. The null hypothesis for one specific subject at one specific sample is that the average metric for that subject at that time sample is equal to chance level (i.e. 0.5 for binary classification). The procedure for one time point is as follow: we regroup the computed metrics of the n subjects and subtract them by the chance level. We sample a random number k between 1 and n. Then, we sample k random indexes idx in the range $1, ..., n$ and for each of these index, we flip the sign of the corresponding metric's value. After that, we average the modified n metrics to get a sample of the null distribution. We repeat those steps 10000 times to get the null distribution. Once the null distribution is obtained, the p-value corresponding to the current time sample is the proportion of samples in the null distribution that are greater than the Mean (mean - chance level) of the n original metrics at a particular time sample. Due to the fact that there are multiple comparisons, significance level α was adjusted with the False Discovery Rate (FDR) correction to obtain α_{corr}. Correction was only based on the number of samples post-stimulus. Samples with p-values lower than α_{corr} were considered statistically significant with significance level α.

Chapter 4

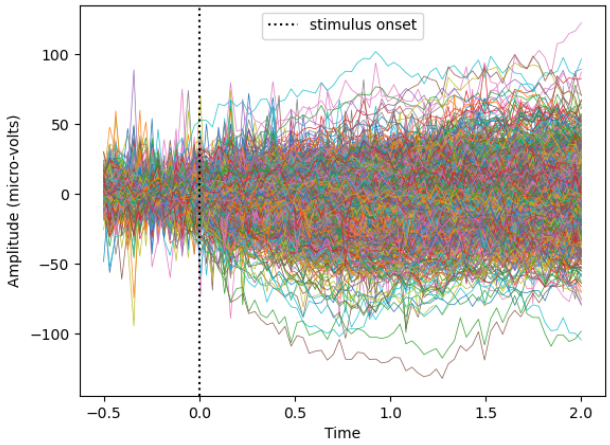
Results and analysis

The following chapter starts by introducing the setup of the experiment with the data collection procedure and the preprocessing applied on the data. Afterwards, we present the algorithm used to compute final metrics. Results are then exposed with plots of accuracy and AUC vs time and accompanied by a description and analysis for different scenarios.

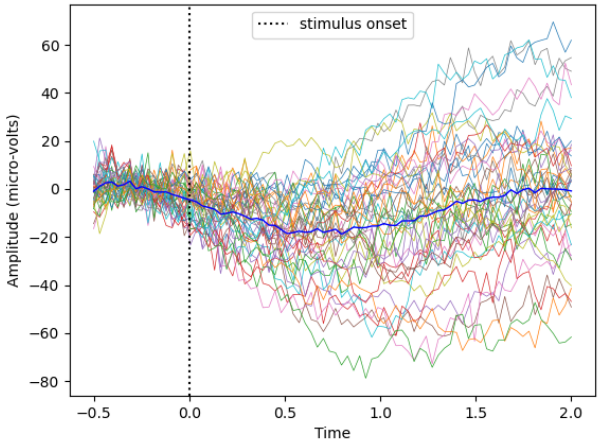
4.1 Data

4.1.1 Collection

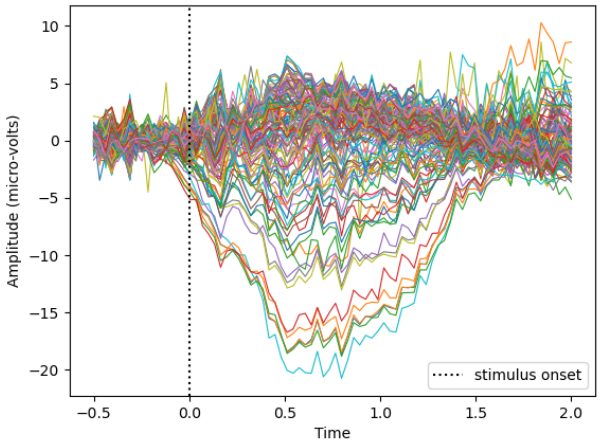
A study was conducted with 15 subjects who listened to sounds stimuli. EEG signals were recorded across 128 electrodes (or channels) meanwhile with a certain sampling frequency (i.e. the number of time points sampled per second by the EEG device. 1024 Hz in this case). The montage of electrodes is referential. We recorded the EEG responses of subjects to 24 different sounds. These sounds can be divided in four categories: animals, humans, environments and objects. Thus, in each category, we had six individual unique sounds (or exemplars). The stimuli are the inherent sounds that the entities make (e.g. barking for a dog). Each subject underwent a session of 60 minutes during which he listened to 1080 trials divided in four 15-minutes runs of 270 trials. Participants performed a task unrelated to the aims of the experiment to encourage them to maintain vigilance. The task was to press a button whenever two trials were the same in succession. Each sound lasted for $1s$ and there are $4 \times 6 = 24$ different exemplars. Therefore, we have $\frac{1080}{24} = 45$ stimuli per exemplar. For data visualization purpose, we plotted different representative values for one specific subject. Figure 4.1a plots all the trials for all electrodes for one specific sound across time. On figure 4.1b, you can see all the 45 trials of a specific exemplar sound corresponding to a particular electrode. The ERP of each channel has also been computed and is plotted on figure 4.1c, we can observe that ERPs react to the onset of the stimulus.



(a) All trials for all electrodes within one subject



(b) All trials corresponding to one sound within one subject and one electrode



(c) ERPs of the 128 electrodes for one sound listened by one subject

Figure 4.1: Data visualization

4.1.2 Preprocessing

1. **Band-pass filtering** The signals are band-pass filtered (0.1–200 Hz);
2. **Epoching** We sliced the data into epochs (i.e. trials) time-locked to the onset of the stimulus (-500 to 2000 ms);

3. **Baseline subtraction** was applied to remove drifts due to DC components of the signal;

4. **Downsampling** Signals were downsampled at a frequency of 64Hz. Meaning that we are left with $2.5 \times 64 = 160$ time samples. For the 15 subjects, we are left with a dataset of shape of $15 \times 24 \times 45 \times 160 \times 128$ (subjects $\times$ sounds $\times$ trials $\times$ time points $\times$ channels). Other downsamplings were tried, but this 64Hz allowed for a tradeoff between computational time and reasonable quantity of information.

5. **Artifacts removal** Trials were visually examined to remove the ones with obvious artifact(s). We finally had a minimum of 41 trials for each exemplar sound. Therefore, each sound constituted between 41 and 45 trials.

6. **Categories** This preprocessing step was added for this specific work. The exemplars can be divided in four categories. There are $\frac{24}{4} = 6$ exemplars per category. For the binary classification purpose, we will use four datasets, one for each category. For a specific category, we kept all the exemplars corresponding to this category with all its corresponding trials: the shape is $6 \times 45 \times 160 \times 128$ (exemplars $\times$ trials $\times$ time points $\times$ channels) for one subject. NB: as artifacts were removed, the number of trials varies between 41 and 45 but we wrote down 45 everywhere for the sake of simplicity.

4.2 Algorithm for metrics computation

This section describes the algorithm followed to acquire the final metrics at each time point.

The classifiers used are the ones implemented by the sklearn python's library [46] and supplemented with a probabilistic output method to compute AUC.

To observe the time influence on classification performance, we made a time by time classification: at each time point $t \in \{0, ..., 160\}$, we built one classifier and computed the metrics value with their corresponding dataset $d(t)$ that is the vector of shape 540×128 built from the concatenation of the two 270-vectors of category A and B with their 128 features. The target is a 540-vector made of 270 zeros followed by 270 ones. The metric value for one specific comparison corresponds to the average metric of the cross-validation fed with data corresponding to this comparison. To obtain the final metrics value for one subject and for one time point, we average metrics from each of the six comparisons at that time point and for this subject. Once all time points metrics computed, we get a time-varying measure of stimulus/condition decodability evaluated by the accuracy or the AUC ROC. Those steps are summarized in the pseudo-code 1. The method `get_dataset(t, p, s)` returns X and Y i.e. respectively concatenated data for both categories of the dataset pair p and corresponding target (270 zeros and 270 ones) for time t and subject s. The method `compute_metrics(X, Y, m, cv)` performs a cross-validation (see section 2.2.4) with X, Y, classification model m, cross-validation cv and returns the metrics' mean across each fold. The metrics computed by `compute_metrics(X, Y, m, cv)` are the accuracy and the AUC, details are hidden for the sake of simplicity. Eventually, `ground_accuracy` and `ground_auc` are vectors of 160 elements that contain the mean accuracy and AUC of all subjects respectively for each time points. This algorithm was adapted and modulated to be able to run different scenarios.

