

École polytechnique de Louvain

Machine learning for brain networks : task identification based on fMRI BOLD signals

Author: **Charles FRÈRE**

Supervisor: **Jean-Charles DELVENNE**

Readers: **Frédéric CREVECOEUR, Michel VERLEYSEN**

Academic year 2020–2021

Master [120] in Mathematical Engineering

UNIVERSITÉ CATHOLIQUE DE LOUVAIN

Abstract

Ecole Polytechnique de Louvain
Applied Mathematics

Master [120] in Mathematical Engineering

Machine learning for brain networks: task identification based on BOLD fMRI signals

by Charles FRÈRE

Data has become one of the main issues of the 21st century and their potential contribution to society is huge. One of the most interesting sources of data could come from our brain. It is composed of about one hundred millions of neurons each connected to other neurons by up to fifteen thousand synapses and forming a gigantic network responsible for the treatment of signals coming from sensors from all over our body and leading our actions in everyday life. Detailed access to brain activity could help us understand the most important organ of the human body, which is still the subject of lots of interrogations. The last decades saw the emergence of neuroimaging techniques capable to give us more and more precise brain networks representations. The introduction of these data into a machine learning framework takes all its interest. Classification of brain networks can help us understand some brain disorder responsible for pathologies like Alzheimer or schizophrenia, but also brain behaviour when we perform different actions. In this thesis, we will use functional magnetic resonance imaging data from the human connectome project to classify resting-state fMRI and seven other task fMRI from these hundred subjects with three different methods : (i) using functional connectomes (ii) using raw time series of the fMRI and (iii) using autocorrelation of these time series.

Acknowledgements

I would like to tell all my gratitude to my supervisor, Mr Jean-Charles Delvenne who guided me with his support and his advice to accomplish the realisation of this master thesis.

I also thank the readers of this manuscript, Mr Michel Verleysen and particularly Mr Crevecoeur who gave me his more neuroscientific point of view to interpret the results.

All the results detailed in this manuscript are based on the dataset Benjamin Chiêm provided me, alongside some precious advice to inspire me. Thanks to him for his help.

Finally, I would like to thank my loved family who gave me all their moral support during the accomplishment of this work and from the beginning of my studies.

Contents

Abstract	iii
Acknowledgements	v
1 Introduction	1
1.1 Motivations	2
1.2 Thesis organization	2
2 Data acquisition and state of the art	5
2.1 Historical overview of brain connectivity analysis	5
2.2 fMRI BOLD technique	6
2.3 Available databases	7
2.3.1 Different brain atlases	7
2.4 State of the art	8
3 Machine learning framework of classification	11
3.1 Classifiers	11
3.1.1 Support Vector Machine (based on [37])	11
Binary classification	11
Kernel trick	13
Different kernels	14
Multi-class classification [28]	15
3.1.2 Random Forest (from [15])	15
Bagging	17
Tuning parameters	17
3.1.3 Multilayer Perceptron	18
Activation functions	19
Learning of MLP	19
Main parameters	20
3.2 Model selection	20
3.2.1 Performances evaluation	21
3.2.2 Choice of a performance metric	22
3.3 Projection and dimensionality reduction methods	24
3.3.1 Principal component analysis	24
Dimensionality reduction with PCA	25
3.3.2 Recursive feature elimination	25
3.3.3 t-distributed stochastic neighbor embedding	26
4 Classification of tasks from BOLD fMRI	29
4.1 Dataset	29
4.2 First method : Functional connectome	30
4.2.1 Different node association measures	30
Pearson's correlation coefficient	30

	Mutual Information (MI) between time series	31
4.2.2	Vectorization of the functional connectomes	33
4.2.3	Curse of dimensionnality	34
4.2.4	Results	35
	Impact of the association metric	35
	Best classifier	37
	Label-specific results	37
	Impact of dimensionality reduction	38
	Clustering results	39
4.3	Second method : time series based classification	40
4.3.1	Time series reduction techniques	41
	Trivial selection	41
	Resampling	41
4.3.2	Results	42
	Label-specific results	43
	Impact of feature selection	43
	Clustering results	44
4.4	Third method : classification based on autocorrelation between time series	45
4.4.1	Autocorrelation for time series	45
4.4.2	Methodology and results	46
4.4.3	Discussion	47
	Stationarity of resting state	47
	Inefficiency of feature selection	48
4.5	General discussion	49
4.5.1	Best cortical brain atlas	49
4.5.2	Limitations of dataset	50
4.5.3	Subject identification	50
4.5.4	Pertinence of feature selection with RFE	50
5	Conclusion	53
A	Detailed results obtained with DES parcellation	55
	Bibliography	57

List of Figures

2.1	Representation of the two available cortical atlas	8
3.1	Illustration of the margin maximization of the SVM	12
3.2	Illustration of the kernel trick to allow nonlinear separation of the data points. The function Φ matches the points in another feature space. . .	13
3.3	Example of an MLP with three layers of neurons. Inspired from the example in the MLP part of [37]	18
3.4	Representation of the train/test split and the 5-fold cross-validation . .	22
4.1	Examples of Pearson correlation coefficients between two datasets . . .	31
4.2	Venn diagram explaining the links between entropy and mutual information	32
4.3	Representation of a case of overfitting	35
4.4	Examples of functional connectome based with Pearson correlation (A) and mutual information (B) between ROIs signals	37
4.5	(A) Confusion matrix resulting from the the classification of the Pearson correlation based FC by the SVM classifier. (B) Table of the classification results for each class	38
4.6	2D projections of the 800 functional connectomes using t-SNE algorithm, for both association metric used in method 1 before model selection.	40
4.7	2D projections of the 800 functional connectomes using t-SNE algorithm, for FCs based on Pearson correlation and after dimensionality reduction (A) : PCA (B) : RFE	40
4.8	Comparison between the original signal and the resampled time series from one zone of one subject for the motor task (A) and the resting state (B).	42
4.9	(A) Confusion matrix resulting from the the classification of the data of method 2 with the SVM classifier. (B) Table of the classification results for each class	44
4.10	t-SNE projection of data of the the second method with resampling (A) before feature selection and (B) after recursive feature elimination. . .	45
4.11	(A) Confusion matrix resulting from the the classification of the data of method 3 with the SVM classifier. (B) Table of the classification results for each class	47
4.12	Comparison of the average autocorrelations between the different tasks. Each line represents the average of all autocorrelation of 100 lag values from all ROI from all subjects	48
4.13	Comparison between a raw time series of a ROI from a sujet in emotional processing (A) and its corresponding 100 first lag values of its autocorrelation function.	49
4.14	Correlation between the vectorised FCs of the 100 subjects from 2 different sessions of resting state data acquisition	51

A.1 t-SNE projection of data of the the third method with (A) Desikan-Killian and (B) Destrieux atlas	56
---	----

List of Tables

3.1	Parameters tested for model selection	21
3.2	Binary confusion matrix	22
3.3	Confusion matrix from a dataset of G classes	22
4.1	Selected values of the models parameters after a grid search cross-validation for connectomes built with Pearson correlation and mutual information.	36
4.2	Results of the different classifiers applied to the data of the first method 4.2.3 for functional connectomes built with Pearson correlation and MI between nodes.	36
4.3	Results of the different classifiers applied to the data of the first method for functional connectomes built with Pearson correlation. Data is reduced to a feature space of size 100 with PCA and RFE	39
4.4	Results obtained with method 2 for both time series values selection methods.	43
4.5	Results obtained with an optimal SVM model after two dimensionality reduction methods : (1) dimensionality reduction to a feature space of size 100 with PCA and (2) RFE to select 500 best features	43
4.6	Results obtained with method 3 for the optimal number of lagged values per ROI before and after feature selection with RFE to 500 features	46
4.7	Comparison between the main results obtained with DK and DES parcellation	49
A.1	Results of the different classifiers applied to the data of the first method for functional connectomes built with Pearson correlation for DES parcellation	55
A.2	Results obtained with method 2 before and after RFE with DES atlas	55
A.3	Results obtained with method 3 for the optimal number of lagged values per ROI before and after feature selection with RFE to 500 features with the DES atlas	55

List of Abbreviations

HCP	Human Connectome Project
MRI	Magnetic Resonance Imaging
ROI	Region Of Interest
DES	Destrieux
DK	Desikan-Killiany
FC	Functionnal Connectome
SVM	Support Vector Machine
RF	Random Forest
MLP	Multilayer Perceptron
MI	Mutual Information
PCA	Principal Component Analysis
RFE	Recursive Feature Elimination
RS	Resting State
EMO	Emotional processing
GAM	Gambling
LAN	Language processing
MOT	Motor task
REL	Relational processing
SOC	Social cognition
WM	Working Memory

Chapter 1

Introduction

Over the past recent years, the amount of data generated in any field of our everyday life has reached some unimaginable level. Data are everywhere and worldwide technologies have never been so developed in the human era. These technologies allow generating a huge amount of data, that we can efficiently store in databases for some potential use. Since the early 1990', the field of "big data" has been more and more studied over the years and we now fully realize all their potential. There are big data applications in a large variety of domains like marketing, decision making, optimization, economy, meteorology or medicine. This emergence of this field comes from the development of new data management techniques to efficiently store and retrieve data, but also thanks to the continually increasing computer's power, which allow us to make some computations once were too heavy to carry out.

One of the largest potential sources of data is our brain. Our brain is the most complex and most important organ of our body. It has inspired lots of researches, has been the source of lots of questions that sometimes have not yet been answered. Our brain has a very unique and complex structure. It is composed of about one hundred million neurons linked by synapses and forming a gigantic network responsible for the treatment of signals coming from sensors from all over our body and leading our actions in everyday life. All we knew about its structure once was only due to autopsies and physical observation of post-mortem inactive brain organ. These observations were quite imprecise and incomplete compared to all the knowledge carried by the brain. With the arrival of medical imaging techniques, allowing the scientific community to better understand the neural activity, some questions have been answered and other challenges have been raised. Different data acquisition techniques have been developed since then and brain structure analysis disposes of powerful tools.

In this context, the research in neurosciences has drastically risen over the past few years and more and more scientific papers are published on related studies. Neurosciences are the study of the nervous system, its structure and behaviour from the molecular level to the organs scale and its effects on the human body. It is a transdisciplinary domain that uses a large variety of research fields from biology to mathematics or bio-informatics, each aiming to understand some specificities of the brain. With the evolution of the precision of neuroimaging and data quantity due to more and more efficient sensors, the need for strong mathematical tools was felt. New neuroscience domains were opened since then, like connectomics, which studies the connectivity of brain regions, a connectome being the mathematical model representing the interaction between these elements of the brain. From this point of view, the human brain can be seen as a high dimensional graph on which we can apply some mathematical theorems, algorithms or machine learning techniques to retrieve

relevant information.

This is where comes the interest in machine learning models and methods. These strong mathematical data analysis techniques constitute important tools to allow us to understand better the behaviour of this organ from which we still have so much to learn.

1.1 Motivations

The recent progress of neuroimaging has brought to us more complete and precise brain representation than ever. We dispose then of a powerful base of information which, alongside supercomputers and state of the art machine learning techniques, constitute a huge potential of research to understand human behaviour, to identify the source of pathologies and find ways to cure what can't be for now. These neuroimaging techniques provide us with representations of the brain in the form of numerical values over space and time. Each activity of a region of the brain has the form of a time series, a list of values indexed in time separated by a fixed time step. The interest of using mathematical models to treat those data is to find the best way to extract relevant pieces of information that could allow us to classify brains according to some criteria relative to our study.

Classification is one of the main objectives of machine learning techniques. This process consists of the attribution of a label to each data treated by a model. It is currently used in computer vision to determine whether an image represents a cat, a dog or anything else, in natural language processing where an algorithm can tell the sentiment of a given text, and in many other fields. The interest in applying machine learning and data sciences to neurosciences seems unquestionable. We could classify healthy brains from those presenting some disease, prevent future pathologies and determine the cause of some malfunctioning, but also just better understand how our brain works.

Besides all the medical interests of those researches which potentially will lead us to cure some misunderstood diseases like Alzheimer, Parkinson or schizophrenia, we can also focus on more philosophical questions that always have been some kind of fantasies on the collective imagination. One of the main question we could ask over the potential of this tremendous amount of data is if it could eventually be possible to understand what is happening on a functional brain, what a person is doing or thinking. In simple words, is it possible to read in our minds? This is the question to which we will try to find an answer.

1.2 Thesis organization

In chapter 2, we will give a historical overview of the principal data acquisition techniques used to represent brain activity, their development and their current state. We will detail a part of the state of the art in brain networks classification to set the background in which the current research is taking place. We detail all the processes leading to the elaboration of the dataset that will be used in the current work and precisely describe it to understand the matter used in the main part of this work.

Chapter 3 will bring the theoretical background necessary for classification. A description of classification processes and models will be detailed, we will explain the methodology and metrics to assess the results, but also the projection and feature selection techniques.

Chapter 4 is the core of this master thesis. We will focus on a machine learning framework applied to the treatment of brain functional magnetic resonance imaging signals. We will compare three different methods of data processing and perform classifications of brain signals to build a model that would get the best results at predicting a task carried out by a subject based on its brain activity.

Chapter 5 will be the conclusion of this work where we will summarise the most important results, discuss the potential future work on this subject and conclude with some methods of interest that were not covered in this thesis but could be the subject of future studies.

Chapter 2

Data acquisition and state of the art

From the biological phenomena occurring in our brains to the database that will follow us during this work, a lot of research has been necessary to obtain the precision of the images we can generate today. Starting from a neuroimage, different processes are needed to build a clean database of brain networks and different processes have existed before reaching the mastery of the actual technologies.

2.1 Historical overview of brain connectivity analysis

The anatomical study on the nervous system takes its sources in the Antiquity with Hippocrate or Dalien passing through Descartes who will lead the way for lots of future progress in anatomy and physiology like those of André Vésale in the Renaissance. Understanding the origin of intractable brain disorders like schizophrenia or dementia has been a consistent challenge to find some ways to cure them. This is the principal reason people got involved in the research on the brain over time. The term "neurosciences" appears in the 1960', to designate the study of the nervous system from an electrophysiological point of view. A major realisation of this time was the Nobel price obtained by David Hubel and Torsten Wiesel for their work [23] on the neurons electric response from the visual cortex of cats when seeing different images. The technique used in their work provided encouraging results but used invasive methods and was focused on too small areas of the brain which induced repeated experiments to obtain sufficient results.

The arrival of non-invasive methods (do not require the injection of radioisotope agent) like positron emission tomography (PET) in the 1980' and functional magnetic resonance imaging (fMRI) in the 1990' allowed reasonably precise representations of the brain. This lead to the development of the network sciences applied in this domain. The central concept used in brain networks is the connectome. This term was first used in the structural description of the human brain by Sporns et al. [34] in 2005 to represent pairwise interactions between elements of the brain. Originally, the term connectome supposed a perfect knowledge of the neuron scale brain interactions. They were studied based on micro-scale invasive imaging techniques. But since this date, the number of researches increased exponentially and lead the research to also study macroscale interactions between brain zones. Macroscale connectomics use subdivision on the brain into multiple zones called Regions Of Interest (ROIs). This allows relatively good brain representations using non-invasive methods. The current work will rely on macroscale connectomes of the human brain whose signals

are recorded with the fMRI BOLD technique.

2.2 fMRI BOLD technique

Intuitively, we could consider the brain as an extremely complex network, where neurons are the nodes and the axons representing the edges between them. Our logic could lead us to think to measure the activity of each neuron individually. But as we said in the introduction, a brain is composed of more than one hundred billion neurons, each linked to other neurons by hundreds of synapses. These intuitive methods, in addition to being extremely difficult to implement in practice, would generate a tremendous amount of data that would be extremely hard to manipulate. All the mathematical models we would use, even with supercomputers, would suffer from a much too long execution time. Here is where comes the interest of macroscale connectomics.

As said earlier, the data we will be using come from the Blood Oxygen Level Dependent functional Magnetic Resonance Imaging (BOLD fMRI) technique (well detailed in [18]). This is a non-invasive method of brain representation that generates macroscale data measured at preselected ROIs. Since its introduction in 1990, it has been used in lots of researches from different fields ranging from psychology to neurosciences.

fMRI is a process based on the detection of physiological changes in the brain. fMRI acquires data by detecting changes in regional brain activity induced by a neural activation. The first one is the increase of the cerebral blood flow (CBF) which can be non invasively detected with arterial spin labelling. The second is the change of oxygenation of the blood (Blood Oxygen Level Dependent, also called BOLD contrast. At rest, brain tissues are well oxygenated and haemoglobin is magnetically indistinguishable from brain tissues, but after activation of a region, fully deoxygenated haemoglobin becomes highly paramagnetic and these changes influence the magnetic susceptibility of the brain tissues which is reflected in the magnetic resonance image.

Once the experiment is over, data still has to be processed as the BOLD contrast between activity and control state is quite small (less than 1%). Moreover, there is often a lot of noise during experiments coming from small body motions, breathing, cardiac activity, all parasite signals that add noise to the results which then have to be processed. These pre-processing steps include time-slice correction, body motion registration to subtract the induced noise, spatial smoothing to improve the signal-to-noise ratio and other techniques to finish with denoised signals for each ROIs of the chosen brain atlas. A clear activation map of the brain based on a threshold activation value can be obtained based on these time series.

A major advantage of BOLD fMRI compared to other methods is its high spatial resolution. However, its principal downside is its low temporal resolution due to the low frequency of BOLD contrasts which have a reaction time of around 3s after a stimulus. But for the current purpose, as we have to study and classify tasks performed during long periods, we dispose of a sufficient quantity of data to overcome the negative aspects of this technique.

2.3 Available databases

In the domain of neurosciences, data are not as easy to collect as it necessitates costly materials like 3T scanners. In addition to this cost parameter, subjects are also very limited. The need for collaboration of the scientific community is mandatory. Relevant studies can only be made by analysing and comparing a sufficiently large amount of data. This situation leads to the apparition of the **Human Connectome Project** (HCP). The idea of the HCP was introduced in 2009 by the NIH Neuroscience Blueprint Institutes and Centers and lead to a consortium in 2010 led by Washington University, the University of Minnesota, and Oxford University (called the “WU-Minn” HCP consortium).

The WU-Minn HCP consortium aims to build a database of human brain connectivity composed of 1200 healthy adult subjects between 22 and 35 years old and measured by the state of the art technology. The MR signals recorded in these databases can be structural MRI, resting-state fMRI, task fMRI or diffusion MRI. Other technologies are also part of this database but we mainly focus on fMRI signals. Every subject participating in experiments must perform two sessions of resting-state fMRI and also two sessions of every other task fMRI. This repetition makes sure sufficiently high-quality data are acquired.

The main purpose of the HCP was to make powerful and precise databases freely available to the scientific community. There is free access to raw data but also to data after different steps of pre-processing. HCP databases have already been the raw material of lots of studies ([1, 12, 29] for example) and in the future, the development of the HCP dataset and related methods can only improve the potential of the research in neuroscience. In this work, we will be using the “100 unrelated subjects” dataset that is precisely described in the classification chapter 4.1. It will allow us to study the signals of 100 subjects recorded while processing 7 different tasks and a resting state, which we choose to consider as a “low activity” task to study the differences of its mechanisms compared to other task fMRI.

2.3.1 Different brain atlases

As explained in chapter 2.2, even if the human brain is composed of about 100 billion neurons, we can’t study its structure as a huge dimensional graph with that many nodes, unfortunately, we don’t have the tools to measure brain network at this level of granularity. When we study the activity of the brain, we can remark that every activity of a person activates certain parts of the brain. As discussed in the state of the art, this had already been observed in the visual cortex of cats by Hubel and Wiesel in their work [23] and more recent studies highlight the human brain zones associated with certain tasks, like Hasson and al. in [21]. We conclude from this fact that it is more interesting to study the behaviour of the brain for well-delimited zones and to consider this zone as a node representing the global behaviour of all the neurons in this area.

In the dataset from the Human Connectome Project, we have two types of cortical parcellation: The Desikan-Killiany which separates the brain into 84 zones and the Destrieux brain atlas with 164 zones of the brain¹. We can visualize the different parcellations on figures 2.1a and 2.1b.

¹Images available on <https://surfer.nmr.mgh.harvard.edu/fswiki/CorticalParcellation>

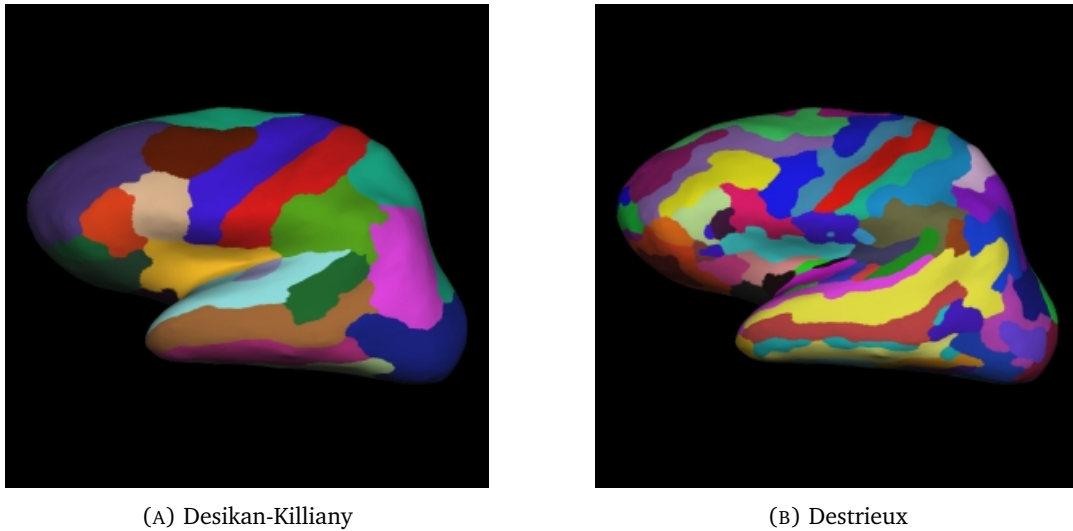


FIGURE 2.1: Representation of the two available cortical atlas

There exist many other cortical atlases² but we will focus our work on these two particular ones. Applying the same methods on two different datasets can be interesting, a certain method could give us better results with one atlas and another could work better with the other type of parcellation.

2.4 State of the art

Until the emergence of precise brain representations techniques like MRI, magnetoencephalography (MEG) or electroencephalography (EEG), lots of researches have been published over brain networks and the classification of brain disorder has become a major subject of interest. The work of Bullmore and Sporns [11] published in 2009 was a complete graph theoretical analysis of brain networks that inspired lots of later publications on brain networks. The technique of construction of functional connectomes based on fMRI signals was later used for different applications [12].

The application of machine learning methods on human connectome based on MRI data became very current during the last 15 years. A comparative structured summary of all machine learning related methods based on functional MRI (fMRI) or diffusion MRI (dMRI) is detailed in [9], where the authors highlight the exponential growth of MRI based studies and the associated growth of interest from 2008 to 2015.

More recently, different techniques based on functional connectomes from fMRI data of the Human connectome project were developed. Most of the recent researches on fMRI have the purpose to identify the neurological disorder and classify brains following their health state. Brain network classification is used for autism diagnosis, which has been the subject of a machine learning competition in 2020 [5], a very challenging task due to the heterogeneity of the brains: maximal the maximal accuracy reached 70%. Classification of Alzheimer's disease is also a recurrent subject, with techniques like convolutional neural networks [33] or with minimum spanning tree of functional network on fMRI data [19]. There is also recent deep learning

²An exhaustive list is described on this website <https://www.lead-dbs.org/helpsupport/knowledge-base/atlasresources/cortical-atlas-parcellations-mni-space/>

methods applied to hyperactivity disorder classification [27].

Other studies focus on the analysis of brain network classification between healthy subjects to try to understand mechanisms of the brain between people and between different activities. We can cite graph embedding for functional fingerprinting (GEEF) [1] published in 2020 which uses group-level decomposition of FCs into eigenvectors to perform subject and task identification, also using the 100 subject databases of the HCP. The GEEF method for task classification was compared to the FCs and obtained very good accuracy. But the improvement of these technologies and the low price of data storage induce the storage of more and more individuals data. This raises the issue of the privacy of datasets. Indeed, actual data science methods are very powerful as the amount of data get bigger every day and people can get concerned about the privacy of their brain data which are made public. These databases are meant to be anonymous but de-anonymization on neuroimaging datasets can be possible. In the work of Ravindra and Grama [29], this risk is explained by performing Subject and task identification attacks with a technique of novel matrix analysis used to extract discriminating features in neuroimages. Task identification is also proved to be efficient and they exposed the danger with data de-anonymization: if one dataset is de-anonymized, we can do the same for any other database containing the same set of subjects.

Also in 2020, a structure-informed functional connectivity study of Chiem et al. [12] introduces a framework for the identification of potential control centres in the brain. This work is also based on the "100 unrelated subjects" dataset of the Human Connectome Project. It highlights some state-specific control regions of the brain based on functional fMRI.

Chapter 3

Machine learning framework of classification

3.1 Classifiers

The main purpose of this work will reside in comparing three different methods in term of manipulation of the fMRI data. But for each of these approaches, what we will mainly study is the classification of these data. We will look for a model that can get the best results at correctly determining what is the activity of the brain. In the following section, we will then describe the functioning, the benefits and the interests of three different types of classifiers which we will then compare the results in the next sections. The three classifiers we will be interested in are the Support Vector Machine (SVM), Random Forest and Multilayer Perceptron (MLP).

3.1.1 Support Vector Machine (based on [37])

Support Vector Machines (SVMs) are well known supervised models for classification and regression analysis. They were first developed in the 1990' by Vapnik, Boser and colleagues in the ATT laboratories [7, 36]. SVMs are one of the most robust methods, based on a statistical learning framework. Their objective is to find the best boundary between different sets of data having to be separated from each other depending on the labels assigned to the data. This can be done by a linear separation or by mapping the available data into another space, resulting in a defined function called kernel. We will first describe the basic principles of SVMs for two classes classification and then extend the discussion on how it is used when data points are to be classified in more than two classes.

Binary classification

We consider a set of points described with a certain number of features. These points have to be classified in a class C_1 or another class C_2 according to their respective labels. The SVM is a linear classifier whose objective is to find a hyperplane of dimension $n - 1$ in a n -dimensional space that maximizes the size of the border (the margin) between the points (if points are linearly separable). The points whose distance to the hyperplane is minimal are called the *support vectors*. Indeed, many hyperplanes could separate the 2 sets of data but we want the maximal margin since in general, the larger the margin is, the lower the generalization error becomes so the accuracy is better.

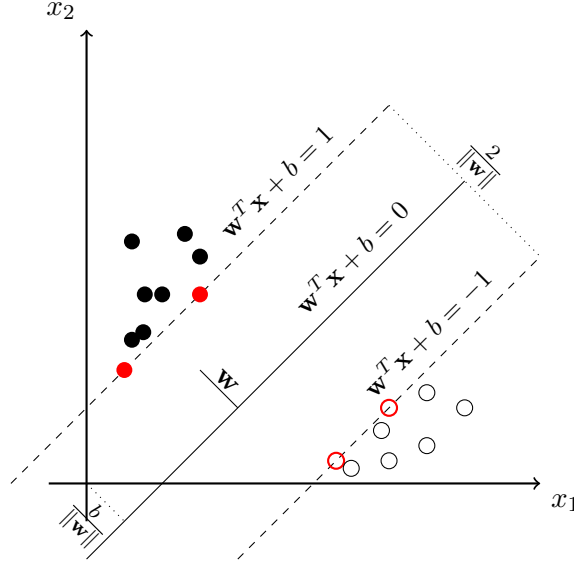


FIGURE 3.1: Illustration of the margin maximization of the SVM

In figure 3.1, we have a particular case of a hyperplane in R^2 , where the red line, a hyperplane of dimension 1 linearly separates the two sets of points (green and blue). Following the notations of this figure, the width of the margin is defined as

$$\text{width} = \frac{2|k|}{\|\mathbf{w}\|} \quad (3.1)$$

The problem consisting to find the maximal margin width that linearly separates the two classes of data can be formulated as an optimization problem :

$$\begin{aligned} \max_{\mathbf{w}, b} \quad & \frac{2|k|}{\|\mathbf{w}\|} \\ \text{subject to} \quad & \mathbf{w}^T \mathbf{x} + b \geq k, \forall \mathbf{x} \in C_1 \\ & \mathbf{w}^T \mathbf{x} + b \leq -k, \forall \mathbf{x} \in C_2 \end{aligned} \quad (3.2)$$

In this optimization problem, the two constraints are used to ensure that data points of both class are located in the good side of the hyperplane. We can notice there is a scale and unit data for which $k = 1$, that will simplify the notations by indexing the data points and associating them a new variable y with a superscript $i \in 1, 2, \dots, N$. Class C_1 correspond to 1 and class C_2 corresponds to -1 :

$$y^i = 1 \quad \forall \mathbf{x}^i \in C_1 \quad (3.3)$$

$$y^i = -1 \quad \forall \mathbf{x}^i \in C_2 \quad (3.4)$$

Then, this optimization problem 3.2 becomes :

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \|\mathbf{w}\| \\ \text{subject to} \quad & y^i (\mathbf{w}^T \mathbf{x}^i + b) \geq 1, \quad \forall i \in 1, 2, \dots, N \end{aligned} \quad (3.5)$$

The problem is convex, so if the problem is feasible, there is a unique minimum value and a unique minimizer $(\mathbf{w}, b)$ providing the minimum. Moreover, this problem can be solved using quadratic programming techniques, which explains why SVMs

are so attractive and widely used in machine learning.

However, it can happen that the data are not linearly separable, which makes the problem not feasible. To overcome this problematic, we introduce slack variables ξ^i , which allow some data instances to fall within the margin or in the wrong class, but penalizing these data ($\xi^i = 0$ for correctly classified points). With the introduction of these slack variables, the formulation of the problem becomes :

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & ||\mathbf{w}'||^2 + C \sum_{i=1}^N \xi^i \\ \text{subject to} \quad & y^i(\mathbf{w}'^T \mathbf{x}^i + b) \geq 1 - \xi^i, \quad \forall i \in 1, 2, \dots, N \\ & \xi^i \geq 0, \quad \forall i \in 1, 2, \dots, N \end{aligned} \quad (3.6)$$

Where the constant C is introduced to trade-off the margin width and the impact of misclassifications. This problem is called the **soft-margin** formulation, while the first formulation of the problem (3.2), called the **hard-margin** problem, doesn't accept any misclassification. The introduction of these slack variables is essential as it always gives a solution to the problem, by minimizing the sum of distances to the margin, moreover, as C tends to infinity, we get closer to the hard-margin solution. We will then only consider soft-margin SVMs from now.

Kernel trick

To go further, let us speak about the limitations of linear decision surfaces. Indeed, sometimes hard-margin separations can be found if we modify the problem to allow nonlinear separation of the data. Let's have an example to illustrate this situation.

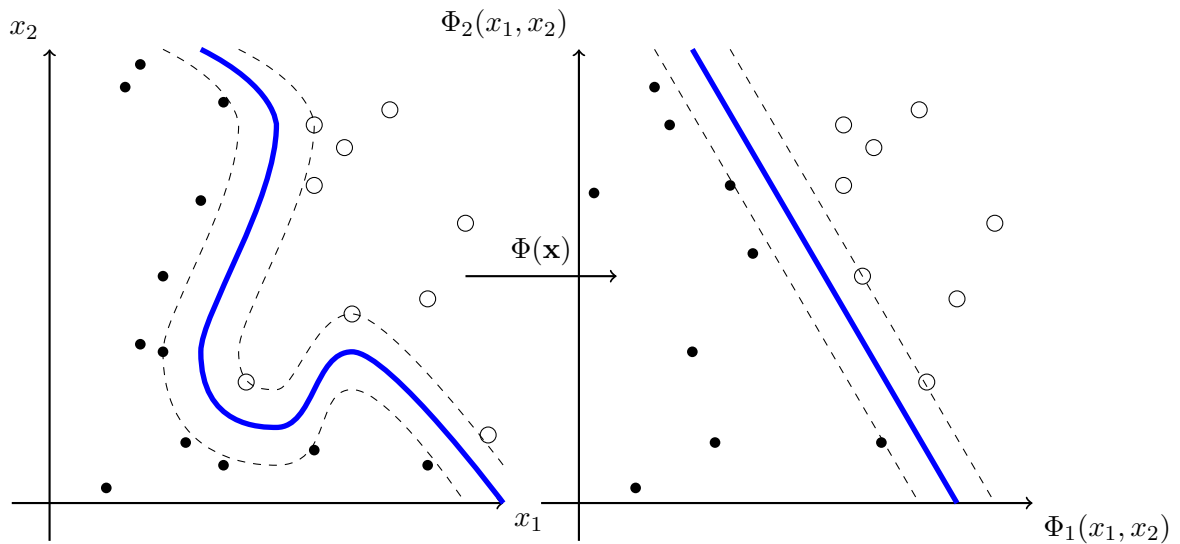


FIGURE 3.2: Illustration of the kernel trick to allow nonlinear separation of the data points. The function Φ matches the points in another feature space.