NB: the `mean` operator used here performs a mean along the first axis. E.g. `mean([[1, 2], [3, 4]]) = [2, 3]`

Algorithm 1 Metrics computation

```
acc_all_subj, auc_all_subj = list(), list()
for all subjects s do
  acc_subj, auc_subj = list(), list()
  for all dataset pair p do
    acc_comp, auc_comp = list(), list()
    for all time samples t do
      X, Y = get_dataset(t, p, s)
      acc, auc = compute_metrics(X, Y, m, cv)
      acc_comp.add(acc)
      auc_comp.add(auc)
    end for
    acc_subj.add(acc_comp)
    auc_subj.add(auc_comp)
  end for
  acc_all_subj.add(mean(acc_subj))
  auc_all_subj.add(mean(auc_subj))
end for
ground_accuracy = mean(acc_all_subj)
ground_auc = mean(auc_all_subj)
```

4.3 Results

Tools and techniques aforementioned in section 3 were tried with different parameters and this chapter contains the results followed by a discussion. The results will be represented as time vs metrics plots. Measured metrics are accuracy and AUC at each time point. On each time vs metrics plot, each curve represents the mean metrics of the 15 subjects (the metric for one subject is the mean for the 6 comparisons metrics see section 4.2 for details), accuracy and AUC are plotted side by side where relevant, statistical assessment was done with significance level $\alpha = 0.05$ (FDR corrected, see section 3.5). A dot of color at a certain time-point, means that the accuracy at this time-point is significantly greater than chance-level for the same color curve.

4.3.1 Methodology

In order to compare different combinations of techniques, we will play with a series of parameters that you can find on table 4.1.

Parameter	Classifier	Cross-validation	# splits	Pseudo-trials
Possible values	LDA, RBF, Linear	Strat, LEO, L1TEO	15, 30	0, 3, 6, 9
Parameter	Time-window avg	Feature extraction	Feature selection	Multiclass
Possible values	0, 3, 5, 7	None, PCA, KPCA	None, F-test, MI	0, 1

Table 4.1: Parameter of a scenario and their possible values

Here are their meaning:

- Classifier: the classifier used. Either LDA, SVM RBF or SVM linear. Section 2.2.2 and 2.2.3;
- Cross-validation: cross-validation strategy used (strat stands for stratified). Section 3.4.1;
- # Splits: in the case of stratified cross-validation, the corresponding number of splits;
- Pseudo-trials: if greater than 0, the number of trials averaged to build pseudo-trials. Section 3.2.3;
- Time-window avg: if greater than 0, the number of time samples averaged within a sliding window. Section 3.2.2;
- Feature extraction: the feature extraction algorithm used either PCA or KPCA. Section 3.2.4;
- Feature selection: univariate selection of feature applied either with F-test or Mutual Information (MI). 2.2.5
- Multiclass: multiclass classification if 1, binary classification otherwise. Section 3.3.3.

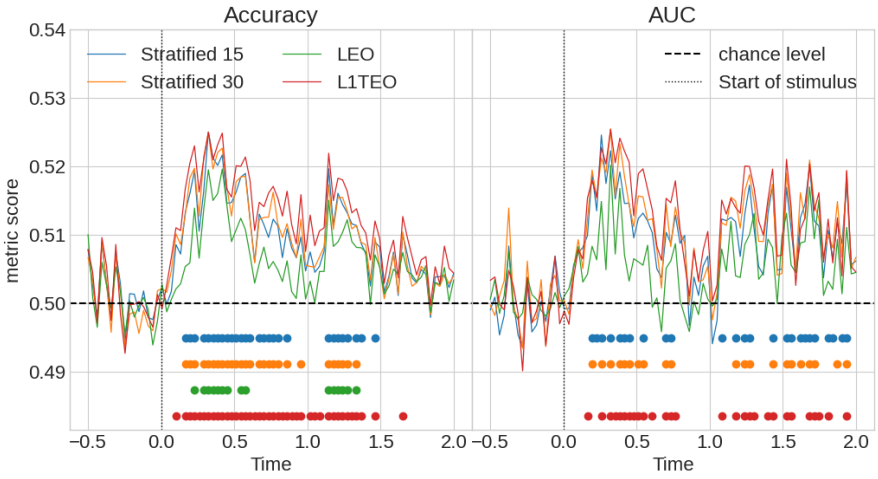
Once all the parameters are fixed to a value, we call that a scenario and we launch the python code corresponding to the pseudo-code 1 with those parameters values. The result is a time-varying accuracy and AUC that can be statistically compared to other runs or to chance-level in order to highlight differences in performances.

4.3.2 Classifier and cross-validation

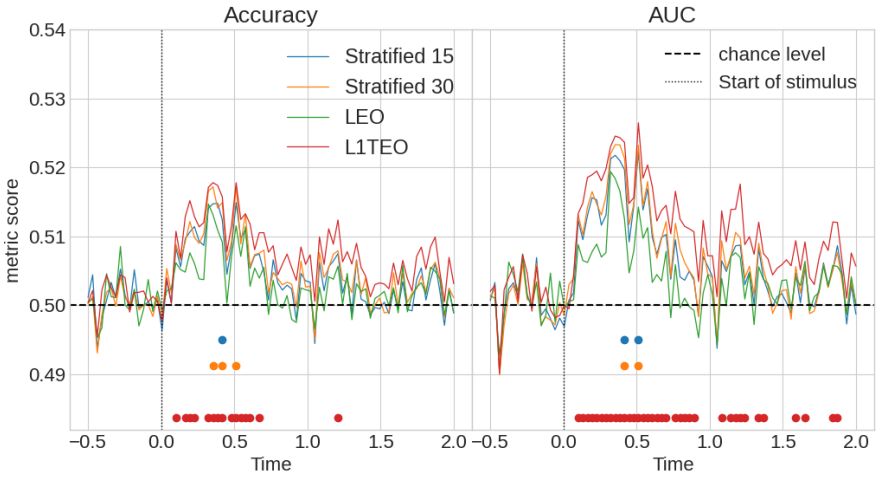
The simple KFold was discarded as the test sets do not have the original proportion of both classes (50-50). This could lead to bias in the metrics. As an answer, we used stratified KFold. We used 15 splits and 30 splits to observe if there were significant differences. LEO and L1TEO cross-validation strategies were also tried. LEO for its ability to evaluate the classifier performances when presented unseen sounds as test sets. L1TEO for evaluating the classifier when the test sets keep the distribution of the original dataset. We ran 3 different classifiers: SVM (linear kernel), SVM (RBF kernel) and LDA. We tried all possible classifier-cross-validation combinations which makes a total of 12 scenarios. We made two types of figures: In the first one (figure 4.3 and 4.4), each cross-validation strategy is separated as an individual sub-figure and all the 3 classifiers are plotted. This will serve for highlighting differences between classifiers. The second one (figure 4.2), is the opposite, we have one sub-figure per classifier and all the four cross-validation strategies are plotted on each figure. This will be used for highlighting differences between cross-validation strategies.