On the example of figure 3.2, we can see that in the left situation, the feature space of the components of the points $\mathbf{x}^i$, the two classes are not linearly separable. Thanks to an appropriate linear mapping of these data, we can transform the feature

space in order to make the data linearly separable by finding a function $\Phi(\mathbf{x})$ to map to a different space. The SVM formulation then becomes :

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi^i \\ \text{subject to} \quad & y^i(\mathbf{w}^T \Phi(\mathbf{x}^i) + b) \geq 1 - \xi^i, \quad \forall i \in 1, 2, \dots, N \\ & \xi^i \geq 0, \quad \forall i \in 1, 2, \dots, N \end{aligned} \quad (3.7)$$

On the example the feature space has 2 dimensions but it can potentially have an infinite number of dimensions. Intuitively, the more dimensions the feature space has, the easier it is to find a separation, but the main downside resides in the computation of $\Phi(\mathbf{x})$: the explicit mapping can be very expensive if $\Phi(\mathbf{x})$ is high dimensional. To explain how to deal with this, let's consider the dual of problem (3.7) :

$$\begin{aligned} \min_{\alpha^i} \quad & \frac{1}{2} \sum_{i,j=1}^n \alpha^i \alpha^j y^i y^j (\Phi(\mathbf{x}^i)^T \Phi(\mathbf{x}^j)) - \sum_{i=1}^N \alpha^i \\ \text{subject to} \quad & 0 \leq \alpha^i \leq C, \quad \forall i \in 1, 2, \dots, N \\ & \sum_{i=1}^N \alpha^i y^i = 0 \end{aligned} \quad (3.8)$$

In this formulation, the mapping ϕ of the data only appears once through the form of a scalar $\Phi(\mathbf{x}^i)^T \Phi(\mathbf{x}^j)$. From this scalar product, we define the kernel function as

$$K(\mathbf{x}^i, \mathbf{x}^j) = \Phi(\mathbf{x}^i)^T \Phi(\mathbf{x}^j) \quad (3.9)$$

This function evaluates the inner product of the images of the data in the induced feature space. Consequently, as solving the primal (3.7) is equivalent to solving the dual (3.8), if we know the kernel value $K(\mathbf{x}^i, \mathbf{x}^j)$, we do not need to know the explicit mapping Φ of the data into the high-dimensional space to solve the optimization problem, which provides in general huge computational savings over explicit mapping [37]. This process is known as the "kernel trick". There exist several different types of kernel, which we are going to detail further. However, not all functions K can be used as a kernel. James Mercer proposed the following condition to be a kernel : if the Gram matrix $G^{ij} = K(\mathbf{x}^i, \mathbf{x}^j)$ is always semi-positive definite, then there is a feature space $\Phi(\mathbf{x})$.

Different kernels

During the model selection phase, it is interesting to compare the results obtained with different kernel functions. Below are the most classical we will use and compare in our models.

The first and the most simple kernel is the linear kernel. It doesn't perform any mapping Φ of the data in an induced feature space. This, mapping function Φ is the identity matrix I . It has to be used when the data are mostly linearly separable. Its main advantage is that it doesn't require much computation cost.

The second kernel we are interested in is the Gaussian kernel (3.10) (also called radial basis function kernel). It is also a widely used kernel as it is often used when

there is no prior knowledge about the data. It maps the instances to an infinite-dimensional space with the hyperparameter σ has to be adjusted. It is a very interesting kernel to use with SVMs as SVM is firstly a linear separator, when the problem is nonlinearly separable, it projects the data into a high dimensional space and thus there is a high chance to find a linear separation.

$$K(\mathbf{x}^i, \mathbf{x}^j) = \exp - \frac{\|\mathbf{x} - \mathbf{z}\|}{2\sigma^2} \quad (3.10)$$

There are a lot of other types of kernels as the sigmoid, Bessel function kernel or ANOVA kernel, but we will detail another interesting and widely used one: the polynomial kernel. The polynomial kernel of degree p is defined as

$$K(\mathbf{x}^i, \mathbf{x}^j) = ((\mathbf{x}^i)^T \mathbf{x}^j + 1)^p \quad (3.11)$$

It is a commonly used kernel in Natural Language Processing (NLP), the most commonly used degree is $p = 2$ as a larger degree could lead to overfitting for NLP problems.

Multi-class classification [28]

After this theoretical introduction on SVMs, let's dive into their main usage we will make of them: multi-class classification. Instead of separating 2 data classes, we will assign a label to all data for more than two classes. There are different approaches to this problem : The **one-versus-all** method trains n binary classifiers, each time with one class against all the others. The final predicted class is the one with the best classification results. The **one-versus-one** generates $n(n - 1)$ classifiers, it classifies each class with all of the others and then assigns a label following a defined strategy. Finally, there is the **truly MultiClass SVMs** which are a generalized SVM formulation for multiple classes. Among all described strategies, empirical experiments [17] conclude that there is no clearly better method than the other.

3.1.2 Random Forest (from [15])

The second model we seek is a multipurpose tool, a supervised learning algorithm that is applied both for classification and regression problems. Random forests were introduced by Leo Breiman first with the name of bagging/regression trees [8]. Random forests are defined in [8] as classifiers consisting of a set of tree-structured independent classifiers $\{h(\mathbf{x}, \Theta_j), j = 1, 2, \dots\}$ where the $\{\Theta_j\}$ are independent identically distributed random vectors and each tree decides the most popular class for each input $\mathbf{x}$.

Given a p -dimensional random vector $X = (X_1, \dots, X_p)^T$, the real-value input and a corresponding random variable response Y , and an unknown joint distribution $P_{XY}(X, Y)$, the objective of the algorithm is to find a prediction function $f(X)$ to predict Y . This function is determined by a loss function $L(Y, f(X))$ which is defined to minimize the expected value of the loss :

$$E_{XY}(L(Y, f(X))) \quad (3.12)$$

Typically, this loss function is defined as the *squared error loss* $L(Y, f(X)) = (Y - f(X))^2$ for regression and the *zero-one loss* for classification :

$$L(Y, f(X)) = I(Y \neq f(X)) = \begin{cases} 0, & \text{if } Y = f(X). \\ 1, & \text{otherwise.} \end{cases} \quad (3.13)$$

A principal advantage of random forests is that thanks to the law of large numbers, we can show that they always converge so overfitting is not a problem. Moreover, as it is a kind of "teamwork" of multiple decision trees, it gives an effect similar to the k-fold cross-validation. Minimizing 3.12 for squared error loss gives the following conditional expectation, also called *regression function* :

$$f(x) = E(Y|X = x) \quad (3.14)$$

For classification then, the minimization of 3.12 for zero-one loss gives the *Bayes rule* :

$$f(X) = \operatorname{argmax}_{y \in \mathcal{Y}} \sum_{j=1}^J I(y = h_j(x)) \quad (3.15)$$

The most frequently predicted class can be established thanks to the "base learners" $h_1(x), \dots, h_J(x)$, which combined together give the "ensemble predictor" $f(x)$:

$$f(x) = \operatorname{argmax}_{y \in \mathcal{Y}} P(Y = y|X = x) \quad (3.16)$$

where $\mathcal{Y}$ is the set of all possible values of Y . In the final form (3.16), the j^{th} base learner is a tree $h_j(X, \Theta_j)$. A summary of the method is presented in the box below.

Box 3.1: Random forests classification

Let $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$ be the training data with $x_i = (x_{i,1}, \dots, x_{i,p})^T$
 For $i = 1$ to J :

- Take a bootstrap sample $\mathcal{D}_i$ of size N from $\mathcal{D}$
- Using the bootstrap sample $\mathcal{D}_j$ as the training data, fit a tree using binary recursive partitioning (Sect. 2.1):
 - Start with all observations in a single node.
 - Repeat the following steps recursively for each unsplit node until the stopping criterion is met:
 1. Select m predictors at random from the p available predictors.
 2. Find the best binary split among all binary splits on the m predictors from step 1.
 3. Split the node into two descendant nodes using the split from step 2.

Then, to predict result on new data point x :

$$\hat{f}(x) = \underset{y}{\operatorname{argmax}} \sum_{j=1}^J I(\hat{h}_j(x) = y)$$

where $\hat{h}_j(x)$ is the prediction of the response variable at x using the j th tree

Bagging

The idea of the bagging technique, or bootstrap aggregating, is a combination of a lot of learning models to increase the overall result. Random forests build multiple decision trees and merge them to obtain more accurate predictions. Given a training set $X = \{x_1, x_2, \dots, x_n\}$ and the respective labels $Y = \{y_1, y_2, \dots, y_n\}$, bagging is repeated B times and each time, a training set is randomly sampled with replacement. Each time trees are fit to this training sample X_b, Y_b , a classification tree f_b is trained based on X_b, Y_b . Once all the bagging steps are done, we can make predictions on new data x' by taking the majority vote of all the trees.

Tuning parameters

Even if the random forest classifier already works well with standard parameters, three parameters are interesting to tune to get better results :

- m : the amount of randomly selected predictor variables chosen at each node. For classification, its standard value is $m = \sqrt{M}$, M being the number of predictors. It seems to be the parameters to which random forest is the most sensitive.
- J : The number of trees in the forest. Generalization error initially decreases when this value increases, but at a certain point, too large values of J can lead to overfitting.

- *tree size* : maximum number of terminal nodes/smallest node size for splitting. Sometimes forest of too large trees overfits and this issue can be solved by tuning one of these two parameters.

3.1.3 Multilayer Perceptron

In a certain range of situations, linear classifiers can provide extremely poor results both for regression and classification. Indeed, for small datasets, a single outlier can lead to a huge negative influence on the model. That's where comes the interest for nonlinear models, and a famous one in the machine learning community is the multi-layer perceptron (for the following, we shorten it to MLP). MLP is a type of feed-forward artificial neural network, which uses the concept of perceptron (a supervised learning algorithm for binary classification), first introduced by Frank Rosenblatt in 1958 at the Cornell Aeronautical Laboratory [31]. Combinations of perceptrons and their organisation in several layers lead to the concept of neural networks. Later, the implementation of back-propagation by David Rumelhart in 1986 [32] allowed to solve nonlinear problems and MLP became a commonly used algorithm. MLP is now used for a wide range of application like function approximation, image compression, time series forecasting but in this context, we will look at its classification abilities.

An MLP consists of a minimum of three layers of fully connected nodes (an input layer, a hidden layer and an output layer) and two layers of weights $\mathbf{w}_{ij}^k$ between them, as represented on figure 3.3.

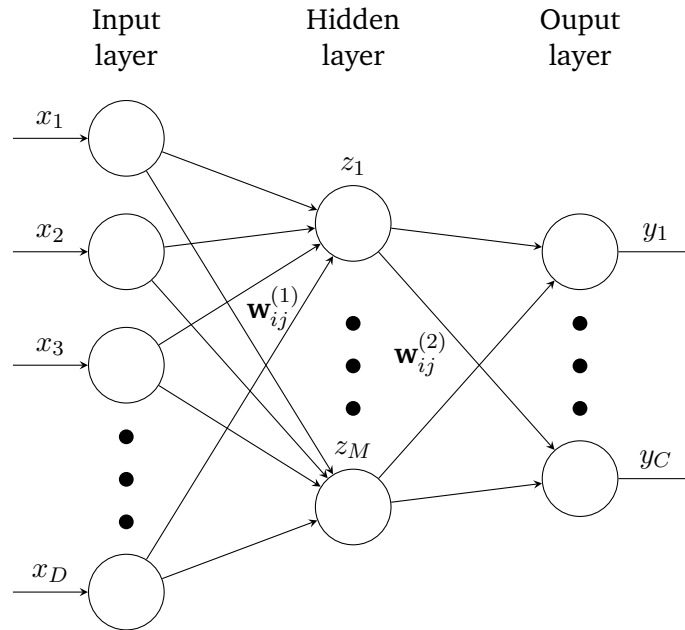


FIGURE 3.3: Example of an MLP with three layers of neurons. Inspired from the example in the MLP part of [37]

In such a model, each node is a neuron using a nonlinear activation function σ such that the output of a neuron is $y = \sigma(\mathbf{w}^T \mathbf{x})$. Each of these neurons is connected to all neurons of the next layer with a certain weight w_{ij} . The output y of an MLP can be define as a function of the input data x by :

$$y_k(\mathbf{x}) = h \left(\sum_{i=0}^M w_{ki}^{(2)} g \left(\sum_{j=0}^D w_{ij}^{(1)} x_j \right) \right) \quad (3.17)$$

where g and h can be the threshold of continuous units and g can't be linear otherwise it the system get back to a single layer situation.

Activation functions

There exist different nonlinear activation functions we can use with MLP. The 2 most common are two sigmoids : the hyperbolic tangent and the logistic function, which is used to estimate posterior probabilities :

$$y(v_i) = \tanh(v_i) \quad \text{and} \quad y(v_i) = \frac{1}{1 + e^{-v_i}} \quad (3.18)$$

Recently, deep learning development suggested to use the rectified linear unit (ReLU) as it could solve numerical problems (like the vanishing gradient) due to sigmoids:

$$y(v_i) = \max\{0, v_i\} \quad (3.19)$$

In the case of classification with continuous inputs, data become binary after the first layer.

Learning of MLP

MLP works by minimizing an error criterion on every perceptron and adjusting the corresponding weights. First, MLP defines the error criterion as

$$E = \frac{1}{P} \sum_{p=1}^P E_p(y_1, \dots, y_C) \quad (3.20)$$

with $E = (t_i - y_i)^2$

Then, the weights $\mathbf{w}$ are adjusted according to the derivatives of E . This evaluation of the weights is called *error back-propagation*. To define this process, let's first introduce the following notation : $a_i^{(l)} = \sum_j w_{ij}^{(l)} z_j^{(l-1)}$. The gradient descent term is then expressed as :

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}^{(l)}} &= \frac{\partial E}{\partial a_i^{(l)}} \frac{\partial a_i^{(l)}}{\partial w_{ij}^{(l)}} \\ &= \delta_i^{(l)} z_j^{(l-1)} \end{aligned} \quad (3.21)$$

In this equation, the term δ is to be evaluated both for output units 3.22 and hidden units 3.23 where the error δ is expressed as a combination of errors in the next layer.

$$\begin{aligned} \delta_i^{(l)} &= \frac{\partial E}{\partial a_i^{(l)}} = \frac{\partial E}{\partial y_i^{(l)}} \frac{y_i^{(l)}}{\partial a_i^{(l)}} \\ &= \frac{\partial E}{\partial y_i^{(l)}} g'(a_i^{(l)}) \quad (\text{for output units}) \end{aligned} \quad (3.22)$$

$$\delta_i^{(l)} = g'(a_i^{(l)}) \sum_k (\delta_k^{(l+1)} w_{ki}^{(l+1)}) \quad (\text{for hidden units}) \quad (3.23)$$

Then, the weight adjustment can be done thanks to the gradient descent 3.24 or a second order method like Newton's method 3.25:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \alpha \frac{\partial E}{\partial \mathbf{w}} \Big|_{\mathbf{w}(t)} \quad (3.24)$$

$$\delta \mathbf{w} = - \left(\frac{\partial^2 E}{\partial \mathbf{w}^2} \Big|_{\mathbf{w}(t)} \right)^{-1} \frac{\partial E}{\partial \mathbf{w}} \Big|_{\mathbf{w}(t)} \quad (3.25)$$

With the learning rate α that can be fixed or adaptive ($\alpha(t)$). This back-propagation process is summarised by the algorithm below.

Box 3.2: Error back-propagation algorithm (from [37])

- First, propagate a vector $\mathbf{x}^k$ through the network to activate all activations $z_i^{(l)}$ and neuron outputs $a_i^{(l)}$.
- Evaluate the error terms $\delta_i^{(0)}$ of the output layer
- Back-propagation of the error term $\delta_i^{(l)}$ to find $\delta_i^{(l-1)}$
- Compute and evaluate all the derivative terms $\frac{\partial E}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} z_j^{(l-1)}$
- Use gradient descent to update the values of the weights

As a solver of the differential equation, there are also different options but the most appropriate to the data of this thesis is surely an optimizer in the family of quasi-Newton methods (lbfgs from the sklearn package) as it converges faster and performs better for small datasets (800 data is a small dataset in the machine learning universe). This quasi-Newton method is a version of the Newton method where the Hessian matrix is approximated. This idea has been confirmed by experiments: models generated with a stochastic gradient descent solver quickly reached the maximum iteration numbers before converging.

Main parameters

The first parameter we can tune is the number of layers of the MLP. Generally, we use 2 layers of weights as the universal approximation property tells that a two-layer MLP can approximate arbitrarily well any function, provided a sufficient number of hidden layers M . However, large values of hidden layer units M can also lead to overfitting. This overfitting can be avoided by a pertinent choice of the learning rate α . We can also, as explained earlier, choose an activation function, we will use the three functions mentioned above in subsection 3.1.3.

3.2 Model selection

First, for all of the three models we will test, we have to select the best parameters for the model. To make this choice, we can perform a grid search: we arbitrarily select some parameters and some pertinent values of these parameters and we generate the model several times for all combinations of parameters [10]. Indeed, the

execution time of a grid search grows exponentially with the number of tested values so we have to think about the most relevant combinations for our research. Here, we perform this selection thanks to the *scikit-learn* python package ¹.

The table below summarizes all the parameters tested per model.

Classifier	Parameter	Values
Support Vector Machine	C from equation 3.7	$C \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$
	Kernel	{linear, polynomial, rbf}
Random Forest	m	$\{\sqrt{M}, \log_2(M)\}$
	J	{10, 50, 100, 200}
	tree size	{1, 2, 5, 10}
MLP	hidden layer sizes	{(50, 50), (50, 100, 50), (100)}
	activation function	{ logistic, tanh, ReLU }
	α	0.0001, 0.01, 0.05

TABLE 3.1: Parameters tested for model selection

Once the grid search has selected the best combination of parameters for the model, this model is then fed with new unknown data and we can assess its performances.

3.2.1 Performances evaluation

We will detail the method we use to test the performances of the models. First of all, we randomly shuffle all the dataset and their corresponding list of labels (here the associated number of the task $1 \rightarrow 8$). To be sure our model can generalise to any new sample of data we could give it, we have to divide our database into two parts: a training set, that will be used to train the model allowing it to fit the data, and a test set, that will serve to test how precisely the models classify the data. It is important not to use all the dataset for training as it could result in overfitting and biases in our results and not well represent the behaviour of the model on future data.

To generate our model, we divide the training set into 5 parts of equal size, taking one of them as a validation set and the other 4 as a learning set. The model is generated with some parameters, the error is computed thanks to the validation set. We then repeat 5 times this experiment and take the average error. This process is known as 5-fold cross-validation [10, 14] and is represented on figure 3.4.

For the test phase, we also perform 5-fold cross-validation. We alternatively feed the trained model with a test set and compute the classification error. Repeating test phases and taking the average of the results is essential to get sufficiently robust results. Indeed, one single execution of a test phase could lead to unusually deviating results that could not correspond to the average values returned if the experiment is repeated.

¹https://scikit-learn.org/stable/modules/grid_search.html

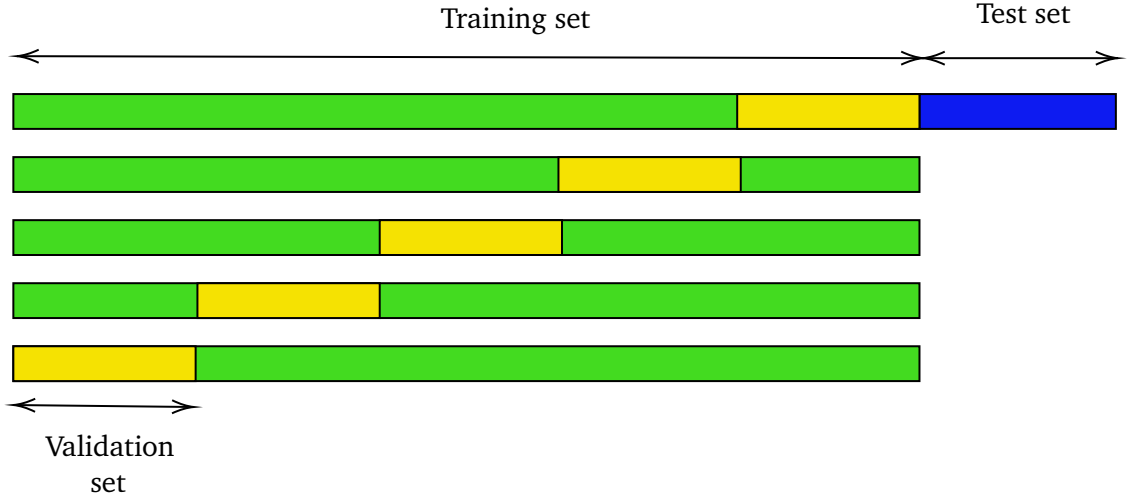


FIGURE 3.4: Representation of the train/test split and the 5-fold cross-validation

3.2.2 Choice of a performance metric

There exist plenty of metrics to evaluate classification results. One of the most commonly used tools to visualize classification is the confusion matrix. This is a squared matrix of size equal to the number of different classes in the data, where the rows and columns represent the predicted and actual classes, as explained in [2]. Such a matrix is represented in table 3.2. In this example, the number of classes is only 2, it's a binary classification, an element can either be classified as positive or negative. This simple example is a good base to illustrate the different metric concepts we will discuss. Figure 3.3 is a generalization of the confusion matrix for cases where we have more than two classes (let's say G classes) in which classifying our data. In the notations of this matrix, the number of samples is n , n_g is the number of samples belonging to the class g and n'_g the number of samples predicted in the class g .

	Actual : positive	Actual : negative
Classified : positive	True positive (TP)	False positive (FP)
Classified : negative	False negative (FN)	True negative (TN)

TABLE 3.2: Binary confusion matrix

		Predicted class				
		1	2	...	G	
Actual class	1	c_{11}	c_{12}	...	c_{1G}	n_1
	2	c_{21}	c_{22}	...	c_{2G}	n_2
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
	G	c_{G1}	c_{G2}	...	c_{GG}	n_G
		n'_1	n'_2	...	n'_G	n

TABLE 3.3: Confusion matrix from a dataset of G classes

The confusion matrix encodes the number of both correct and incorrect classifications of our model for each class. From this matrix, we can establish a certain

number of class indices like accuracy, sensitivity, precision, recall, and some more. Each of these indices should not be considered singularly to assess the prediction results of our model as they all describe a certain aspect of the classification [2]. The best solution is always to compare several of these values to be more confident of the strength of the quality of the predictions.

Among all these measures of the classification estimation, **accuracy** (Acc) is the most commonly used, surely because it is one of the most easily understandable. It is also known as the overall success rate. It corresponds to the proportion of well-classified samples (diagonal elements of the matrix c_{ii} from table 3.3, with N classes) among all samples classified by the model ($n = \sum_{i=1}^G \sum_{j=1}^G c_{ij}$) :

$$Acc = \frac{\sum_{i=1}^N c_{ii}}{n} \quad (3.26)$$

Or in the binary case :

$$Acc = \frac{TP + TN}{TP + FP + FN + TN} \quad (3.27)$$

Accuracy takes values between 0 and 1 (1 being the perfect classification result) and is used both for binary and multiclass classification. This coefficient is perfectly suited when the classes are well balanced, which is the case here as all classes are composed of 100 samples each. Its complementary is the misclassification error, which is the proportion of samples classified in the wrong class.

Then, the precision is the proportion of samples that are really positive among samples classified as positive. It tells us precise results for each class, high precision means the classifier rarely classify samples in a class if it shouldn't be :

$$Precision = \begin{cases} \frac{c_{gg}}{n_g} & \text{for a class } g \text{ in the multiclass case} \\ \frac{TP}{TP+FP} & \text{for the binary case} \end{cases}$$

There is also the Recall, or sensitivity in the multiclass case, which indicates the proportion of samples classified in a certain class among all samples of this class. It measures the ability of a classifier to correctly classify samples of a certain class.

$$Sensitivity_g = \frac{c_{gg}}{n_g} \quad \text{for a class } g \text{ in the multiclass case}$$

$$Recall = \frac{TP}{TP + FN} \quad \text{for the binary case}$$

The three indices detailed above each have their strength and their drawback, the drawback of the accuracy being small as the classes are identically distributed. But precision and sensitivity can give quite different results for the same confusion matrix.

Here is why we introduce the **F1-score** (also called F-score or F-measure), which is defined as a harmonic mean of the Precision and the sensitivity as explained by Cuadros-Rodriguez in [14] :

$$F1\text{-score} = 2 \times \frac{Precision \cdot Sensitivity}{Precision + Sensitivity} \quad (3.28)$$

This F1-score conveys the balance between sensitivity and precision. Indeed, if one of these 2 metrics is lower than the other, the F1-score tends to be closer to the lowest value. A high F1-score is therefore a good quality criterion for classification. In the case of multilabel classification, an F1-score can be computed for each class based on the definition of the precision and sensitivity of the class. The mean of these results is then computed to obtain the overall F1-score. This is called the macro-average F1-score, which differs from the micro-average, less robust to unbalanced classes, which first computes the average of the precision and recall before computing f1-score. The macro-average F1-score will be used here along with the standard deviation between classes.

There exist other interesting coefficients used for classification models, like **Cohen's kappa** that Ben-David compares to ROC in [4], but it is best suited for unbalanced classes, which is not the case in our current work. We will then focus our analysis on the **accuracy** and the **F1-score** of the different classifications.

3.3 Projection and dimensionality reduction methods

When we have access to huge databases, we want to quickly get information on those, but at a first sight, it is impossible to retrieve information by looking at the data. Here comes the interest in visualisation and projection methods. Projecting data into a lower-dimensional space allows us to understand the links between the data, analyse their distribution, the presence of possible clusters or outliers and other interesting features to analyse. It can also be useful to do some feature selection when we face datasets with too many features for the model we are using.

Two projection and one feature selection methods will be used in this work. The first one will serve to reduce the dimensionality of the time series of our database while keeping the most discriminant features to optimize classification results. Another will be used to make projections into a two-dimensional space to visualise the clusters present in our data.

3.3.1 Principal component analysis

Principal components analysis (as explained in [37]) is one of the oldest and the most used tool of data analysis. It is useful in lots of fields like data processing for data decorrelation or whitening, multivariate orthogonal linear regression and dimensionality reduction, which is the main application of PCA we will use in this thesis.

To understand what PCA does to the data, let's consider a matrix-like dataset of M rows and N columns (or N observations of M dimensions) $\mathbf{X} = [x_k]_{1 \leq k \leq N}$. The goal of PCA is to find new decorrelated variables thanks to some linear transformations, keeping only variables with the highest variance and minimizing the loss of information after backpropagation to the initial system of coordinates. Mathematically, PCA is an orthogonal linear transformation that maps the data into a new coordinate system where the first coordinate (also called principal component) corresponds to the axis of highest variance, the second coordinate corresponds to the second-highest variance and so on. This purpose makes PCA an interesting tool for dimensionality reduction as it can select dimensions that have the highest variance, which are the

most discriminant features in the new feature space.

Dimensionality reduction with PCA

Let U an orthogonal squared matrix of size $M \times M$, we define a matrix $U_P = [u_i]_{1 \leq i \leq P}$ with $P \leq M$. $U_P^T U_P = I$ remains true. The product $\mathbf{y} = U_P^T \mathbf{x}$ projects vector $\mathbf{x}$ from space R^M to R^P . We can then say that the matrix U_P^T is a linear transformation that reduces the dimension of a vector.

If we consider the case $P = 1$, as $\mathbf{X}$ has zero-mean, the variance along u can be expressed as

$$\hat{\Sigma}_y = \frac{1}{N} \mathbf{Y} \mathbf{Y}^T = \frac{1}{N} (u^T \mathbf{X}) (u^T \mathbf{X})^T = (u^T \hat{\Sigma}_x u) \quad (3.29)$$

then, the vector associated to the maximal variance is solution to the maximization problem for a symmetric matrix A :

$$\begin{aligned} \max_{\mathbf{x}} \mathbf{x}^T A \mathbf{x} \\ \text{s.t. } \mathbf{x}^T \mathbf{x} = 1 \end{aligned} \quad (3.30)$$

If we perform an eigenvalue decomposition (EVD) of the variance, we get $\hat{\Sigma}_y = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^T$, then the variance $\mathbf{Y}$ reaches its maximal value with $\mathbf{u} = \mathbf{v}_1 = \lambda_1$, which is the biggest eigenvalue. For $P \geq 1$, V_P can get the P -dimensional linear subspace that keeps the maximum fraction of variance, which is expressed as

$$\frac{\sum_{i=1}^P \lambda_i}{\sum_{i=1}^M \lambda_i} \quad (3.31)$$

Once we reduced the dimensionality of the data, we can still send back the data sample $\mathbf{Y}$ into the original space with back-propagation :

$$\tilde{\mathbf{Y}} = \mathbf{V}_P \mathbf{Y} = \mathbf{V}^T \mathbf{V} \mathbf{X} \quad (3.32)$$

The goal of the PCA projection and back-propagation is to minimize the Frobenius norm $\|\mathbf{X} - \tilde{\mathbf{X}}\|_F$. To summarise the idea of dimensionality reduction with PCA, we can say that $\mathbf{Y}$ is the P -dimensional linear projection that best reconstructs the original data $\mathbf{X}$.

3.3.2 Recursive feature elimination

Recursive feature elimination (RFE) (developed by Guyon et al. in [20]) is a powerful supervised feature selection method that works by recursively removing less meaningful features and building again models on the remaining features. At each iteration, this procedure is trained with a given training set. After this training phase, features are ranked according to the accuracy metric. The best features are chosen based on the ranking provided by the training phase and the remaining ones are ignored. Then, the process starts again until we reach the desired number of features. For each iteration, the number of features to be removed can be a fixed value or a percentage of the remaining set of features. A disadvantage of this method is that it doesn't stop removing features if the accuracy begins to drop. We must then process

incrementally when dropping features to avoid doing worse than better.

However, this method also has strong advantages. An important one is that it doesn't map the data into another feature space to reduce dimensionality. Features are recursively dropped at each step and we can visualize the most discriminant features and at each step, we can see less important features as they are ranked. This is useful to put a name on the particular features responsible for the classification. In this context, it can be used to identify brain ROI responsible for task identification among the others.

3.3.3 t-distributed stochastic neighbor embedding

This method, which we call t-SNE, was created in 2008 by van der Maarten and Hinton [26] and is currently used for a large variety of researches like music analysis, bioinformatics, research on cancer, or biomedical signals treatment. It is a dimensionality reduction technique used to visualise data of very high dimension $\mathcal{X} = \{1, \dots, n\}$ in a two or three-dimensional space, where the data can then be visualised as a points cloud in a scatter plot. The algorithm works by looking for an optimal configuration according to information theory criteria that respect relative positions between points: two points close from each other in the original space must have a similar relative position in the reduced space, and so with more distant points. t-SNE can capture most of the local structure of the high-dimensional data while making it easier to see global structure such as the presence of clusters at several scales.

This algorithm is based on the previous Stochastic Neighbor Embedding (SNE) method. It works by first transforming high dimensional Euclidian distances into conditional probabilities that represent similarity :

$$p_{j|i} = \frac{\exp -||x_i - x_j||^2 / 2\sigma_i^2}{\sum_{k \neq i} \exp -||x_i - x_k||^2 / 2\sigma_i^2} \quad (3.33)$$

Where the parameter σ_i is given by a binary search that computes a P_i with a perplexity $Perp(P_i) = 2^{H(P_i)}$ set by the user (H being the Shannon entropy detailed further in part 4.2.1). Similar conditional probabilities are also set for low-dimensional counterparts y_i, y_j of the high-dimensional datapoints x_i and x_j :

$$q_{j|i} = \frac{\exp -||x_i - x_j||^2}{\sum_{k \neq i} \exp -||x_i - x_k||^2} \quad (3.34)$$

Remark that as only pairwise similarities are interesting, the terms $q_{i|i}$ and $p_{i|i}$ are set to 0. Then, a measure of the quality with which $q_{j|i}$ models $p_{j|i}$ is expressed with the Kullback-Leibler divergence that has to be minimized thanks to a gradient descent method. The cost function of the modelling is :

$$C = \sum_i KL(P_i || Q_i) = \sum_i \sum_j p_{j|i} \log \left(\frac{p_{j|i}}{q_{j|i}} \right) \quad (3.35)$$

where P_i and Q_i are the respective conditional probabilities over all other datapoints x_i and all other map points given map point y_i . The gradient of this cost with respect to variables y_i has to be minimized.

t-SNE is a variant of SNE that differs in two ways : (1) t-SNE uses a Student-t distribution instead of a Gaussian to compute similarities between points. (2) it uses

a symmetrized version of the original SNE cost function C that makes the gradient easier to compute.

$$C = \sum_i KL(P||Q) = \sum_i \sum_j p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) \quad (3.36)$$

with pairwise similarities becoming

$$q_{ij} = \frac{\exp -||x_i - x_j||^2}{\sum_{k \neq l} \exp -||x_l - x_k||^2} \quad (3.37)$$

$$p_{ij} = \frac{\exp -||x_i - x_j||^2 / 2\sigma_i^2}{\sum_{k \neq l} \exp -||x_l - x_k||^2 / 2\sigma_i^2} \quad (3.38)$$

But as the expressions of p_{ij} can lead to problems when facing outliers, we decide to set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$. t-SNE then uses the Student-t distribution to define the distribution of points x_i in the low dimensional map :

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l} (1 + ||x_l - x_k||^2)^{-1}} \quad (3.39)$$

Once those 2 expressions are fixed, the two final steps of the algorithm are the computation of the gradient of the Kullback-Leibler divergence between P and Q (3.40),

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1} \quad (3.40)$$

and finally, the update of the set of low dimensional points $\mathcal{Y}$:

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\partial C}{\partial \mathcal{Y}} + \alpha(t)(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}). \quad (3.41)$$

This updating step is repeated a number of times T fixed by the user. Each time, the term q_{ij} (from equation 3.39) and the gradient (3.40) are updated too. In addition to the complexity parameter, the learning rate η and the momentum $\alpha(t)$ are parameters users can tune to get better visualisation results.

Chapter 4

Classification of tasks from BOLD fMRI

This chapter will constitute the core of the interest of this master thesis. With several methods, we will try to maximise the accuracy and f1-score of our task classifications in order to reach a model whose performance are the best we can get with the processes we are going to use.

At first, we will focus on the data we have in our hands to perform this task, what it contains, what is possible to do with these data and how it will be used for our research. Then a focus on the three main methods used will detail all the data processing, the model selection for each case. The results of these three methods will then be compared and discussed to determine the most accurate way to classify the tasks for each brain.

4.1 Dataset

The start of all machine learning project is to understand the available data, its features, its dimension and its limitations. Here, a set of fMRI BOLD signals of 100 subjects has been retrieved from the Human Connectome Project [35]. The subjects who participated to the experiments are composed of 54 females 46 males between 22 and 36 years old.

For each of these 100 subjects, recorded data of 2 sessions of resting state (RS) fMRI and seven other different tasks (task-fMRI) are available. These tasks (and their abbreviations in brackets) are labeled as emotional processing (EMO), gambling (GAM), language processing (LAN), motor task (MOT), relational processing (REL), social cognition (SOC) and working memory (WM)).