Classifier

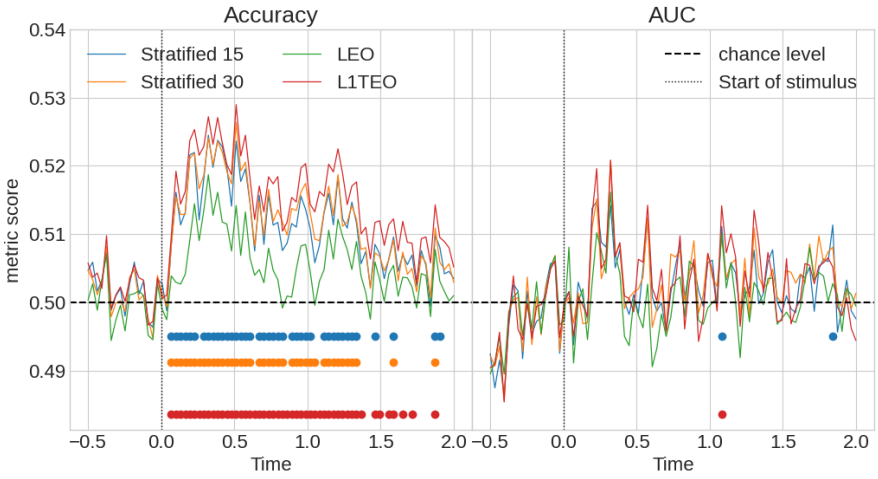
A first observation that we can make in almost all of the 12 first scenarios, is that metrics reach above chance level for a short period of time after stimulus onset. Statistical significance depends on classifier and cross-validation employed and on the measured metric. Both metrics peak at approximately 0.53 in the best case. This appears to be low but it is a typical order of growth achieved when decoding sound stimuli with EEG [4].



(a) SVM RBF



(b) LDA



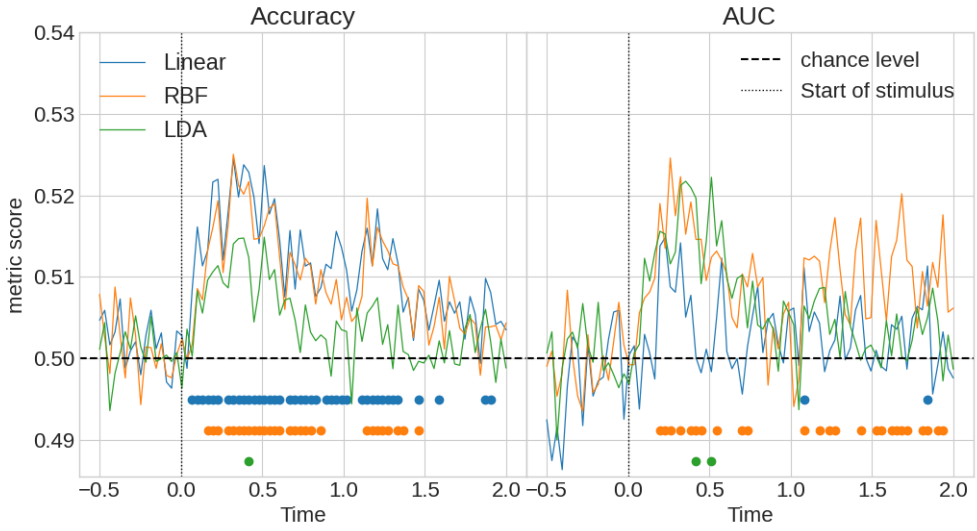
(c) SVM Linear

Figure 4.2: The 3 classifiers with all cross-validation strategies

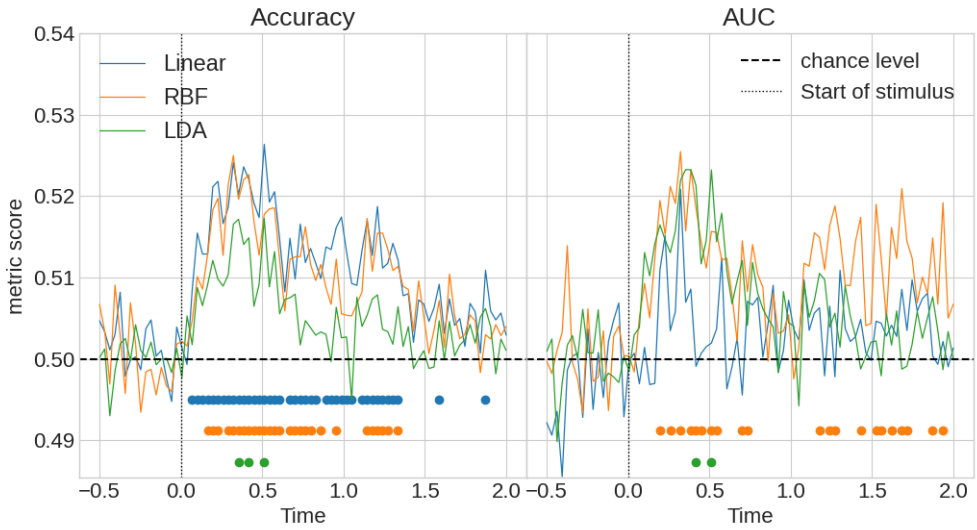
As more clearly observed on figure 4.2a and 4.2c, SVM linear and RBF agree on significant post-stimulus performances and also on the presence of significant time points between $1s$ and $1.5s$ (except for linear results with AUC, for which we have almost no statistical significance). It may suggest that information about the discrimination of the sound could

be present, right after the hearing of the stimulus but also after the offset of the sound since the length of the sound presented was 1 second.

With stratified 15 and 30 cross-validation (figures 4.3a and 4.3b), SVM linear and RBF seem to be better than LDA from an accuracy point of view whilst for AUC, SVM linear and LDA scores appear to be swapped.



(a) Stratified 15 splits



(b) Stratified 30 splits

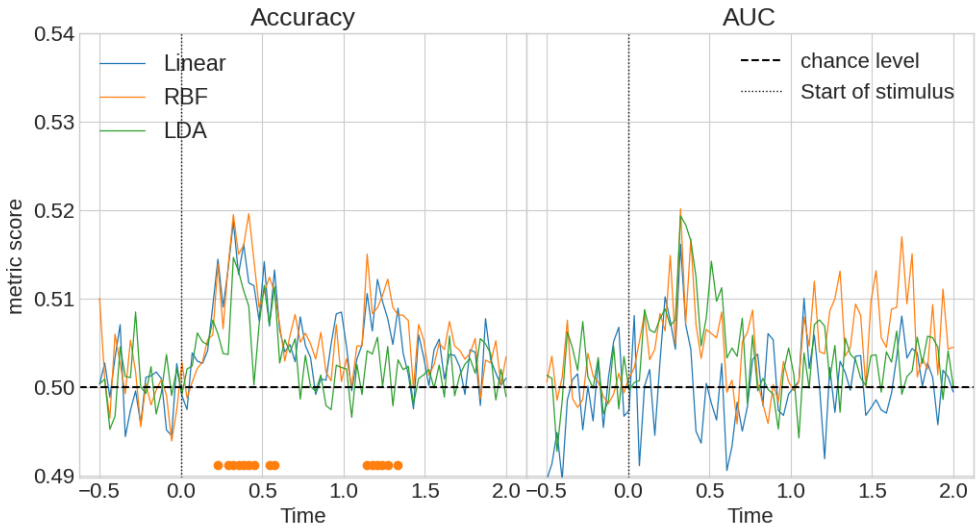
Figure 4.3: Stratified cross-validation strategies for each classifier

For LEO (figure 4.4a), we could not distinguish any significant differences between the 3 classifiers whether for accuracy or AUC except that linear classifier had worse AUC results than other classifiers.