These data comes to the form of pre-processed and normalized fMRI time series associated to every ROI. All of these time series are in the same range of values and are mean-centered. Every task specific time series has its own size, ranging from 176 for the emotional processing task to 1200 for the resting state, which need longer acquisition sessions. The overall dimensions of this database is 8 tasks $\times$ 100 subjects $\times N_{ROI}$ Regions of interest $\times N_{\Delta t}$ time series values.

The two available brain parcellations are described in section . The Desikan-Killiany with 84 zones or the Destrieux atlas with 164 zones. Both have interests and drawbacks. A first reasoning would be to prefer the Desikan-Killiany atlas to reduce computation cost in the different classification methods, but when it is possible,

comparing the results on these 2 datasets could reveal an unexpectedly clear interest for one of these two atlas. If relevant results are obtained using Destrieux atlas, both results will be compared but the based of the current study will focus on the DK parcellation.

4.2 First method : Functional connectome

As explained above in section 4.1, we have access to a huge amount of data (more than 1GB of matrices). The challenge will be to find techniques to handle these the best way, reducing its size while trying to keep the most interesting information it can contain. Then, after appropriate manipulations, these data will feed our different models and we will compare the results of the methods and discuss about them. As a reminder, this huge amount of data represents the following information : for 100 people, we have access to data from 8 activities and for every zone of the selected atlas (Region of interest of the brain), we have between 176 and 1200 regional BOLD time series.

A first approach would be to construct a matrix of the pairwise association between all time series of the zones (method used by Bullmore and Sporns in [11]). This is a matrix where association between nodes can be quantified by many continuously variable metrics but we prefer to call it functional connectome. These functional connectomes are then represented as a squared symmetric matrix of size $N \times N$ where N is the number of zones, or regions of interest (ROI) of the brain atlas. This method is the most commonly used for fMRI analysis (method also used in [1, 12]). With this method, we are not interested in the nodes of the networks but in the similarity coefficient of the edges between nodes.

4.2.1 Different node association measures

The coefficients of our functional connectome represent the weights of the edges of the induced graph. It can be computed with several similarity measures. The choice of the computing method of this FC can lead us to different quality of results, as some coefficients could retrieve less information between two time series. Here we compare 2 main coefficients :

Pearson's correlation coefficient

This coefficient, also called bivariate correlation, measures the linear correlation between two sets of data or two random variables. Mathematically, it is computed as the covariance of the two random variables divided by the product of their respective standard deviation (explained in the theoretical background of [16]) :

$$\rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \quad (4.1)$$

where X and Y are two random variables and σ_X, σ_Y are their respective standard deviations. Its values range between -1 and 1 , -1 being the extreme case where data is oppositely correlated, meaning that a raise of the first variable induces a diminution of the value of the second variable, 0 meaning no linear correlation and 1 the opposite case which would mean perfect correlation (both extreme being unrealistic in practice).

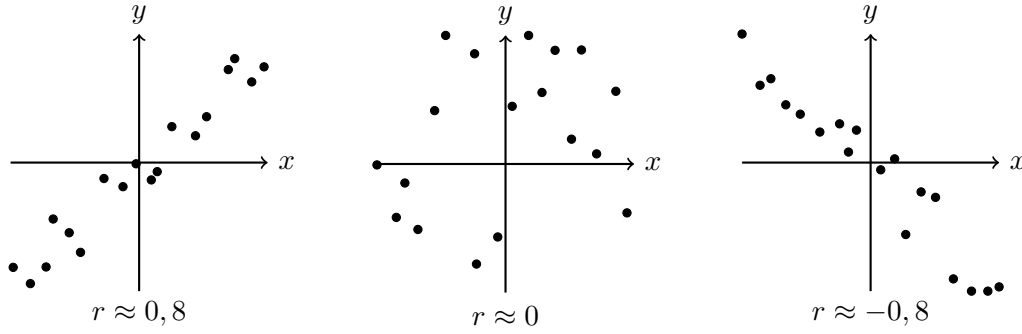


FIGURE 4.1: Examples of Pearson correlation coefficients between two datasets

Mutual Information (MI) between time series

Mutual Information¹(based on [13]) is a well-known measure of dependence in information theory. Here, we are interested in MI between time series of all the regions of interest, which is another tool to build our functional connectome. It measures the amount of mutual dependence between two time series. It tells us the amount of information we can get on a signal by observing another one. It is closely related to the concept of Shannon entropy, another important concept in information theory which can be seen as the measure of uncertainty of a random variable. Its equation in the discrete case is :

$$H(X) = - \sum_{x \in \mathcal{X}} p_X(x) \log(p_X(x)) \quad (4.2)$$

where p is the probability mass function of the random variable X . Based on this definition, the MI can be expressed as² :

$$I(X, Y) = \sum_{x, y} p_{XY}(x, y) \log \frac{p_{XY}(x, y)}{p_X(x)p_Y(y)} \quad (4.3)$$

$$= H(X) - H(X|Y) \quad (4.4)$$

$$= H(Y) - H(Y|X) \quad (4.5)$$

$$= H(X) + H(Y) - H(X, Y) \quad (4.6)$$

Where $H(X, Y) = - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_{XY}(x, y) \log(p_{XY}(x, y))$ is the discrete joint entropy of X and Y . The links between the different entropies and mutual information is depicted on the Venn diagram of figure 4.2 from page 22 of [13].

¹We only focus on the discrete formulation as we won't use the continuous one

²details of the development can be found on page 21 of [13]

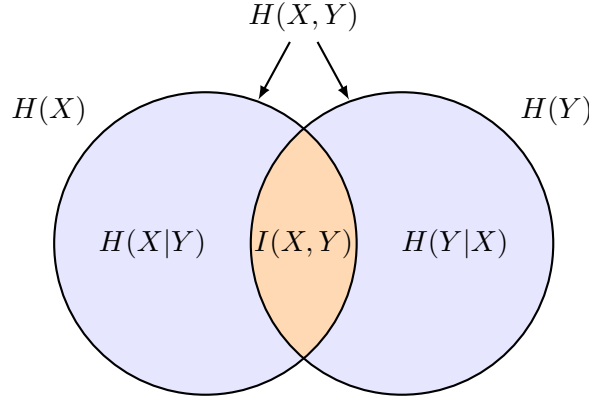


FIGURE 4.2: Venn diagram explaining the links between entropy and mutual information

It can be interpreted as the reduction of the uncertainty of X due to the knowledge of Y . Moreover, it is a symmetric operator, which means X tells us as much on Y than Y on X . Unlike the Pearson coefficient described above, the mutual information takes values from 0 (which indicates that the two variables X and Y are independent) to the infinity. The higher is the MI, the more information is shared between the variables.

However, a major problem for both entropy and mutual information with time series is that it is often difficult and sometimes impossible to compute in practice, especially for high-dimensional time series. Indeed, in the case of large scale time series of real values, each value is present only once in the set of values and so their respective probability of presence is extremely low and common to all values. To overcome this problem, there exists a variety of approximation methods, each coming with its strength. However, evaluating $I(Y, X)$ is often very difficult. Most of the principal methods like histograms or splines suffer from the curse of dimensionality, which is a problem faced with the fMRI BOLD time series.

First, **approximate entropy (ApEn)** was a method introduced to handle entropy approximation, which was before not well suited to study noisy dynamical systems time series. ApEn was a set of measures closely related to the entropy easily applied to time series but lead to inconsistent statistical results. This was because ApEn suggested more similarity than there really was. Then, a new complexity measure was introduced : the **Sample Entropy (SampEn)**, described by Richman and Moorman in [30]. Its main advantages is that it reduces the bias introduced by self-matching in ApEn and it maintains relative consistency.

Mathematically, the sample entropy is defined as

$$SampEn = -\log \frac{A}{B} \quad (4.7)$$

where :

- $X_m(i) = x_i, x_{i+1}, \dots, x_{i+m-1}$
- A = number of template vector pairs such that the Chebychev distance d meet $d[X_{m+1}(i), x_{m+1}(j)] < r$

- B = number of template vector pairs such that the Chebychev distance d meet $d[X_m(i), x_m(j)] < r$

For a given embedding dimension m , a tolerance r and a number of data points N . In the current research, as we are interested in fMRI data, empirical results on resting state data suggested to used parameters $m = 3$ and $r = 0,6\sigma_X$ (std of the data). Wang Et al. found in [38] that these coefficient obtained best results to differentiate signal from noise while retrieving maximal information. Now we have a good approximation of the entropy, we can easily compute the MI with formula 4.6. A main downside of this method is its complexity, it is much too slow to generate 800 functional connectomes with this way to compute MI.

Another approach to approximate the MI for high-dimensional time series is to use a **k-nearest neighbor (k-NN) based estimator**. Such an estimator has been developped by Kraskov, Stögbauer and Grassberger in 2004 [24] and is now widely used under the name of KSG method. Its principle is based on the ranking of the points $(x_i, y_i) = z_i$ according to the maximal distance to their neighbors, where any norm could be used.

$$||z - z'|| = \max\{||x - x'||, ||y - y'||\}$$

The distance between z_i to its k^{th} neighbor is denoted by $\epsilon(i)/2 = \max\{\epsilon_x(i)/2, \epsilon_y(i)/2\}$ where $\epsilon_x(i)/2$ and $\epsilon_y(i)/2$ are the distances between the same points projected into the X and Y subspaces. Then we count the number $n_x(i)$ of points x_j that are at a distance from x_i inferior to $\epsilon_x(i)/2$ and same for y instaead of x . These assumption lead us to the following formulation of the KSG estimator of the MI, with N the size of the vectors, k the number of neighbors to consider and ψ is the digamma function $\psi(z) = \frac{\Gamma'(z)}{\Gamma(z)}$.

$$I(X, Y) = \psi(k) + \psi(N) - \langle \psi(n_x + 1) + \psi(n_y + 1) \rangle \quad (4.8)$$

The MI approximation computed here will be generated thanks to the pybilt package [40], which implement a variation of the KSG method with distances computed with unit balls instead of the maximal norm computed in the original algorithm.

MI is quite different than Pearson correlation, unlike the cosine similarity, which would give too close results from this first one. It will then be interesting to study the impact of a different coefficient metric to generate weights of the edges of our functional connectome. At which point will it influence the classification performances ?

4.2.2 Vectorization of the functional connectomes

Once we have formed our functional connectomes for each subject and each task, our database is represented as a list of matrices as follows :

$$X = [FC_1^1, FC_2^1, \dots, FC_n^1, \dots, FC_1^8, FC_2^8, \dots, FC_n^8] \quad (4.9)$$

Where the superscripts 1, 2, ..., 8 indicates the task ID and the indices refer to the different subjects. This format is not convenient in order to perform classification methods on it. indeed, we still have 3-dimensional data and we wish to have a set of

vectors, or in machine learning terms, a set of elements and a set of features. To overcome this problem, we are going to linearize these matrices by concatenating all rows of the matrices to form large dimension feature vectors, also called a bag-of edges (as denoted in [39, 12]). As the Functional connectomes are symmetric, we can only take the upper triangular parts of matrices to obtain vectors of size $n(n-1)/2$ where n is the number of brain parts of the atlas (voxels).

$$X_{linearized} = \begin{bmatrix} x_1^1 \\ x_2^1 \\ \vdots \\ x_{100}^1 \\ \vdots \\ x_1^8 \\ \vdots \\ x_{100}^8 \end{bmatrix} \quad (4.10)$$

Finally, with this first method, we have to classify 800 FC vectors (8 tasks for 100 subjects), each of size 3486 for the DK atlas and 13366 for the Destrieux atlas, which in both cases is a much higher values than the amount of subjects per class of task. But will it be a problem for our research ?

4.2.3 Curse of dimensionnality

The main consequence of these high dimensional vectors is the risk of overfitting, a famous phenomenon in machine learning [3]. Overfitting is a phenomenon occuring when the amount of features of our data is very large. This makes the model fit so well the data that when we give it a new unknown sample, it can not classify them well. The model is so well fitted for the training data that it can not generalise anymore. This situation is represented³ on figure 4.3 where the green line represents an overfitted model while the black one is a more regular one. We can figure out with this figure than if we fit a new set of data, if we assume that it will be equally distributed, the amount of misclassified data (red/blue) will be larger with the green model than with the black one.

This phenomenon can be avoided thanks to feature selection/elimination methods that can either be supervised or unsupervised [22]. In our particular case, as we know every correspondence between connectomes and tasks, we can either use supervised feature selection methods like RFE, or unsupervised dimensionality reduction methods like PCA.

³Image from <https://fr.wikipedia.org/wiki/Surapprentissage#/media/Fichier:Overfitting.svg>

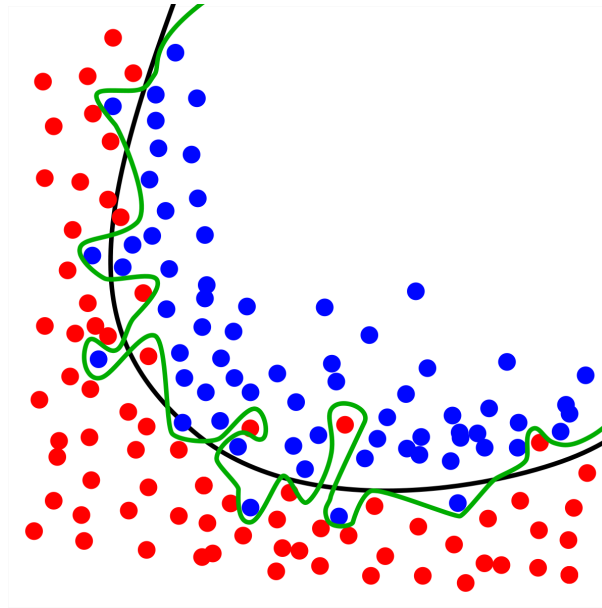


FIGURE 4.3: Representation of a case of overfitting

Box 4.1: Summary of method 1

- In the beginning, we have time series recordings for each region of interest of the brain
- Then, compute the Functional connectome based on a chosen association metric between nodes.
- Take the upper triangular part and turn it into a feature vector.
- (Optional) Dimensionality reduction with PCA or RFE
- Feed the models with these feature vectors.

4.2.4 Results

Now our first method is well described, we can begin to generate our models and test this method on the data described above [4.10](#). First of all, we perform model selections for each type of classifier, whose different combinations of values are listed in [table 3.1](#). Their final values are compiled in the [table 4.1](#).

The final results of the classifiers fed with the data processed with method 1 and without any feature selection are summarized in [table 4.2](#) :

Impact of the association metric

A first remark when observing the results ([4.2](#)) is the clear advantage of using the Pearson correlation coefficient as association metric between nodes to build the functional connectomes. For both of the three classifiers, the models generated perform better when the edges of the connectome forming the feature vector are computed as a correlation between the time series.

Classifier	parameter	value (Pearson)	value (MI)
SVM	Kernel	linear	rbf
	C	0,01	10
RF	m	$\sqrt{M}$	$\sqrt{M}$
	J	100	100
	tree size	5	2
MLP	hidden layer size	(100)	(50, 100, 50)
	activation function	tanh	ReLU
	α	0,05	0,0001

TABLE 4.1: Selected values of the models parameters after a grid search cross-validation for connectomes built with Pearson correlation and mutual information.

	Pearson		
	SVM	RF	MLP
Accuracy (%)	85,6 ± 2,47	76,32 ± 3,51	84,25 ± 2,14
f1-score (%)	84,56 ± 11,21	75,66 ± 15,83	83,89 ± 11,41
	MI		
	SVM	RF	MLP
Accuracy (%)	73,17 ± 3,9	73,65 ± 4,61	46,7 ± 4,14
f1-score (%)	72,217,99	72,37 ± 17,13	45,72 ± 22,26

TABLE 4.2: Results of the different classifiers applied to the data of the first method 4.2.3 for functional connectomes built with Pearson correlation and MI between nodes.

Another observation is the differences between the best hyperparameters in the two cases. This can be explained by the smaller differences of values between features when using the MI which may not be the best metric to describe the relations between the ROIs of the brains based on high-dimensional time series. This lack of variance between mutual information of time series is illustrated by the example below (Figure 4.4). The connectome built with Pearson correlation seems to have more discriminating potential than the other built with MI. The values of the edges (off-diagonal values) on functional connectome (A) have a standard deviation of 15,66% of the range of possible values while edges values of MI-based functional connectome (B) have standard deviation of 8,88%.

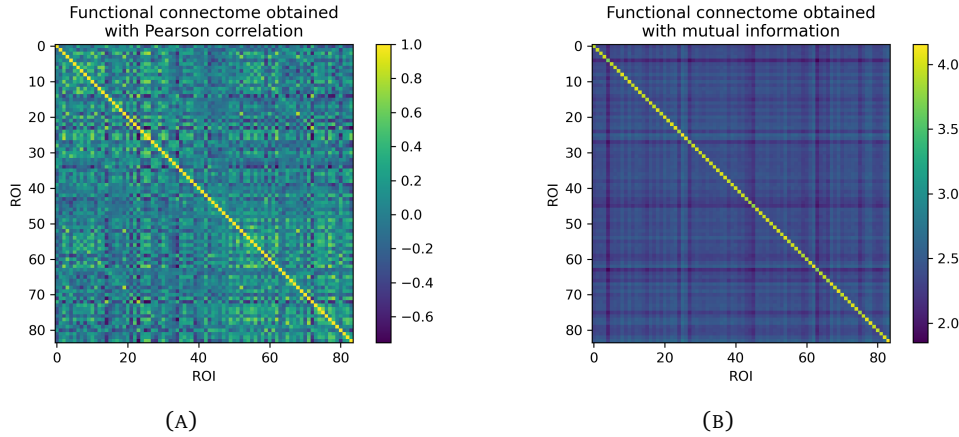


FIGURE 4.4: Examples of functional connectome based with Pearson correlation (A) and mutual information (B) between ROIs signals

Best classifier

The averaged results of the experiences repeated 5 times each confirms the fact that support vector machines are best suited for datasets with a very high dimensional space, which confirms experimental studies which compared SVM and MLP [42]. It produces the best accuracy and F1-score, with a minimum standard deviation. Moreover, it is noticeable that the training was extremely efficient, the time needed to train the models was really quick compared to MLP, the second-best classifier, which produces nearly as good results but the training phase was extremely costly and took a lot of time. Moreover, for a certain combination of the hyperparameters, it was frequent that the solver of the MLP did not converge after the maximum number of iterations we set. Finally, the results of the random forest classifier are less good but still give decent classification results even if in theory, it is not the best option of classifier for such an amount of features.

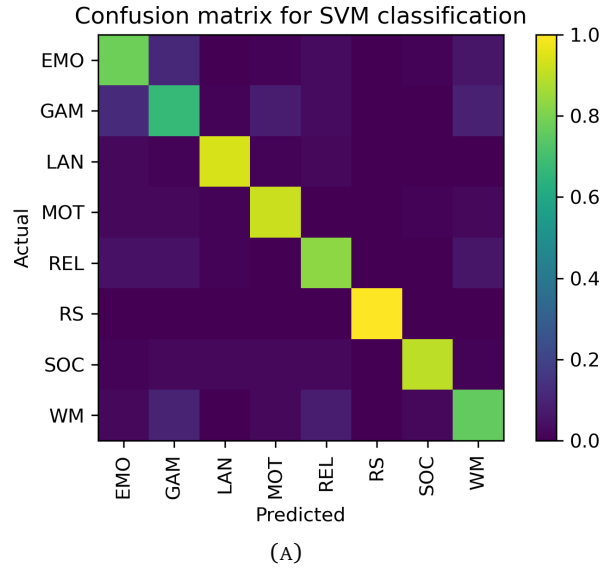
Label-specific results

Beyond the overall satisfying results of the SVM classifier, it is interesting to note the different results obtained for each class and detailed in figure 4.9a. Indeed, some tasks are way more easily classified by the SVM than others. The clearest observation is the resting state, which is always well classified and none of the other tasks is misclassified in resting-state. This result confirms the assumptions we discussed in 2.3, where we underlined the difference between RS from the other tasks. Indeed, fMRI signals of resting-state are weaker than those obtained while the subject is performing a task and the correlation between ROI signals are more similar. A discriminating factor of resting-state is its average correlation between ROI of 0.0029 while edges of task functional connectomes have average values between 0,01 and 0,016.

Besides particular results obtained with resting state, three tasks obtain relatively good results: language processing, motor task and social cognition. For these tasks, functional connectomes must have more discriminating features, which means clearer patterns of correlations between certain zones.

Finally, the gambling task obtained the worst results with this method (69%). This could potentially be explained by the different reactions subjects could have

during the experiments due to the nature of this experiment, described in [25], where subjects play a card game. The subjects may have a relatively different activity, which could result in non-recurrent activity patterns between subjects.



Metric	EMO	GAM	LAN	MOT	REL	RS	SOC	WM
Sensitivity (%)	78	69	93	90	79	100	90	83
f1-score (%)	78	69	93	89,55	81,03	100	93,26	78,67

(B)

FIGURE 4.5: (A) Confusion matrix resulting from the the classification of the Pearson correlation based FC by the SVM classifier. (B) Table of the classification results for each class

Impact of dimensionality reduction

The experiments were performed again with the data of this first method, but this time, after reducing the dimensionality with two methods, to figure out the potential interest of reducing the dimensionality of our data.

First, data was mapped to another feature space of dimension 100 with PCA. This method completely changes data and keeps dimensions with the highest variance. The reduced dimension was set to 100 after testing the classifiers with a grid of different values between 20 and 1000. The results we got here 4.3 are a little bit counterintuitive. We usually expect that reducing the number of features lead to better results, but it seems that PCA projection to a feature space of dimension 100 leads to slightly worse results. We can also notice that SVM and MLP still got close and reasonable results.

However, the effect was the opposite while using a recursive feature elimination method. With RFE, the best results were obtained with a selection of the 500 most relevant features, or the most meaningful correlations between nodes. Results of SVM and MLP are still the best among the models tested and their results both increase by 5% compared to classification with all data. We can then say that an appropriate RFE

process has a positive impact on the classification of the tasks.

	PCA (reduction to 100 features)		
	SVM	RF	MLP
Accuracy (%)	81,27 $\pm$ 3,1	69 $\pm$ 2,69	81,02 $\pm$ 1,97
f1-score (%)	81,13 $\pm$ 13,48	68,3 $\pm$ 18,4	80,81 $\pm$ 13,34
	RFE (500 features kept)		
	SVM	RF	MLP
Accuracy (%)	90,52 $\pm$ 2,02	79,8 $\pm$ 2,38	89,65 $\pm$ 2,22
f1-score (%)	90,24 $\pm$ 8,24	79,29 $\pm$ 13,47	89,42 $\pm$ 8,45

TABLE 4.3: Results of the different classifiers applied to the data of the first method for functional connectomes built with Pearson correlation. Data is reduced to a feature space of size 100 with PCA and RFE

Moreover, the tendencies among classes are the same as those obtained without dimensionality reduction. Resting-state is still always well classified and the other tasks obtain slightly increased results, but their score ranking remains unchanged.

Clustering results

It is interesting to visualise the data we manipulate by projecting them into a two-dimensional space. A first observation is the positive impact of dimensionality reduction on clustering results. All the task clusters are more easily distinguishable after reducing dimension.

Another remark is the link we can do with the label specific results (4.2.4). By projecting data on a plane, we see that some tasks are more linearly separable than others. For example, the resting state on figure 4.7a forms a clear and linearly separable cluster. Language processing and social cognition tasks form a distinguishable cluster. These three tasks are those that obtained the best accuracy and f1-score for the SVM classifier. The t-SNE projection allows us to confirm the validity of the results.

But the projection is still quite foggy with this first method, data points are too much mixed between labels, which means that we didn't use all the power time series can give us. That's what the second method will focus on.

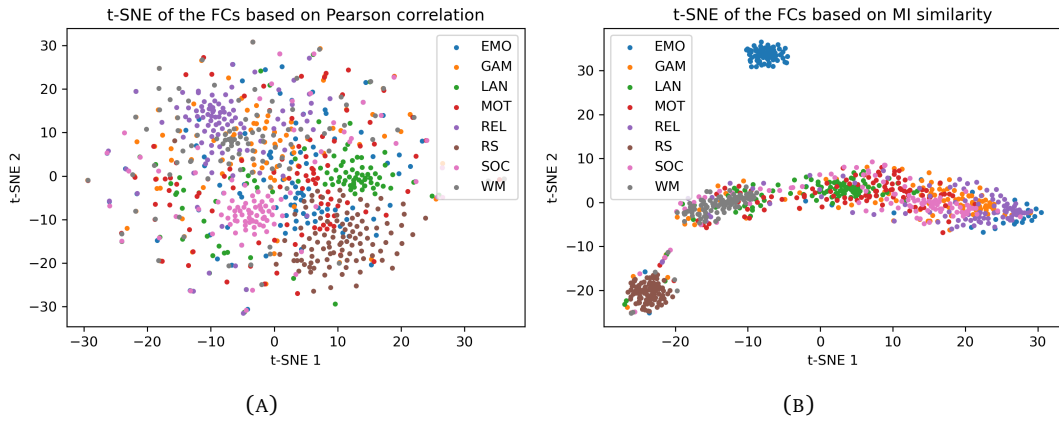


FIGURE 4.6: 2D projections of the 800 functional connectomes using t-SNE algorithm, for both association metric used in method 1 before model selection.

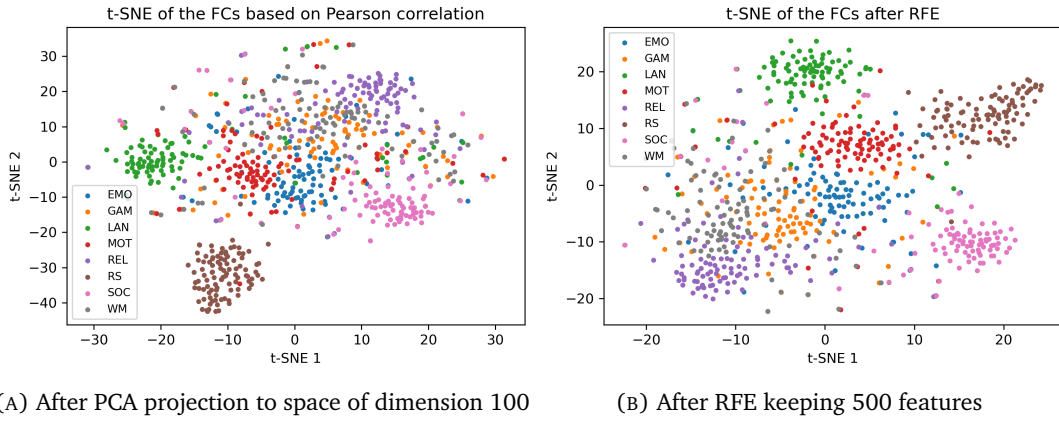


FIGURE 4.7: 2D projections of the 800 functional connectomes using t-SNE algorithm, for FCs based on Pearson correlation and after dimensionality reduction (A) : PCA (B) : RFE

4.3 Second method : time series based classification

Even if the obtained classification results are already quite good with this first method, the clustering results are not sufficiently satisfying and the classification scores could be better. That's why we completely change our way to treat the data to see if we can improve our results. In this second method, we decide to no longer use the concept of functional connectome. Indeed, even if this representation of data may be more easily understandable as it is a simple complete graph, transforming high-dimensional time series into a simple coefficient between each of them constitutes a massive loss of information.

Here is why it can be interesting to treat these "brute state" time series as features to feed our different models. This decision will massively increase the dimension of our data.

The principal problem with using the time series this way for classification is that data acquisition sessions have a different duration for each task performed by the subjects and so time series have different lengths. Furthermore, these differences are not negligible (176 time steps for emotional processing, 405 for the working memory task and 1200 for the resting state). We must then find a way to transform these data to reduce them to a common size such that we can feed our models with it. This common size is the size of the smallest time series. Indeed, we choose to not create additional data as it could bring too much noise to the fMRI data. All the time series will then be shortened to a size of 176, by different choices of selection.

4.3.1 Time series reduction techniques

"Choosing is losing", this quote takes all his sense here. Indeed, as we must reduce all time series to 176 data acquisitions, we will have to throw away a major fraction of the available data. This is not the same situation as the feature selection step in machine learning. Here, we don't choose data to get better results, but we have to select them to be able to obtain results.

Trivial selection

This first selection method, which we call trivial, consists of choosing 176 consecutive time series for each region of the brains and each task. It can be either the first ones, the last ones or any consecutive window of values in between. This method has the advantage to be extremely easy but maybe less representative of the data as certain tasks may have more intense periods than others. For example, the low activity of the resting-state will be capture whatever the selection, but a certain period could be more active than others for gambling task for example, and choosing the wrong window would mean giving up some interesting discriminant time features.

Resampling

Another way to select a subset of time series data is to use a resampling method. The objective of resampling methods in this context will be to reduce the size of the time series while keeping the general patterns in the data, we want the statistical properties to be similar to the original signal. An example of this fact is represented in figure 4.8 where we can see that the reduced size sample in blue still describes sufficiently good the initial curves of the full sample in orange.

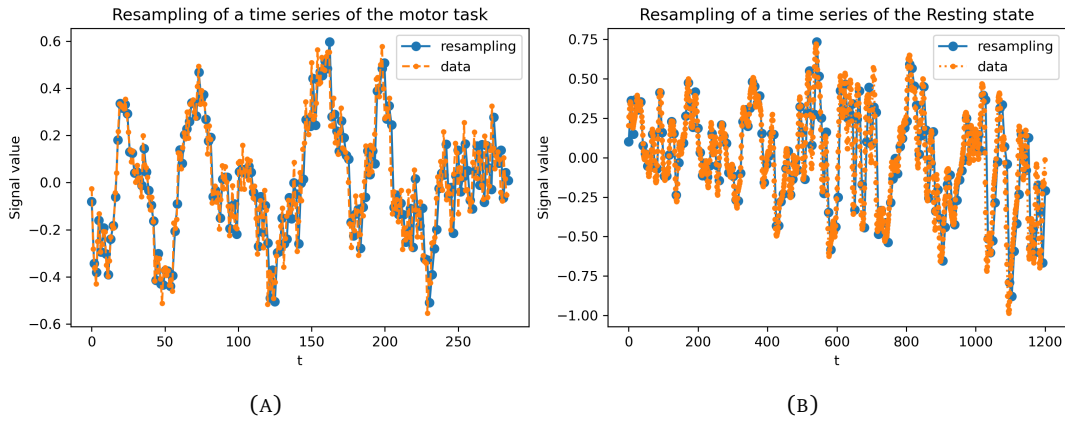


FIGURE 4.8: Comparison between the original signal and the resampled time series from one zone of one subject for the motor task (A) and the resting state (B).

For the examples above, we use resampling based on the **Fourier method**. This is also called downsampling, or decimation, as we reduce the size of the signal. We have in the beginning a signal $x[n]$ of length L with a sampling period ΔT and we want to reduce its size by a factor N so that the remaining signal becomes $y[n] = x[nN]$ with a sampling period $N\Delta T$. This is equivalent to compress the signal in the time domain or expand it in the frequency domain. To perform this reduction, we use the discrete-time Fourier transform of the signal to express the reduced signal in the frequency domain :

$$Y(e^{j\omega}) = \frac{1}{N} \sum_{k=0}^{N-1} X(e^{j(\omega - 2\pi k)/N}) \quad (4.11)$$

Going back to the time domain, we have a downsampled version of our signal which kept as much information as possible.

Once all time series for each ROI has the same time, we perform the same kind of vectorization as in method 1. We stack all resampled time series from all ROI to obtain a database composed of 800 data points with 14784 features each for the DK parcellation (84 zones times 176 time series values).

4.3.2 Results

Yet again, we can see on table 4.4 that the SVM classifier obtains the best results, which are extremely closed to other models due to the high overall classification scores obtained with this method. The best combination of processes is to use the SVM with the resampling method to select time series values. It gets the best accuracy and f1-score with the smallest standard deviation among the classes. Results don't change a lot with the resampling method but slightly increase, random forest and MLP classifier get a 1% improvement though.

An interesting point to notice is that during the model selection process, we obtained extremely low accuracy values for certain combinations of parameters. One of the worst choices of parameters was the polynomial kernel for the SVM classifier that lead to 15% accuracy. For such a number of features, linear kernels are the best and in general, the overall impact when changing a parameter was huge.

	Window method		
	SVM	RF	MLP
Accuracy (%)	98,37 $\pm$ 1	96,12 $\pm$ 0,83	96,75 $\pm$ 1,79
f1-score (%)	98,25 $\pm$ 2,28	96,06 $\pm$ 4,79	96,67 $\pm$ 3,88
	Resampling		
	SVM	RF	MLP
Accuracy (%)	98,47 $\pm$ 0,79	97,32 $\pm$ 0,96	97,47 $\pm$ 1,34
f1-score (%)	98,47 $\pm$ 2,2	97,26 $\pm$ 3,19	97,32 $\pm$ 3,69

TABLE 4.4: Results obtained with method 2 for both time series values selection methods.

Label-specific results

We can see on the results below (figure 4.9) that classification is in general extremely accurate. All classes obtain a nearly perfect f1-score, which was the main objective of the research. This time series resampling combined with an appropriate SVM classifier is our best way to perform task classification.

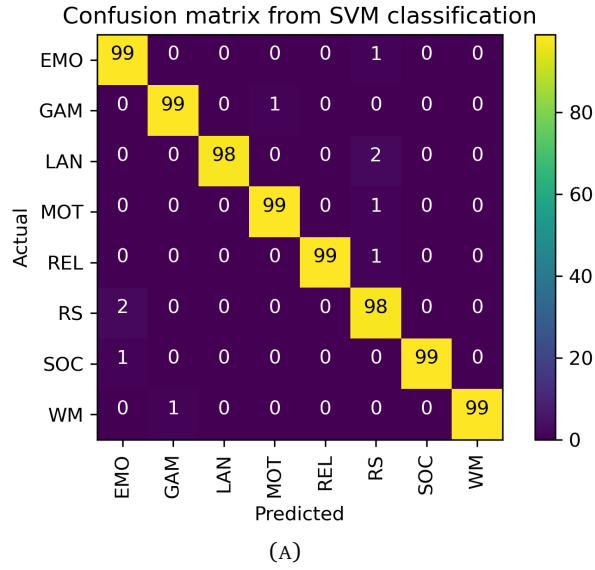
However, the resting state is once again not producing results identical to other tasks. In method 1, it was the most often well-classified task. In this section, it is the opposite, resting-state obtains the lowest f1-score and the percentage of misclassified data from the other tasks are more often predicted as resting state. This could be explained by some outliers in the recording phase of the fMRI BOLD data where certain people could have less active brain than the average of patients for some activities or a smaller oxygen-level increase after activity in certain areas of the brain. This happens for tasks more likely to vary between people: 2 language processing signals classified as resting state, 2 resting-state classified as emotional processing.