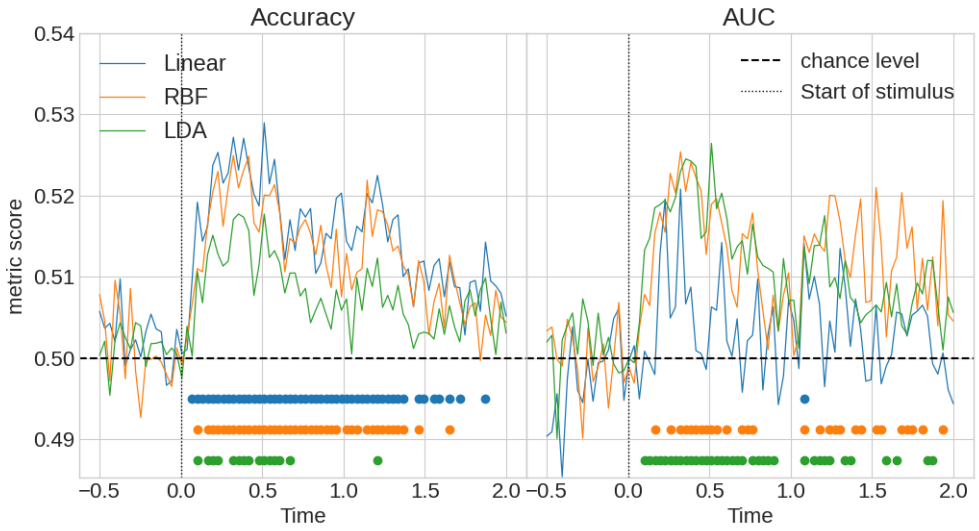
L1TEO (figure 4.4b) gave same ordering as stratified 15 and 30 for the classifiers performances. Again, AUC plots were different and performances of linear kernel were way lower than accuracy ones.

Overall, cross-validation strategies with accuracy metric seem to be quite coherent in-between them, but accuracy results differ a lot from the ones of AUC. AUC flips LDA and

linear performances compared to accuracy. We can draw the conclusion from these graphs that accuracy may not be such a go-to metric. Particularly for SVM linear and LDA where metrics scores are literally swapped between accuracy and AUC.



(a) LEO



(b) L1TEO

Figure 4.4: LEO and L1TEO cross-validation for each classifier

Regarding the choice of a classifier, LDA is a less popular classifier in the literature compared to the other ones. Moreover, LDA metrics were lower and showed lesser significant time points with both stratified strategies. For those reasons, we decided to exclude LDA from our classifier choice for the following scenarios. To choose between linear and RBF kernels, we based our decision on literature review. The most used in EEG decoding is linear SVM 3.2 and we kept it as default classifier for the rest of our computations.

Cross-validation strategy

With SVM RBF (figure 4.2a), all cross-validation strategies gave significant accuracy scores right after-stimulus. Interestingly, between $1s$ and $1.5s$ post-stimulus, there was an other

bunch of samples that gave higher chance level decoding.

AUC with SVM RBF had also, in a more moderate way, near stimulus significant decoding scores except for the LEO strategy. There were also other significant samples sparsely situated after $1s$ for AUC.

SVM linear (figure 4.2c) had a strict difference between accuracy metrics and AUC metrics. For accuracy, all the cross-validation strategies (except LEO) agreed on a metric significantly higher than chance level from $0s$ to $1.5s$ almost without interruption whilst AUC had almost no significant sample.

LDA (figure 4.2b) gave similar results for accuracy and AUC: almost no significant time points except with L1TEO strategy. We can observe that AUC metrics are higher than accuracy ones.

Regarding all those results, we can first observe that LEO strategy metrics scores were lower than other strategies for each classifier and gave almost no significant samples except for RBF. In stratified KFold classifications, each test fold can contain trials from different exemplars. Accordingly, the remaining training sets contain trials from all exemplar sounds. It gives the classifier the ability to learn patterns about all exemplars in the training. While with LEO, all the trials concerning two specific exemplar sounds (one from each class) are put away for testing; meaning that the classifier has no clue about the sounds that will be used for testing. Consequently, accuracies for LEO are lower but more reliable in evaluating how our model generalizes to new exemplar sounds. Nonetheless, LEO has only 6 folds compared to 15, 30 and 45 for stratified (15 splits), stratified (30 splits) and L1TEO respectively. This implicates a significantly lower running time as there is less cross-validation iterations, which can be a good point to reduce execution time when working with a lot of data. However, less folds means that we have less training data and more testing data at each split; thus less data to learn and a bigger place to error and a more severe cross-validation. We can slightly observe this effect between stratified 15 and stratified 30. Concerning L1TEO, it gave the highest results just above stratified 30. Unsurprisingly, it means that our classifiers have less difficulties in predicting test sets with distribution similar to the training set distribution. But as said above, it could be also due to the fact that we have more folds and thus, more iterations in cross-validation with more data to learn and less space to error (as iterative test sets are smaller).

We can observe that there were no considerable differences between all the curves. We can notably observe that almost all cross-validation curves agreed on a metrics peak (but not specially significant) after the stimulus onset. The differences lied more in the height of the whole curve (e.g. lower for LEO) but the shapes were quite similar. This gives us confidence about the reliability and consistency of our evaluation.

All the cross-validation strategies have their pros and cons and can be relevant in different situations. For comparison and simplicity purposes, we decided to adopt the Stratified KFold along this work as it is the most used cross-validation strategy in other paper. Stratified KFold with 30 splits is more computationally intensive than 15 splits and leads to almost similar results. That is why we stick with stratified KFold with 15 splits.

4.3.3 Time-window

We used 3 time-window sizes: 3, 5 and 7. We can observe on figure 4.5 that on the one hand accuracy scores were not specially impacted except that we lose statistical significance and on the other hand, AUC levels are boosted by the use of a time-window with statistically significant above chance level scores, reaching accuracy above 0.54 for time-window of size 7.

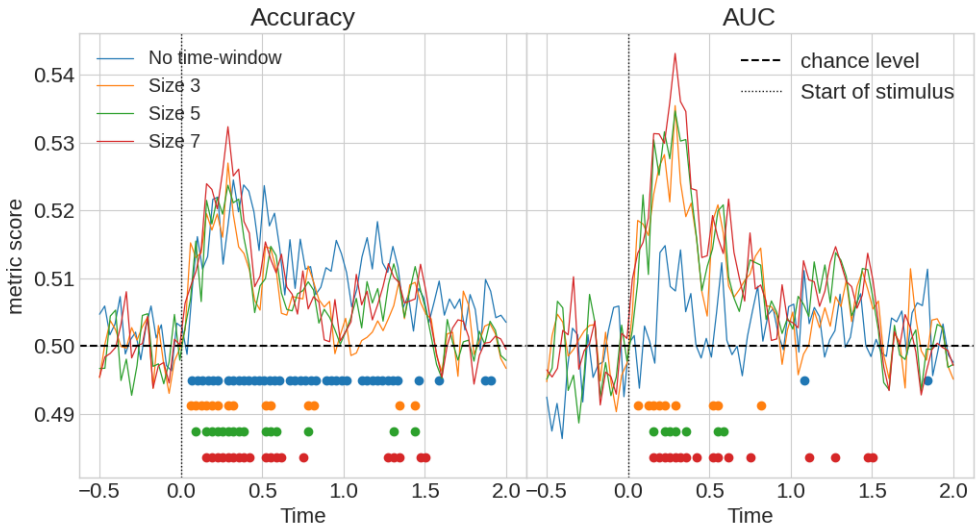


Figure 4.5: Time-window of different sizes

4.3.4 Feature extraction

PCA

We ran PCA to our dataset. The number of components was chosen such that the PCs explained approximately 90% of the variance for the projected dataset. The corresponding number of components was found by interpolation of the number of components - explained variance ratio curve. We plotted this curve at 3 time points (-500ms, 750ms, 2000ms) to check that it was consistent across time. We can see on figure 4.6 that the curves have the same look. To obtain the number of components, we averaged the 3 interpolations from the 3 time-points. The corresponding number of PCs is 18 for 90% of the variance explained. However, it is important to note that the number of components - explained variance ratio curve varies also across subjects. Thus, the final number of components is an approximation. PCA was applied just before putting data into the cross-validation process, transforming the data dimensions from 540×128 to 540×18 (for one subject, one comparison and one time point).

We compared the PCA-run against a run without PCA. We can observe on figure 4.7 that both runs have almost the same results for both metrics. The classifiers performances are almost insensitive to feature extraction with PCA.

KPCA

As PCA performances are limited for high dimensional data. We also ran KPCA with the same number of components to improve efficiency. It employs a non-linear conversion to enhance the separability of the data points within the transformed space [39]. As we can

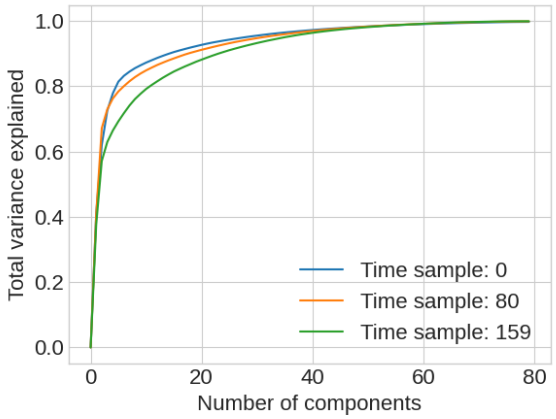


Figure 4.6: Total variance explained in function of the number of components for 3 different time points

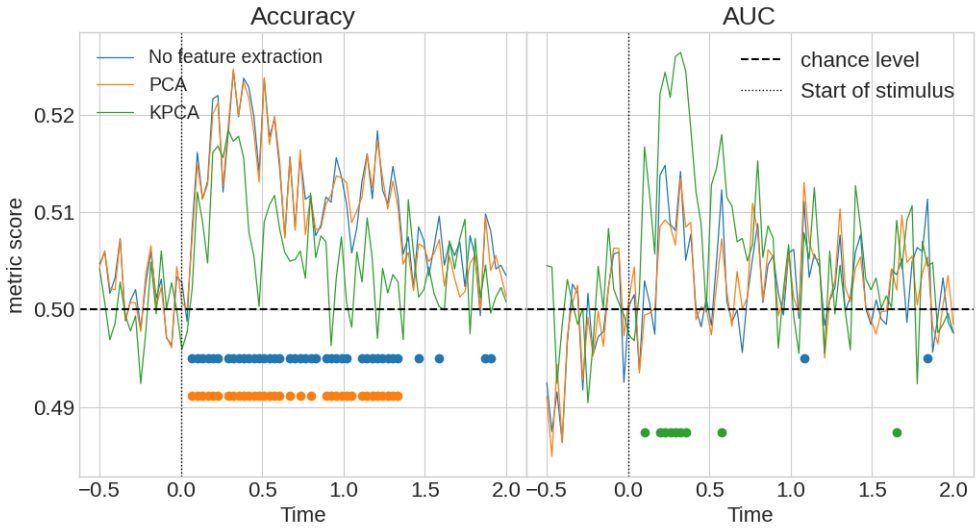


Figure 4.7: PCA and KPCA results

see on figure 4.7, while accuracy scores were worsened, AUC went up with the use of KPCA.

Feature extraction did not show satisfying results. Yet, running those machine learning algorithms led to lower running times. For that reason, we decided to keep on applying PCA with 18 PCs. PCA was chosen over KPCA because PCA scores were similar to the ones without PCA.

4.3.5 Univariate feature selection

As stated before, PCA can separate out noise and artifacts into their own specific components (section 3.2.4). Those components should not be correlated much with the target as they are unrelated to the category of a sound heard. Accordingly, we complemented the PCA feature extraction with a feature selection based on univariate scores with targets. The 15 features among the 18 ones with highest scores were selected. Two different univariate scores were attempted: the F-test statistics and Mutual Information (MI) as there are the unique ones provided by the sklearn library for classification purposes. Figure 4.8 suggests that results for both metrics are worsened by a feature selection based on the Mutual In-

formation (MI) criterion, but interestingly, it provides for the most statistical significance (maybe due to the fact that the variance across the 15 subjects is the lowest: 5.38×10^{-5}, 2.34×10^{-5} and 7.74×10^{-5} for no feature selection, MI and F-test respectively). F-test kept similar results for accuracy and improved AUC results to 0.535 approximately with statistical significance.

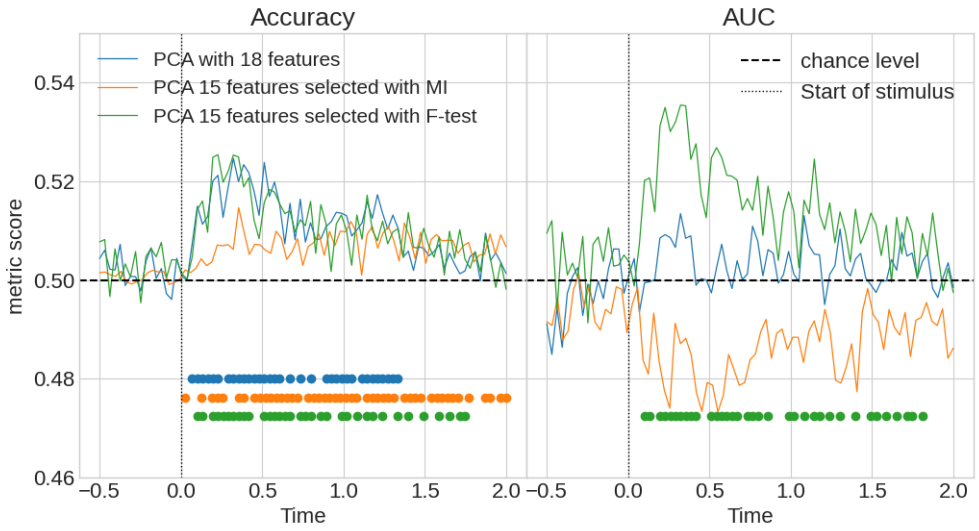


Figure 4.8: Feature selection (15 features) with MI and f-test

4.3.6 Hyperparameter

Linear SVM has one hyperparameter that is C. In typical classification problem, this parameter is tuned to obtain highest metrics as possible. There are a bunch of ways to efficiently do it: grid search, genetic algorithm, ants colony, [47], [48] ... It turns out that tuning C time by time is not relevant in our specific neuroscientific context. Indeed, the requirements here are in a moderate way, the performances, but mostly the coherence. Typically, before stimulus onset, we must have an accuracy around 0.5. When tuning C for each classifier of each time point, we would probably reach an accuracy far above 0.5; among the infinite possibility of classifiers, there must be one that may be reaching for instance 0.7, which is totally incoherent with the expected reality during the pre-stimulus time as the subject has not even heard the sound. Therefore, we did not use a parameter tuning algorithm. Still, we found it interesting to test different order of magnitude for C (that we maintain constant across all time points) to find out which value adapts the most to our specific problem time-independently. Range of C values are usually in a logarithmic scale, we tested the following ones: $\{10^{-5}, 10^{-2}, 10^{-1}, 1\}$ (figure 4.9).

We did not test for higher C values as results did not seem to improve with further increment and the subsequent increase complexity of models led to excessively high running times. We can observe that more complex models ($C = 0.1$ and $C = 1$) worsen accuracy results whilst AUC is pushed to approximately the same values as with RBF kernel. $C = 10^{-2}$ makes a fair compromise between accuracy and AUC results.

4.3.7 Pseudo-trials

We averaged trials n by n to build pseudo-trials with three different n: $3, 6, 9$. We can observe that for $C = 10^{-5}$ (figure 4.10a), using pseudo-trials leads to deplorable results.

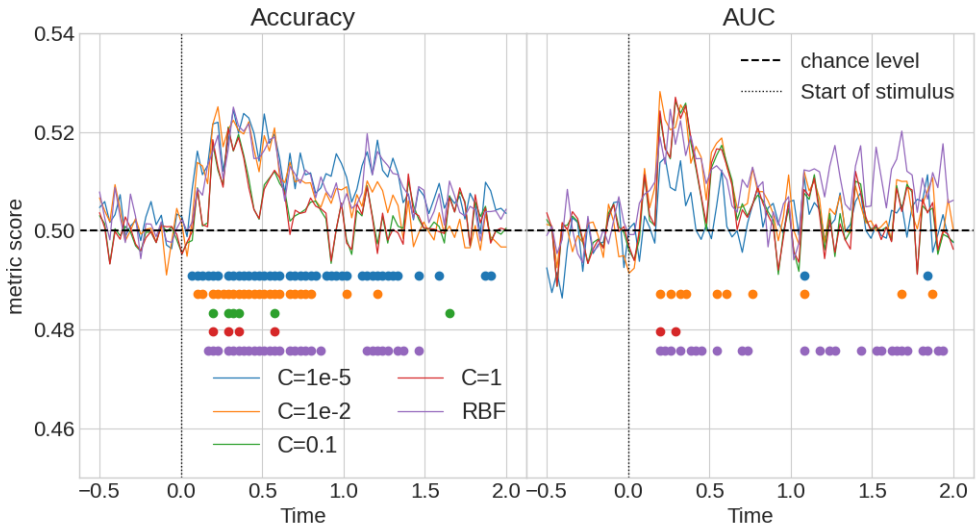
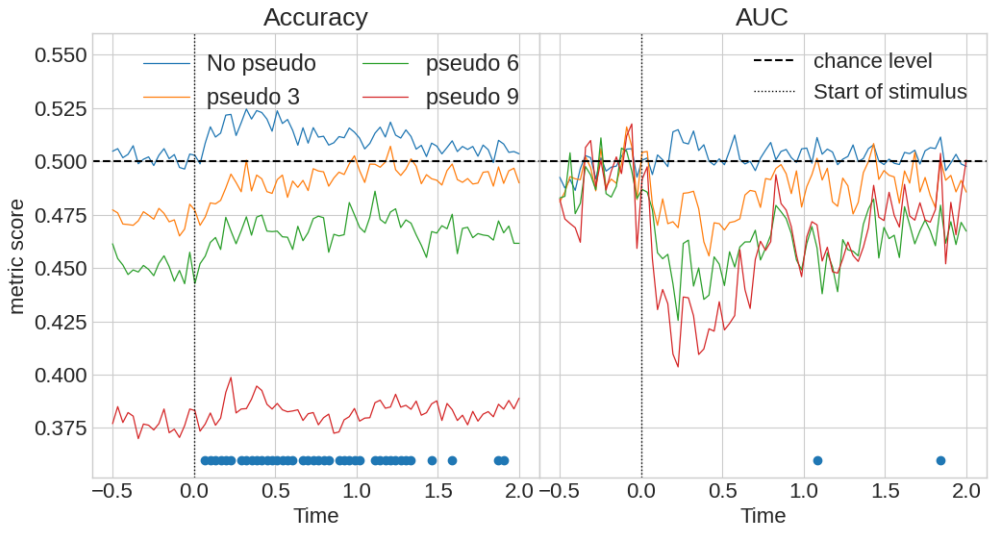
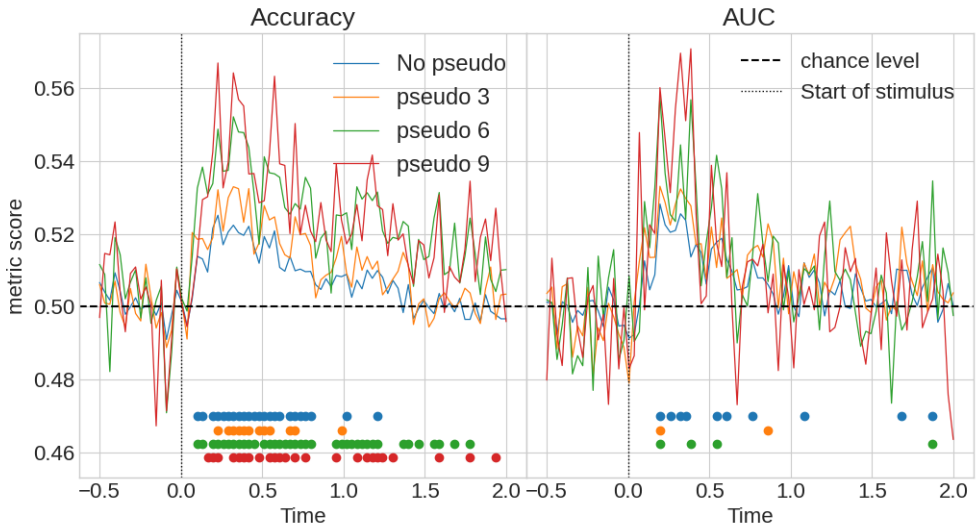


Figure 4.9: Results with different values for C parameter of linear kernel and comparison with RBF kernel

It lowers accuracy and AUC, even down the chance level. This could be explained by the fact that the model struggles to generalize with this fewer number of data caused by the trials-averaging. In combination with the low C value that encourages SVM to find wider margin, the model probably underfits for accuracy and AUC. Nevertheless, after increasing C ($C = 10^{-2}$), we observed that the results were way more better and even surpassed the scenario without pseudo-trials. This increase in C led to more computation time but the fact that we had fewer number of trials (divided by 3, 6 or 9) compensated. With a higher C value, the model with pseudo-trials was able to generalize correctly and exceeded the metrics of single-trials scenario with the same C value. This reinforces the idea that the model was indeed underfitting with lower C. We can further observe that 9 average pseudo-trials with adjusted $C = 10^{-2}$ value loses in stability and seems to oscillate, it also loses in statistical significance. It suggests that averaging too much trials do not improve performances further.



(a) $C = 10^{-5}$



(b) $C = 10^{-2}$

Figure 4.10: Pseudo-trials results for different C values

Interestingly, we find again significant accuracy performances for samples around $1s$ and $1.5s$ for 6 and 9 averaged pseudos-trials, reinforcing the idea that discrimination information in the brain could appear at that particular moment.

4.3.8 Time-window averaging

Averaging does not seem to have any significant effect on the performances, for accuracy as well as for AUC (see figure 4.11). We can solely establish the report that the sawtooth pattern seems to vanish when the average number increases.

4.3.9 Multiclass classification

We tried multiclass classification to observe the performances of our classifier in "real conditions": we wanted to evaluate its ability to discriminate among all the 4 conditions instead

Figure 4.11: Time-window averaging

of 2. The chance level was set at 0.25 for the 4 classes. We did not compute AUC because it was harder to compute as the confusion matrix was 4×4 instead of 2×2. Similarly to binary classifications, we can observe on figure 4.12 above chance-level decoding after stimulus (to analyze potential differences between binary classification and multiclass classification, we plotted accuracy - chance level on figure 4.12 for both strategies). Interestingly, OvO and OvR did not cross each other and OvO had a better post-stimulus accuracy. Moreover, OvR had higher execution time than OvO. This might be because kernel algorithms like SVM don't scale well with the number of data points as they need to stock the kernel matrix. Therefore, it might be beneficial for SVM to train 6 classifiers with 540 data points each as in OvO than 4 classifiers with the whole dataset (1080 data points) during OvR. The peak adjusted accuracies (for time around 0.25s to 0.7s) are in the same order between OvO and binary classifications but we observe that for time points after 0.7s, they start to diverge and OvO maintains statistically significant accuracies until approximately 1.5s.

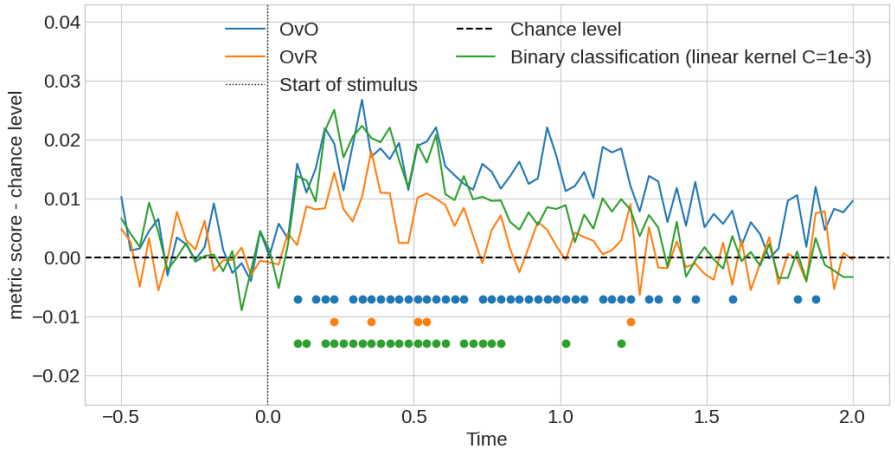


Figure 4.12: Metric - chance level for multiclass strategies and binary classification

Chapter 5

Conclusion and perspectives

In this thesis, we implemented different parameters to visualize how they change decoding performance. The main aim was to find the optimum parameters to obtain high decoding performance and neurophysiological interpretability.

The first objective was to identify a supervised classification algorithm well suited for this kind of work. In parallel to this, we tried multiple cross-validation strategies to elect a reliable one. About the classifier algorithm, SVM remains the standard reference in those kind of studies. LDA is faster and gave pretty good scores on AUC results but its performances were rarely significant and accuracy results were not convincing. SVM RBF showed great scores whether for accuracy or AUC, whilst SVM linear had bad AUC results for low C values ($C = 10^{-5}$) (figure 4.2a and 4.2c). It suggests that the non-linearity of RBF kernel brings the model's complexity that linear kernel could lack to separate the two classes in a more qualitative way and to generalize. But this complexity can be also obtained by increasing the order of magnitude of the regularization parameter. Indeed, we observed that changing the scale of the regularization parameter C led the linear SVM to better results with AUC (figure 4.9). In further researches, results could be improved even more by the tuning of a general C (kept along all time samples).

Regarding the cross-validations, we would recommend to stick with the basics ones such as Stratified KFold: it is the most used and it gives a good overview of the performances with both classes represented equally in all folds. LEO cross-validation can be worth to run to validate the capacity of a model to generalize for handling new exemplars but a lot of data is needed to avoid losing the statistical significance (figure 4.2). L1TEO did not differ so much from stratified strategies. Moreover, it could overestimate results as the test folds are distributed similarly as the train folds. In addition to its difficulty of implementation when different number of trials per exemplar are pressent (due to artifact removal), we wouldn't recommend it.

Once cross-validation strategy and classifier were chosen, we wanted to try different processing techniques on the data in order to observe their influence on the results. PCA results showed that it is an effortless way to reduce the running time while keeping most of the information. With appropriate work and context, we think that it is a good practice to apply it (or another well suited feature extraction technique) before decoding, with a reasonable number of components that explains enough variance. However, contrary to results obtained in [37], it has not showed any improvement in the results. Except a little increase in AUC, Kernel PCA did not show any significant improvement neither. The results ob-

tained by pseudo-trials (figure 4.10) suggested that, like EEG decoding with visual stimuli, the main criterion for good EEG classification remains the quality of acquired signals over quantity, even if this quality is artificially obtained. Having more data can also enhance gradually the performances because it allows for the creation of more pseudo-trials. We strongly encourage to build pseudo-trials early in any machine learning pipeline as it also reduces considerably the execution times.

The time-window approach led to a strict improvement for the AUC metric but no major changes for accuracy. This suggests that AUC benefit from a model that learns in a bigger space with more temporal context. Regarding time-window averaging, it did not change significantly the results but the usual sawtooth pattern is vanishing when we average enough time samples.

Concerning multiclass classification, we can clearly observe on figure 4.12 that OvO strategy gave better results than OvR. OvO gave similar results as binary classification until around 0.7s post-stimulus time. After that, while statistical significance is disappearing for binary classification, multiclass with OvO extends it until around 1.5s post-stimulus. It suggests that multiclass OvO strategy could better decode late post-stimulus information about category discrimination that would be present in the brain. A report that we can add to the use of multiclass classification, especially OvO in this context, is that it is less impacted by the eventual case where a specific comparison between two categories would perform way better or way worse than the other ones. Indeed, accuracies reported for binary classification at a certain time point are the average of the 6 comparisons accuracies for this time point, it is thus sensitive to any "outsider": if one of the comparison was to be straightforwardly (or very hardly) decodable in a general manner, it would impact a lot the final metric in a positive (or negative way), leading to a bias if we consider the metric as a measure of all the comparisons performances. Whereas for OvO multiclass classification, an easily—or hardly—decodable pair of categories will only enhance the reliability—or non-reliability—of one of the vote, limiting the impact on the overall accuracy. This could be analyzed in further researches.

About the measured metrics, we observed that accuracy can not be blindly followed like one used. Indeed, accuracy and AUC results differ quite often. For instance on figure 4.9, accuracy is not much impacted by the change in C value for linear kernel whilst AUC grows significantly. The same applies for 4.5, where accuracy is insensitive to the time-window whereas AUC grows significantly. Therefore, if feasible, it would be interesting to complement accuracy analysis with an other metric such as AUC so as to re-check obtained results.

5.1 Improvements of the results

Regarding literature and above-mentioned results, SVM seems well-suited for the domain presently studied. Still, our exploration of the C hyperparameter is limited and could be further investigated with more computational power. RBF kernel gave good results as well (figure 4.4) and using it with appropriate hyperparameters could be interesting too.

PCA, even if diminishing the computational power needed, did not show any improvement in the metrics. It appears that PCA is limited for high dimensional data. Kernel-PCA

(KPCA) was tried in response to this shortcoming but did not show any significant change neither. There exists a lot of other feature extraction techniques that are often used in EEG processing, whether for artifact removal or feature reduction: Independent Component Analysis (ICA), AutoEncoder (AE), Fourier Transform (FT), Wavelet Transform (WT), etc. ICA is frequently preferred over PCA for artifact removal due to the challenging nature of ensuring that artifact components remain uncorrelated with EEG data. This condition is often difficult to meet despite being important for PCA to separate out artifacts. Additionally, in scenarios where the potential for drifts and EEG data are comparable, PCA struggles to effectively distinguish these interfering factors. As a result, researchers in subsequent studies prefer the utilization of alternative and more adaptable techniques, such as ICA [3]. It could be tried as feature extraction algorithm and complemented by feature selection for artifacts removal.

The use of pseudo-trials showed notable improvement when used with appropriate parameters in this work. Yet, we observed an obvious underfit for small C values. To address this issue, it is possible to draw trials with replacement to create more pseudo-trials (bootstrapping): more data (even if artificially created) could limit the risk of underfitting and would maybe improve the results [12].

5.2 Future works

It has been observed in [49] that sound categories encode themselves more reliably in the brains of blind people than in those of sighted ones. It suggests also that sounds trigger categorical responses in the Ventral Occipito-Temporal Cortex (VOTC) of congenitally blind and sighted people that partially match the topography and functional profile of the visual response. Using functional Magnetic Resonance Imagery (fMRI) and its inherent high space resolution, the above-cited paper focused mainly on the brain's space dimension importance in decoding analysis. Following it, one could elaborate the study with a deeper understanding of the time-dimension importance, e.g. by answering this question: when does the brain express the most apparent categorical concepts discrimination? Electroencephalography (EEG) meets the requirements for a time-analysis: despite having a lower space-resolution, it has a higher time-resolution than fMRI. Along with machine learning, EEG would be the ideal partner for answering the above-mentioned question.

Bibliography

[1] Maham Saeidi, Waldemar Karwowski, Farzad V. Farahani, Krzysztof Fiok, Redha Taiar, P. A. Hancock, and Awad Al-Juaid. Neural decoding of eeg signals with machine learning: A systematic review. *Brain Sciences*, 11(11), 2021.

[2] Mohammad-Parsa Hosseini, Amin Hosseini, and Kiarash Ahi. A review on machine learning for eeg signal processing in bioengineering. *IEEE reviews in biomedical engineering*, 14:204–218, 2020.

[3] Xiao Jiang, Gui-Bin Bian, and Zean Tian. Removal of artifacts from eeg signals: a review. *Sensors*, 19(5):987, 2019.

[4] João M. Correia, Bernadette Jansma, Lars Hausfeld, Sanne Kikkert, and Milene Bonte. Eeg decoding of spoken words in bilingual listeners: from words to language invariant semantic-conceptual representations. *Frontiers in Psychology*, 6, 2015.

[5] Alexander M. Chan, Eric Halgren, Ksenija Marinkovic, and Sydney S. Cash. Decoding word and category-specific spatiotemporal representations from meg and eeg. *NeuroImage*, 54(4):3028–3039, 2011.

[6] Elia Formisano, Federico De Martino, and Giancarlo Valente. Multivariate analysis of fmri time series: classification and regression of brain responses using machine learning. *Magnetic Resonance Imaging*, 26(7):921–934, 2008. Proceedings of the International School on Magnetic Resonance and Brain Function.

[7] Stanislas Chambon, Mathieu N. Galtier, Pierrick J. Arnal, Gilles Wainrib, and Alexandre Gramfort. A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(4):758–769, 2018.

[8] Andrea Biasiucci, Benedetta Franceschiello, and Micah M Murray. Electroencephalography. *Current Biology*, 29(3):R80–R85, 2019.

[9] Fernando Lopes da Silva. Eeg and meg: relevance to neuroscience. *Neuron*, 80(5):1112–1128, 2013.

[10] Eeg (electroencephalogram). `https://www.mayoclinic.org/tests-procedures/eeg/about/pac-20393875`. Accessed: 2023-08-07.

[11] learning eeg: montages technicalities. `https://www.learningeeg.com/montages-and-technical-components`. Accessed: 2023-08-01.

[12] Tijl Grootswagers, Susan G. Wardle, and Thomas A. Carlson. Decoding Dynamic Brain Patterns from Evoked Responses: A Tutorial on Multivariate Pattern Analysis Applied

to Time Series Neuroimaging Data. *Journal of Cognitive Neuroscience*, 29(4):677–697, 04 2017.

[13] Hallowell Davis, Pauline A Davis, Alfred L Loomis, Edmund Newton Harvey, and G Hobart. Electrical reactions of the human brain to auditory stimulation during sleep. *Journal of Neurophysiology*, 2(6):500–514, 1939.

[14] Lin Wang and Gina R Kuperberg. Better together: integrating multivariate with univariate methods, and meg with eeg to study language comprehension. *Language, Cognition and Neuroscience*, pages 1–29, 2023.

[15] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, December 2002.

[16] Lipo Wang. *Support vector machines: theory and applications*, volume 177. Springer Science & Business Media, 2005.

[17] Verleysen M. Support vector machines.

[18] What is the kernel trick? why is it important? `https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d`. Accessed: 2023-06-10.

[19] Kernel functions-introduction to svm kernel. `https://data-flair.training/blogs/svm-kernel-functions/`. Accessed: 2023-06-10.

[20] Support vector machine (svm). `https://dinhanhthi.com/support-vector-machine/`. Accessed: 2023-07-25.

[21] Ml | linear discriminant analysis. `https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/`. Accessed: 2023-07-26.

[22] Suresh Balakrishnama and Aravind Ganapathiraju. Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*, 18(1998):1–8, 1998.

[23] Cross-validation: evaluating estimator performance. `https://scikit-learn.org/stable/modules/cross_validation.html`. Accessed: 2023-05-16.

[24] Receiver operating characteristic. `https://en.wikipedia.org/wiki/Receiver_operating_characteristic`. Accessed: 2023-05-30.

[25] Understanding auc - roc curve. `https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5`. Accessed: 2023-05-30.

[26] Principal component analysis. `https://en.wikipedia.org/wiki/Principal_component_analysis`. Accessed: 2023-05-30.

[27] Linda L. Chao and Alex Martin. Representation of manipulable man-made objects in the dorsal stream. *NeuroImage*, 12(4):478–484, 2000.

[28] JF Alonso, S Romero, MR Ballester, RM Antonijoan, and MA Mañanas. Stress assessment based on eeg univariate features and functional connectivity measures. *Physiological measurement*, 36(7):1351, 2015.

[29] Kenneth A. Norman, Sean M. Polyn, Greg J. Detre, and James V. Haxby. Beyond mind-reading: multi-voxel pattern analysis of fmri data. *Trends in Cognitive Sciences*, 10(9):424–430, 2006.

[30] Björn O. Peters, Gert Pfurtscheller, and Henrik Flyvbjerg. Mining multi-channel eeg for its information content: an ann-based method for a brain–computer interface. *Neural Networks*, 11(7):1429–1433, 1998.

[31] Lucas Parra, Chris Alvino, Akaysha Tang, Barak Pearlmutter, Nick Yeung, Allen Osman, and Paul Sajda. Linear spatial integration for single-trial detection in encephalography. *NeuroImage*, 17(1):223–230, 2002.

[32] Maxime Cauchoix, Gladys Barragan-Jason, Thomas Serre, and Emmanuel J Barbeau. The neural dynamics of face detection in the wild revealed by mvpa. *Journal of Neuroscience*, 34(3):846–854, 2014.

[33] C1 and p1. https://en.wikipedia.org/wiki/C1$_a nd_P$1. *Accessed* : 2023 − 08 − 12.

[34] N170. https://en.wikipedia.org/wiki/N170. Accessed: 2023-08-12.

[35] Leyla Isik, Ethan M Meyers, Joel Z Leibo, and Tomaso Poggio. The dynamics of invariant object recognition in the human visual system. *Journal of neurophysiology*, 111(1):91–102, 2014.

[36] Masaya Misaki, Youn Kim, Peter A. Bandettini, and Nikolaus Kriegeskorte. Comparison of multivariate classifiers and response normalizations for pattern-information fmri. *NeuroImage*, 53(1):103–118, 2010.

[37] Vikrant Doma and Matin Pirouz. A comparative analysis of machine learning methods for emotion recognition using eeg and peripheral physiological signals. *Journal of Big Data*, 7(1):1–21, 2020.

[38] Akshaya R Mane, SD Biradar, and RK Shastri. Review paper on feature extraction methods for eeg signal analysis. *Int. J. Emerg. Trend Eng. Basic Sci*, 2(1):545–552, 2015.

[39] Sayeh Mirzaei and Parisa Ghasemi. Eeg motor imagery classification using dynamic connectivity patterns and convolutional autoencoder. *Biomedical Signal Processing and Control*, 68:102584, 2021.

[40] Plot different svm classifiers in the iris dataset. `https://scikit-learn.org/stable/auto_examples/svm/plot_iris_svc.html`. Accessed: 2023-06-30.

[41] Maximal margin classifier. https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/. Accessed: 2023-06-30.

[42] Siuly and Yan Li. A novel statistical algorithm for multiclass eeg signal classification. *Engineering Applications of Artificial Intelligence*, 34(C):154–167, 2014.

[43] AS Muthanantha Murugavel, S Ramakrishnan, K Balasamy, and T Gopalakrishnan. Lyapunov features based eeg signal classification by multi-class svm. pages 197–201, 2011.

[44] Pavan Ramkumar, Mainak Jas, Sebastian Pannasch, Riitta Hari, and Lauri Parkkonen. Feature-specific information processing precedes concerted activation in human visual cortex. *Journal of Neuroscience*, 33(18):7691–7699, 2013.

[45] Jacobo Diego Sitt, Jean-Remi King, Imen El Karoui, Benjamin Rohaut, Frederic Faugeras, Alexandre Gramfort, Laurent Cohen, Mariano Sigman, Stanislas Dehaene, and Lionel Naccache. Large scale screening of neural signatures of consciousness in patients in a vegetative or minimally conscious state. *Brain*, 137(8):2258–2270, 2014.

[46] 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*, 12:2825–2830, 2011.

[47] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[48] Cheng-Lung Huang and Chieh-Jen Wang. A ga-based feature selection and parameters optimizationfor support vector machines. *Expert Systems with applications*, 31(2):231–240, 2006.

[49] Stefania Mattioni, Mohamed Rezk, Ceren Battal, Roberto Bottini, Karen E Cuculiza Mendoza, Nikolaas N Oosterhof, and Olivier Collignon. Categorical representation from sound and sight in the ventral occipito-temporal cortex of sighted and blind. *eLife*, 9:e50732, feb 2020.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**