Impact of feature selection

Even if the results are already very satisfying, we try to go even further by making some feature selection. First, like in method 1, we looked at the impact of a PCA applied to these data. The results were quite similar to the initial ones. Then, a recursive feature elimination to reduce the data dimension to 500 lead to even better results (4.5). An SVM applied to the data of method 2 reduced to a feature space of 500 with RFE gives the overall best classification results, which could hardly be better than this. However, it could be interesting to see if a much larger database with a 2% outliers removal would get perfect results.

metric	PCA	RFE
Accuracy (%)	98,27 $\pm$ 0,77	99,47 $\pm$ 0,46
f1-score (%)	98,27 $\pm$ 2,28	99,48 $\pm$ 1,15

TABLE 4.5: Results obtained with an optimal SVM model after two dimensionality reduction methods : (1) dimensionality reduction to a feature space of size 100 with PCA and (2) RFE to select 500 best features



Metric	EMO	GAM	LAN	MOT	REL	RS	SOC	WM
Sensitivity (%)	99	99	98	99	99	98	99	99
f1-score (%)	98,02	99	98,99	99	99,5	96,55	99,49	99,5

(B)

FIGURE 4.9: (A) Confusion matrix resulting from the the classification of the data of method 2 with the SVM classifier. (B) Table of the classification results for each class

Clustering results

As we did for method 1, we project data that generated the results of the second method on a 2D plane. First, the t-SNE projection of the original data after resampling (figure A.1a) already shows clear task-specific clusters, especially for relational and social processing. Other tasks are closer but still distinct. But the most interesting projection to study is without a doubt the t-SNE obtained after recursive feature elimination (A.1b), where clusters are clearer than those obtained in [12] (except for RS) but just a bit less separated than the t-SNE shown in [29]. On this plot, tasks all form clearly separated clusters, even if some very few outliers points are projected closer to points of a different task, which could be due to the resampling phase which can drop meaningful data. Beyond the very satisfying clustering results compared to the same experiment from method 1, the resting state behaves differently from what was observed in method 1. In method 1, it was the clearest cluster among all tasks. Here, it is the opposite situation: all points from active tasks form distinguishable clusters and the resting state points are much more distributed on the plane. this explains why the resting state obtains the poorest results with method 2. Neural activity differs more between subjects in the resting state, each people having different patterns of activity, even with a quite stationary activity. After all, resting state is not a task properly speaking, it was our choice to consider it as a task and the results confirm what we could expect. This dispersion could also come from the reduction of the size that was imposed on the data : the resting-state time series lost more than 85% after the resampling while other tasks lost much less.

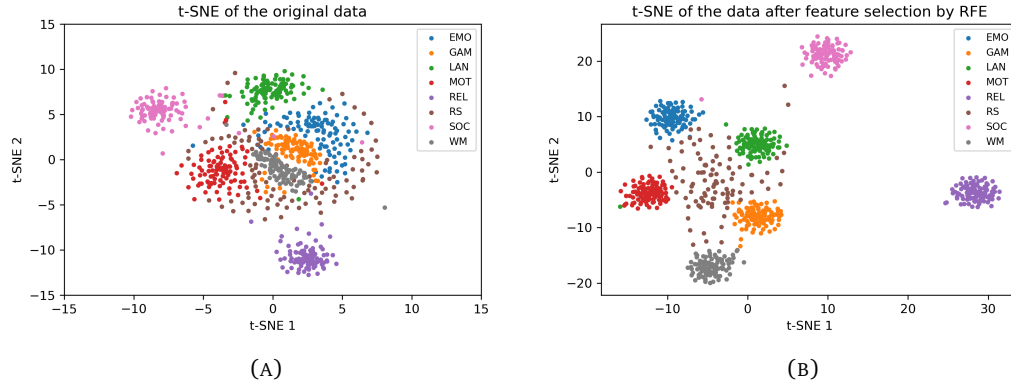


FIGURE 4.10: t-SNE projection of data of the the second method with resampling (A) before feature selection and (B) after recursive feature elimination.

4.4 Third method : classification based on autocorrelation between time series

As explained in the description of the second method, the main drawback of this one comes from the inevitable reduction of time series sizes which forces to drop some values of the database. Even if the results of this last method are extremely satisfying, we want to use all the power of our data, we then decide to treat the time series a third way. We will study and classify their autocorrelation.

4.4.1 Autocorrelation for time series

Times series is basically just real values indexed in time that often carry some meaningful patterns and information if they describe real processes. Autocorrelation is a common tool used in the field of time series analysis that can help to visualize and retrieve this information from time series. It can highlight some repeated features by removing the negative effect of noise on the data. fMRI data treatment is one example of the application of autocorrelation and some research has already be done on this topic [41]. Basically, autocorrelation is the measure of the correlation between a signal and a lagged version of itself by a number of time steps τ .

$$\text{Autocorrelation} = \text{Corr}(y[t], y[t + \tau]) \quad \tau = 1, 2, \dots \quad (4.12)$$

The principal interest in using this measure for our data is to find some seasonality or trending patterns in the data that could be some kind of discriminative features useful to identify task-specific patterns in fMRI signals. In the case of discrete-time series of size N with known mean and variance like those we dispose of, Woolrich et al. proposed in [41] an unbiased estimation of this autocorrelation function of a discrete variable X :

$$\hat{R}_{XX}(\tau) = \frac{1}{(N - \tau)\sigma_X^2} \sum_{t=1}^{N-\tau} (X_t - \mu_X)(X_{t+\tau} - \mu_X) \quad (4.13)$$

Where μ_X is the mean of the variable X and σ_X its standard deviation. Meanwhile, the Pearson correlation coefficient is the most current to compute these autocorrelation values.

4.4.2 Methodology and results

We are going to use this autocorrelation function to build vectors of autocorrelation values for different values of the lag ($\tau = 0, 1, 2, \dots, 100$). For each ROI, we build new vectors of autocorrelation values as a function of the lag. The same technique as before is necessary to linearise the matrix form of data per subject into a high dimensional vector composed of a stacked autocorrelation vector for each ROI. The optimal number of lag values per zone is selected empirically by testing the models with data built with different lag values ranged from 1 to 150. The feature vectors then become, for a subject s and a task t :

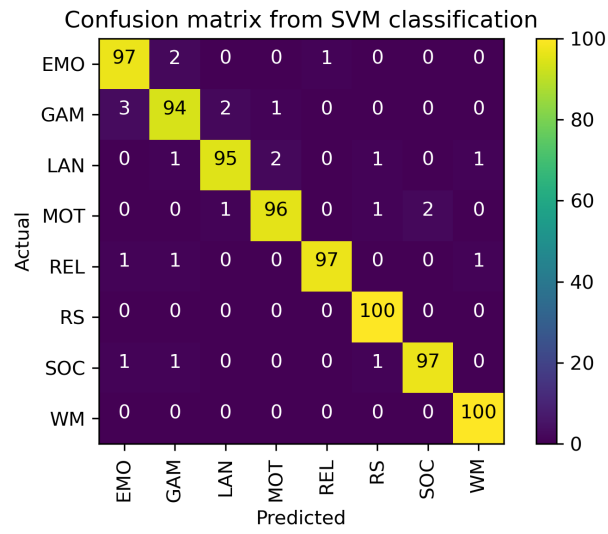
$$FV_s^t = [R_{XX}(1), R_{XX}(2), \dots, R_{XX}(\tau_{max})] \quad (4.14)$$

The results we got from this method are in between those of the first and the second method. This time the MLP classifier with a single hidden layer of size 100 gets the best classification results but SVM is close to its result. The feature selection with RFE doesn't have such an impact on classification while PCA was not used anymore as it slightly reduced the quality of the results.

	100 lag values		
	SVM	RF	MLP
Accuracy (%)	96, 6 $\pm$ 1, 15	94, 7 $\pm$ 1, 33	97, 12 $\pm$ 1, 25
f1-score (%)	96, 53 $\pm$ 3, 35	94, 69 $\pm$ 4, 89	97, 07 $\pm$ 2, 86
	RFE		
	SVM	RF	MLP
Accuracy (%)	96, 9 $\pm$ 1, 5	94, 8 $\pm$ 1, 73	97 $\pm$ 1, 06
f1-score (%)	96, 85 $\pm$ 3, 6	94, 78 $\pm$ 5, 26	96, 92 $\pm$ 3, 38

TABLE 4.6: Results obtained with method 3 for the optimal number of lagged values per ROI before and after feature selection with RFE to 500 features

Even if those results are not as good as previous method, task-specific results are still interesting to compare. Resting state and working memory are always perfectly classified but do not have a perfect f1-score as some other tasks are sometimes classified as one of them.



(A)

Metric	EMO	GAM	LAN	MOT	REL	RS	SOC	WM
Accuracy (%)	97	94	95	96	97	100	97	100
f1-score (%)	96,04	94,47	95,96	96,48	97,98	98,52	97,49	99,01

(B)

FIGURE 4.11: (A) Confusion matrix resulting from the the classification of the data of method 3 with the SVM classifier. (B) Table of the classification results for each class

4.4.3 Discussion

Stationarity of resting state

The easiest way to recognise the resting state among the other tasks is its stationarity. When plotting autocorrelation and lag plots of resting state, the correlation between a signal and the lag 1, 2 or 3 versions of itself is remarkable.

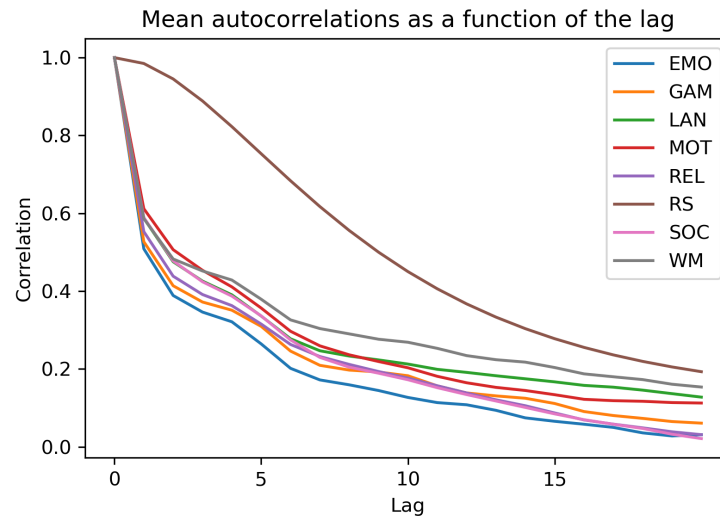


FIGURE 4.12: Comparison of the average autocorrelations between the different tasks. Each line represents the average of all autocorrelation of 100 lag values from all ROI from all subjects

Results shown by figure 4.12 confirm our intuition that the resting state (brown line) behaves very differently from the other tasks. Its average autocorrelations with the 10 first lag versions of itself are much higher compared to the other tasks. The resting-state signals have a much longer memory than every other tasks fMRI and it may explain why in most other researches on fMRI, resting state is considered differently. The interest now could be to study most of the less autocorrelated zones from other tasks to identify the most active ROI per task and compare those results to the physiological theory.

Inefficiency of feature selection

In the context of this third method, the analysis of the results indicated that neither PCA nor RFE induced better classification results. It could maybe be explained by the fact that autocorrelation reduces noise in the data. Moreover, computing the autocorrelation of the data is already a dimensionality reduction in itself and the modifications it makes to the data can be sufficient. To illustrate this hypothesis, we can compare characteristic samples of raw time series data 4.13a and their corresponding autocorrelation values 4.13b after data processing of method 3. In this example, the autocorrelation function is a lot smoother than the raw original data. Once all correlations corresponding to ROI are concatenated, feature selection could bring additional noise originally removed by the preprocessing of method 3. This could explain the lack of interest in feature selection in this particular case.

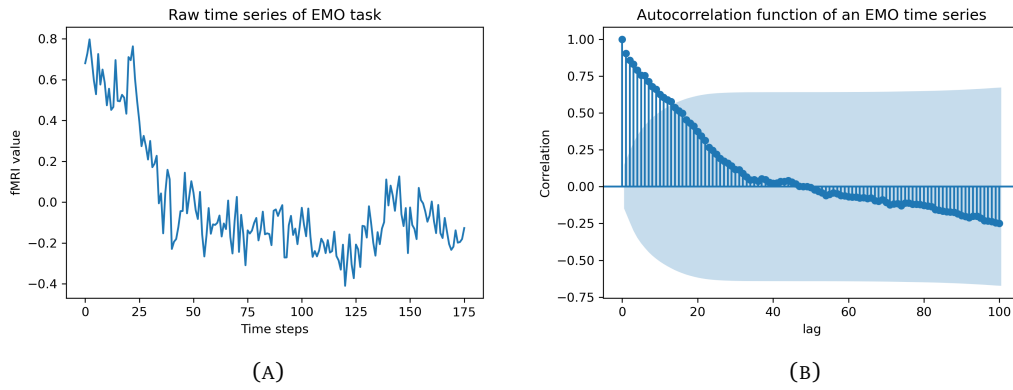


FIGURE 4.13: Comparison between a raw time series of a ROI from a subject in emotional processing (A) and its corresponding 100 first lag values of its autocorrelation function.

4.5 General discussion

4.5.1 Best cortical brain atlas

All the above experiments were run again with the Destrieux cortical parcellation composed of 164 ROIs. The detailed results are summarised in the annexe to avoid redundancy in the presentation of the results but the main differences between the results obtained with the Desikan-Killiany atlas are compared here.

	F1-score DK atlas (classifier)	F1-score DES atlas (classifier)
Method 1	90, 28 (SVM)	93, 61 (MLP)
Method 2	99, 47 (SVM)	99, 77 (SVM)
Method 3	97, 12 (MLP)	96, 85 (SVM)

TABLE 4.7: Comparison between the main results obtained with DK and DES parcellation

A first general remark is that the behaviour of our methods didn't change a lot between the data coming from one atlas or the other. F1-scores and accuracy slightly change between the two cases and the general tendencies followed by the results of the models coincides between DK and DES. Moreover, the best hyperparameters of every model of each method generally correspond for the 2 versions of the data, all the remarks above still stand when using the Destrieux brain atlas. The t-SNE clustering obtained doesn't provide visible change either.

For method 1, changing the atlas allows us to get the best result 3% higher with the MLP classifier and about 1% for SVM and RF (see A.1). As both datasets are reduced to 500 features, feature selection from a four times bigger set (3486 in DK to 13366 with DES) have more potential to find discriminating features as there are more potentially interesting edges in the functional connectome with 164 nodes.

For Method 2, the same kind of remarks can be made, hyperparameters remain unchanged and the results of classifiers remain similar (see A.2). Compared to method 1, the size is only doubled in the original feature space (84 to 164 zones, each with 176 times series values) and a bigger number of parcellations doesn't seem

to change to overall classification results. However, after RFE, the classification results become nearly perfect with the Destrieux atlas : 7 among 8 classes are perfectly classified and only 1 point of the relational processing task is classified as gambling.

For method 3 now, models generated all have the same hyperparameters (except MLP slightly better with logistic activation function) than before but the DK parcellations seem to be the best fit for the autocorrelation method. As the dimension of the data is two times bigger, the SVM classifier with linear kernel gets the best results with DES, while MLP performs better with DK.

The overall best classification results are obtained with the Destrieux parcellation data from method 2, classification based on raw time series resampling performs better with more data. The functional connectome method (1) has the best results increase among the three methods. But both atlases may have their strength in certain situations, autocorrelation methods were more precise with a smaller amount of ROIs and in general, comparing the results with both datasets will always be interesting.

4.5.2 Limitations of dataset

Even if the results detailed here are satisfying for the objective of task classification, the main drawback of the current dataset is its size. We only dispose of 100 subjects per task fMRI session, which is small compared to the feature space dimensions discussed in this work. Moreover, only seven tasks and the resting state are made available in the Human Connectome Project. access to more variety of potential labels would allow us to investigate further the limitations of the power of our methods. Till how many different tasks classification results would be satisfying?

4.5.3 Subject identification

Another remark we can do is the high similarity of the neural activity of the same person when recorded several times. Here we dispose of 2 sessions of each task so we could check if Subject identification obtained good results. It is the case, If we train a model with data from a session of resting-state and predict the subjects ID of a second and different dataset of resting-state, people are always well predicted. Indeed, functional connectomes of the subject for 2 sessions of the same task are highly correlated, as we can see in figure 4.14. However, between-task subject identification doesn't work at all, with an accuracy of at most 5%. This cross task subject identification with multiple tasks in the learning set has already been discussed in [1] and some experiments based on method 1 give the same trends in the results: resting state in the training set induces poor subject identification and other combinations of tasks get better results but still much lower (around 45%) than detailed by Abbas et al.

4.5.4 Pertinence of feature selection with RFE

In addition to improving the accuracy of task identification in the two first methods, the interest of recursive feature elimination is to figure out and interpret which part of the data is the most relevant. RFE selects features where the difference between classes is the clearest to classify them. In the current context, it is interesting to compare the results of feature selection from a machine learning perspective with the

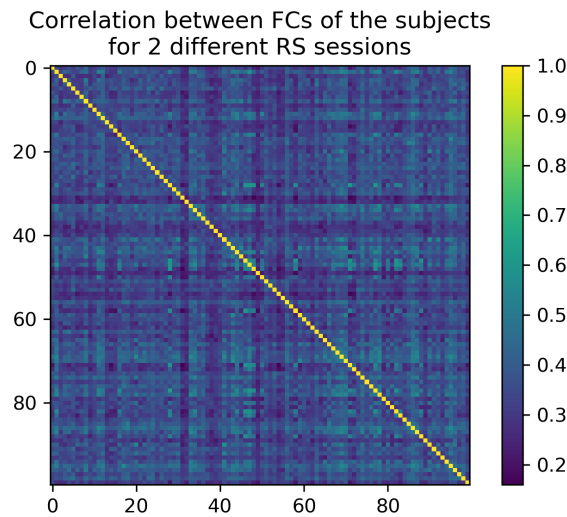


FIGURE 4.14: Correlation between the vectorised FCs of the 100 subjects from 2 different sessions of resting state data acquisition

relevance of these features from a biological point of view. In other words, we can investigate if the discriminative features selected by RFE to discriminate a certain brain activity are coherent with neuroscientific expectation.

We investigated the 500 best features of the data of the second method when discriminating only the motor task among the others. We then counted the occurrences of each zone in the list of 500 features and sorted the number of occurrences. The ROI that was most present in the 500 best features was the Post Central region of the left hemisphere, related to the somatosensory cognitive system. This is coherent from the neuroscientific point of view as the impact of the motor tasks activity on this cognitive system has already been studied in [6]. Even if we study the fMRI signals from a purely mathematical perspective, we can get similar results to a more biological study. The same experiment could also be done with edges of the functional connectome to see on the correlation between the activity of some regions can confirm or infirm neuroscientific theory.

Many comparisons could be studied in this context and the similarity of discriminant features of machine learning models and brain activated regions is a strong potential for understanding brain activity.

Chapter 5

Conclusion

The main objective of this research was to build a method that would get the best results at classifying tasks processed by the subjects of our database, based on the observations of the neural activity from the BOLD fMRI signals. This straightforward objective also hides some underlying questions and reflections. We wanted to compare different approaches to manipulate the raw fMRI data to use them the best way. A focus was made on comparing the classification ability of three different classifiers used by the machine learning community.

Throughout this thesis, we focused on the interest and the potential of the use of machine learning tools in the research in neurosciences. After a brief introduction to the technologies currently available and the main studies already published on the classification of fMRI data, we focused on the techniques used to represent the brain with the BOLD fMRI technique and detailed the different possible parcellation of the brain into a set of distinct regions of interest (ROIs). After introducing the database composition, we then discussed different ways to treat these fMRI data to understand and predict what was happening in our brains and the characteristics of signals relative to the different tasks we are doing.

A first method transformed the time series associated with the ROIs into a functional connectome based on the mutual information or Pearson correlation between the signals. We empirically showed that Pearson correlation was the best metric to characterise relations between fMRI signals from ROIs. A recursive feature elimination method applied on the edges of the functional connectomes from the Destrieux dataset ROI lead to classification with an F1-score of 93,61% for the SVM classifier with linear kernel. We confirmed from this first approach that the resting state behaved differently from other tasks as it was perfectly classified.

With the next method, we tried to use as much data as possible by classifying the raw time series after an appropriate resampling method. We considered the concatenated resulting time series as a huge dimensional feature space. For all the classifiers we used, the results were extremely good. We managed to fit an SVM model which classifies nearly perfectly all the tasks after a recursive feature elimination with an F1-score of 99,47% for the Desikan-Killiany brain parcellation and 99,77% with the Destrieux atlas where seven among the eight tasks were perfectly classified.

A final method of data processing focused on the study of the autocorrelation of the time series on the first hundred lag values. This process took the role of both time series denoising and dimensionality reduction. The data from this method were better classified with a multilayer perceptron and the DK atlas with an F1-score of 97,12%. It also allowed us to highlight the long memory of the average signals of resting-state compared to the other tasks, which was perfectly classified.

At the end of all the experiments with each of the different methods, we showed the SVM classifier with a linear kernel was excellent to classify high dimensional data as BOLD fMRI signals. The multilayer perceptron produced the closest results to SVM and after a pertinent feature selection method and the appropriate hyperparameters, it could get similar results, or even better in some situations.

The general advantages of all the methods detailed here are multiple: in addition to giving to the high accuracy of the task identification, it is not necessary to have any knowledge about neurosciences to select the most discriminant features. Moreover, these methods can be applied to any future fMRI time series that could be released.

Finally, at the question to know if would be able to read in our minds one day, we could answer that given a finite set of experiments, it is possible to train a model to would accurately identify what a person is doing based on his neural activity measured by a BOLD fMRI technique. And as we are still at the beginning of the emergence of connectomics and machine learning application to neurosciences, we can be confident in the future development of technologies that may seem futuristic nowadays.

Potential future works

During the different steps of this thesis, lots of interesting ideas were discussed. The potential of fMRI data is so vast that we could have taken lots of different paths towards the objective.

One potential method that could be applied to realize this objective in the future is to use an image processing method directly on the fMRI images to see if images features could perform as well as methods described in this study.

The principal limitation in fMRI study is the size of datasets and the low variety of experiments. We know we can classify eight tasks from the observation of neural activity, but to which point would the classification results be satisfying, how many different activities would it be possible to identify? This is a question that would be a good follow-up to what was done here.

Another discussion of interest would be to investigate further if the features selected by RFE for each data pre-processing method coincide with the real activation zones of the brain. Potential solutions to treat some neural disorder could be found by pushing further the analysis of these BOLD fMRI images.

As a conclusion, we can be confident in the fact that neurosciences will continue to take an important place in the future of scientific research. The continuously evolving technologies will continue to provide always more precise representations of the brain connectivity and the combination of the power of these data with graph theory, signal processing and machine learning algorithms will surely lead us to unveil lots of secrets of our brains.

Appendix A

Detailed results obtained with DES parcellation

PCA (reduction to 100 features)			
	SVM	RF	MLP
Accuracy (%)	81,87 $\pm$ 3,34	68,5 $\pm$ 3,34	81,75 $\pm$ 2,86
f1-score (%)	81,76 $\pm$ 13,16	67,32 $\pm$ 18,62	81,56 $\pm$ 12,44
RFE (500 features kept)			
	SVM	RF	MLP
Accuracy (%)	91,52 $\pm$ 2,1	82,47 $\pm$ 2,94	93,7 $\pm$ 1,65
f1-score (%)	91,327,83	82,1 $\pm$ 11,39	93,61 $\pm$ 5,48

TABLE A.1: Results of the different classifiers applied to the data of the first method for functional connectomes built with Pearson correlation for DES parcellation

Resampling method, no RFE			
	SVM	RF	MLP
Accuracy (%)	98,6 $\pm$ 0,9	96,82 $\pm$ 1,4	96,92 $\pm$ 1,41
f1-score (%)	98,6 $\pm$ 2,26	96,76 $\pm$ 3,57	96,78 $\pm$ 4,71
Resampling method, RFE to 500 features			
	SVM	RF	MLP
Accuracy (%)	99,77 $\pm$ 0,35	98,25 $\pm$ 1,07	99,22 $\pm$ 0,8
f1-score (%)	99,77 $\pm$ 0,76	98,19 $\pm$ 2,42	99,2 $\pm$ 1,68

TABLE A.2: Results obtained with method 2 before and after RFE with DES atlas

100 lag values			
	SVM	RF	MLP
Accuracy (%)	96,9 $\pm$ 1,5	93,97 $\pm$ 1,94	94,72 $\pm$ 1,45
f1-score (%)	96,85 $\pm$ 3,6	94,01 $\pm$ 5,31	94,57 $\pm$ 4,43
RFE			
	SVM	RF	MLP
Accuracy (%)	96,37 $\pm$ 1,13	95,2 $\pm$ 1	94,72 $\pm$ 1,45
f1-score (%)	96,33 $\pm$ 3,74	95,13 $\pm$ 4,35	94,57 $\pm$ 4,43

TABLE A.3: Results obtained with method 3 for the optimal number of lagged values per ROI before and after feature selection with RFE to 500 features with the DES atlas

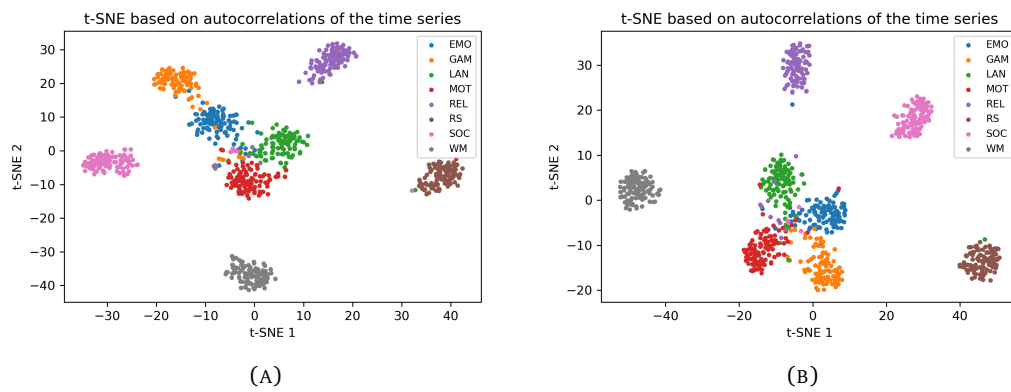


FIGURE A.1: t-SNE projection of data of the the third method with (A) Desikan-Killian and (B) Destrieux atlas

Bibliography

- [1] Kausar Abbas et al. “GEFF: Graph embedding for functional fingerprinting”. In: *NeuroImage* 221 (2020), p. 117181.
- [2] Davide Ballabio, Francesca Grisoni, and Roberto Todeschini. “Multivariate comparison of classification performance measures”. In: *Chemometrics and Intelligent Laboratory Systems* 174 (2018), pp. 33–44.
- [3] R. Bellman, Rand Corporation, and Karreman Mathematics Research Collection. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957. ISBN: 9780691079516. URL: <https://books.google.be/books?id=wdtoPwAACAAJ>.
- [4] Arie Ben-David. “About the relationship between ROC curves and Cohen’s kappa”. In: *Engineering Applications of Artificial Intelligence* 21.6 (2008), pp. 874–882.
- [5] Ismail Bilgen, Goktug Guvercin, and Islem Rekik. “Machine learning methods for brain network classification: Application to autism diagnosis using cortical morphological networks”. In: *Journal of neuroscience methods* 343 (2020), p. 108799.
- [6] Anna Bodegård et al. “Somatosensory areas in man activated by moving stimuli: cytoarchitectonic mapping and PET”. In: *Neuroreport* 11.1 (2000), pp. 187–190.
- [7] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. “A training algorithm for optimal margin classifiers”. In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 144–152.
- [8] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.
- [9] Colin J Brown and Ghassan Hamarneh. “Machine learning on human connectome data from MRI”. In: *arXiv preprint arXiv:1611.08699* (2016).
- [10] Jason Brownlee. *Machine learning mastery with Python: understand your data, create accurate models, and work projects end-to-end*. Machine Learning Mastery, 2016.
- [11] Ed Bullmore and Olaf Sporns. “Complex brain networks: graph theoretical analysis of structural and functional systems”. In: *Nature reviews neuroscience* 10.3 (2009), pp. 186–198.
- [12] Benjamin Chiêm, Frédéric Crevecoeur, and Jean-Charles Delvenne. “Structure-informed functional connectivity driven by identifiable and state-specific control regions”. In: *BioRxiv* (2020).
- [13] Thomas M Cover and Joy A Thomas. “Information theory and statistics”. In: *Elements of Information Theory* 1.1 (1991), pp. 279–335.
- [14] Luis Cuadros-Rodríguez, Estefanía Pérez-Castaño, and Cristina Ruiz-Samblás. “Quality performance metrics in multivariate classification methods for qualitative analysis”. In: *TrAC Trends in Analytical Chemistry* 80 (2016), pp. 612–624.

- [15] Adele Cutler, D Richard Cutler, and John R Stevens. "Random forests". In: *Ensemble machine learning*. Springer, 2012, pp. 157–175.
- [16] Catherine E Davey et al. "Filtering induces correlation in fMRI resting state data". In: *Neuroimage* 64 (2013), pp. 728–740.
- [17] Kai-Bo Duan and S Sathiya Keerthi. "Which is the best multiclass SVM method? An empirical study". In: *International workshop on multiple classifier systems*. Springer. 2005, pp. 278–285.
- [18] Gary H Glover. "Overview of functional magnetic resonance imaging". In: *Neurosurgery Clinics* 22.2 (2011), pp. 133–139.
- [19] Hao Guo et al. "Alzheimer classification using a minimum spanning tree of high-order functional network on fMRI dataset". In: *Frontiers in neuroscience* 11 (2017), p. 639.
- [20] Isabelle Guyon et al. "Gene selection for cancer classification using support vector machines". In: *Machine learning* 46.1 (2002), pp. 389–422.
- [21] Uri Hasson, Howard C Nusbaum, and Steven L Small. "Task-dependent organization of brain regions active during rest". In: *Proceedings of the National Academy of Sciences* 106.26 (2009), pp. 10841–10846.
- [22] Khadidja Henni, Neila Mezghani, and Charles Gouin-Vallerand. "Unsupervised graph-based feature selection via subspace and pagerank centrality". In: *Expert Systems with Applications* 114 (2018), pp. 46–53.
- [23] David H Hubel and Torsten N Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". In: *The Journal of physiology* 160.1 (1962), pp. 106–154.
- [24] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. "Estimating mutual information". In: *Physical review E* 69.6 (2004), p. 066138.
- [25] Xiangrui Li et al. "The Iowa gambling task in fMRI images". In: *Human brain mapping* 31.3 (2010), pp. 410–423.
- [26] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).
- [27] Zhenyu Mao et al. "Spatio-temporal deep learning method for adhd fmri classification". In: *Information Sciences* 499 (2019), pp. 1–11.
- [28] Eddy Mayoraz and Ethem Alpaydin. "Support vector machines for multi-class classification". In: *International Work-Conference on Artificial Neural Networks*. Springer. 1999, pp. 833–842.
- [29] Vikram Ravindra and Ananth Grama. "De-anonymization attacks on neuroimaging datasets". In: *arXiv preprint arXiv:1908.03260* (2019).
- [30] Joshua S Richman and J Randall Moorman. "Physiological time-series analysis using approximate entropy and sample entropy". In: *American Journal of Physiology-Heart and Circulatory Physiology* (2000).
- [31] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.
- [32] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

- [33] Saman Sarraf and Ghassem Tofghi. "Classification of alzheimer's disease using fmri data and deep learning convolutional neural networks". In: *arXiv preprint arXiv:1603.08631* (2016).
- [34] Olaf Sporns, Giulio Tononi, and Rolf Kötter. "The human connectome: a structural description of the human brain". In: *PLoS Comput Biol* 1.4 (2005), e42.
- [35] David C Van Essen et al. "The WU-Minn human connectome project: an overview". In: *Neuroimage* 80 (2013), pp. 62–79.
- [36] Vladimir N Vapnik. "The support vector method". In: *International Conference on Artificial Neural Networks*. Springer. 1997, pp. 261–271.
- [37] Michel Verleysen and John Lee. *Lecture notes of the course LELEC2870 - Machine Learning : regression, dimensionality reduction and data visualization*. 2020.
- [38] Ze Wang et al. "Brain entropy mapping using fMRI". In: *PloS one* 9.3 (2014), e89948.
- [39] Avani Wildani, Ethan L Miller, and Lee Ward. "Efficiently identifying working sets in block i/o streams". In: *Proceedings of the 4th Annual International Conference on Systems and Storage*. 2011, pp. 1–12.
- [40] Blake A. Wilson, Arvind Ramanathan, and Carlos F. Lopez. *LoLab-VU/PyBILT: A Python toolkit to analyze molecular simulation trajectories of lipid bilayer systems*. Version v0.2.0. First release with version 0.2.0! Sept. 2019. DOI: [10 . 5281/zenodo.3426128](https://doi.org/10.5281/zenodo.3426128). URL: <https://doi.org/10.5281/zenodo.3426128>.
- [41] Mark W Woolrich et al. "Temporal autocorrelation in univariate linear modeling of FMRI data". In: *Neuroimage* 14.6 (2001), pp. 1370–1386.
- [42] EA Zanaty. "Support vector machines (SVMs) versus multilayer perception (MLP) in data classification". In: *Egyptian Informatics Journal* 13.3 (2012), pp. 177–183.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl