Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación (GITT)

EEG signal classification for MI-BCI applications

Autor: David García Ramos

Tutor: Sergio A. Cruces Álvarez

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación (GITT)

EEG signal classification for MI-BCI applications

Autor:
David García Ramos

Tutor:
Sergio A. Cruces Álvarez
Catedrático de Universidad

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado: EEG signal classification for MI-BCI applications

Autor: David García Ramos
Tutor: Sergio A. Cruces Álvarez

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

El desarrollo de este trabajo ha sido posible gracias a la inestimable ayuda de mi tutor, Sergio A. Cruces Álvarez, quien me ha introducido al fascinante tema de las interfaces cerebro-ordenador y ha sido mi guía al dar mis primeros pasos en esta materia. Siempre que he tenido dificultades ha estado disponible para mí, solventando todas mis dudas y realizando una inconmensurable contribución a mi formación académica.

Por supuesto, este documento no es únicamente producto de mi esfuerzo, sino del de todos aquellos que vinieron antes y prepararon el camino a través del estudio de diversos asuntos relacionados con las matemáticas, la medicina, la física o la anatomía. En concreto, tengo que mencionar la investigación realizada por F. Javier Olías Sánchez, cuyo trabajo ha servido de precedente a este.

Además, la aportación de mi pareja, Anna-Lena, ha sido imprescindible, ya que no solo me ha dado todo el apoyo que he necesitado para avanzar durante mi grado y en especial durante el desarrollo de este trabajo, sino también porque ha tenido suficiente paciencia como para revisar la memoria y corregir mis faltas de ortografía.

Asímismo, no hubiera llegado aquí si no fuera por la formación excepcional que he tenido la suerte de poder recibir gracias a los profesores que me han acompañado en mi etapa universitaria, tanto en la Universidad de Sevilla como el año de mi estancia en University College Cork. En especial tengo que dar mi agradecimiento a aquellos profesores del Departamento de Teoría de la Señal y Comunicaciones, que han despertado en mí la pasión por esta materia que hace cuatro años ni sabía que existía.

Durante esta etapa nunca he estado solo y por eso estoy agradecido a todos mis amigos y familiares, quienes me han acompañado siempre, incluso cuando estaba lejos de ellos.

Finalmente, traslado mi más sincera gratitud a mis padres, María del Mar y Miguel Ángel, quienes han sido los principales contribuyentes a mi formación, académica y personal. Siempre han apoyado mis decisiones y me han ayudado en todo lo que he necesitado y más. Es difícil resumir lo que han hecho por mí en tan pocas palabras, pero sin ellos no sería quien soy hoy y desde luego este trabajo nunca hubiera llegado a ser.

Resumen

Una Interfaz Cerebro Ordenador (o BCI por sus siglas en inglés) es un sistema que permite al usuario producir instrucciones que pueden ser interpretadas por una máquina sin necesidad de interacción física. Funciona como un puente entre el cerebro humano y un ordenador y es una solución prometedora que permitiría a personas con discapacidad motora interactuar con sus alrededores de una nueva manera, mejorando drásticamente su calidad de vida.

Existen múltiples tipos de BCI que se basan en diferentes señales cerebrales que pueden ser registradas con distintos dispositivos. Entre estas BCI, una de las más interesantes es la Interfaz Cerebro Ordenador de Movimiento Imaginario (MI-BCI): el usuario piensa en efectuar un movimiento repetidamente y el sistema tiene que identificar de qué movimiento se trata de entre un conjunto de posibilidades, por ejemplo, movimientos de mano izquierda y derecha, pies o lengua.

Este sistema se basa en que, cuando una persona se imagina a sí misma haciendo un movimiento, se activa una respuesta en el cerebro similar a la que se produce cuando el movimiento se realiza de verdad. Esta actividad cerebral se puede registrar de múltiples maneras; sin embargo, el transductor más común usado para grabar señales cerebrales es el Electroencefalógrafo (EEG), que es un sistema económico, portable y no invasivo que se compone de múltiples electrodos colocados en el cuero cabelludo del usuario y conectados a un amplificador.

El problema presentado no es sencillo. Las señales de EEG tienen una baja resolución espacial, baja relación señal a ruido y se ven afectadas por interferencias producidas por movimiento ocular o muscular entre otros. Además, se introduce redundancia porque estas señales se propagan a través de las diferentes capas que componen la cabeza y llegan a la vez a varios electrodos. Como resultado, debemos emplear múltiples potentes técnicas de procesado de señal para obtener un sistema MI-BCI robusto.

Comenzamos introduciendo brevemente algunas BCI existentes, así como el conjunto de datos que analizaremos en este documento, que es el MI-BCI dataset 2a, de la BCI Competition IV. Este está disponible para uso público y ha sido estudiado previamente en profundidad. A continuación, presentamos el problema de MI-BCI como un problema general de reconocimiento de patrones y lo dividimos en sus partes principales: preprocesado de señales, extracción de características o reducción de dimensionalidad y finalmente clasificación, que será el centro de atención de este trabajo. Las primeras dos fases preparan las señales y extraen de ellas la información más relevante, mientras que en la última se realiza la predicción. Para este paso una solución que se ha usado extensivamente en el pasado ha sido el Análisis de Discriminantes Lineales (LDA), pero también existen otras alternativas.

En el siguiente capítulo intentamos ilustrar el funcionamiento interno de algunos de estos clasificadores y finalmente realizamos un experimento para comparar los resultados obtenidos con los diferentes algoritmos en el problema de MI-BCI.

Antes de realizar ninguna predicción, los clasificadores pasan por una fase de entrenamiento en la que se calculan los parámetros internos que gobiernan su comportamiento. Esto se realiza con observaciones similares a aquellas que después se desean clasificar. En este documento el problema no solo se aborda desde la clásica situación supervisada, en la que durante el entrenamiento la información sobre cuáles eran las intenciones del usuario es conocida, sino que también se trata el caso no supervisado, en el que esta información se desconoce.

Abstract

A Brain Computer Interface (BCI) is a system that allows the user to produce commands that can be interpreted by a machine without the need for physical interaction. It acts as a bridge between a human brain and a computer, and it is a promising solution that would allow people with motor disabilities to interact with their surroundings in a new way, drastically improving their quality of life.

Several different types of BCI exist, based on multiple distinct brain signals that can be registered with different physical devices. Amongst these BCIs, one of the most interesting is Motor Imagery BCI (MI-BCI). The user thinks about performing a movement repeatedly, and the system has to identify which movement the person is imagining amongst a defined set of possibilities, for example left and right hand, feet or tongue movements.

The working principle behind this is that when a person imagines themselves performing a movement, some of the same responses that happen when the movement is actually performed are triggered. This brain activity can be registered in multiple ways, however, the most common transducer used to record brain signals is the Electroencefalogram (EEG), which is a portable, affordable and non invasive system composed by electrodes placed on the scalp of the subject and attached to an amplifier.

The problem presented here is not simple. The EEG signals have low spatial resolution, low signal to noise ratio and are affected by artifacts produced by eye or muscle movement. Furthermore, redundancy is introduced, as the same signals propagate through the different layers of the head and reach multiple electrodes at the same time. As a result, we must use several powerful signal processing techniques in order to obtain a robust MI-BCI.

To begin with, we introduce some of the different existing BCIs briefly, as well as the data set that will be analysed in this work, which is the publicly available and extensively studied MI-BCI dataset 2a from the BCI Competition IV. Next, we present the MI-BCI problem as a general pattern recognition problem, and divide it into its main parts, signal preprocessing, feature extraction or dimensionality reduction and finally classification, the main focus of this work. The first two stages prepare the signals and extract the most relevant information in them, whereas in the latter the prediction is made. For the classifying step, a solution known as Linear Discriminant Analysis (LDA) has been widely used previously, however, other less explored methods do exist.

In the next chapter, we strive to provide some insight into the principles in which this classifier as well as other less common approaches are sustained, and finally we perform an experiment in order to compare the different solutions that these algorithms achieve when used in a MI-BCI problem.

Prior to making any predictions, a training stage is needed, in which the internal parameters that govern the behaviour of the algorithm are obtained. These parameters are calculated using observations similar to those that we will be required to classify. We study not only the classic supervised situation in which during training the intentions of the user are known, but we aim to expand the problem to the less studied unsupervised scenario, where this information is not provided.

Contents

Resumen III
Abstract V
Notation IX

1 Introduction 1
1.1 The brain: signals 1
1.1.1 P300 2
1.1.2 SSEVP 3
1.1.3 Motor Imagery 4
1.2 Physical interfaces 4
1.3 The data set 5
1.4 Summary 7

2 Motor Imagery detection as a pattern recognition problem 9
2.1 Modeling the EEG signals 10
2.2 Signal preprocessing 11
2.3 Dimensionality reduction 12
2.3.1 Common Spatial Patterns 13
2.3.2 Understanding the spatial filters 15
2.4 Classification 16
2.5 Summary 17

3 Classifiers 19
3.1 Supervised classifiers 19
3.1.1 Bayesian classifier 19
3.1.2 Linear Discriminant Analysis 23
3.1.3 K Nearest Neighbours 25
3.1.4 Nearest archetype 27
3.1.5 Gradient descent 28
3.1.6 Classifiers in a Riemannian manifold 31
3.2 Unsupervised classifiers 31
3.2.1 K-means 32
3.2.2 Gaussian EM 33
3.3 Label matching 38
3.4 Summary 38

4 Experimental results 39
4.1 Defining the experiment 39
4.2 Individual algorithms and their variants 40
4.2.1 Bayesian classifier 40
4.2.2 Linear discriminant analysis 40

4.2.3 K Nearest Neighbours 41
4.2.4 Nearest archetype 42
4.2.5 Gradient descent 43
4.2.6 Classifiers in a riemannian manifold 44
4.2.7 K-means 46
4.2.8 Gaussian EM 48
4.2.9 K-means on a Riemannian manifold 49
4.3 Comparing classifiers 50
4.3.1 Supervised classifiers 50
4.3.2 Unsupervised classifiers 54
4.4 Summary 55

5 Conclusions 57
5.1 Future lines of research 58
5.2 Closing statement 58

List of Figures 59
List of Tables 63
Bibliography 65
Glossary 69

Notation

k	Index marking a class from a set of possible categories
τ	Index marking a trial
$x, \mathbf{v}$	An observation, scalar or vectorial
$\mathbf{a}^T$	Transpose of $\mathbf{a}$
$\mathbf{A}^{-1}$	Inverse of the matrix $\mathbf{A}$
$\|\mathbf{A}\|$	Determinant of the matrix $\mathbf{A}$
$\|\|\mathbf{v}\|\|$	Euclidean norm of the vector $\mathbf{v}$
$\|\|\cdot\|\|_F$	Frobenius norm
$\log(\cdot)$	Natural logarithm
$\nabla f(\cdot)$	Gradient of the function f
$\mathbf{\Sigma}$	Covariance matrix or estimated sample covariance matrix
σ^2	Variance or estimated variance
$\boldsymbol{\mu}$	Mean or estimated mean
$P(\cdot)$	Probability mass function
$p(\cdot)$	Probability density function
$H(\cdot)$	Entropy
$p(A\|B)$	Conditional probability of A given B
$p(A,B)$	Joint probability of A and B
$N(\boldsymbol{\mu},\mathbf{\Sigma})$	Normal distribution defined by the mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$
$\mathbf{X}_\tau$	EEG recordings corresponding to trial τ
$\mathbf{w_i}$	Single spatial filter i
$\mathbf{y}_\tau$	Signal filtered with a single spatial filter
$\mathbf{W}$	Matrix composed by multiple concatenated spatial filters
$\mathbf{Y}_\tau$	Signal filtered with multiple spatial filters
$P(w_k)$	Prior probability of an element of class k appearing
Z_k	Decision region for class k
$g_k(v)$	Discriminant function for class k

1 Introduction

In summary, there are no small problems. Problems that appear small are large problems that are not understood

Santiago Ramón y Cajal

During the last decades, computers have become an integral part of our daily lives, changing drastically the ways in which we comunicate with the world and with each other. We do this through interfaces, which are the bridge between user and machine, the component that allows us to transmit our intentions to a computer system and interpret the information that it presents us with.

In the present, when a user desires to input a command into a computer, this instruction is first formulated on the brain of the subject and translated into a series of motor actions. These actions are converted into electric signals through a set of transducers such as a keyboard, mouse, touch screen or microphone. These signals are then processed and interpreted by the machine. In a similar manner, when the computer has information to comunicate to the user, it might do so through a different interface such as a screen or speakers, may it be through sounds or synthesised speach. The user then has to perceive this information and interpret it.

It is reasonable to think that if we could directly read these commands from the brain and write them in their natural form, these intermediate steps would be innecessary and we could comunicate with computers through the fastest and most natural interface. A system of this nature, that connects brain and machine and therefore eliminates the need of physical actions from the user is known as a Brain Computer Interface.

Due to the complex and delicate nature of the human brain, a system that satisfies these requirements does not exist and does not seem to be part of the near future. The stage of this process that carries a greater risk is to input signals into the brain, and although treatments for different disseases that do this in a very rudimentary level are currently implementable [36], this is not common. The existing interfaces provide enough information, are convenient and multiple accesibility options exist for people that cannot use them in their original form. As a result, this step becomes a secondary goal and conventional interfaces are used in most BCI projects.

Reading signals from the brain can be a less invasive process as it will be seen later. The existing solutions rely on brain patterns that a person can produce at will, however, this process is usually slow and inconvenient, and they do not result in an upgrade from the classic computer interfaces for most users. This being said, a common factor amongst most existing systems is the necessity to perform a movement to trigger them, which forbids a group of people with motor disabilities from using them. In these cases a BCI, a system that allows the user to input commands to a machine without physical interaction, would result in a drastic improvement in their quality of life, providing them with a new way to interact with their surroundings.

1.1 The brain: signals

The basic unit that forms the nervous system is a cell called neuron[1]. This type of cell is formed by a main body or soma, which receives signals from a set of protuberances known as dendrites. A long elongation

[1] Different kinds of neurons have been observed, some of which may present a different structure than the one described here and might perform different functions.

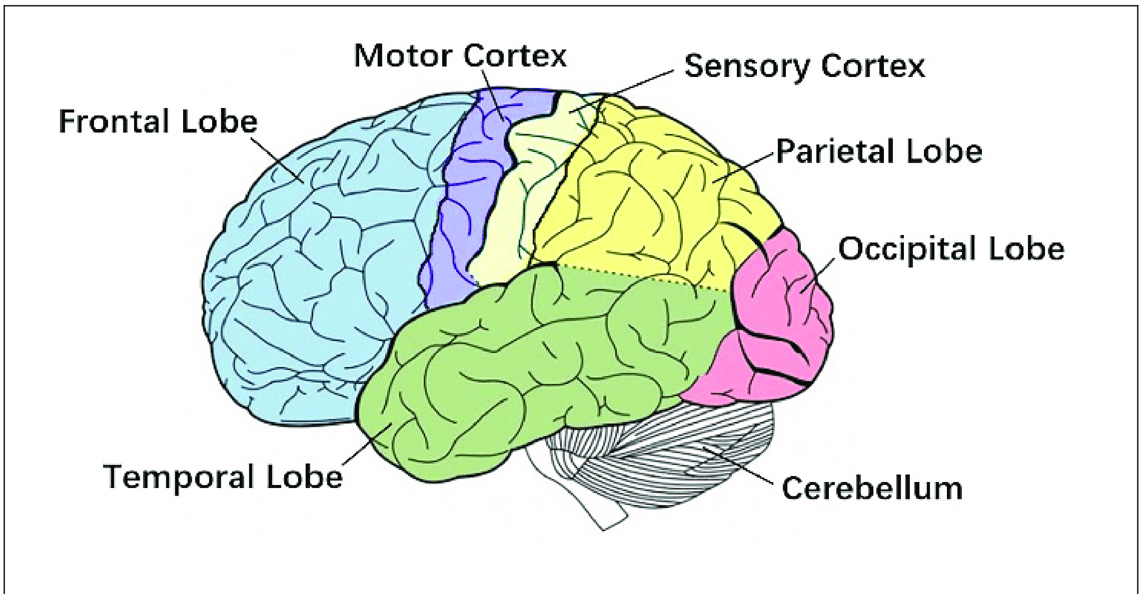

Figure 1.1 Computer image of a brain, with some of the principal regions marked in different colours. Image extracted from [28].

called axon extends from the soma and is finished by the synaptic terminals, which relay the signal to the next element, usually the dendrite of another neuron.

A synaptic connection may excite or inhibit the activity on a dendrite of the next neuron, and if a threshold is reached, the target neuron will release a pulse of electrochemical nature through its axon, transmiting the signal to other connected neurons.

When a large number of these cells interact, complex patterns emerge. Some of these are what we perceive as information from the senses, language processing, thinking or even something as abstract as conciousness.

In the cortex, the most external layer of the human brain, there are billions of neurons, and although it is not possible or practical to observe the individual activity of these cells, there are techniques to observe the evoked potentials that they produce on a more global level.

The cortex has been observed to be a distributed system, divided into different parts dedicated to distinct tasks [41]. Some of the more relevant for BCI are the visual cortex, situated on the back of the brain (occipital lobe), or the sensor and motor cortex, two joint strips that go from one side of the brain to the other, from left to right at approximately the center point and dedicated to perceiving stimulus and controlling voluntary movements.

The human brain has been studied extensively and although many questions still remain to be answered, some characteristic signals have also been observed under particular circumstances that can potentially be used in a BCI.

1.1.1 P300

If a subject is actively waiting for an event that will occur but the timing is random, when it happens a signal is triggered in the brain about 300ms after the stimulus. This electric signal named P300 can be registered with sensors on the cortex or scalp[2].

The P300 event related potential has been used to create different systems that have proved to be able to improve the quality of life of a patient such as a drawing program [20] or that have a high potential to do so like a music composing application [34]. However, all of these systems are based on the same working principle and are better described by the P300 speller [29].

In this system, a matrix of letters and commands is presented in front of the user. Some cells are highlighted randomly. This has been done in different ways. Row and column wise, one element at a time, a group of random elements simultaneously[3], presenting the marked element with a brighter light, inverting the colours of the element and background and so on. The subject focuses on the command that they wish to comunicate

[2] The different types of brain activity sensors will be discussed in the next chapter.

[3] These elements are not presented truly randomly as there are some limitations on adyacent and consecutive flashes.

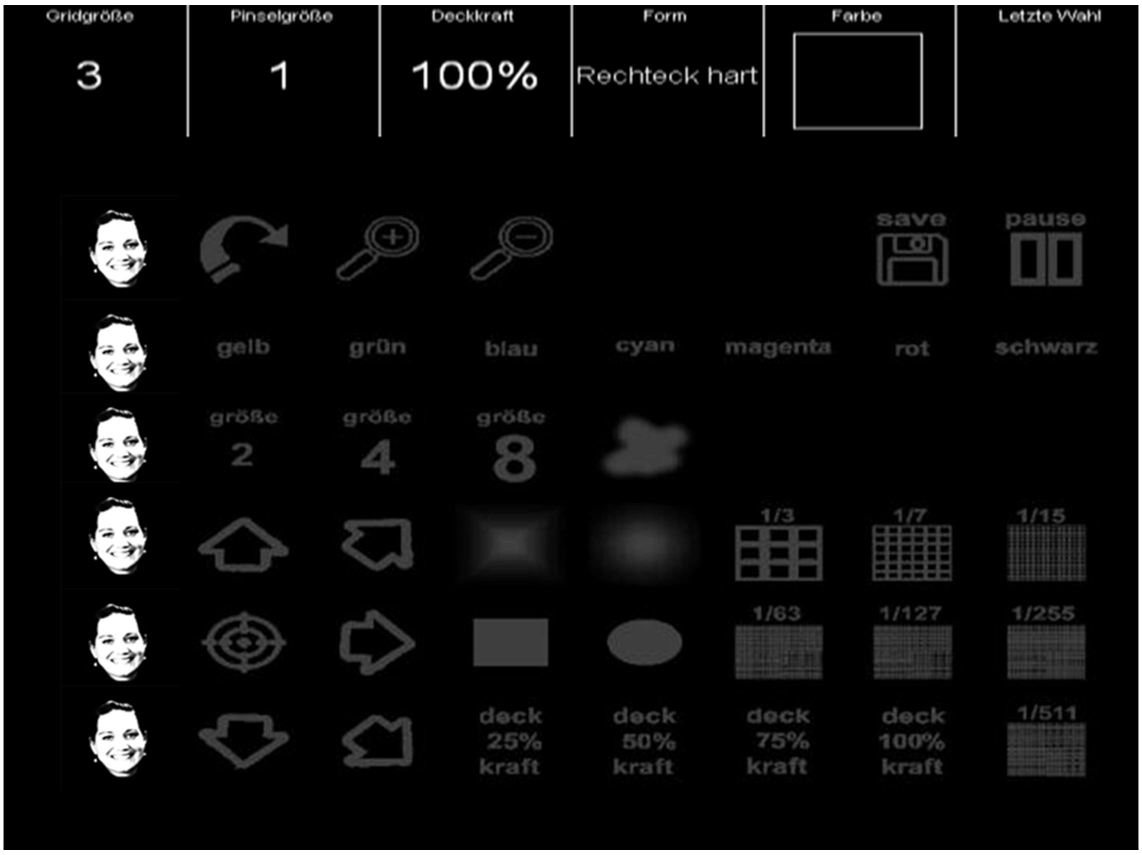

Figure 1.2 Matrix used in a P300 BCI. Instead of letters, commands such as brush shape, transparency, colour or zoom allow the user to paint. The instructions are highlighted by flashing the face of a famous character, Albert Einstein in this case, as this triggers a stronger response. The actual image was ommited in the original article due to copyright reasons. Image extracted from [20].

and approximately 300ms after the desired element is marked, the P300 signal is triggered inside the brain and registered by a transducer. The process is repeated a few times before making a decision to compensate for inaccuracies.

This system has been improved on using language models to predict the most likely letters in context in order to improve the prior probabilities of those letters appearing or adding some autocompletion sugestions as commands to make the system faster. Despite these efforts, the number of letters per minute typed with this BCI is very low.

In the case of other applications using the P300 BCI, the system is very similar, and the main change is the substitution of letters for other commands. The speed limitations of the system are concerning, however, the main problem is still accuracy, as the number of incorrect decissions is currently very high, which can make it frustrating for the user. Nonetheless, as it was mentioned earlier, this system provides an alternative for people in a locked-in state[4], and it has proven across several studies to have a positive impact on their lives.

1.1.2 SSEVP

When a person focuses their sight on a flickering light, a signal of the same frequency can be detected near the area of the cortex dedicated to the processing of optical stimulus. The BCI systems that employ this signal are known as Steady State Visual Evoked Potential (SSVEP) BCI. These usually work by detecting where the user is gazing, flashing lights at different frequencies near the multiple commands that the user may select.

The main advantages of this technique are the relatively robust signal detection, no need of user training and low level of mental exhaustion, as the user is not required to actively perform any tasks beyond looking at a screen. The discomfort produced by long periods of exposure to flickering lights has been an adressed

[4] The locked-in state refers to the situation in which a person is conscious but not capable of moving [24]

problem and alternative solutions have been proposed such as the use of moving objects or patterns instead of flickering lights [40].

Another system based on activity registered in the visual cortex which works with a different signal is the movement Visual Evoked Potential BCI (mVEP-BCI). This employs the N200, an evoked potential produced when the user looks at a moving object [27]. Its main use is the N200 speller, which has a similar setup to the P300 speller, but the cells are highlighted by a moving marker going through the different rows and columns in order.

1.1.3 Motor Imagery

The motor cortex is a part of the brain that controls voluntary movement, and movements of each part of the body can be mapped to activity in a different region of the motor cortex. This activity cannot be directly observed easily, however, a set of signals of unknown origin named Sensorimotor Rhythms (SMR)[5] may be used to perceive changes associated with this region. When there is no motor activity, the brain areas related to movement are at rest, and the SMR signal, which can be detected with several sensors as it will be seen later, is strong. This is called Event Related Sinchronization (ERS) and reflects an idle state. On the other hand, when the user performs a movement, the corresponding part of the cortex is activated and the signal fades near the area where the activity is located. This is known as Event Related Desinchronization (ERD).

This signal is even more interesting because a similar response is triggered when the movement is not phisically performed but imagined, creating the possibility of a Motor Imagery Brain Computer Interface. With this system, the user imagines a movement without performing it and a device is capable of reading the intention to perform this action, transforming it into a command for a computer. This opens the possibility of using it in cases where the user is in a locked-in state. Due to the potential of this technique, this will be the subject of this work.

Beyond this application, MI-BCI is also a promising tool for helping patients recover motor function after a stroke [43].

1.2 Physical interfaces

In order to detect Motor Imagery, the first step is to register a signal from the brain that contains the relevant information. For this purpose a transducer is necessary: hardware capable of converting a physical signal into an electrical signal that can be digitalized and registered in a computer where it can be processed and analysed.

Several sensors capable of detecting brain activity have been developed and improved over the last century[6], however, due to the large differences between them, not all of them are equally suitable or convenient for Motor Imagery detection.

Two techniques used in medicine are the functional Magnetic Resonance Imaging (fMRI) and Magnetoencelography (MEG). Although both of these provide good spatial resolution, they require machines which are expensive and non portable in order to register the signal. Moreover, the obtained temporal resolution is very low. In spite of these limitations that make these systems unsuitable for our task, the possibility of using them has been previously explored [45].

Another more extended approach to MI-BCI has been the Electrocorticography (ECoG). For this recording of electrical brain activity, a set of electrodes are placed directly on the surface of the cortex.

This method has some mejor advantages such as being portable, having high temporal resolution, good signal to noise ratio and large bandwidth. The system does not require a time consuming setup before every session and the electrodes are always located in the same place. All these features make ECoG signals a great candidate for the implementation of a MI-BCI system, however, there are some important drawbacks that prevent ECoG from being our case of study.

A craneotomy, a delicate medical procedure in which a part of the craneum is removed is required in order to access the cortex. Furthermore, the electrodes have to be placed on the brain directly and remain in contact without damaging it. This have resulted in most studies on ECoG being carried out with patients that suffered other illneses and had to be subjected to a craneotomy for a different reason, and the periods of study were limited due to the risk of infection. Now, efforts are being made towards creating long term solutions, as this poses another challenge beyond the analysis of the signals [31, 32].

[5] There are contributions to this signal from outside the motor cortex, specially from the sensory cortex, another region responsible for perceiving stimulus [25]

[6] Signals from the brain were recorded before the first digital computers, this data was stored on paper [1].

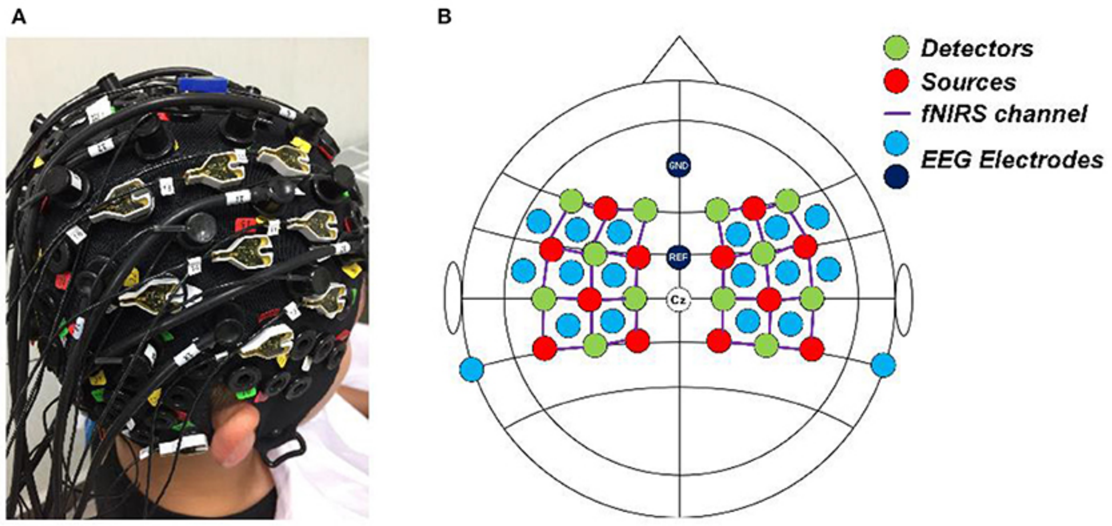

Figure 1.3 This figure shows a setup for a hybrid EEG-fNIRS system. Image extracted from [26].

If we ignore these concerns and argue that, once a reliable complete system is developed, the procedure to implant it might be worth considering for a person that has a certain condition and whose quality of life might be improved drastically by this surgery, some other problems remain. The first one is user rejection, as a possible user may not agree to be subjected to this procedure, may it be because of the objective reason that it can be dangerous, inconvenient or uncomfortable or for any other subjective motivation such as considering having a brain implant as alien, immoral or unethical.

Once that we decide to move away from invasive procedures, other non invasive methods exist. The first one is Near-Infrared Spectroscopy (fNIRS). This imaging technique, instead of measuring the electromagnetic signals produced by the brain, uses electromagnetic sources in the near infrarred wavelength, which penetrate the skull, scatter and bounce, and can be registered by optic sensors in order to detect the hemodynamic response, blood flow towards groups of active neurons in the cortex. The resulting system is portable and relatively affordable. Despite having a good temporal resolution, it cannot provide good spatial resolution and its application in MI-BCI has been limited [16].

The most extended non invasive method of obtaining brain signals for MI-BCI is the Electroencefalogram (EEG). This is similar to the ECoG in the sense that it registers evoked potentials produced in the cortex. The main difference with the ECoG is that the signals are registered on the scalp. This results in a lower signal to noise ratio, narrower bandwidth and greater interference from other sources such as ocular or muscular movement. It also requires a longer time to install before each session as the electrodes have to be placed carefully on the scalp of the user. The main advantage is that the process is non invasive, making it safer, easier to test and more accessible to potential users.

Finally, the possibility of using functional Near-Infrared Spectroscopy (fNIRS) in conjunction with EEG signals has been proposed [44], as both of these methods are compatible and can be used simultaneously. Unfortunately, researching this technique would require obtaining a large and reliable data set, which is outside of the scope of this work.

In conclusion, the chosen physical interfacce for this study will be EEG. The reasons that motivate this decission are the relative low cost, portability, and it being a non invasive technique. Despite not being the only method satisfying these conditions, EEG was the most reliable and previously studied amongst them.

1.3 The data set

Part of the challenge of testing processing techniques on EEG signals is obtaining a reliable data set. We will use the one known as Data Set 2a [7], part of the BCI Competition IV [6]. It was provided by the Institute for Knowledge Discovery, part of Graz University of Technology.

The advantage of using this data set is that it has been extensively analysed before, and rather than worrying about the quality of the obtained data, we may focus our efforts towards the signal processing task. Furthermore, it is possible to compare our work with other studies that use the same data set, and it is easier for others to verify our results and compare theirs with our own.

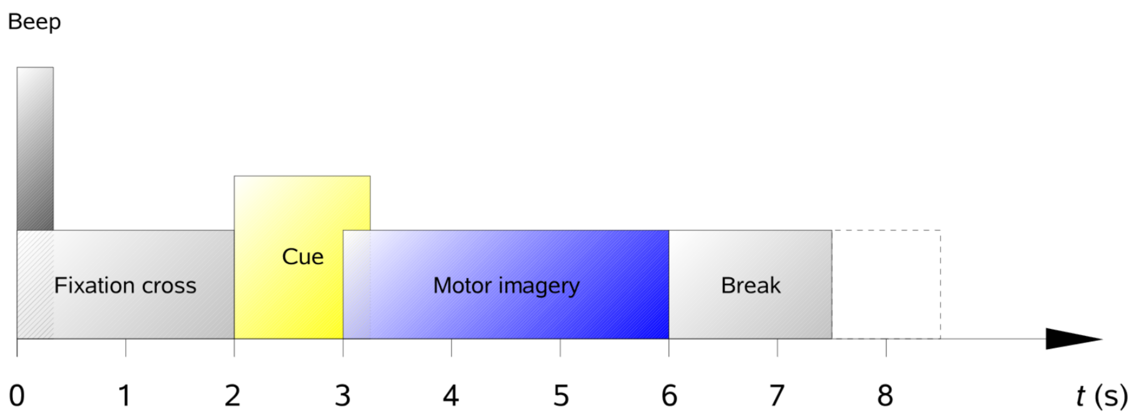

Figure 1.4 This figure shows the timeline of a single trial. Extracted from [7].

The following explanation of the experiment is a summary of the description provided with the data set [7]. Nine different subjects participated on the recording of the signals, and they were seated on a comfortable armchair in front of a screen during the recolection procedure.

The process was divided into two sessions expanding across two days. The first day the training data set was registered, and the second day the test session was recorded. Although this was the original purpose of these sessions, we may treat them independently as it will be seen later in chapter 4.

The structure of every session was similar. First, a few minutes of Electrooculography (EOG) signals were rescorded, as part of the challenge was to use this information in order to reduce the influence of the eye movement on the EEG signal [38]. The subjects were asked to look at a black screen with a fixation cross on it for two minutes, then during another minute they remained with their eyes closed, and finally one minute of eye movement was registered. After that, six runs with short breaks in between were conducted.

Each run was composed of fourty eight trials, twelve from each of the four possible movements: left hand, right hand, feet or tongue. This amounts to a total of 288 trials per session, 72 for each class.

The trials were registered following the next scheme. A short sound along the apperance of a fixation cross on screen indicated the beginning of the trial. This fixation cross remained for two seconds on screen until it was replaced by a visual cue that indicated the movement that the subject should perform. The arrow pointing in one of four directions remained on screen for 1.25 seconds. The subject would begin the Motor Imagery task and the fixation cross would reappear for six seconds during which the subject would continue with the task. Then, a short break lasting one and a half seconds would follow before the begining of the next trial.

The EEG signals were recorded using 22 Ag/AgCl electrodes placed on the scalp of the subjects following the international 10-20 system [22]. All the signals, EEG and EOG, were sampled at 250Hz and bandpass filtered between 0.5Hz and 100Hz. A notch filter[7] was used to remove the interference of the 50Hz power line. The sensitivity of the amplifier was configured to be $100\mu V$ in the case of the EEG signals and $1mV$ for the EOG signals.

[7] A notch filter is a narrow band-rejection filter

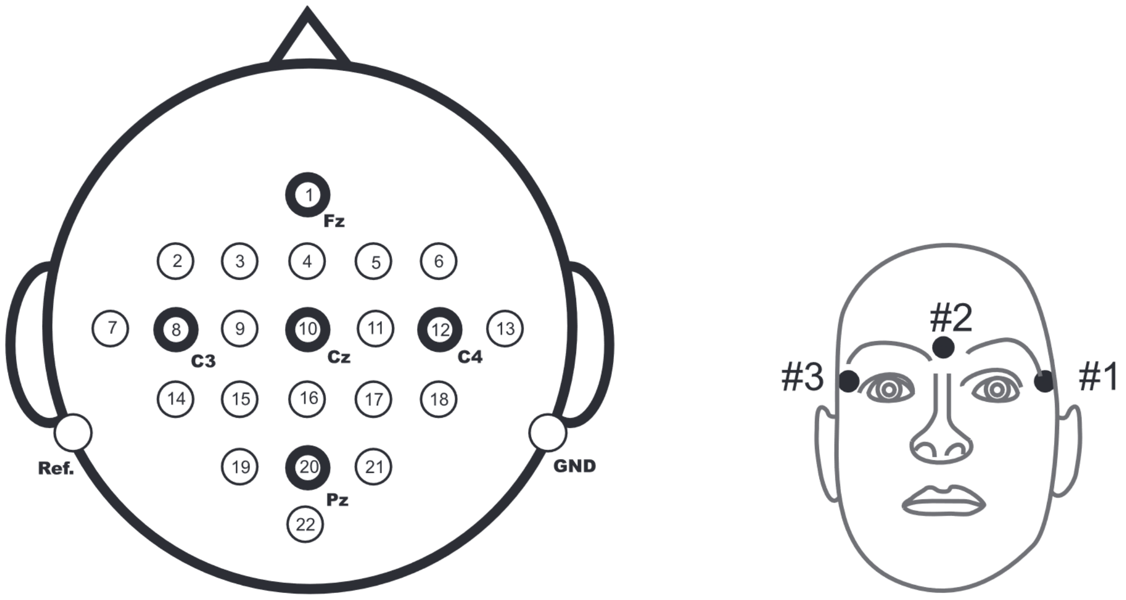

Figure 1.5 This figure shows the positioning of the EEG and EOG electrodes. Extracted from [7].

1.4 Summary

In this chapter we have introduced the basic concepts on what constitutes a Brain Computer Interface, what motivates the development of such systems and the signals that have been traditionally used for this purpose. Amongst these, Motor Imagery BCI was the most promising system and therefore it was chosen to be the subject of this work.

We have also analysed the main physical interfaces that are capable of recording brain activity. Most of these were very different and had their advantages and disadvantages, and we came to the conclusion that the technique known as Electroencefalogram (EEG) was the one that better suited our needs due to being affordable, portable and non invasive.

Finally, we described the data set that we will use in our experiments, which is the one known as Data Set 2a [7] from the BCI Competition IV [6]. This provides a reliable testing ground for our experiments, allowing us to verify our results.

2 Motor Imagery detection as a pattern recognition problem

From rainbows, river meanders, and shadows to spider webs, honeycombs, and the markings on animal coats, the visible world is full of patterns that can be described mathematically.

John A. Adam

The objective of a Motor Imagery Brain Computer Interface is to monitor brain activity and make a prediction about the intention of the user, who is imagining an action amongst a set of possible movements.

This problem in which some form of signal is received as input to the system and a prediction about the category in which it belongs to has to be made is part of the pattern recognition framework, and, in particular, a subbranch of this known as statistical classification.

These problems share common features and are of great interest, as the solutions can be applied in a wide range of fields such as communications, biometrics, computer vision, medicine or data science [19].

Independently from the application, there is an underlying procedure when approaching a problem of this kind that can be summarised in three steps.

The first one is signal preprocessing. This consists on operating on the signals in order to eliminate information that could hinder the process on a later stage. Some examples could be applying a notch filter[1] to an Electrocardiogram (EEC) signal[2] to remove the 50Hz or 60Hz noise induced by the power supply, low-pass filter an image to reduce Gaussian noise or use a high-boost filter on a picture to facilitate the recognition of edges[3]. An extended problem belonging to the preprocessing of signals for pattern recognition in pictures is image segmentation, where the relevant parts of the signal are extracted from the background and separated from other important elements.

It is clear that, due to the large number of signals that we can study and the intrinsic differences between them, this stage varies drastically with the type of signal being analysed, and multiple procedures could be carried out in succession in order to achieve a better result.

Another action that could be performed before continuing with the pattern recognition task and that is distinct, yet related to signal preprocessing is cleaning the data set. Any signal that was corrupted during the capture process or that is not relevant can be removed to prevent it from interfering with the correct operation of the system. An example of this would be to eliminate the trials in which the microphone was saturated and the signal clipped from a data set intended for musical instrument separation from a single track[4], as these might not be representative of the intended use of the system.

The second stage of signal preprocessing is known as feature extraction or dimensionality reduction, names which point to its two main objectives. The first one is to reduce the signal we wish to classify to a limited set of relevant characteristics that describe it called features.

[1] A notch filter is a narrow band-rejection filter

[2] An EEC is a recording of heart electric activity

[3] A high boost filter emphasizes high frequencies, which contain information about edges in a picture, as these are characterized by sudden changes in intensity

[4] This is not a classification problem per se as the task does not involve categorizing, but it is a pattern recognition problem.

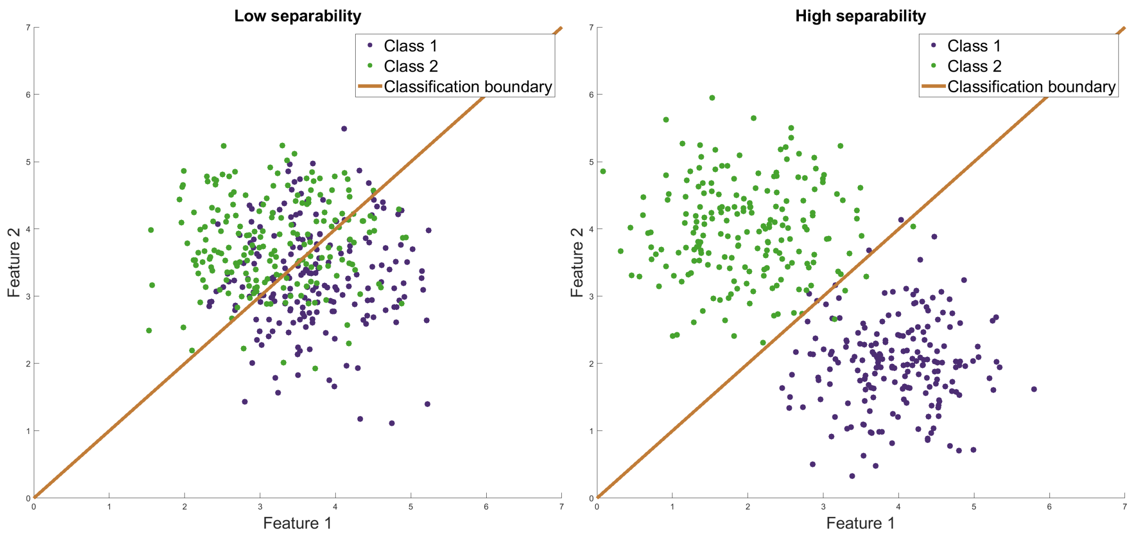

Figure 2.1 By comparing these two representations of classifiers using synthetic data, it is clear that the separability of the classes limits the performance of the classification algorithm, and therefore selecting appropriate features is a crucial step in the process .

The other name, dimensionality reduction, points to the fact that the obtained features are a representation of the original signal in a lower dimensional space. A significant reduction in the volume of information makes it easier or even computationally feasable for an algorithm to learn about the particularities that define each class. This being said, the large amounts of information publicly available and advancements in computing power in the last few decades have resulted in algorithms that required a vast amount of samples or that once were not possible to compute to be implemented. An example of this are convolutional neural networks, that may be able to receive pictures composed of hundreds of thousands of pixels and learn the patterns in them in an acceptable time, bypassing the feauture extraction step. However, that is not completely true as part of the learning process consists of finding out which are the best features and extracting them, incorporating this stage in the classifier itself.

Here we may introduce the concept of separability. Our features need not only to be disctintive of one class but different from those of the other. For example, if we wanted to create a system that classified pictures of lemons and bananas, the colour would not be a good choice for a feature, as even if lemons can be recognised easily for their yellow colour, so are bananas. A possible parameter to add to our system would be the ratio between the longest axis and the shortest axis[5]. As bananas are long and thin, this would yield a relatively large number, and lemons are almost round, resulting in a ratio closer to one. In general, the distinct selected features should not be repeated or correlated, as this would not facilitate the process of classification. Obtaining the set of features that would result in a better performance is a complex task, and specific algorithms to approximate this problem do exist [35].

The final stage of the process is classification, in which feature vectors containing the particular values that the different features take for a certain sample are used as the input for a classification algorithm, which outputs a prediction of the corresponding class. This phase can again be divided into two. The first one is a learning stage, during which the classifier extracts the relevant information from a set of training samples in order to configure the internal parameters that will gobern the behaviour of the algorithm. The second one is predicting, in which new samples never observed before can be assigned to a class based on the traits that they share with the training samples.

2.1 Modeling the EEG signals

Before starting the classification process, it is important to have an understanding of the underlying model of the observations, as this can be of great help when deciding which are the most important characteristics to extract or which processes will preserve the most relevant information for classification.

[5] These values could be obtained by the use of the geometrical moments of order two as well as by other more complicated means [13].

The brain activity registered with an Electroencefalogram can be understood as being mainly produced by a set of localized sources, which we may denote as $s_j(t)$, on distinct parts of the cortex. The number of sources may vary and we will assume that there are N_s of them. Each one is atenuated due to the propagation inside the head of the user before reaching an electrode and being registered as part of the signal $x_i(t)$, where i represents the number of the EEG channel from 1 to N_x. Each channel is therefore a linear combination of the sources, plus some random noise that we will denote as $n(t)$ [33].

Each registered signal can therefore be written as

$$x_i(t) = \sum_{j=1}^{N_s} a_{ji} s_j(t) + n(t) \tag{2.1}$$

Where a_{ji} is a coefficient that models the attenuation from the jth source to the ith electrode. Although this value could vary due to movements of the EEG helmet or the gel[6] drying, we will assume that it is approximately costant during an experiment.

The EEG signals are then sampled, as they will be processed by a computer. This means that a single channel can be represented with a vector $\mathbf{x_i} = [x_i(t_0), x_i(t_0+T_s), ..., x_i(t_0+(L-1)T_s)] \in \mathbb{R}^{1\times L}$, where T_s is the sampling period, t_0 is the starting time of the recording and L is the number of samples registered. Although this is done indirectly, the information registered from the active sources suffer a similar sampling process and each source can be interpreted as a vector $\mathbf{s_j} \in \mathbb{R}^{1\times L}$:

$$\mathbf{x_i} = \sum_{j=1}^{N_s} a_{ji} \mathbf{s_j} + \mathbf{n} \tag{2.2}$$

If we define $\mathbf{a_i} \in \mathbb{R}^{1\times N_s}$ as $\mathbf{a_i} = [a_{1i}, ..., a_{N_s i}]$ and $\mathbf{S} \in \mathbb{R}^{N_s \times L}$ as the row wise concatenation of the different sources, this linear combination can be written more conveniently.

$$\mathbf{x_i} = \mathbf{a_i S} + \mathbf{n} \tag{2.3}$$

Now we have N_x different EEG signals that, similarly to what we did with the sources, we may concatenate row wise into the same matrix, obtaining the complete sampled EEG signal $\mathbf{X} \in \mathbb{R}^{N_x \times L}$. We may write the EEG observations in matrix form as a combination of the active signals in the brain.

$$\mathbf{X} = \mathbf{AS} + \mathbf{N} \tag{2.4}$$

Where $\mathbf{A} \in \mathbb{R}^{N_x \times N_s}$ contains the different coefficients a_{ji}. Although the subindex ji was appropriate to represent the attenuation due to the propagation from the source j to the electrode i, in this case the term a_{ji} will be located on the ith row and the jth column of A.

This last expression is only possible because the process of propagation of the signals can be considered instantaneous and therefore there is no delay between a source being registered in different electrodes.

As it will be seen in the next section, the EEG observations will be spliced into trials, each one of which will be denoted as $\mathbf{X}_\tau$. Another relevant parameter that will be used later is the covariance of the different EEG signals in each trial, denoted as $\mathbf{\Sigma}_\tau$ and the covariance of the signals across all trials belonging to the same class, $\mathbf{\Sigma}_k$, where k is the corresponding category. The calculation of these will be detailed in the following section, as this is a delicate process that significantly affects the performance of the classifiers.

2.2 Signal preprocessing

As it was described in the previous chapter, the EEG signals used on the competition were preprocessed before being published. After recording, the signals were bandpass filtered between 0.5Hz and 100Hz. Furthermore, a notch filter removed the interference of the 50Hz line signal.

Each session was provided as a continuous recording, where along the EEG and EOG channels there was a data track providing information about different events such as when a trial started or ended, a cue appeared on screen or a new run begun. Other fields included further information such as if an artifact affected a trial

[6] Usually, a conductive gel or paste is used to connect the electrodes and the scalp, providing a low impedance path for an accurate measurement.

or the true labels of the trials. Although this last information was not included at first for the second group of sessions[7], when the competition ended it was released to the public, and we may use this information.

In order to make it easier to work with this amount of data, the sessions were spliced into trials, and the two seconds which were considered to be the most relevant for Motor Imagery detection, that is 500 samples, were saved in a separate matrix for each trial. The selected window starts 0.5 seconds after the Motor Imagery begins. Before being stored, the trials were bandpass filtered between 8Hz and 32Hz. This is the band in which it is estimated that most of the SMR signals can be found. The signals were treated to make the process of testing different algorithms easier and faster, as the different trials may be accessed randomly in a convenient way and it is not necessary to repeat the filtering step for each experiment.

This process of splicing, filtering and converting the data into a more convenient format was carried out by the Department of Signal Theory and Communications from the University of Seville and although the basics have been discussed here, further detail may be found in their publications [42, 30].

During the description of the feature extraction stage it will be seen that a good estimate of the class conditional covariance matrices[8] and the covariance matrices for each trial are necessary. Although this could be considered part of the dimensionality reduction step, we may calculate these here in order to reduce the data and computational load on latter stages, as the line separating these may be blurred in favour of efficiency.

Moreover, these matrices will be estimated using the instantaneos power normalization of the sources described in [33]. The reason for doing so is that artifacts, which are undesired interferences produced by external factors such as muscle or eye movement, may introduce large fluctuations on the signals over time, hindering the estimation of the covariance matrices of the trials. Furthermore, a few trials affected by these artifacts might dominate the estimation of the class conditional covariance matrices. With the normalization, this is partially averted, as every trial contributes in an homogeneous amount to this estimation. As we are implementing a protection against artifacts, an algorithm to reduce the effect of eye movement using EOG signals as it was originally intended on the competition is no longer needed.

2.3 Dimensionality reduction

Now that the EEG signals have been prepared for classification, the next step is to extract the most relevant features from them. As we saw in the previous chapter, we may think about the Sensorymotor Rhythm (SMR) signals as being generated in a localised area of the cortex responsible for the motor task that the user is performing. When an imaginary movement is carried out, two events happen. The first one is an Event Related Desinchronization (ERD) in the area corresponding to the movement. The second is an Event Related Synchronization (ERS) in the location of the cortex related to the other class, as there is no motor activity in this region. The ERS can be located due to an increase in the power of the signals, and the ERD has the opposite property.

Originally, the signals were filtered to eliminate the components under 8 Hz, however, after splicing them into trials, we centered them, assuring that the trials had zero mean. This has an advantage. If we consider that every value that the signal may take is equally likely, the mathematical formula for the variance of the signals is proportional to the energy of the windowed signal. Furthermore, if we were to compute the covariance matrix of all the EEG signals belonging to the same trial, we would not only have information about the energy recorded in the different EEG channels but also about how they relate to the others. Due to the propagation of the SMR inside the skull, scalp and other internal layers that protect the brain from physical damage, the same signals are registered in multiple electrodes at different distances from the source, which is a phenomenon that we must account for, specially when using a large number of electrodes.

Now, we could use this information in order to train a classifier directly, yet these features might not be the most relevant, and as we mentioned, the variance of the signals from the different channels contain redundant information. This hinders the performance of those classifiers that assume that the features are uncorrelated, and in the case of a large number of electrodes, the high dimensionality of the feature space makes the classification task even more difficult. Although there are multiple methods to select relevant features in EEG signals, the most widely used is known as Common Spatial Patterns (CSP).

[7] The second session of the first subject was used as an example and it was completely available since the begining of the competition.

[8] The term class conditional covariance matrix is used here to refere to the covariance matrix that is representative of all the trials belonging to the same class

2.3.1 Common Spatial Patterns

The dimensionality reduction algorithm known as Common Spatial Patterns was originally proposed as an extension of the Principal Component Analysis (PCA) method. This is based on the concept of finding a linear transformation of the original feature space in which the most defining characteristics would lie along the direction of the generational vectors of the space which have one element equal to one[9].

As the most relevant components are found first, the last dimensions might contain very little information and may be discarded, reducing the number of features that are input to the classifier. In general, these combinations would lack any interpretable meaning. As we will see later, this is not the case for CSP applied to EEG signals.

In our case, we had a set of trials belonging to two different classes. Each one of them is a sampled EEG signal, a matrix $\mathbf{X}_\tau$ of the form $\mathbb{R}^{C\times L}$, where L is the length of the recording in samples and C is the number of channels.

We may look for a filter $\mathbf{w} \in \mathbb{R}^{C\times 1}$ that combines the information of the multiple channels into a single signal, thus reducing the dimensionality of the features and the amount of redundant information. This is done with a linear combination of the EEG channels, and the resulting signal $\mathbf{y}_\tau \in \mathbb{R}^{1\times L}$ can be obtained as:

$$\mathbf{y}_\tau = \mathbf{w}^T\mathbf{X}_\tau \tag{2.5}$$

This filter is called a spatial filter because the combination is only done along the spatial dimension, corresponding to the different EEG channels, and not the temporal dimension.

As we introduced previously, the most relevant information was not in the EEG signals themselves but in their sampled covariance matrix $\mathbf{\Sigma}_\tau \in \mathbb{R}^{C\times C}$. We may also define the class conditional covariance matrices as the covariance matrices obtained for all the trials belonging to the same class, $\mathbf{\Sigma}_k, k = 1,2$. The effect that the spatial filter has on these matrices is:

$$\sigma_k^2 = \mathbf{w}^T\mathbf{\Sigma}_k\mathbf{w} \tag{2.6}$$

Where σ_k^2 is a real positive number, the variance for each class.

In order to find a value of $\mathbf{w}$ that is relevant, we may define an objective function $R(\mathbf{w})$.

$$R(\mathbf{w}) = \frac{\sigma_1^2}{\sigma_2^2} \tag{2.7}$$

This is justified because when a movement belonging to certain class is carried out, the variance of the signal proceeding from that area of the cortex should be minimum due to the ERD and the variance of the signal proceeding from the opposite region should be maximum as a result of the ERS. The ratio of variances reflect these two conditions. Both maximums and minimums correspond to a maximum difference in variance.

Replacing (2.6) in (2.7):

$$R(w) = \frac{\mathbf{w}^T\mathbf{\Sigma}_1\mathbf{w}}{\mathbf{w}^T\mathbf{\Sigma}_2\mathbf{w}} \tag{2.8}$$

And finding the gradient, accounting for the fact that the covariance matrices are symmetric[10]:

$$\nabla R(\mathbf{w}) = \frac{2(\mathbf{w}^T\mathbf{\Sigma}_1\mathbf{w}\mathbf{\Sigma}_2\mathbf{w} - \mathbf{\Sigma}_1\mathbf{w}\mathbf{w}^T\mathbf{\Sigma}_2\mathbf{w})}{(\mathbf{w}^T\mathbf{\Sigma}_2\mathbf{w})^2} \tag{2.9}$$

The optimal value of the spatial filter $\mathbf{w}_*$ is either a maximum or a minimum. In order to find it we may equate this gradient to zero to locate the critical points:

$$\mathbf{\Sigma}_1\mathbf{w}_*\mathbf{w}_*{}^T\mathbf{\Sigma}_2\mathbf{w}_* - \mathbf{w}_*{}^T\mathbf{\Sigma}_1\mathbf{w}_*\mathbf{\Sigma}_2\mathbf{w}_* = 0 \tag{2.10}$$

$$\mathbf{\Sigma}_1\mathbf{w}_*\mathbf{w}_*{}^T\mathbf{\Sigma}_2\mathbf{w}_* = \mathbf{w}_*{}^T\mathbf{\Sigma}_1\mathbf{w}_*\mathbf{\Sigma}_2\mathbf{w}_* \tag{2.11}$$

[9] As a result of the definition of generational vector, the other elements would be zero.

[10] The covariance matrices are Symmetric and Positive Definite (SPD), which justifies the fact that $\mathbf{w}^T\mathbf{\Sigma}\mathbf{w}$ is always a positive number. This also implies that the matrix is symmetric, which is the property that is needed here.

$$\mathbf{\Sigma}_1\mathbf{w}_* = R(\mathbf{w}_*)\mathbf{\Sigma}_2\mathbf{w}_* \tag{2.12}$$

This is a generalized eigenvalue problem [17], where the resulting eigenvectors are $\mathbf{w}_*$ and the associated eigenvalues are the corresponding values of $R(\mathbf{w}_*)$.

If instead of having an objective function $R_1(\mathbf{w}) = \sigma_1/\sigma_2$ we were to maximize the variance of the second class and minimize the variance of the first class, we would have the function $R_2(\mathbf{w}) = \sigma_2/\sigma_1$. As $R_1(\mathbf{w}) = 1/R_2(\mathbf{w})$, each local maximum of $R_1(w)$ is a minimum of $R_2(\mathbf{w})$ and vice versa, meaning that both functions share critical points. Both problems are essentially the same, however, the eigenvalues obtained with one method or the other will be reciprocal.

$$\mathbf{\Sigma}_2\mathbf{w}_* = R_2(\mathbf{w}_*)\mathbf{\Sigma}_1\mathbf{w}_* = \frac{1}{R_1(\mathbf{w}_*)}\mathbf{\Sigma}_1\mathbf{w}_* \tag{2.13}$$

There are as many lineally independent solutions as the dimension of the original matrices, that is, as as many solutions as EEG channels. This being said, not all of them are equally relevant. Those eigenvectors associated with extreme eigenvalues contain more information. This is because as the eigenvalues were related to the value that the function $R(\mathbf{w}_*)$ takes, and therefore a larger value corresponds to a larger local maximum, and a smaller eigenvalue is related to a smaller local minimum. It is therefore usual to choose an even number of spatial filters p, equal to or in most cases less than the number of EEG channels. These are selected alternatively from those related to the highest and lowest available eigenvalues. Furthermore, this selection eliminates inflection points from the possible solutions, as by equating the gradient to zero in (2.10) these were obtained as critical points along with the local maxima and minima.

A matrix of spatial filters $\mathbf{W} \in \mathbb{R}^{C\times P}$ can be formed as the concatenation of the spatial filters, $\mathbf{W} = [\mathbf{w_1},\ldots,\mathbf{w_{p/2}},\mathbf{w_{c-p/2}},\ldots,\mathbf{w_c}]$, where the vectors $\mathbf{w_i}$ are ordered according to their associated eigenvalues, c is the total number of channels and p is the desired number of filters.

We may now filter the EEG signals by extending equation (2.14):

$$\mathbf{Y}_\tau = \mathbf{W}^T\mathbf{X}_\tau \tag{2.14}$$

Where $\mathbf{Y}_\tau \in \mathbb{R}^{L\times P}$. Now, as the filters were eigenvectors and therefore orthogonal, the different signals in $\mathbf{Y}_\tau$ do not contain redundant information, and we may use the variance of these signals as features. This being said, and as it was hinted at during the description of the preprocessing stage, there is another efficient method for computing these same features without the need of using the original signals. If we have the covariance matrix of a trial Σ_τ:

$$\mathbf{\Sigma}_{f\tau} = \mathbf{W}^T\mathbf{\Sigma}_\tau\mathbf{W} \tag{2.15}$$

Where $\mathbf{\Sigma}_{f\tau}$ is the covariance matrix of the filtered trial. The main diagonal of this matrix contains the variance of the filtered signals. This is convenient because instead of working with the trials, which are large matrices, the covariance matrices of the trials can be calculated at the beginning of the dimensionality reduction step, or even during the preprocessing stage. This allows for a normalization of the power of the sources, which in turn leads to a better estimation of the class conditional covariance matrices and better spatial filters [33].

Instead of working with the variance of the signals directly, it is the standard procedure to use the natural logarithm of these values as features. This is because the variance of the signals as a random variable is not normally distributed, but its logarithm is, which is an interesting property that can be used during the classification stage. This is possible as well because the logarithm is a monotonically increasing function, and as a result, if $A > B$, $\log(A) > \log(B)$.

Unfortunately, CSP has some limitations. The first one is that it is a supervised algorithm, meaning that the samples used to calculate the spatial filters must be labeled. An alternative version that is not constrained by this has been proposed [30]. This is based on the idea that for a mixture of normal distributions and under the same conditions imposed over CSP, the kurtosis[11] shares critical points with the CSP objective function. This is then expanded to other mixtures of elliptic distributions not necessarily Gaussian.

Another problem is that, although this will be the case of study of this work, CSP can only be used if there are two different classes. Despite this fact, alternatives for the multiclass problem do exist, including both extensions of CSP [18] and other methods that are not directly related to this technique [12].

[11]This is the fourth order normalised moment of the distribution projected onto one dimension, $k_y(\mathbf{w}) = E[y^4]/E[y^2]^2$, where $E[\cdot]$ represents the expected value.

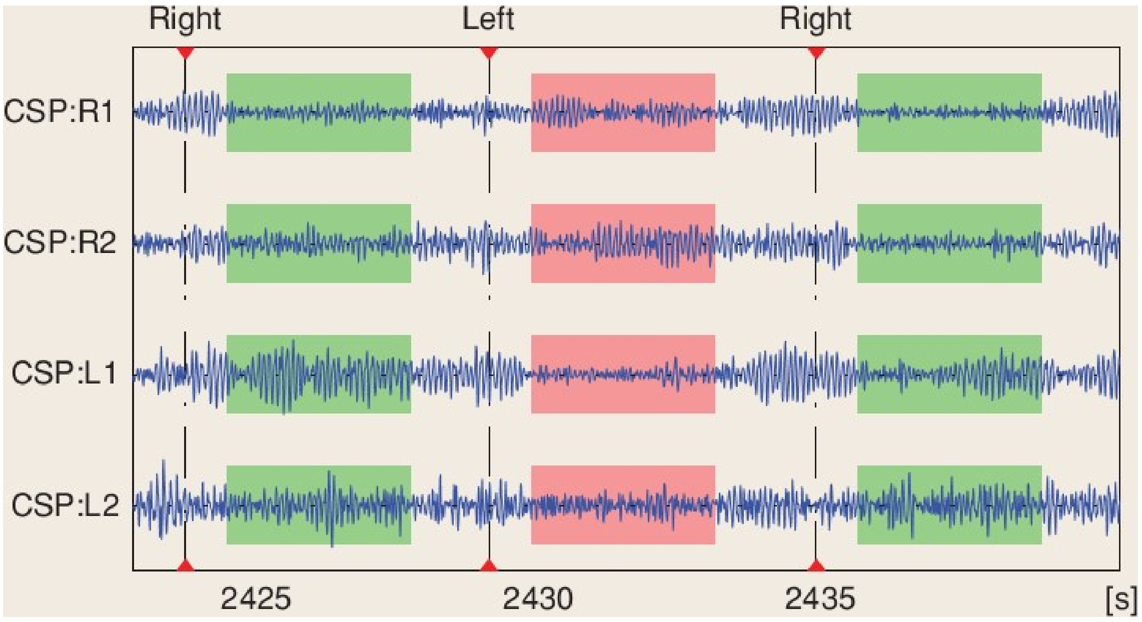

Figure 2.2 EEG signals filtered with four different spatial filters are shown. The first two filters minimize the variance of the right hand movements (in green) and the second two minimize the variance of the left hand movements (in red). Image extracted from [5].

2.3.2 Understanding the spatial filters

One great advantage of CSP is that, unlike other dimensionality reduction algorithms, the resulting spatial filters have a clear interpretation, and therefore this information can be used for different purposes. The first one is to better understand the underlying principle that supports this algorithm. Furthermore, in the context of a practical use of a BCI system, a professional may use a graphic representation of the filters to interpret whether the user is performing the Motor Imagery task as it is intended. This is specially relevant in the use of MI-BCI for rehabilitation purposes, where if done incorrectly, the recovered movement may be abnormal and even hurtful for the patient [43].

It has been stablished that the SMR can be trazed to localized regions of the cortex where the sensorymotor actions are processed. These signals then propagate through the protective layers of the head and reach the different electrodes after suffering an attenuation, which will be larger for those sensors located further away from the source. These signals will be registered at approximately the same time in all electrodes, as they propagate as an electromagnetic wave, nearing the speed of light in vacuum, and they are measured in different electrodes centimetres appart. Therefore, we may then consider them to be registered at the same sampling time in all channels.

The filters obtained with CSP solve the mixing problem proposed in 2.1, finding two signals, one with maximum variance and other with minimum, and tracking them to their original sources inside the brain. By solving this problem of reverting the process of propagation of the signals, the result is more relevant for classification than the original mixture.

As the resulting filtered signals are linear combinations of the original EEG channels, it is possible to create a visual representation of the estimated gain applied to the signals proceding from different locations of the brain. At each point, the gain is calculated as the coefficient that multiplies the signal registered at each channel divided by the square of the distance that separates the point from the corresponding electrode. This is to account for the propagation of the signals inside the head. Regions near electrodes with low coefficients will have a low gain, and therefore they are associated with places where an ERD occurs. The opposite is also true; regions with a high gain are close to electrodes with a larger coefficient and they mark the place where an ERS happens.

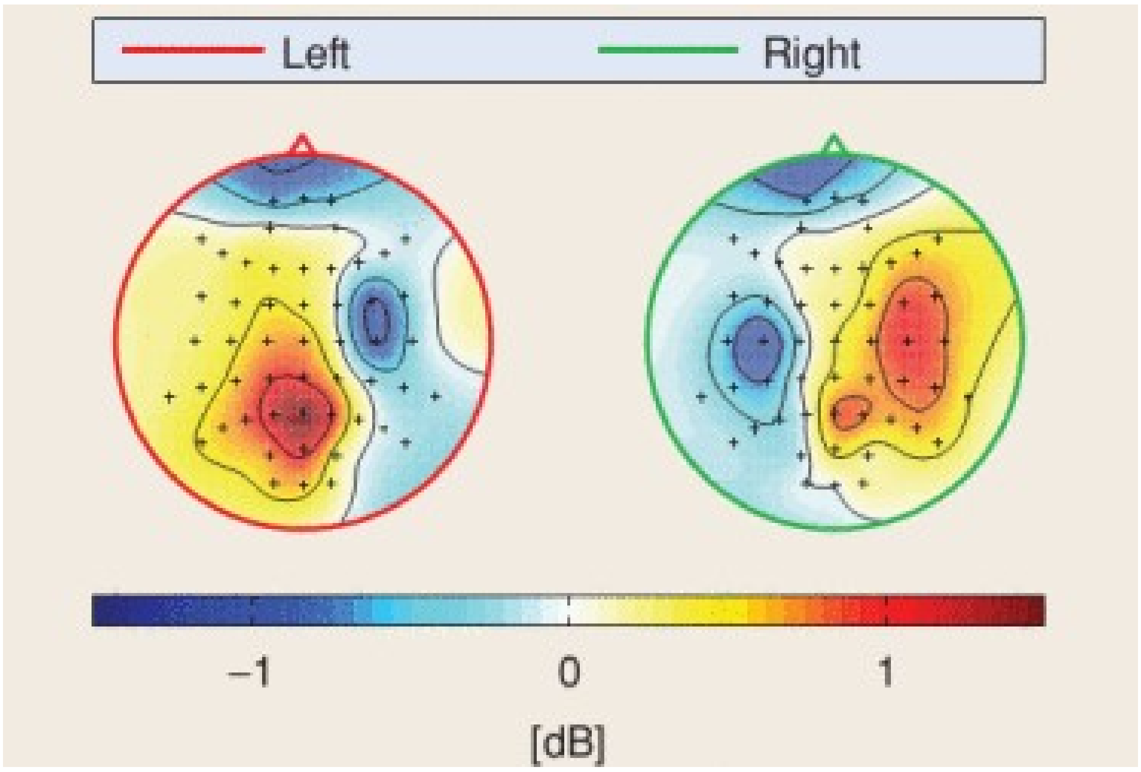

Figure 2.3 This figure represents the estimated gain applied to each source depending on its origin. The first filter maximizes the variance of the right hand movements while minimizing the variance for the left hand movement. The second filter does the opposite. The minimums correspond to ERD, the origin of the Motor Imagery. It can be observed that left hand movements are produced on the right side of the brain and vice versa. Modified image, original extracted from [5].

2.4 Classification

Once the EEG signals have been treated and the representative features extracted, these can be used in order to train a classifier. This algorithm can then be used to predict the label of any new possible observation.

Many different algorithms that serve this purpose exist, and there is no unique answer about which is the best one to use. This usually depends on the distribution of the data set as well as other restrictions such as computational cost or memory usage. In our case, we will focus mainly on accuracy, however, all of the algorithms implemented in our work would run practically in real time after training in order to classify a single trial on any modern computer.

Our study will be divided into two different types of algorithms. The first group is that of supervised classifiers. During the training stage of these algorithms, the labels of the trials are known, which makes it possible to learn the defining features of each class. The second set is composed by the unsupervised classifiers, which do not have the labels of the training samples. These find clusters of points in the training data, which should correspond to the different categories. As these algorithms are blind in the sense that all of the samples are perceived as equal, without a distinction of category, the parameters obtained during training tend to be less accurate approximations of the underlying model that generates the data.

Moreover, as it was mentioned earlier, there are limitations to the maximum accuracy achievable by classifiers, because even if the model is known with absolute certainty and the best possible prediction can be made, the classes are not completely separable. This means that there is a degree of overlap between the features that define the different categories and therefore the error probability of a prediction will not be zero[12].

The classifiers that we will use will be described in more detail during the next chapter, as the main goal of our experiment is to explore the possibility of using different classification algorithms on MI-BCI applications and as such this subject is considered important enough to have a dedicated chapter.

[12]A more detailed explanation and mathematical proof can be found in the next chapter, in the discussion about the Bayesian classifier.

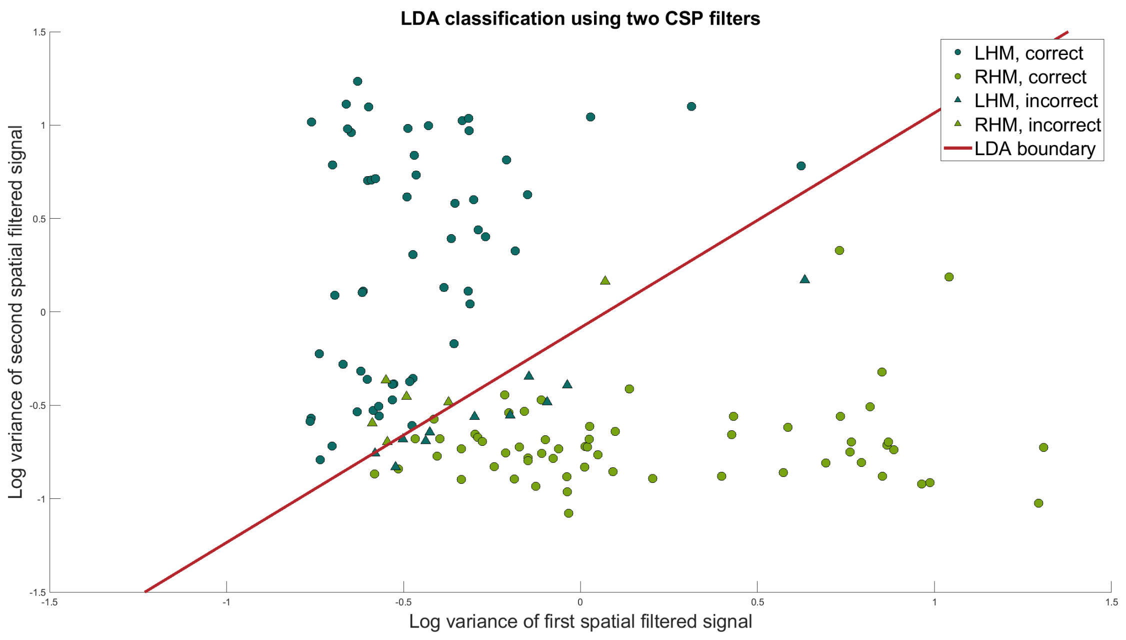

Figure 2.4 This is a representation of a LDA classifier in an R^2 space, using two CSP filters. The data is from the first sesion of the first user in the Data Set 2a. The trials corresponding to both classes, left hand movements (LHM) and right hand movements (RHM) are marked in different colors and different shapes are used for those predictions that would be correct or incorrect. This is not significant as the data used to train the classifier is used to plot the graph, which could cause overfitting as it will be discussed later.

2.5 Summary

The process involved in solving a general classification problem was decomposed into three stages, signal preprocessing, feature extraction or dimensionality reduction and classification, and the particular methods employed in this work for each one of them was detailed.

During the first stage, the EEG recordings were spliced into trials and filtered between 8Hz and 32Hz. Then, the covariance matrices of the trials were calculated using an instantaneous normalization of the power of the sources [33].

The dimensionality reduction stage consisted mainly on the application of the Common Spatial Patterns algorithm, used to create the spatial filters that extract the most relevant information for Motor Imagery detection from the EEG signals. The log function is then applied to the variance of these filtered signals and this information is used as input for our classifiers. The modifications that are necessary in order to make CSP an unsupervised algorithm were briefly mentioned.

Lastly, an overview of the classification stage was given, as this subject will be studied in depth in the next chapter.

3 Classifiers

The theory of probability combines commonsense reasoning with calculation. It domesticates luck, making it subservient to reason.

Ivars Peterson

A classifier can be understood as a mathematical structure that takes a set of observations from a particular element that we desire to classify as input and, based on previous experiences acquired during a training stage, is capable of making an educated guess of the category in which the element belongs to. This set of observations or features must not be understood as the subjective notations that we take from nature, but rather as a set of numbers that objectively describe a property. Effectively, a classifier acts as a mapping that relates every possible observation to the category in which it belongs to.

3.1 Supervised classifiers

The peculiarity that makes a supervised classifier distinct is that for every sample used during training, the class to which it belongs to is known. This category is represented on a label, and therefore we can say that the training samples are labeled.

3.1.1 Bayesian classifier

For each element that has to be categorized, we can estimate how likely it is for the received observation to belong to each class. Once this assesment is complete, assigning the trial to the most likely class should lead to the highest success rate.

This intuitive concept can be formally written in the Bayesian statistics framework and implemented algorithmically in order to create a classifier.

In order to illustrate this idea, we will synthesise a simpler, more intuitive data set than the one analised in this work. In our imaginary example, there is a citrus processing plant that receives shipments of oranges and lemons[1]. Unfortunately, they were mixed in the cleaning process and need to be separated again before being converted into lemonade and orange juice. As the next fruit that will arrive to the classifier is considered to be arbitrary, let w be a random variable that takes the value $w = w_o$ if the next fruit is an orange or $w = w_l$ if the next fruit is a lemon.

We do not know the value that w takes at any given point in time, but $P(w_o)$ (that is, $P(w = w_o)$, the prior probability of the next fruit being an orange) and $P(w_l)$ can be estimated[2]. We also know that as there are no other kinds of fruits in the plant, the sum of these prior probabilities must ammount to one, $P(w_o) + P(w_l) = 1$.

With no further information, the best guess would be to classify the next fruit as an orange if $P(w_o) > P(w_l)$ and as a lemon if $P(w_o) < P(w_l)$. This yields a better result than ramdomly guessing one or the other with equal probability for each trial, and the greater the difference between $P(w_o)$ and $P(w_l)$ the better the result will be, but it creates the problem that every fruit will be put into the same category, which is far from ideal.

[1] This explanation is loosely based on the sea bass and salmon example from [10]

[2] It is possible to have an accurate guess of the number of oranges and lemons that enter the plant if the weight input and the average weight for each fruit is known. A good estimation of $P(w_o)$ is $P(w_o) = \frac{\text{Number of oranges}}{\text{Number of fruits}}$.

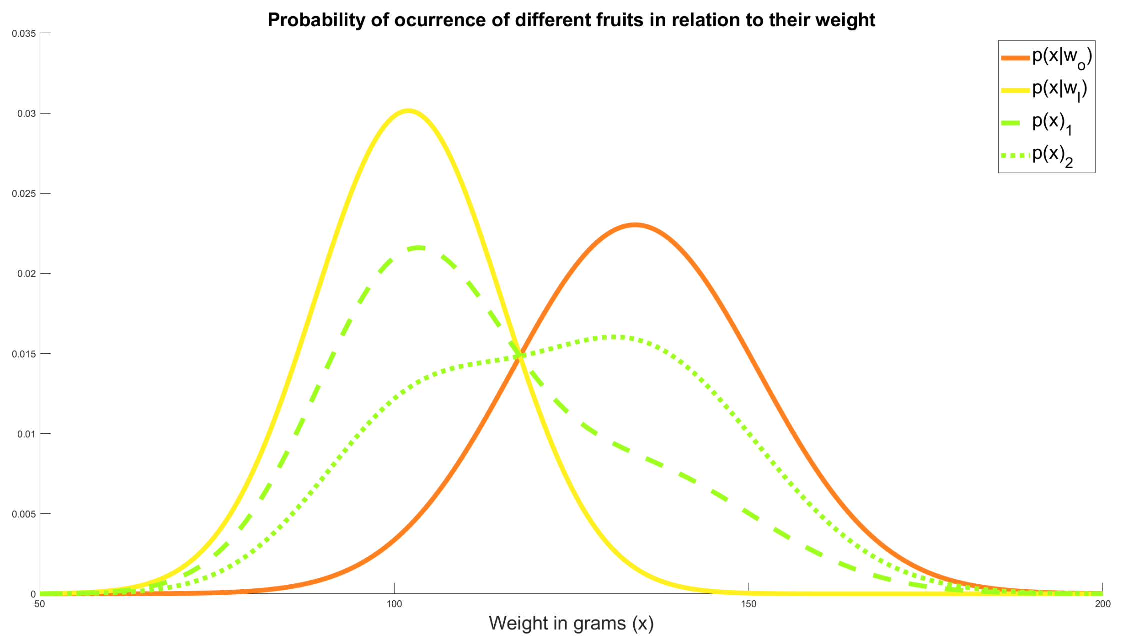

Figure 3.1 $p(x|w_o)$ represents the probability of finding an orange that weights x grams, and $p(x|w_l)$ is the equivalent for lemons. $p(x)$ is the probability of finding either an orange or a lemon of weight x. In the case of $p(x)_1$, finding a lemon was twice as likely as finding an orange (this is represented by the prior probabilities, $P(w_l) = 2P(w_o)$, which are also constrained by the condition $P(w_l) + P(w_o) = 1$), whereas the opposite is true for $p(x)_2$.

In order to improve the chance of a correct guess and solve the problem of only being able to produce one beverage, we can add an observed feature to the classifier such as the weight of the fuit as the random variable x, measured with a scale in the conveyor belt.

By performing some experiments, we can obtain the estimated probability density function of the weight of an orange, which can also be interpreted as the probability density function of the observation conditioned by the fact that the next fruit is an orange, $p_X(x|w_o)$. In the same way, $p_X(x|w_k)$ can be approximated. These functions $p_X(x|w_k)$, where k can either be o or l, are called likelihood functions and will be expressed from now on as $p(x|w_k)$ for the sake of simplicity.

Furthermore, the probability density function of an observation can be written as a mixture of the two previous probability density functions, $p(x) = p(x|w_o)P(w_o) + p(x|w_l)P(w_l)$ following the law of total probability.

The probabiliy of an observation being made and a fruit belonging to a certain class at the same time, $p(w_k,x)$ can be expressed in two different ways, $p(w_k,x) = P(w_k|x)p(x) = p(x|w_k)P(w_k)$. Rearranging these terms, we obtain Bayes' formula [10]:

$$P(w_k|x) = \frac{p(x|w_k)P(w_k)}{p(x)} \tag{3.1}$$

The term $P(w_k|x)$ indicates the probability of the next fruit being either an orange or a lemon conditioned by the fact that a certain observation has been received. As it was previously stated, intuitively for any given value of x, the best classification decision would be to choose w_o if $P(w_o|x) > P(w_l|x)$, or w_l otherwise. In order to prove this, we will calculate the error probability associated with this decision. We can define decision regions Z_k such as if $x \in Z_k$, we decide that the observation belongs to the kth category[3].

To calculate the total error probability we integrate over every possible value of the observation [10].

$$P(error) = \int_{\mathbb{R}} P(error,x)dx = \int_{\mathbb{R}} P(error|x)p(x)dx \tag{3.2}$$

[3] In order to properly define the decision regions, the union of all decision regions must amount to $\mathbb{R}$ and the intersection of any two regions must yield the empty set.

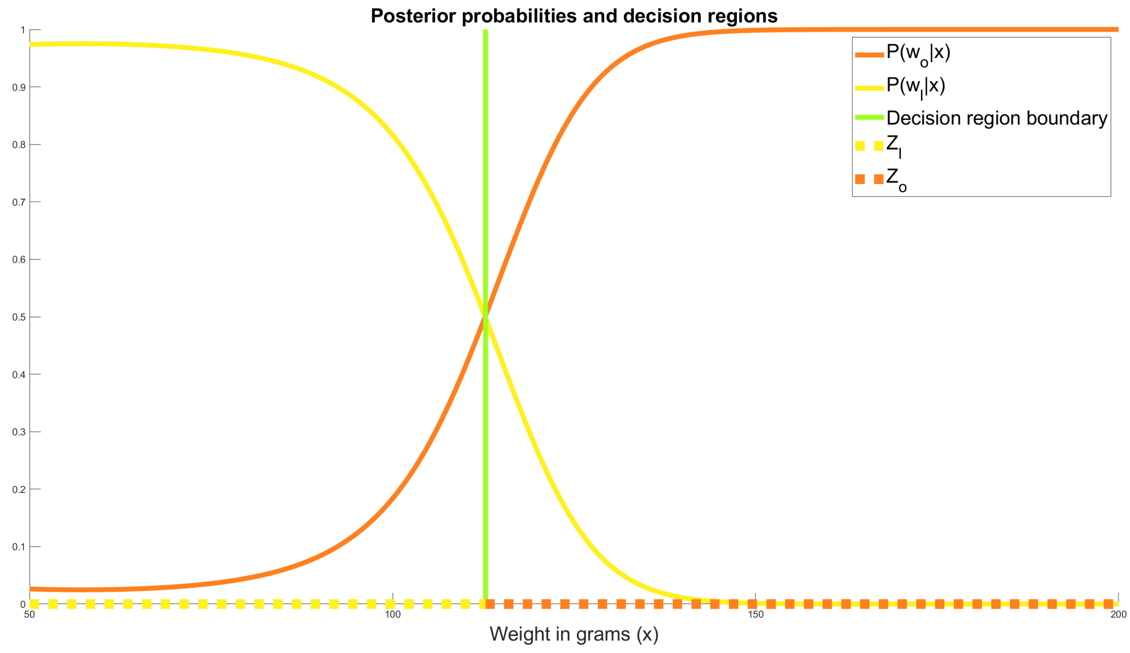

Figure 3.2 For any given value of x, the class choice that maximizes the success rate is the category with the highest posterior probability. This allows us to define the optimal decision regions. The sum of the posterior probabilities of all classes for any particular value of x ammounts to one. In this case, the graph was plotted using the prior probabilities $P(w_o) = \frac{2}{3}$ and $P(w_l) = \frac{1}{3}$ and the likelihood functions from 3.1.

It is possible to write the error probability as a function of the decision regions. In order to do so, note that once x has been observed, in Z_k we will choose the kth class and therefore the probability of making a mistake will be[4] $P(error|x) = P(w_{\neg k}|x)$, the probability of the observation belonging to a class that is not k.

$$P(error) = \int_{\mathbb{R}} P(error|x)p(x)dx = \int_{Z_o} P(w_l|x)p(x)dx + \int_{Z_l} P(w_o|x)p(x)dx \tag{3.3}$$

The decision regions that guarantee the minimum value of these integrals are $Z_k = \{x \in \mathbb{R}/P(w_k|x) > P(w_{\neg k}|x)\}$, as if we were to change any subset of one region to another, the value of the integral of the destination region would be incremented in a greater ammount than the origin one would be reduced. This finding lines up with the intuitive result stated earlier: for any given observation, the optimal choice is the one with the highest posterior probability.

These results can also be extended to include observations with multiple features. This time, the observation is a multivariate random variable $\mathbf{v} \in \mathbb{R}^n$ where $\mathbf{v} = [x_1,...,x_n]$. Furthermore, $p(\mathbf{v}|w_k)$, $p(\mathbf{v})$ and $P(w_k|\mathbf{v})$ are joint probability density functions. The total error probability calculation can be rewritten to include these changes, where the decission regions are now subsets of a $\mathbb{R}^n$ space.

$$P(error) = \int_{\mathbb{R}^n} P(error|\mathbf{v})p(\mathbf{v})dv = \int_{Z_o} P(w_l|\mathbf{v})p(\mathbf{v})dv + \int_{Z_l} P(w_o|\mathbf{v})p(\mathbf{v})dv \tag{3.4}$$

Following the same reasoning, we arrive at the conclusion that the regions that minimize $P(error)$ are[5]:

$$Z_k = \{\mathbf{v} \in \mathbb{R}^n/P(w_k|\mathbf{v}) > P(w_{\neg k}|\mathbf{v})\} \tag{3.5}$$

Now that we have a theoretical understanding of the inner workings of this classifier, the problem on how to analitically find the boundaries that separate the decision regions remains. We define the discriminant functions $g_k(\mathbf{v}) = P(w_k|\mathbf{v})$. From now on, the decision regions will be referenced to the discriminants, and

[4] Although this result is for two variables only, the reasoning is easily expanded to the multiclass problem as $P(error|x) = \sum_{i \neq k} P(w_i|x)$

[5] In the multiclass case, the decision regions can be written as $Z_k = \{\mathbf{v} \in \mathbb{R}^n/P(w_k|\mathbf{v}) > P(w_i|\mathbf{v}), \forall i \neq k\}$

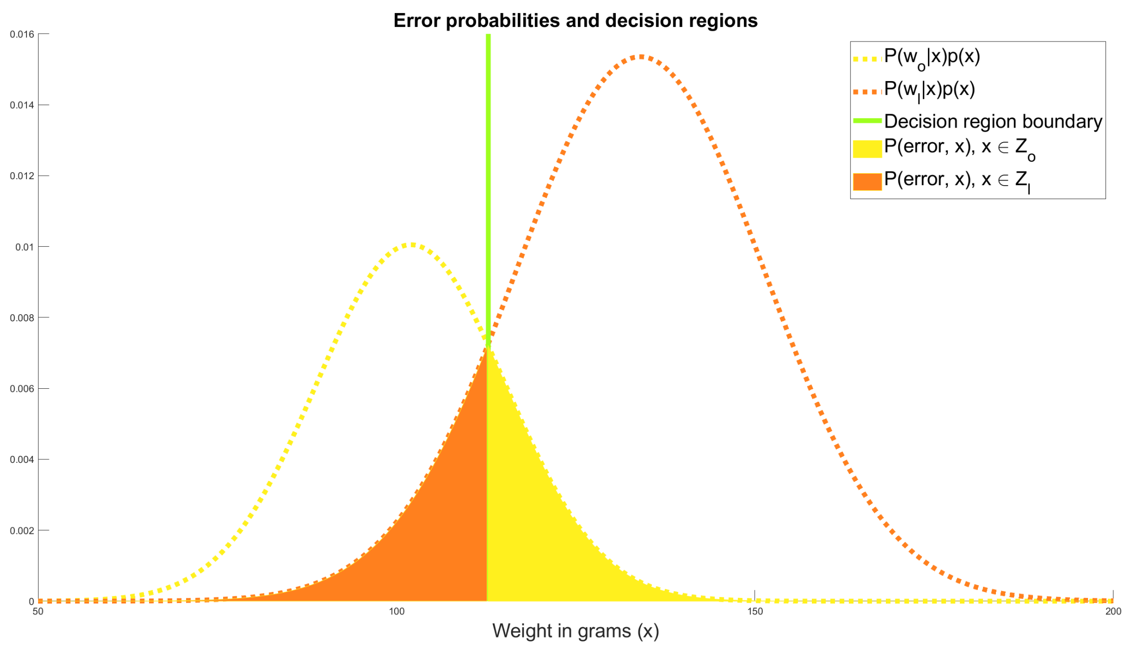

Figure 3.3 $P(w_k|x)p(x)$ is equivalent to $p(x|w_k)P(w_k)$, that is the likelihood functions scaled by the prior probabilities. The integral that calculates the error probability is represented by the solid coloured area, and it depends on the likelihood functions, prior probabilities and decision regions. As it is not possible to control the first two (at least in our system), the error probability can only be minimized by optimizing the decision regions. The graph was plotted using the prior probabilities $P(w_o) = \frac{2}{3}$ and $P(w_l) = \frac{1}{3}$ and the likelihood functions from figure 3.1.

the discriminants may be altered in any way as long as the changes do not result in an alteration of the decision regions.

Using equation (3.1),

$$g_k(\mathbf{v}) = \frac{p(\mathbf{v}|w_k)P(w_k)}{p(\mathbf{v})} \tag{3.6}$$

One important detail is that the term $p(\mathbf{v})$ is a common scale factor in every discriminant function, and as a result, removing it will not modify the decision regions. In general, if $f(\cdot)$ is a monotonically increasing function, replacing $g_k(\mathbf{v})$ with $f(g_k(\mathbf{v}))$ will not alter the decision regions either [10].

$$g_k(\mathbf{v}) = \log(p(\mathbf{v}|w_k)) + \log(P(w_k)) \tag{3.7}$$

To continue these simplifications, we must make some assumptions about the probability density functions of our actual data set. We know that both classes are equally likely[6], and therefore the terms $P(w_k)$ are common to every discriminant and can be eliminated. Moreover, our features follow approximately a normal distribution.

$$p(\mathbf{v}|w_k) \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{v}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{v}-\boldsymbol{\mu}_k)\right] \tag{3.8}$$

Where n is the number of features, $\boldsymbol{\mu}_k$ is the expected value of $\mathbf{v}$ when the class is k, $\boldsymbol{\Sigma}_k$ is the covariance matrix of this vector and $|\boldsymbol{\Sigma}_k|$ and $\boldsymbol{\Sigma}_k^{-1}$ are the determinant and inverse of this matrix respectively. These

[6] During the capture phase, the same number of movements were registered for each class. If this was not the case, this value could be estimated as $P(w_k) = (\text{Trials labeled as } k)/(\text{Total number of trials})$. Still, this calculation can be carried out and used in the classifier. However, it will hinder the performance as it is another estimation with a low number of samples.

values cannot be calculated exactly, but an estimation is possible using the labeled trials during the training stage.

$$\boldsymbol{\mu}_k = \frac{1}{m}\sum_{j=1}^{m} \mathbf{v_j} \tag{3.9}$$

Where m is the number of trials labeled as k and $\mathbf{v_j}$ are all the feature vectors labeled as belonging to the kth category.

$$\boldsymbol{\Sigma}_k = \frac{1}{m-1}\sum_{j=1}^{m} (\mathbf{v_j} - \boldsymbol{\mu}_k)(\mathbf{v_j} - \boldsymbol{\mu}_k)^T \tag{3.10}$$

This estimation of the covariance matrix is known as the Maximum Likelihood (ML) [42] and is one of the most popular methods to approximate the covariance matrix because of its simplicity. The result will tend to the true covariance matrix as the number of samples used tends to infinity. However, in order for it to work properly, the number of trials must be greatly superior to the number of variables. As the number of trials is fairly limited in our data set, this estimation is sufficiently accurate to perform as intended, but there are improved covariance estimators [42] that do increase the performance of this algorithm, as well as any other that do rely on the estimation of the sample covariance matrix.

By replacing (3.8) in (3.7), we obtain:

$$g_k(\mathbf{v}) = -\frac{1}{2}(\mathbf{v} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{v} - \boldsymbol{\mu}_k) - \frac{m}{2}\log(2\pi) - \frac{1}{2}\log(|\boldsymbol{\Sigma}_k|) + \log(P(w_k)) \tag{3.11}$$

As it was discussed before, the term $\log(P(w_k))$ can be dropped in our particular case, yet we will keep it in order to avoid the loss of generality. This is not the case of $-\frac{m}{2}\log(2\pi)$, which is also common to all discriminant functions in the more general case.

The parameters that describe the discriminant functions are estimated during the training stage, and after that, for any element that we wish to classify, it is possible to evaluate the discriminant functions. In order to achieve the maximum success rate, or minimum error rate, the observation will be matched with the class which discriminant function yields the greatest value.

3.1.2 Linear Discriminant Analysis

The Linear Discriminant Analysis classifier is based on the same underlying theory as the Bayesian classifier, but instead of using the discriminant functions[7] directly, the boundaries that separate the decission regions are studied.

First, there are some simplifications that might increase the performance of our system. The first one is to consider the sample covariance matrix $\boldsymbol{\Sigma}_k$ to be diagonal. This is because the features are extracted as the natural logarithm of the variance of the EEG signals after filtering with CSP. These spatial filters are orthogonal and the resulting signals should be uncorrelated. As the sample covariance matrix is an approximation, the terms outside of the main diagonal are close to but not exactly zero.

If $\boldsymbol{\Sigma}_k$is a diagonal matrix it is much easier to compute $|\boldsymbol{\Sigma}_k|$ and $\boldsymbol{\Sigma}_k^{-1}$. This last one is also diagonal, making the calculation of $(\mathbf{v} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{v} - \boldsymbol{\mu}_k)$, which has to be computed for every class and trial, significantly faster. This consideration is neither necessary in order to implement the LDA classifier nor exclusive to it and will be tested in chapter 4 for both the Bayesian and LDA classifiers.

The next step is to instead of considering two different covariance matrices, one for each class, use the same one for both classes. This is a feasable assumption for our data set as the covariance matrices for both categories are fairly similar. This leaves the discriminants as:

$$g_k(\mathbf{v}) = -\frac{1}{2}(\mathbf{v} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1} (\mathbf{v} - \boldsymbol{\mu}_k) + \log(P(w_k)) \tag{3.12}$$

As we were previously using approximately half of the labeled samples to compute each covariance matrix, the calculation of this one covariance matrix is conducted with double the amount of observations. This compensates for any performance improvements that having two different but more innacurate covariance matrix estimations for the different classes could have.

Moreover, as the sample covariance matrix is shared, there is only one linear boundary, a $n-1$ dimensional hyperplane that divides the $\mathbb{R}^n$ space into two decision regions.

[7] These discriminant functions are previously discussed in 3.1.1

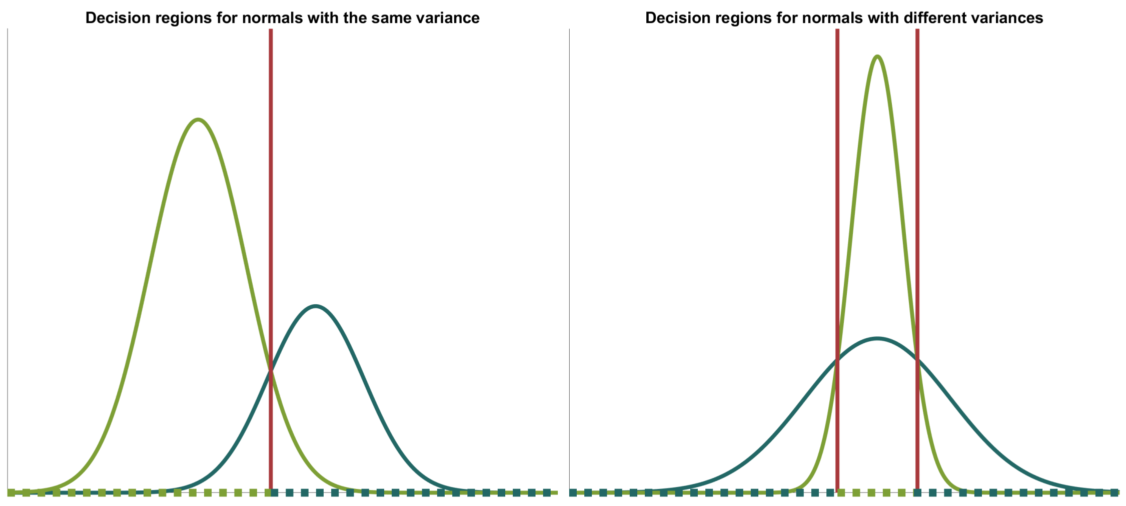

Figure 3.4 This figure shows two different sets of normally shaped distributions (scaled by the prior probabilities) and the decision regions associated to the MAP criterion. This is equivalent to the multivariate case where instead of the variances we would compare the covariance matrices. Provided that the covariance matrix is the same for both distributions, the decision regions are connected and the boundaries are linear. This is not necessarily true for the general case [10].

From (3.12), $(\mathbf{v}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{v}-\boldsymbol{\mu}_k)$ can be expanded to:

$$(\mathbf{v}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{v}-\boldsymbol{\mu}_k) = \mathbf{v}^T\boldsymbol{\Sigma}^{-1}\mathbf{v} - 2\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k\right)^T\mathbf{v} + \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \tag{3.13}$$

The term $\mathbf{v}^T\boldsymbol{\Sigma}^{-1}\mathbf{v}$ is independent of the class k and is therefore dropped.

Until this point, despite our work being centered in the dichotomizer, the equations have remained general enough to be applied directly to the multiclass problem. Now, in order to obtain an equation for the boundary that separates two classes, we will be comparing two discriminant functions. As the condition that defines the first region is $g_1(\mathbf{v}) > g_2(\mathbf{v})$ and the opposite is true for the second region, the points that form the boundary satisfy the condition:

$$g_1(\mathbf{v}) = g_2(\mathbf{v}) \tag{3.14}$$

$$g_1(\mathbf{v}) - g_2(\mathbf{v}) = 0 \tag{3.15}$$

Replacing (3.12) in (3.15) and accounting for the expansion in (3.13),

$$\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1\right)^T\mathbf{v} - \left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2\right)^T\mathbf{v} - \frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \log(P(w_1)) - \log(P(w_2)) = 0 \tag{3.16}$$

Grouping terms and simplifying,

$$\left(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)\right)^T\mathbf{v} - \frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \log(P(w_1)) - \log(P(w_2)) = 0 \tag{3.17}$$

This corresponds to the equation of a hyperplane of the form

$$\mathbf{A}\mathbf{v} + b = 0 \tag{3.18}$$

Where

$$\mathbf{A} = \left(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)\right)^T \tag{3.19}$$

And

$$b = -\frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \log(P(w_1)) - \log(P(w_2)) \tag{3.20}$$

This can also be rewritten as a function of A, which has already been calculated:

$$b = -\frac{1}{2}\mathbf{A}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) + \log(P(w_1)) - \log(P(w_2)) \tag{3.21}$$

For each sample that we want to classify, we compute $\mathbf{Av}+b$. As this is equivalent to calculating $g_1(\mathbf{v}) - g_2(\mathbf{v})$, if the result is greater than zero, the feature vector is located in the region of space where $g_1(\mathbf{v}) > g_2(\mathbf{v})$, that is the decision region Z_1 and the sample is assigned to the first class. The opposite is also true, if the sign of $\mathbf{Av}+b$ is negative, the sample can be classified in the second category.

As in our case $P(w_1) = P(w_2)$, the boundary goes through the midpoint of the two means and, in general, it is not necessarily normal to the segment that connects them. If $P(w_1) \neq P(w_2)$, the hyperplane would be paralel to the boundary obtained with $P(w_1) = P(w_2)$, but it would not intersect the segment that joins the two means in the middle point. It would be biased towards the mean with the lowest a priori probability, and if the difference between both probabilities is big enough, the boundary can be pushed beyond the mean corresponding to this category.

Usually, LDA is used to distinguish between two classes, and this is the main focus of this work. However, it is not difficult to extend this algorithm to the multiclass problem.

If we have d different classes, it is possible to find the linear boundary between any pair of them, that is a total of $j = \binom{d}{2}$ hyperplanes. We create a vectorial boolean variable $\mathbf{u}$ of length j that in the ith possition contains a 1 if the equation for the ith boundary yields a result greater than one and 0 otherwise. This divides the $\mathbb{R}^n$ space into at most 2^j regions that correspond to the maximum 2^j possible states of the variable $\mathbf{u}$. Not all of the logic states are necessarily possible because the maximum number of states is constrained by the number of partitions of a d dimensional space achievable with j hyperplanes. This problem is not trivial, however, we will take an approach that does not require us to solve it.

When a new sample arrives and a state of the vector $\mathbf{u}$ with no class assigned is found, the discriminant functions are checked, and the category of the one that yields the greatest result is registered to that region. From now on, any sample that shares this state of the vector $\mathbf{u}$ can be directly mapped to the corresponding class without the need of checking all the individual discriminant functions.

Wether this is worth doing or it is better to calculate the values that the discriminant functions for each class take instead, it depends on the number of dimensions of the feature vector and the number of classes. The number of boundaries increases faster than the number of discriminant functions with the number of classes, whereas the calculation of a single discriminant function becomes more computationally expensive with the size of the feature vector faster than the calculation of a single boundary does.

3.1.3 K Nearest Neighbours

Although in the literature this classifier is better known as K Nearest Neighbours (KNN), we have used the letter k to denote the different categories of elements that we wish to classify. The letter K on the name of this classifier refers to a different concept, and therefore a substitute will be used in its place for the remainder of this work. From this point forward, this algorithm shall be denoted as the N Nearest Neighbours classifier.

In general, during the dimensionality reduction stage of a classification task, the most significant features to distinguish between categories are extracted. This means that the observed features from all the elements of a single class should be close together in the feature space and as separate as possible from the other class.

The N Nearest Neighbours classifier takes advantage of this geometrical property by finding the N closests labeled trials to the sample that has to be classified. It is most likely for these labeled observations to belong to the same class as the element that we want to classify because they are closer together, and therefore we can assign the sample to the most repeated class amongst the labeled trials.

The distance between the sample that has to be classified $\mathbf{v_s}$ and any labeled sample $\mathbf{v_l}$ is calculated as an euclidean distance in the feature space:

$$\delta_E(\mathbf{v_s}, \mathbf{v_l}) = ||\mathbf{v_l} - \mathbf{v_s}|| \tag{3.22}$$

Where $|\cdot|$ is the euclidean norm.

The choice of N is important and does affect the performance of the classifier. First, it has to be an odd number, as the behaviour is not defined in the case of finding the same number of neighbours belonging to each class. Then, if N is too low, the performance might by hindered by the random nature of the distribution of the samples, as it is possible for a low number of labeled trials of the same class to be close together in a region where it is more probable for the other class to appear.

On the other hand, if N is too large, trials that are located further and further away and that therefore might not be significant when classifying that sample, are considered. The ideal value of N for our case of study will be analysed later in chapter 4.

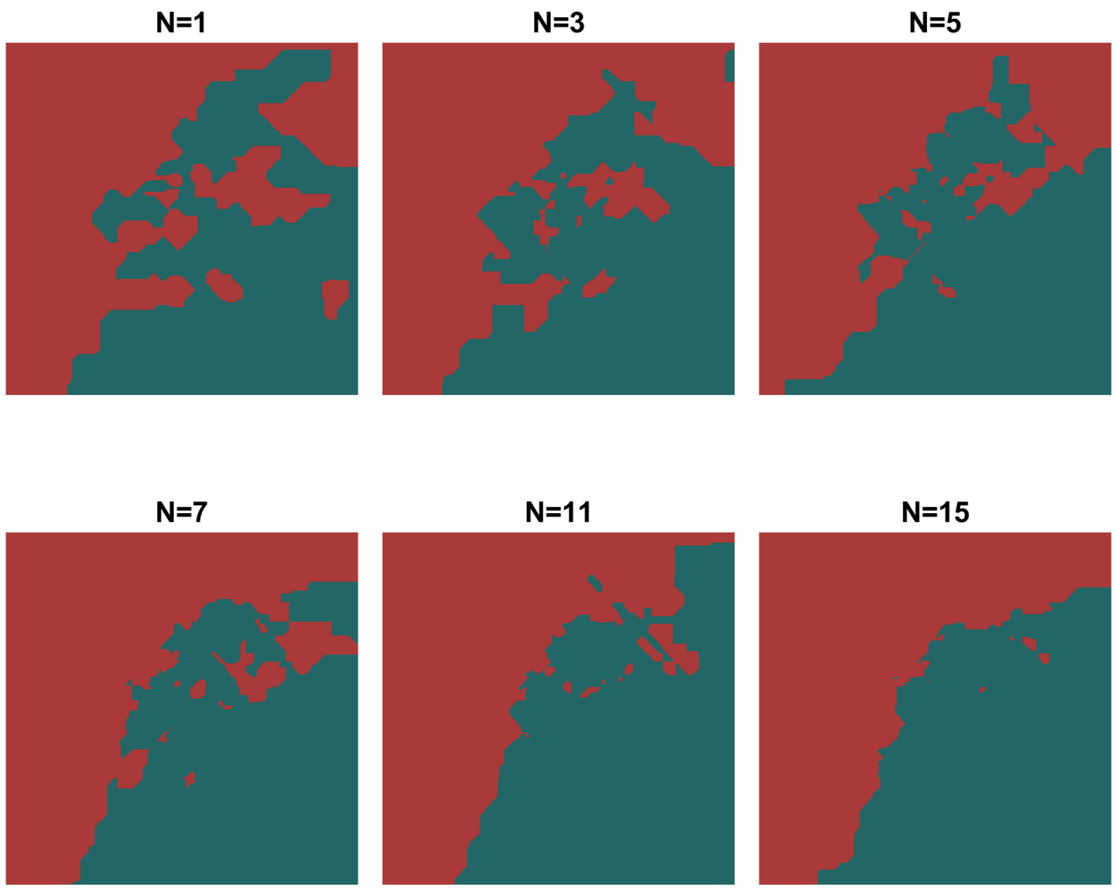

Figure 3.5 This graph shows the effect that the value of the parameter N has on the decission regions of the N nearest neighbours classifier, shown in different colors. In order to obtain these, 75 random elements were drawn from each of two bivariate normal distributions with different means and variances. A relevant portion of the $\mathbb{R}^2$ space is shown.

This idea can be directly expanded to the multiclass classifier by adjusting the limitations on N. This should not take any value that is a multiple of any of the positive integers above one and equal to or lower than the number of existing classes. Without this limitation, it is possible to find the case where the most repeated value is shared by two or more categories, and the behaviour in this scenario is undefined.

The decision regions formed by this method are not necessarily connected and the boundaries are too complex to be described conveniently as it is shown in figures 3.5 and 3.6.

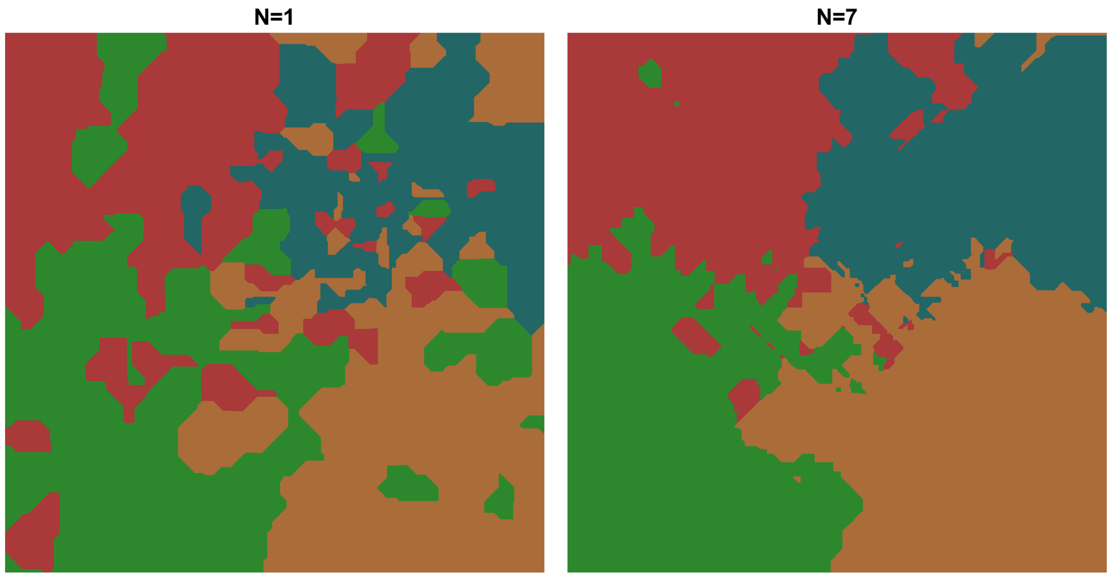

Figure 3.6 This figure represents the decission regions of a N nearest neighbours polichotomizer capable of separating four different categories. The plot was obtained like the ones in figure 3.5, however, this time there are four distributions instead of two.

3.1.4 Nearest archetype

This classifier is based on the idea that for each category there is an archetype[8], that is, an ideal element that represents the features that a perfect sample from that class would have.

This representative should be as close as possible to every labeled trial used during training, and therefore we can develop an error metric to describe this quality.

Although different error functions based on distance do exist, a common procedure is to consider the sum of the squared euclidean distance from the kth archetype $\mathbf{c_k}$ to every labeled sample belonging to the class k.

$$E_k(\mathbf{c_k}) = \sum_{i=1}^{T_k} ||\mathbf{v_i} - \mathbf{c_k}||^2 \tag{3.23}$$

Where T_k is the number of elements labeled as k and $\mathbf{v_i}$ are all the feature vectors corresponding to these elements. It is also worth noting that this error is exclusive to each class and only depends on the elements that are labeled as belonging to that category. As a result, the optimization process for each archetype is independent from the others. This facilitates the extension to the multiclass problem as in order to include more categories, we only need to calculate more representatives independently.

The solution that minimizes the error function presented in (3.23) can be found by using the arithmetic mean. The archetype or centroid $\mathbf{c_k}$ that represents the observations from the kth category can therefore be written as:

$$\mathbf{c_k} = \frac{1}{T_k} \sum_{i=1}^{T_k} \mathbf{v_i} \tag{3.24}$$

For each new sample that has to be classified, the distance from this observation to every centroid is computed. The trial is then assigned to the category represented by the closest archetype.

The resulting boundaries that separate the decision regions are linear, perpendicular to the segment that connects the means and intersects the midpoint of both means. As a result, this dichotomizer is equivalent to the LDA or Bayesian classifier for normal distributions in a particular case: the a priori probabilities are equal, $P(w_1) = P(w_2)$, the covariance matrix is shared between both classes and it is of the form $\mathbf{\Sigma} = \sigma^2\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. In this case the clusters have a normal distribution where the equiprobability

[8] The word prototype is commonly used in the literature to refer to this term, and sometimes the words archetype and prototype are used interchangeably. However, the word archetype will be used here as it is more accurate in this context.

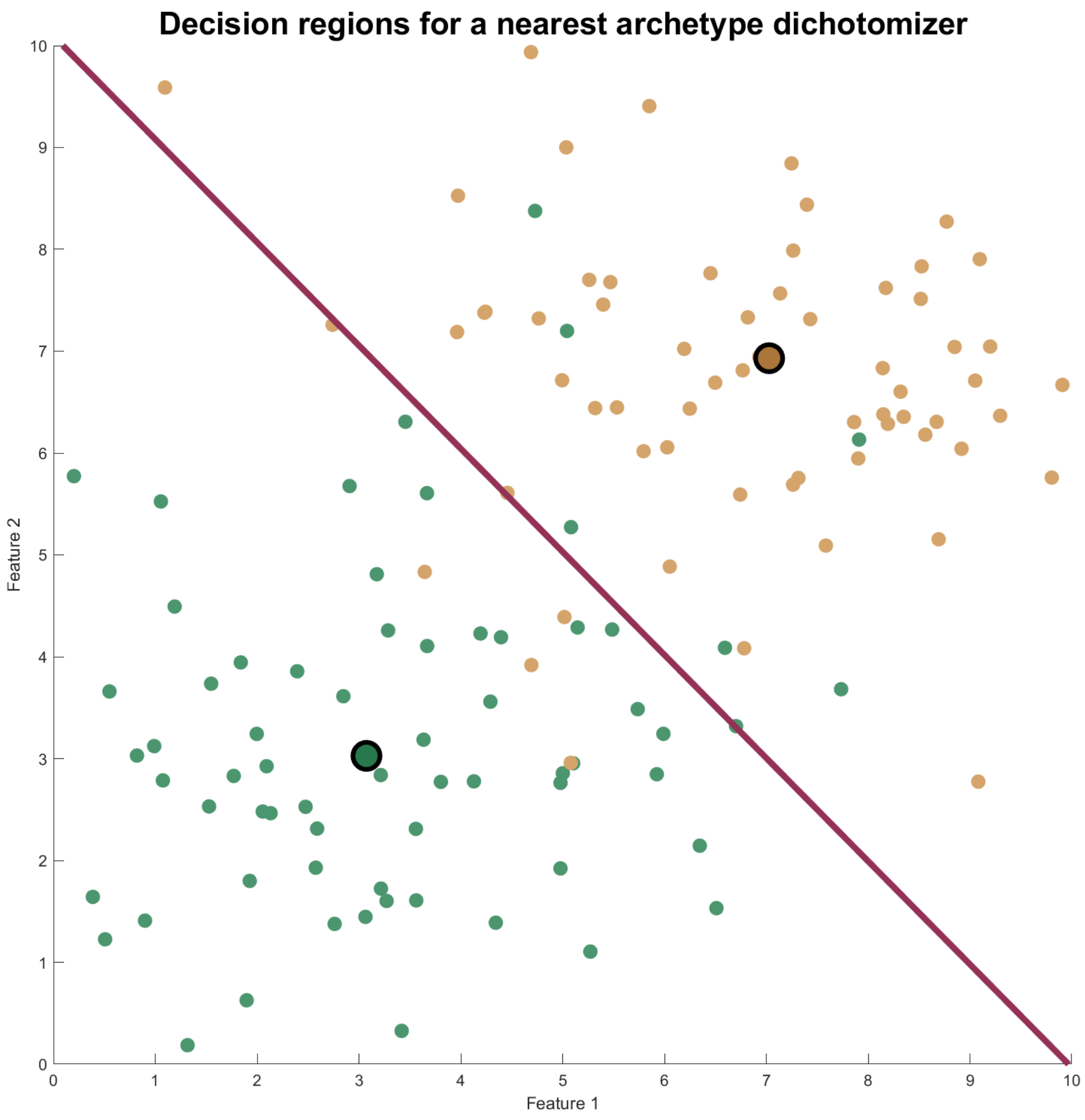

Figure 3.7 The small colored dots represent the labeled trials from each class, whereas the bigger dots are the archetypes obtained for each category. The boundary represented by the line goes through the midpoint of both means and is perpendicular to the segment that joins them. The data set used was extracted randomly from two bivariate normal distributions.

surfaces are spheres[9] and the maximum likelihood[10] criterion can be reduced to the minimum distance to mean criterion.

3.1.5 Gradient descent

Gradient descent is a general technique used to find a local minimum of a given function, usually because the function is too complex for this minimum to be found analitically. A starting point on the function is randomly chosen and the gradient respect to the desired variables is computed at this point. As the gradient points towards the faster growing direction for the function, a step is taken in the opposite direction. This process is repeated until a local minimum is found.

In this linear classifier, we will assume that a perfect representative of each class exists, and that a linear combination of the independent features that conform the first class will ammount to a certain desired value

[9] The term surface and sphere are refered to an $\mathbb{R}^3$ feature space, but in a more general sense, for a $\mathbb{R}^n$ feature space, it is an equiprobability $n-1$ dimensional manifold shaped as an $n-1$ dimensional sphere.

[10] The maximum likelihood criterion is not the basis of the LDA or Bayesian classifier. However, as $P(w_1) = P(w_2)$, these classifiers are essentially comparing the values that the likelihood functions take.

V, whereas the opposite is true for the second class, the same linear combination takes the value $-V$ for its representative.

$$\mathbf{A}\,\mathbf{x_1} + b = V \tag{3.25}$$

$$\mathbf{A}\,\mathbf{x_2} + b = -V \tag{3.26}$$

This is possible because for any two given values of $\mathbf{x}$, a hyperplane in the form $\mathbf{A}\,\mathbf{x} + b = 0$ that satisfies the condition stated in equations (3.25) and (3.26) exists. It goes through the midpoint of both elements and is normal to the segment that joins them.

To be more accurate, a family of equivalent hyperplanes that satisfy the conditions of going through the midpoint of both elements and being normal to the segment that joins them exists and is of the form $c\mathbf{A}\,\mathbf{x} + cb = 0$, $\forall c \in \mathbb{R}$. In general, these will satisfy the equations $c\mathbf{A}\,\mathbf{x_2} + cb = cV$ and $c\mathbf{A}\,\mathbf{x_2} + cb = -cV$, so the exact value of V is not relevant and can be chosen arbitrarily.

For any given trial sample $\mathbf{x_t}$ we can assume that if it belons to a certain class, it must not be very far from the representative $\mathbf{x_k}$, and therefore, the linear combination $\mathbf{A}\,\mathbf{x_t} + b$ is likely to be approximately V (or $-V$, depending on the class, but we will assume our trial belongs to the first class for simplicity). We can develop an error metric for the difference between the actual value that this function takes and the desired value V:

$$E(\mathbf{x_t},\mathbf{A},b) = (\mathbf{A}\,\mathbf{x_t} + b - V)^2 \text{ if } \mathbf{x_t} \text{ belongs to class 1} \tag{3.27}$$

$$E(\mathbf{x_t},\mathbf{A},b) = (\mathbf{A}\,\mathbf{x_t} + b + V)^2 \text{ if } \mathbf{x_t} \text{ belongs to class 2} \tag{3.28}$$

It is possible to calculate the gradient of this function in respect to the variables A and b, which we do not know and wish to adjust automatically in order to reduce the error.

$$\nabla_{\mathbf{A}} E(\mathbf{x_t},\mathbf{A},b) = \nabla_{\mathbf{A}} (\mathbf{A}\,\mathbf{x_t} + b - L)^2 = 2\mathbf{x_t}^T (\mathbf{A}\,\mathbf{x_t} + b - L) \tag{3.29}$$

$$\nabla_b E(\mathbf{x_t},\mathbf{A},b) = \nabla_b (\mathbf{A}\,\mathbf{x_t} + b - L)^2 = 2(\mathbf{A}\,\mathbf{x_t} + b - L) \tag{3.30}$$

Where L is either V or $-V$ depending on wether the sample is from one class or the other.

As it has been previously stablished, the gradient represents the direction in which the function grows faster. We must therefore take a small step in the opposite direction in order to get closer to a local minimum.

For any parameter θ:

$$\theta^{(new)} = \theta^{(old)} - \lambda\, \nabla_\theta E(\mathbf{x_t},\theta) \tag{3.31}$$

And therefore the new parameters will be:

$$\mathbf{A}^{(new)} = \mathbf{A}^{(old)} - \lambda\, 2\mathbf{x_t}^T (\mathbf{A}^{(old)}\,\mathbf{x_t} + b^{(old)} - L) \tag{3.32}$$

$$b^{(new)} = b^{(old)} - \lambda\, 2(\mathbf{A}^{(old)}\,\mathbf{x_t} + b^{(old)} - L) \tag{3.33}$$

Where λ is a parameter known as the learning rate. It dictates the distance of each step, and if it is too large, it is more likely to overshoot the minimum and in the extreme case, the error function may diverge. On the other hand, if it is too small, the convergence time gets longer.

If we did this process over and over for a single training sample, we would indeed find a value of $\mathbf{A}$ and b that makes this error function minimal for this trial. However, this solution would not be optimal. The first reason is that no other condition has been imposed on the second class to be separated from the first one. The second reason is that even if the error is very little for a particular sample from one class, this is not necesarily the optimum, as we are trying to reduce the average error across all the trials.

As we are looking for a value of the parameters that minimize the overall error, we will be alternating amongst the different training samples in order to avoid this overfitting problem. Moreover, it might be convenient to alternate samples from the different classes, because if the system is trained with all the samples from one class and then the other, a local minimum might be found that satisfices only one of the conditions and then forgotten in order to comply with the second condition exclusively.

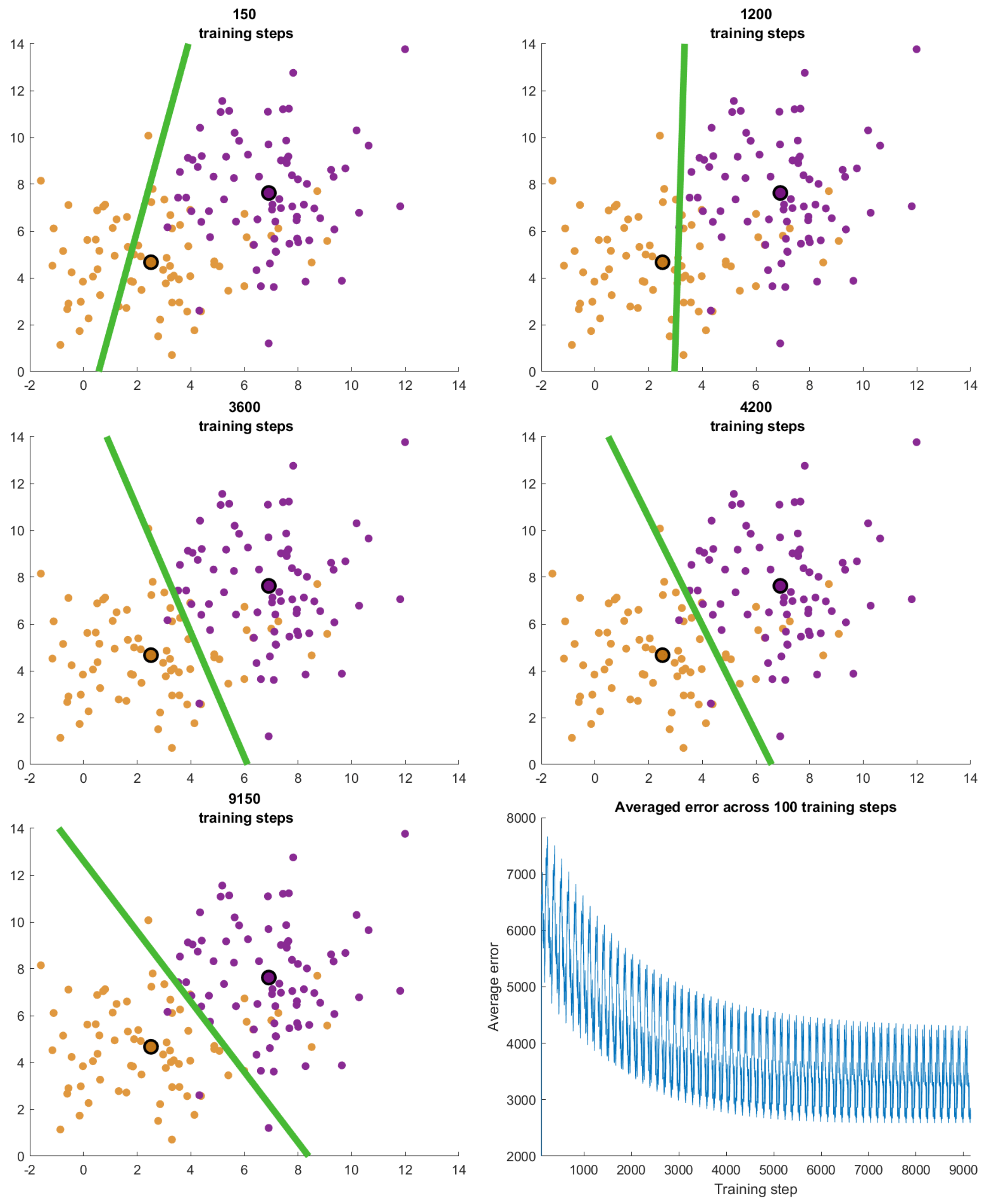

Figure 3.8 This figure shows how after a number of iterations the linear boundary starts to separate the labeled elements from both classes, represented in different colors. The means for each class are also highlighted with a bigger marker in order to facilitate the comparaison with other classifiers, although this value is not used to train this dichotomizer. As the number of labeled elements from each class, extracted from two normal distributions, was limited, the same elements are reused several times in order to train the classifier.

If there are not enough training samples for the error function to converge to a minimum, it is possible to train multiple times with the same set of labeled trials. The order in which these are used can be randomized for every training cycle. This is however not equivalent to having multiple sets of trials and the performance is limited by the available information.

It is worth noting that as we are assuming that the training samples must be very close to the representative element, closer than to the opposite representative, and this is not necessarily true, some iterations may make the overall error greater. However, on average, after many iterations the value of the error function will get closer to a local minimum.

On a second version of this classifier, instead of checking if for a given test sample the value of the linear combination $\mathbf{A}\,\mathbf{x}+b$ is closer to V or $-V$ (which is equivalent to evaluating the sign), two of these classifiers will be trained at the same time. The first one will be trained as it was previously stated whereas the second one will be trained in the opossite way. The first class will now be related to the value $-V$ and the second one will be related to V. Now, we can check which of the two functions yields a greater value for any given sample, and the trial will be matched with this class.

3.1.6 Classifiers in a Riemannian manifold

The previously described classifiers are intended to be used in conjunction with CSP, but in general they do work with any set of features that lie in an euclidean space where the euclidean metrics can be applied.

It is known that the covariance matrix of the EEG signals contains the spatial information required to distinguish between different imagined movements [3]. As this is the case, it is possible to create a classifier that uses these matrices as features. The previously discused classifiers are not suitable for this task because these matrices do not belong to an euclidean space.

The covariance matrices have a particular structure which belong to the Riemannian manifold of the symmetric and positive definite (SPD) matrices [15]. Riemmanian geometry makes it possible to operate with these matrices in their native space, and it is therefore feasable to develop classifiers similar to some of those described in previous sections based on geometric concepts such as distance.

The first step is to obtain an expression for the distance between two points in the SPD matrix space $P(n)$, which is a differentiable Riemannian manifold $\mathscr{M}$ [15]. In general, distance is understood as the length of the shortest diferentiable path that joins two points, where every element along the path belongs to the space. In an euclidean space, the shortest path is a straight line, whereas in $P(n)$ this curve is known as a geodesic. As the path is diferentiable, the distance can be obtained by integrating over this curve in order to calculate its length. Due to the properties of the SPD matrices, the geodesic distance δ_R between two points $\mathbf{P_1}$ and $\mathbf{P_2}$ is known and can be written as [3]

$$\delta_R(\mathbf{P_1},\mathbf{P_2}) = ||\log\left(\mathbf{P_1}^{-1}\mathbf{P_2}\right)||_F = \left[\sum_{i=1}^{n}\log^2(\lambda_i)\right] \tag{3.34}$$

Where λ_i are the real eigenvalues of $\mathbf{P_1}^{-1}\mathbf{P_2}$ and $||\cdot||_F$ is the Frobenius norm.

Another metric that can be described for this space is the Riemannian mean. Analogous to the deffinition of arithmetic mean previously used, the Riemannian mean can be understood as the point of the manifold that minimizes the total sum of the square of the geodesic distance to a set of given points. Unfortunatelly, no anallitic expression to compute this value does exist and optimization algorithms [12] must be used. As a simpler alternative, the arithmetic mean does provide a value that is close enough to the Riemannian mean to be employed in classification algorithms that require this metric.

$$\mathfrak{U}(\mathbf{P_1},\ldots,\mathbf{P_N}) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{P_i} \tag{3.35}$$

With these two deffinitions, it is possible to implement both the nearest archetype and the nearest N neighbours algorithms using the SPD matrices as descriptors of the EEG signal. The only change to the versions of these classifiers previously described is that the euclidean distance is replaced by the geodesic distance, and the arithmetic mean may be replaced by the Riemannian mean if an algorithm to calculate this metric is available.

Although the main advantage of these classifiers is that it is not necessary to calculate the spatial filters, it is also possible to use them in combination with CSP. Instead of classifying the covariance matrices of the EEG signals, the covariance matrices of the filtered signals are used as features.

3.2 Unsupervised classifiers

Unlike with the supervised classifiers, in this case the training samples are not labeled. This limitation does not only apply to the classifier itself but also to the feature extraction method. In general, these classifiers tend to be less accurate than the supervised ones, as less information is provided during the training stage.

However, the motivation for this kind of algorithm still remains. By removing the need of labels, or by reducing the number of labels as it will be seen in 3.3, the data acquisition stage can become more organic, and the line between training and classifying can be blurred as the unlabeled testing trials can be used to improve the performance of the classifier. Although this technique will not be used in this work, the possibility does exist. This would allow to use more trials in order to develop a more accurate classifier over time.

3.2.1 K-means

The idea that supports this classifier is similar to that of the supervised nearest archetype algorithm. The data should be distributed in K distinct clusters, one for each different class that we want to categorize. We introduce a set of vectors $\boldsymbol{\mu}_K$ where the kth vector is a representative of the kth cluster.

In general, elements belonging to the same class will be closer together and elements from different classes should be further appart. In order to formally define this concept, we could write a distortion function[11]:

$$J = \sum_{k=1}^{K} \sum_{i=1}^{I} r_{ik} ||\mathbf{x_i} - \boldsymbol{\mu}_k||^2 \tag{3.36}$$

The term r_{ik} is a binary variable that takes the value 1 if the element $\mathbf{x_i}$ belongs to the kth class and 0 otherwise. If we minimize this function, we ensure that the total sum of the distance squared from each element $\mathbf{x_i}$ used to train the classifier to their corresponding representative is minimal.

This function is described by the variables r_{ik} and $\boldsymbol{\mu}_k$, which are given a value while initializing the algorithm. The optimization process is carried out iteratively in two distinct steps. During the first one, the value of the archetypes is fixed and J is minimized in respect to r_{ik}. In the second stage, the r_{ik} are fixed and the optimal value of $\boldsymbol{\mu}_k$ is calculated. This process is repeated until there are no further changes or a limit number of iterations is reached. This process guarantees that in each iteration the value of J is reduced until the algorithm converges to a local minimum [4].

During the first stage, if we group the terms that share the same i, that is, the distance from every data point to every fixed mean, these groups can be treated independently. Among all the r_{ik} with a common i, one and only one element must take the value 1 and the others will be zero. As a result, this should be the variable whose associated $||\mathbf{x_i} - \boldsymbol{\mu}_k||^2$ is minimal. This is equivalent to matching each element with the closest[12] $\boldsymbol{\mu}_k$.

In the second stage, the r_{ik} elements are fixed, and we may eliminate those terms that are 0. This leaves us with each vector $\mathbf{x_i}$ appearing only once and related to only one $\boldsymbol{\mu}_k$.

$$J = \sum_{i|r_{i1}=1} ||\mathbf{x_i} - \boldsymbol{\mu}_1||^2 + \ldots + \sum_{i|r_{iK}=1} ||\mathbf{x_i} - \boldsymbol{\mu}_K||^2 \tag{3.37}$$

The terms that share a common k may be minimized separatedly by solving the problem of finding the vector $\boldsymbol{\mu}_k$ that minimizes the total sum of the distance squared from every $\mathbf{x_i}$ assigned to the class k to the point $\boldsymbol{\mu}_k$. Once again[13], this problem can be solved by using the euclidean mean.

$$\boldsymbol{\mu}_k = \frac{1}{\sum_i r_{ik}} \sum_{i|r_{ik}=1} \mathbf{x_i} \tag{3.38}$$

This formula is equivalent to saying that the element $\boldsymbol{\mu}_k$ is equal to the mean of all the elements that are assigned to the kth class, and it is also the reason why the algorithm is known as $K-means$.

[11]The distortion function is the name that the objective function for the K-means algorithm receives.

[12]As the square of a possitive value is a monotonically increasing function, comparing A^2 and B^2 is equivalent to comparing A and B, where these are possitive real numbers.

[13]This problem of minimizing the sum of the distance squared from different points to a common point has been studied before in 3.1.4

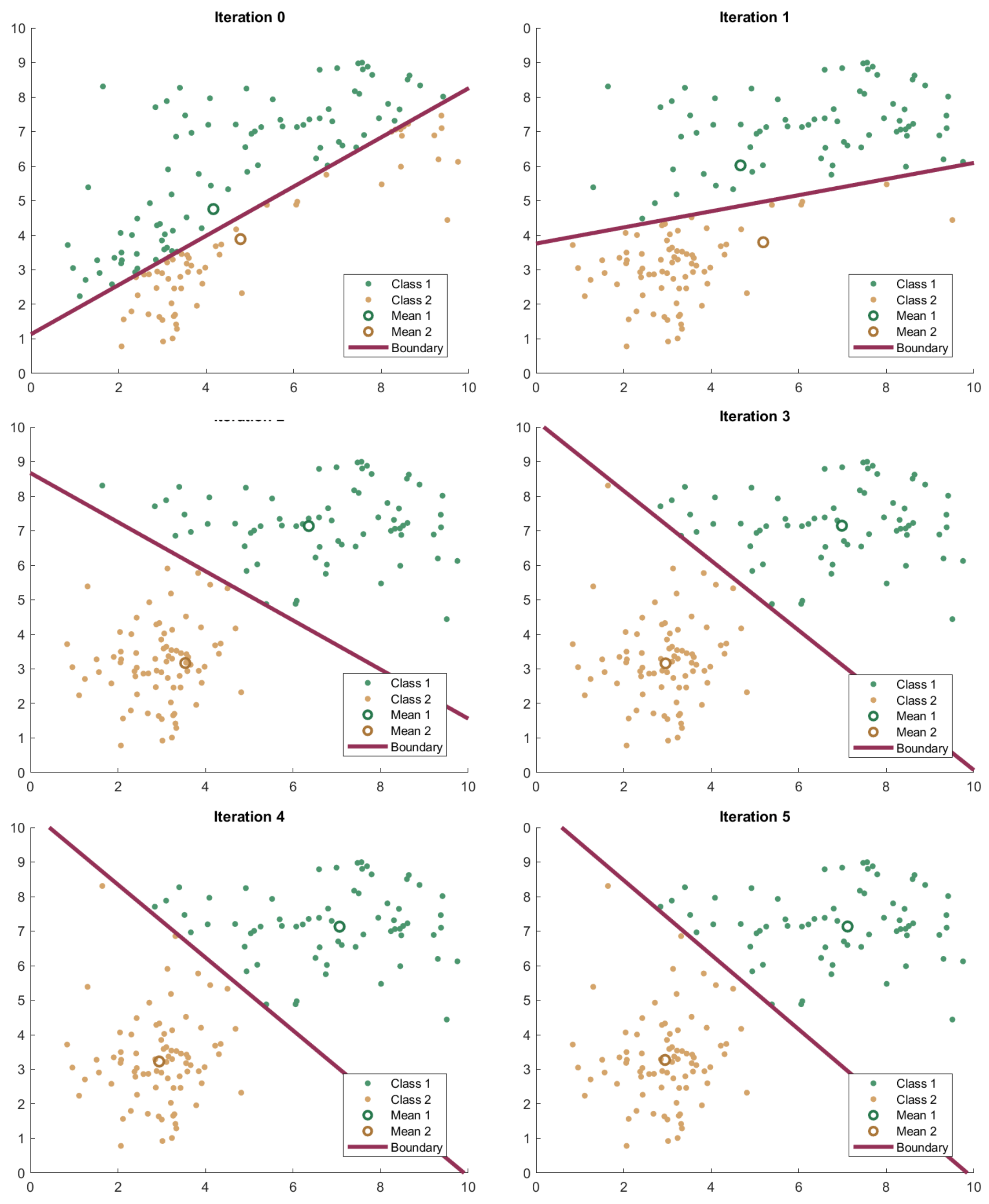

Figure 3.9 After a few iterations, the K-means algorithm finds two possible representatives, separating the training samples into two distinct clusters where all the observations are close together and far away from the other class.

3.2.2 Gaussian EM

Although this is an unsupervised algorithm and we do not know the labels that correspond to each sample, we know that, following the explanation from 3.1.1, every observation from the kth class comes from a random distribution with a probability density function $p(\mathbf{x}|w_k)$. Then, any observation comes from a combination of these functions, ponderated by the probability of each class appearing $P(w_k)$, which for simplicity will be noted as the mixing coefficients π_k. This leads to the mixture model:

$$p(\mathbf{x}) = \sum_k p(\mathbf{x}|w_k)\pi_k \tag{3.39}$$

Furthermore, in our particular case, $p(\mathbf{x}|w_k)$ are normal distributions that can be described by their mean and a shared covariance matrix. As these parameters are hidden, we may call them latent variables. Let θ be the set of all latent variables, composed by the parameters describing the gaussian probability density functions and the mixing coefficients π_k. The goal of the algorithm will be to find the θ that provides the statistical model which best fits the training samples. This fitness can be represented by the normalized log-likelihood function, which for a set of T independent trials of the form $\mathbf{x_t}$ can be written as:

$$l(\theta) = \frac{1}{T}\log p(\mathbf{x}|\theta) = \frac{1}{T}\log\prod_t p(\mathbf{x_t}|\theta) = \frac{1}{T}\sum_t \log p(\mathbf{x_t}|\theta) = \frac{1}{T}\sum_t \log\sum_k p(\mathbf{x_t},w_k|\theta) \tag{3.40}$$

As the parameters will be optimised iteratively, the value these take in the previous iteration will be represented as $\theta^{(old)}$.

$$l(\theta) = \frac{1}{T}\sum_t \log\sum_k p(\mathbf{x_t},w_k|\theta)\frac{p(w_k|\mathbf{x_t},\theta^{(old)})}{p(w_k|\mathbf{x_t},\theta^{(old)})} \tag{3.41}$$

$$l(\theta) = \frac{1}{T}\sum_t \log\sum_k p(w_k|\mathbf{x_t},\theta^{(old)})\frac{p(\mathbf{x_t},w_k|\theta)}{p(w_k|\mathbf{x_t},\theta^{(old)})} \tag{3.42}$$

The logarithm function is concave and therefore Jensen's inequality [23] can be applied in order to find a lower-bound:

$$l(\theta) \geq \frac{1}{T}\sum_t\sum_k p(w_k|\mathbf{x_t},\theta^{(old))}\log\frac{p(\mathbf{x_t},w_k|\theta)}{p(w_k|\mathbf{x_t},\theta^{(old)})} \tag{3.43}$$

$$l(\theta) \geq \left\langle \sum_k p(w_k|\mathbf{x_t},\theta^{(old)})\log p(\mathbf{x_t},w_k|\theta)\right\rangle_t + H(p(w_k|\mathbf{x_t},\theta^{(old)})) \tag{3.44}$$

The entropy term $H(\cdot)$ is not dependent of θ and therefore maximizing the lower bound of $l(\theta)$ is reduced to maximizing the posterior expectation of the joint likelihood of the observations and the hidden variables $Q(\theta,\theta^{(old)})$ [9].

$$Q(\theta,\theta^{(old)}) = \left\langle \sum_k p(w_k|\mathbf{x_t},\theta^{(old)})\log p(\mathbf{x_t},w_k|\theta)\right\rangle_t \tag{3.45}$$

The term $p(w_k|\mathbf{x_t},\theta^{(old)})$ can be interpreted as the responsibility each class takes for explaining the observation $\mathbf{x_t}$. This is equivalent to a soft assignment of each sample to a category. Instead of deciding that each trial belongs to a certain class,this statistical approach makes it possible to consider that an observation could, more or less likely, come from different classes [4].

$$p(w_k|\mathbf{x_t},\theta^{(old)}) = \frac{\pi_k^{(old)}p(\mathbf{x_t}|w_k,\theta)}{\sum_k \pi_k^{(old)}p(\mathbf{x_t}|w_k,\theta)} \tag{3.46}$$

The new values of the latent variables can now be calculated as:

$$\theta^{(new)} = arg\max_\theta Q(\theta,\theta^{(old)}) \tag{3.47}$$

Until now, the procedure has remained general enough to be carried out for mixtures of arbitrary distributions that can be described by latent variables. Now, the gradient of this function with respect to our specific variables that describe the mixture of normal distributions can be obtained. Starting with the means [9]:

$$\nabla_{\boldsymbol{\mu}_k}Q(\theta,\theta^{(old)}) = -\left\langle p(w_k|\mathbf{x_t},\theta^{(old)})\frac{1}{2}(\mathbf{x_t}-\boldsymbol{\mu}_k)\right\rangle_t \tag{3.48}$$

This gradient is then equated to zero in order to find a minimum.

$$\left\langle p(w_k|\mathbf{x_t},\theta^{(old)})\mathbf{x_t} - p(w_k|\mathbf{x_t},\theta^{(old)})\boldsymbol{\mu}_k^{(new)}\right\rangle_t = 0 \tag{3.49}$$

$$\left\langle p(w_k|\mathbf{x_t},\theta^{(old)})\mathbf{x_t}\right\rangle_t = \left\langle p(w_k|\mathbf{x_t},\theta^{(old)})\boldsymbol{\mu}_k^{(new)}\right\rangle_t \tag{3.50}$$

$$\boldsymbol{\mu}_k^{(new)}\left\langle p(w_k|\mathbf{x_t},\theta^{(old)})\right\rangle_t = \left\langle p(w_k|\mathbf{x_t},\theta^{(old)})\mathbf{x_t}\right\rangle_t \tag{3.51}$$

We define T_k as follows, where it can be understood as the effective number of data points assigned to the kth class. As this assignment was soft, there is no need for this value to be an integer.

$$T_k = \sum_t p(w_k|\mathbf{x_t},\theta^{(old)}) \tag{3.52}$$

By replacing (3.52) in (3.51):

$$\boldsymbol{\mu}_k^{(new)} = \frac{1}{T_k}\sum_t p(w_k|\mathbf{x_t},\theta^{(old)})\mathbf{x_t} \tag{3.53}$$

This result can be interpreted as the mean of all the samples, ponderated by the role they had in explaining the kth cluster [4].

The covariance matrix can be improved following the same procedure. This time the gradient is [9]

$$\nabla_{\boldsymbol{\Sigma}^{-1}} Q(\theta,\theta^{(old)}) = \left\langle p(w_k|\mathbf{x_t},\theta^{(old)})\frac{1}{2}\left[\boldsymbol{\Sigma} - (\mathbf{x_t}-\boldsymbol{\mu}_k)(\mathbf{x_t}-\boldsymbol{\mu}_k)^T\right]\right\rangle_t \tag{3.54}$$

Which leaves the new estimate:

$$\boldsymbol{\Sigma}^{(new)} = \left\langle \sum_k p(w_k|\mathbf{x_t},\theta^{(old)})(\mathbf{x_t}-\boldsymbol{\mu}_k)(\mathbf{x_t}-\boldsymbol{\mu}_k)^T\right\rangle_t \tag{3.55}$$

A similar process is used to find the new mixing coefficients. However, this time there is another constraint. All the mixing coefficients must sum to one and therefore a Lagrangian function can be written to represent this problem [9].

$$L(\theta,\lambda) = Q(\theta,\theta^{(old)}) - \lambda(1-\sum_k \pi_k) \tag{3.56}$$

$$\nabla_{\pi_k} L(\theta,\lambda) = \left\langle p(w_k|\mathbf{x_t},\theta^{(old)})\left(\frac{1}{\pi_k}\right)\right\rangle_t - \lambda \tag{3.57}$$

$$0 = \sum_k \pi_k \nabla_{\pi_k} L(\theta,\lambda) = \sum_k T_k - \lambda\sum_k \pi_k = \sum_k T_k - \lambda \tag{3.58}$$

Where T_k was the effective number of samples from each class. Then, the sum of all the samples assigned to each class ammounts to the total number of samples T:

$$0 = \sum_k T_k - \lambda_* = T - \lambda_* \tag{3.59}$$

$$\lambda_* = T \tag{3.60}$$

Replacing in (3.57):

$$\nabla_{\pi_k} L(\theta,\lambda) = \left\langle p(w_k|\mathbf{x_t},\theta^{(old)})\left(\frac{1}{\pi_k}\right)\right\rangle_t - T = \frac{T_k}{\pi_k} - T \tag{3.61}$$

Equating this gradient to zero, the new values of the mixing coefficients are:

$$\pi_k^{(new)} = \frac{T_k}{T} \tag{3.62}$$

The process of calculating the expected posterior probabilities (E step) and then maximizing the likelihood function (M step) is carried out iteratively until either the likelihood function or the value of the parameters converge to a certain number, giving name to the expectation-maximization algorithm.

In this case, we assumed that the covariance matrix was shared by both classes. The main reason for this was given when developing the LDA classifier in 3.1.2, however, there is another reason to do so in this algorithm. If there are no restrictions on both matrices, it is possible for one of the distributions to collapse onto a single data point [4]. The mean of this class will then be equal to the position of this single trial and the covariance matrix will tend to zero. The other distribution will try to explain the rest of the data set. This singularity is not desirable, and one way to avoid it is by detecting when it occurs and reseting the mean to a random value.

The gaussian EM algorithm can be relatively slow, and in order to speed up the convergence, instead of choosing the initial value of the latent variables randomly, K-means may be employed to find a rough estimate of the parameters. These can be then used to initialize the EM algorithm closer to a convergence point [4].

Once the statistical model has been stablished, we can go back to the Bayesian theory of probability to assign every new trial to the most likely category. The most straightforward method is to compute $p(w_k|\mathbf{x_t}) = p(\mathbf{x_t}|w_k)\pi_k$ for each class and choose the class which yields the highest value, but there are other alternatives. In fact, we have all the necessary elements to implement the LDA classifier as it was described in 3.1.2. However, this time we also have an estimate of the prior probabilities for each class in our model, the mixing coefficients, which will bias the border towards one of the means.

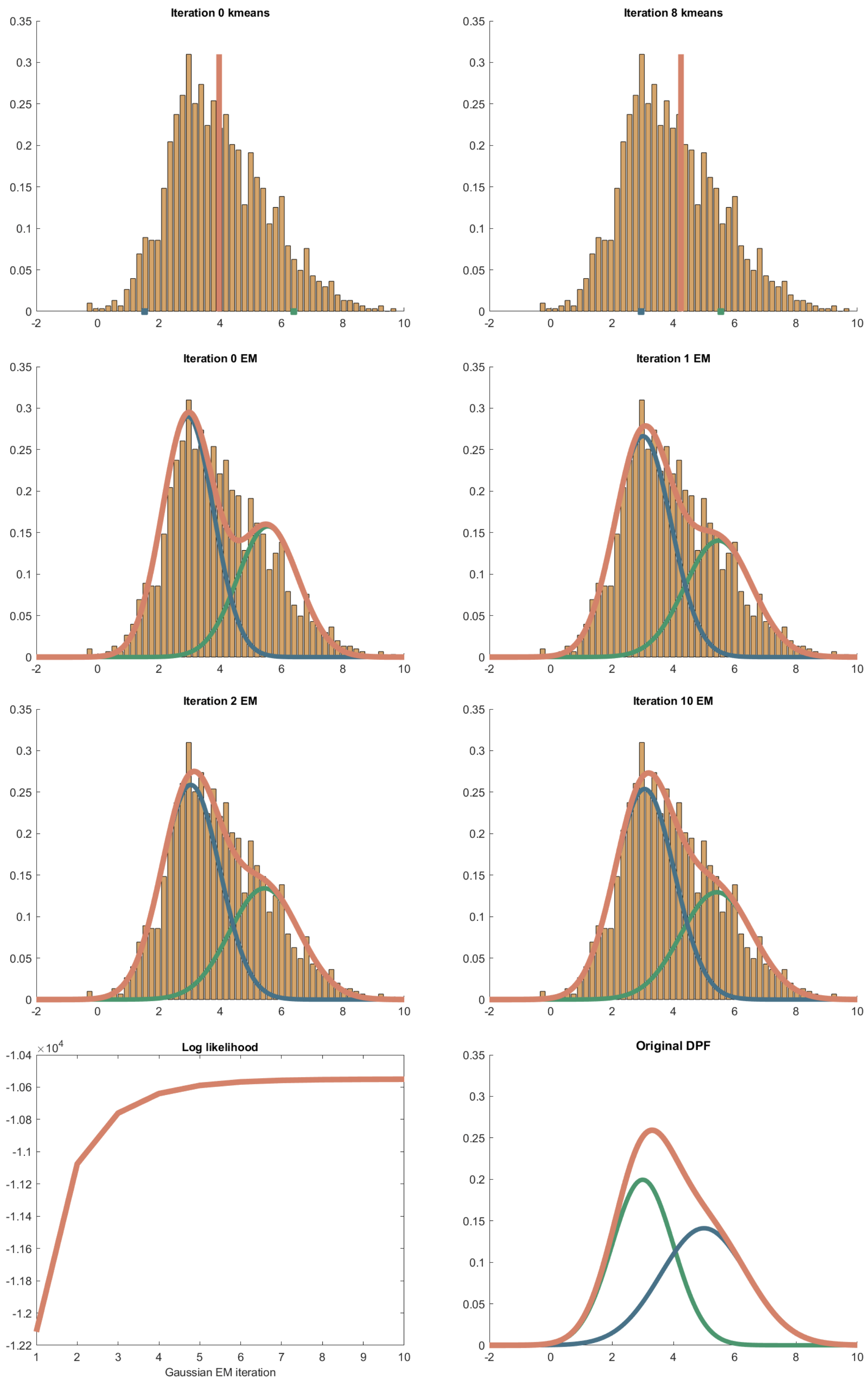

Figure 3.10 The K-means algorithm is used to initialize the model. After that, the classifier finds the most probable likelihood functions and mixing coefficients. The received unlabeled observations are represented by a sampled probability density function and the shown likelihood functions are already pondered by the mixing coefficients. The log likelihood function improves on every iteration until converging to a local maximum.

3.3 Label matching

A difficulty that arises from the use of an unsupervised classifier is the uncertainty problem. If there are k different classes, for every solution there are other equally valid $k! - 1$ solutions that correspond to the different $k!$ ways of matching each class to a category label.

In order to solve this uncertainty, some information about the labels is needed. We will try to achieve this with as few labeled trials as possible, as doing otherwise would go against the purpose of having an unsupervised classifier.

A simple approach consist of obtaining a representative $\mathbf{x_k}$ from each category. For our unsupervised classifiers, the means from each class are a good candidate for this purpose. Then, a set of T_k labeled trials from each class are chosen, and the distance from each representative to all of these T labeled trials is computed. The most common label amongst the closest N neighbours is assigned to the representative $\mathbf{x_k}$ and therefore to all elements that fall into that class. This is very similar to the closest N neighbours supervised classifier, however, in this case there is another restriction that should be noted. It is possible for both classes to be matched with the same label. This is undesirable as the classifier would match every element to the same label.

In order to avoid this problem, a tiebreaker is needed. This could be for example the sum of the squared distance to every element labeled as the contending class. Other approach consist of either decreasing or increasing the value of N until a consensus is reached. The latter method will be the one used in this work because of its simplicity. Still, despite being unlikely, it is possible that this method will not solve the tie, and in this case each class is matched to a different label randomly.

This algorithm is sensible to two different parameters, the number of labeled samples from each class T_k and the starting number of neighbours N. The tuning of these parameters is important as a failure in this stage would not slightly hinder the performance but render the whole classifier useless. The elements categorized correctly by the classifier would now go into the incorrect class and vice versa. During the testing phase in chapter 4, it will be shown how this procedure is robust for a relatively low number of labeled samples in those cases where the performance of the classifier itself is consistent.

3.4 Summary

In this chapter we have studied some well known classification algorithms, the differences between supervised and unsupervised methods and the limitations that accompany the latter, such as the need of an unsupervised dimensionality reduction algorithm or a procedure in order to associate each cluster with the corresponding category using as few labeled samples as possible.

Amongst the supervised classifiers we have analysed two algorithms based on the Bayesian theory of probability and that exploit the Gaussian distribution of the samples, the Bayesian classifier and Linear Discriminant Analysis. These two are related under certain conditions to the nearest archetype classifier which, similarly to the N nearest neighbours and gradient descent algorithms, uses the geometric properties of data clusters such as shorter distances between the elements of the same class. Lastly, we explored similar classifiers that instead of using features in an euclidean space, are capable of working in the natural space of the covariance matrices of the signals both before and after filtering with spatial filters, the Riemannian manifold of SPD matrices.

For our unsupervised methods we chose a clustering algorithm known as K-means. As this classifier tends to find spherical clusters and does not account for the difference in variance of the clusters across dimensions, a statistical approach based on the Gaussian mixture model was used to refine the results with the iterative EM algorithm.

Finally, a simple method for associating clusters with labels was proposed to solve the uncertainty problem.

4 Experimental results

Truth is what stands the test of experience.

Albert Einstein

4.1 Defining the experiment

The objective of this experiment is to test different classifiers against the same Motor Imagery detection task. The data was obtained from nine different users, and two sessions from each user are available.

During each session, 72 movements from each of four possible different movements were registered: tongue, feet, left hand and right hand. A more detailed description of this data set can be found in chapter 1.

It is considered that distinguishing left hand from right hand movement is the most difficult classification task among the possible binary categorization problems in this data set, and therefore this will be the main focus of this work.

As the data set is limited and testing each classifier only once would not be statistically significant due to the random nature of the tests, a Monte Carlo experiment will be carried out. For each session and classifier, the test is repeated a hundred times. Despite a hundred tests not being a very large number, this is sufficient to provide an acceptable approximation of the intended result as it will be reflected on the mean standard deviation shown for each experiment. This limitation is necessary as the process is a computationally expensive task, and increasing the number of tests results in a similar increment on the time necessary to perform them. For each test, the available samples are divided into two subsets. The first one will be used to train the classifier and the second one will be used to test the performance of the trained algorithm.

This division is necessary as if we were to use the same set for both purposes, the results would not reflect the real fitness of the classifiers. This problem where a classifier performs exceptionally well with the training set but cannot categorize the test set correctly is known as overfitting and is a common issue in pattern recognition tasks and machine learning. This is due to the classifier memorising the training set rather than it learning the defining features of each class.

Another important parameter that we have to define is the size of these subsets, as an increased number of training samples will result in a better estimation of the parameters that gobern the behaviour of the classifier, yet the number of test samples must remain large enough to provide a representative estimation of the performance of the classifier. In our case, the training set will be four fifths of the total number of samples, and the remaining fifth will be used to test the performance. This division is carried out randomly for each of the Monte Carlo runs.

As for most of our experiments we will be using CSP, the number of spatial filters used is also a relevant parameter. The first spatial filters provide in general the most significant features. Every added filter will result in added information, however, this might not always improve the performance of the classifiers and in fact it might hinder it. The value that allows for the best classification results varies between algorithms, but we will use four spatial filters as in general this is a robust value and provides a better average performance across algorithms.

4.2 Individual algorithms and their variants

We may start by testing the classifiers proposed in the previous chapter and some of the modifications that could result in a better performance when accounting for the peculiarities of our data set.

4.2.1 Bayesian classifier

We test the two proposed variants of the Bayesian classifier in conjunction with CSP. The first version is the standard algorithm. The covariance matrix is individual to each class and is calculated with the ML estimate [42].

For the other algorithm, only the diagonal elements of the covariance matrices are maintained, and the other entries are set to zero. This is done for two reasons. The first one is that we know that the features extracted from the spatial filters are uncorrelated, and therefore, given enough samples, the ML estimate should tend to the form of a diagonal matrix. The second reason is that, although this is not the main focus of this work, this task is considerably less computationally expensive, which is convenient towards an implementation for a real application.

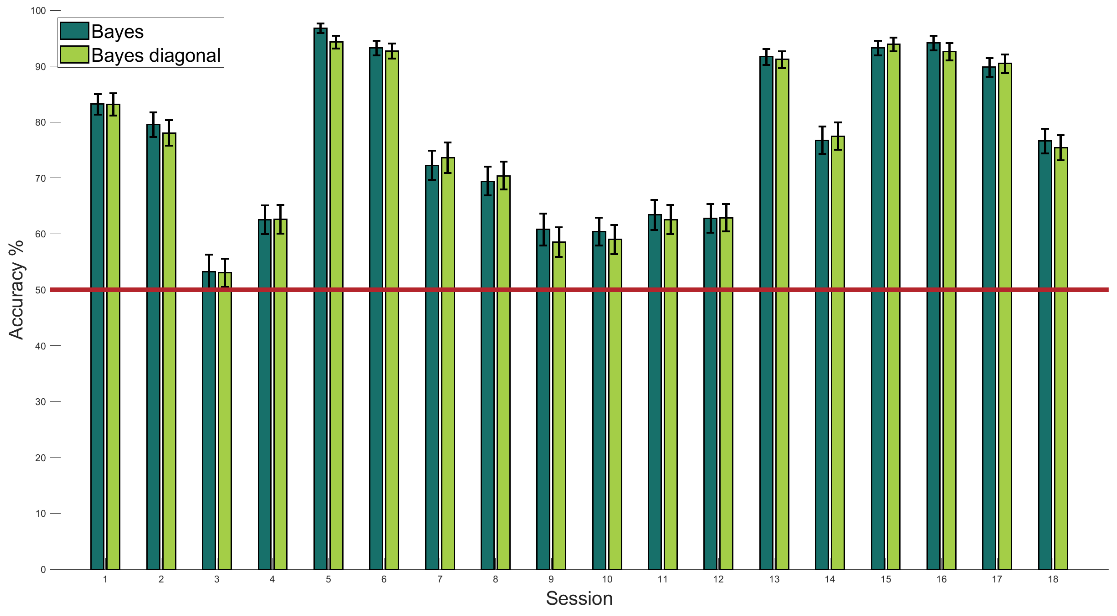

Figure 4.1 The average performance in every session is shown for both versions of the Bayesian classifier. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

Table 4.1 The average performance across sessions and mean standard deviation is written for both versions of the Bayesian classifier, as well as the average for each session.

Classifier	**Accuracy**	**StDev**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**	**S17**	**S18**
Bayes	**76,66**	3,34	**83,20**	**79,56**	**53,21**	62,52	**96,79**	**93,26**	72,22	69,41	**60,81**	**60,41**	**63,41**	62,76	**91,69**	76,75	93,29	94,19	89,83	**76,60**
Bayes diagonal	76,22	3,33	83,17	78,03	53,03	**62,59**	94,33	92,72	**73,58**	**70,37**	58,56	59,02	62,56	**62,87**	91,19	**77,47**	**93,94**	**92,65**	**90,48**	75,39

The standard version seems to perform slightly better during most sessions in this case, however, the difference is small enough to consider using the faster method instead.

4.2.2 Linear discriminant analysis

We will now compare several variants of the LDA classifier. The first one is the standard LDA algorithm, using a covariance matrix shared between classes. As in the previous section, the first modification will be to consider that the sample covariance matrix is diagonal, setting to zero all the elements outside of the main diagonal.

The second alteration is to displace the boundary that separates the decision regions, which is equivalent to altering the prior probabilities. First, all the training samples are projected on the line that intersects the means of the two classes. Then, the median of these trials along this axis is calculated and the boundary is offset to intercept this point without changing the normal vector that defines the hyperplane. The reason behind this alteration is that the median is known to be a more robust metric than the mean against outliers [39]. Finally, a combination of both modifications is tested.

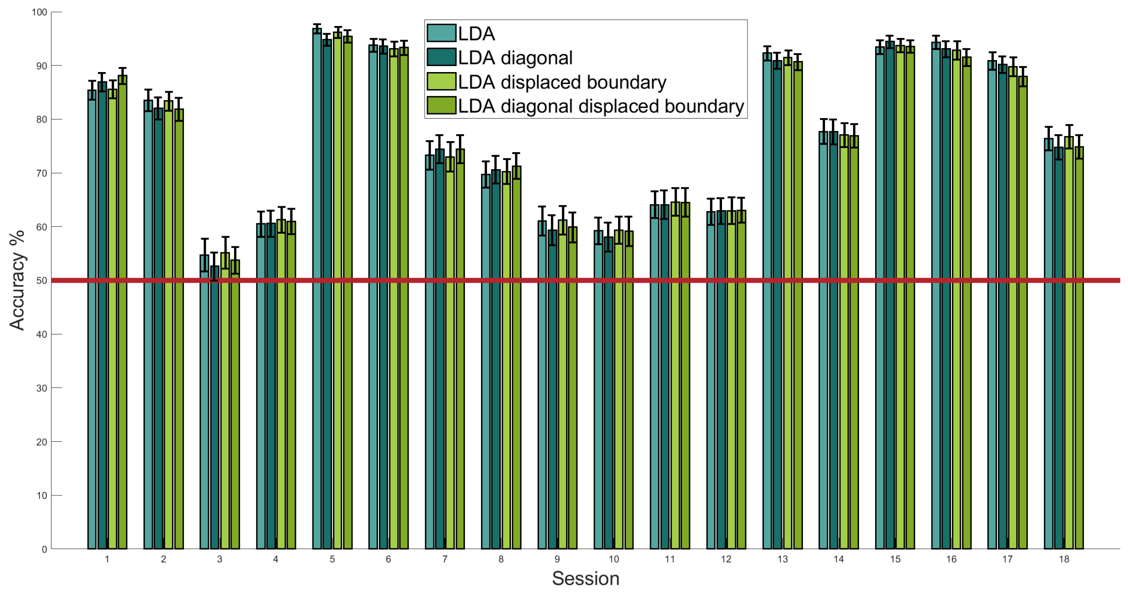

Figure 4.2 The average performance in every session is shown for the different versions of the LDA classifier. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

Table 4.2 The average performance across sessions and mean standard deviation is written for the different versions of the LDA classifier, as well as the average for each session.

Classifier	**Accuracy**	**StDev**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**	**S17**	**S18**
LDA	**77,23**	3,41	85,41	**83,52**	54,74	60,52	**96,86**	**93,80**	73,29	69,69	61,08	59,23	64,09	62,76	**92,30**	**77,72**	93,44	**94,33**	**90,91**	76,39
LDA diagonal	76,72	3,43	86,94	82,06	52,64	60,59	94,83	93,58	**74,40**	70,56	59,34	58,09	64,09	62,91	90,91	77,64	**94,43**	93,08	90,16	74,75
LDA displaced boundary	77,10	3,30	85,59	83,38	**55,17**	**61,34**	96,19	93,12	72,97	70,22	**61,26**	**59,38**	**64,59**	62,98	91,44	77,04	93,72	92,83	89,80	**76,71**
LDA diag.disp. boundary	76,73	3,31	**88,09**	81,85	53,78	60,99	95,44	93,33	**74,40**	**71,26**	59,91	59,17	64,51	**63,05**	90,66	76,89	93,54	91,55	87,95	74,85

As the models obtained by the different variants are fairly similar, there are no great differences between the performances of the algorithms.

4.2.3 K Nearest Neighbours

The K Nearest Neighbours classifier[1] is based on finding the closest labeled trials to every element that we wish to classify. This observation will be assigned to the category which is the most frequent amongst the first N of the nearest labeled samples. As a result, the value that this parameter N takes is important and will dictate the behaviour of the classifier. We will compare the first nine possible values of N.

For most sessions, there is a common trend. The average performance improves quickly with the value of N until getting close to a maximum. Then, the average is approximately maintained with a very slight increase. The maximum is finally reached when comparing 31 neighbours (77% accuracy), and after that the average diminishes slowly until collapsing to a 50%.

The changes between the different values of N above 15 and below 31 neighbours are minimal and therefore any value in this range could be used consistently. The ideal value of N is dependent on the number of labeled samples available, and therefore it would change if we had either more or less samples.

[1] This algorithm is commonly reffered to as the K Nearest Neighbours classifier, however, due to a conflict in notation the number of used neighbours will be denoted as the parameter N instead of K, which is reserved for the different classes in this document

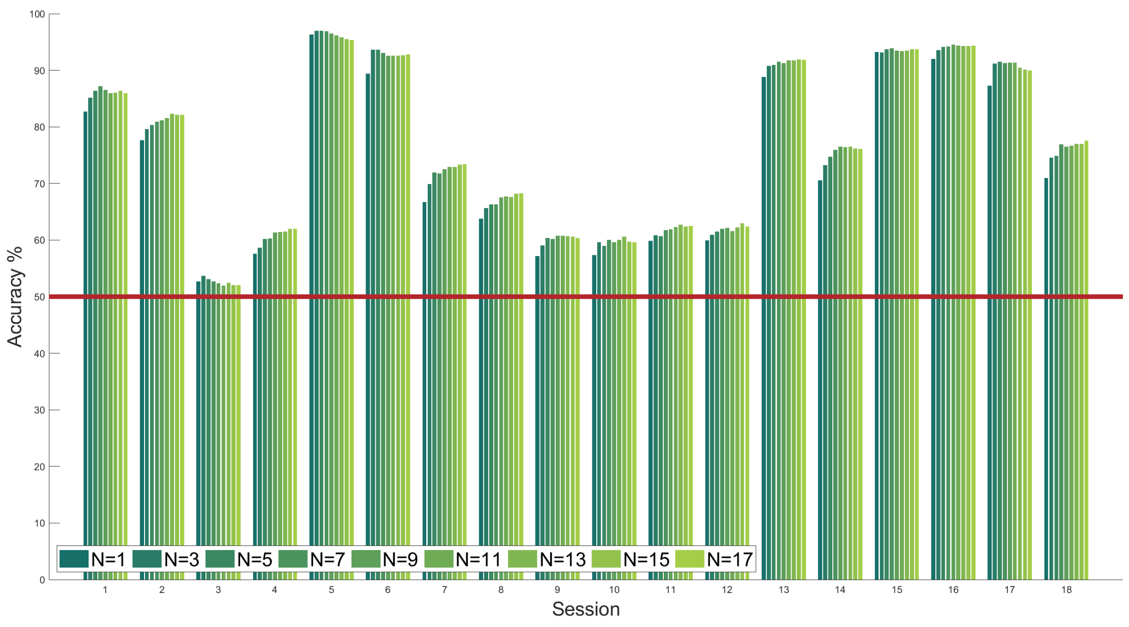

Figure 4.3 The average performance in every session is shown for different values of N. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

Table 4.3 The average performance across sessions and mean standard deviation is written for the different values of N, as well as the average for each session.

N	**Accuracy**	**StDev**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**	**S17**	**S18**
1	73,56	3,51	82,70	77,67	52,67	57,59	96,37	89,37	66,69	63,81	57,20	57,34	59,84	59,91	88,80	70,58	93,22	92,01	87,30	70,94
3	75,58	3,55	85,19	79,63	**53,70**	58,66	**97,01**	**93,62**	69,90	65,66	59,09	59,62	60,84	60,97	90,80	73,22	93,15	93,58	91,19	74,57
5	76,13	3,55	86,41	80,38	53,13	60,17	97,00	**93,62**	71,97	66,30	60,37	58,99	60,67	61,54	90,91	74,69	93,72	94,12	**91,51**	74,89
7	76,49	3,53	**87,19**	80,88	52,67	60,27	96,93	93,08	71,78	66,34	60,20	60,02	61,78	62,01	91,51	75,97	**93,90**	94,19	91,26	76,89
9	76,58	3,48	86,55	81,13	52,35	61,31	96,51	92,58	72,50	67,51	**60,74**	59,66	61,88	62,11	91,30	**76,47**	93,47	**94,54**	91,37	76,53
11	76,61	3,47	85,98	81,59	51,93	61,38	96,15	92,55	72,93	67,73	**60,74**	60,05	62,35	61,62	91,76	76,43	93,40	94,40	91,33	76,64
13	76,71	3,42	86,02	**82,27**	52,43	61,49	95,87	92,55	72,93	67,63	60,67	**60,62**	**62,74**	62,22	91,76	**76,47**	93,47	94,26	90,47	76,96
15	**76,73**	3,43	86,41	82,13	52,04	**61,95**	95,54	92,65	73,32	68,16	60,63	59,74	62,38	**62,94**	**91,94**	76,19	93,72	94,26	90,16	76,96
17	76,69	3,43	85,98	82,17	52,08	**61,95**	95,33	92,83	**73,39**	**68,26**	60,35	59,59	62,49	62,41	91,84	76,11	93,69	94,36	89,98	**77,60**

4.2.4 Nearest archetype

The nearest archetype classifier finds a representative for each class and then assigns every observation to the category which archetype is the closest.

The first presented version calculates this element for each class using the mean, as this metric minimizes the squared distance to every labeled sample.

The second proposed idea consists on calculating the archetypes as the median of all the labeled samples from each class. The reason that supports this concept is that, as it was stated previously, the median can be more robust against outliers than the mean [39]. In this case, the meaning of median in this context should be clarified, as there is no single definition for the median of multiple points in an R^n space.

For simplicity, we will define it as a dimensionwise median, which means that for each dimension of the R^n space, only that dimension will be accounted for while calculating this metric. The reasoning behind this is that the features should be uncorrelated and therefore, as each feature lies in a different dimension, it is possible to calculate this metric for each feature independently. Unlike with the classic deffinition of median in a one dimensional space, the element resulting from this method does not belong in general to the original set of points used to calculate its value[2].

The use of both metrics result in a very similar average performance for every session and both methods are valid in order to implement a classifier.

Another two possible representatives of each class were tested, however, as neither of them resulted in a better classifier[3] and they were reasonably more computationaly expensive and complex to implement, they were not presented here for clarity.

[2] This is supposing that the number of elements in the set is odd, as if it were to be even, the result would not be part of it either, but rather the mean of the two central elements.

[3] Despite these methods not resulting in a higher accuracy, they provided a very similar average, being valid methods as well

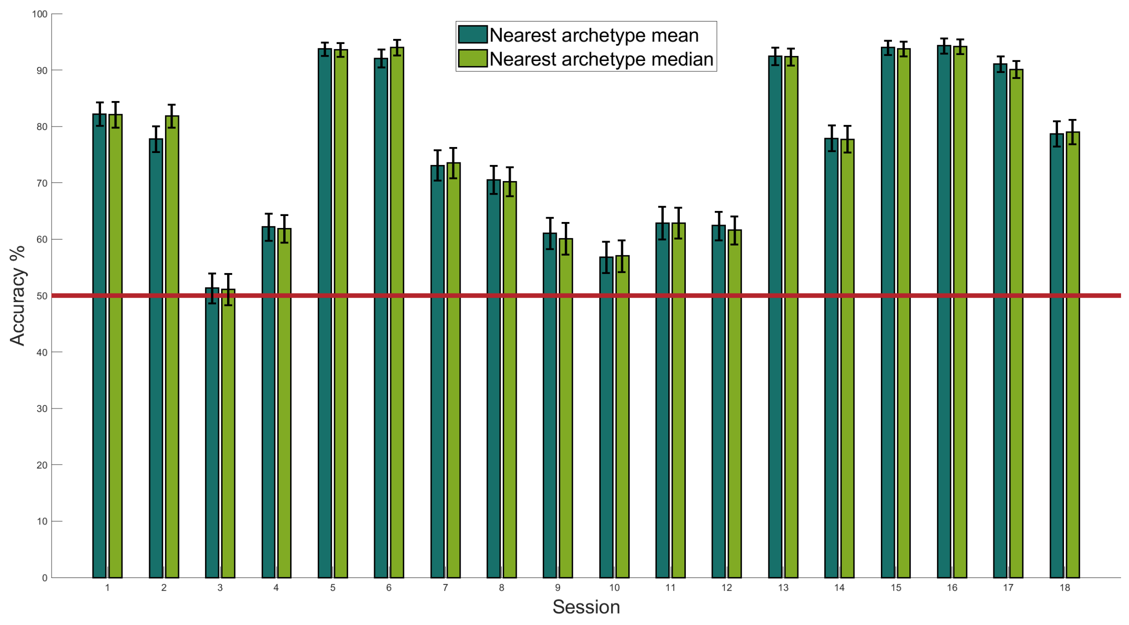

Figure 4.4 The average performance in every session is shown for the different versions of the nearest archetype classifier. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

Table 4.4 The average performance across sessions and mean standard deviation is written for the two different methods of calculating the archetype, the mean and median of the labeled trials. The average for each session is also registered for each method.

Classifier	**Accuracy**	**StDev**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**	**S17**	**S18**
Mean	76,35	3,41	**82,20**	77,75	**51,32**	**62,16**	**93,72**	92,08	73,07	**70,47**	**61,05**	56,80	62,83	62,41	**92,44**	**77,89**	**93,97**	**94,29**	**91,08**	78,71
Median	**76,49**	3,47	82,06	**81,85**	51,10	61,88	93,58	**94,01**	**73,50**	70,15	60,09	**57,02**	**62,87**	**61,59**	92,33	77,71	93,76	94,19	90,12	**79,03**

Both of these were based on the idea of medoid, an element from a group of points that minimizes the total sum of a distance to the remaining points in the set. In our case, both the Euclidean distance and squared Euclidean distance were tested.

4.2.5 Gradient descent

This classifier is implemented in two different variants. The first one is the previously proposed method of training a single classifier using gradient descent.

The second one was also hinted at the end of the section describing this algorithm and consist on training two distinct classifiers in opposite directions and then testing for the greatest output.

For both alternatives, the test is carried out after training with every labeled element once and after repeating this process five times. After some experiments, the learning rate that provides better results is 0.01 with a target value V of ± 10.

Table 4.5 The average performance across sessions and mean standard deviation is written for the two different methods and for one and five training cycles, as well as the average for each session.

Classifier	Accuracy	StDev	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18
Single, 1 cycle	75,01	3,76	87,41	79,20	49,18	58,38	94,33	93,93	67,30	66,76	58,06	52,34	61,91	61,99	91,44	74,00	93,54	94,72	89,48	76,21
Single, 5 cycles	76,57	3,61	87,09	83,52	51,58	**61,02**	**96,54**	94,22	70,87	**69,58**	**59,53**	54,49	**64,23**	**63,38**	**92,51**	75,68	93,65	94,47	90,37	75,60
Adversarial, 1 cycle	75,07	3,79	**87,62**	79,10	48,79	58,41	94,40	**94,54**	67,76	67,15	57,45	52,80	61,94	60,99	91,40	74,50	**93,69**	**95,11**	89,19	**76,32**
Adversarial, 5 cycles	**76,58**	3,61	87,12	**83,84**	**52,01**	60,63	96,47	94,15	**71,30**	69,26	59,13	**54,63**	64,19	63,13	92,44	**76,14**	93,51	94,47	**90,55**	75,46

Both versions of the classifier perform similarly, and training until convergence improves the result slightly in most sessions.

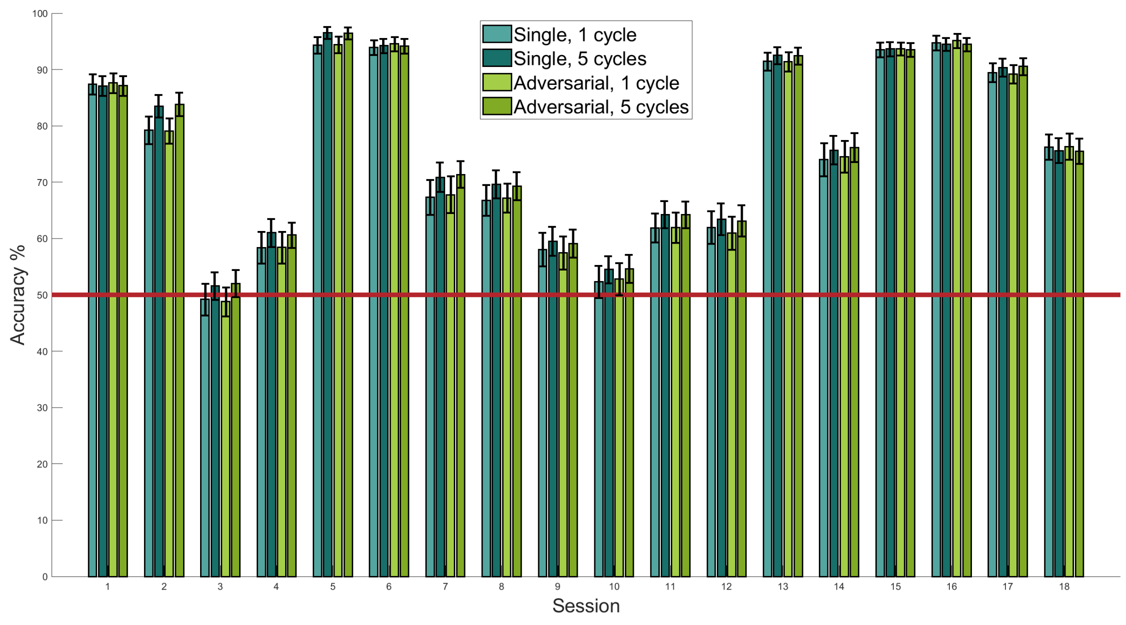

Figure 4.5 The average performance in every session is shown for the gradient descent algorithm and for the adversarial version, where two classifiers are tested against the other. The test is carried out for one and five training cycles of each version. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

4.2.6 Classifiers in a riemannian manifold

We present three different classifiers based on Riemannian metrics for the covariance matrices of the EEG signals.

The first two versions are the Minimum Riemannian Distance to Mean[4] classifier, using both the Euclidean and Riemannian mean[5]. This classifier is equivalent to the nearest archetype algorithm using the mean as the representative element for each of the categories, however, the metrics used in order to calculate distances differ, as the Euclidean metrics are not applicable in the space of SPD matrices.

The last algorithm is the N nearest neighbours for $N = 31$, using the geodesic distance to find the closest labeled elements.

Although the algorithms that use the covariance matrices of the EEG signals as features have the advantage of not needing to calculate the spatial filters, it is also possible to use them in conjuction with CSP.

Table 4.6 The average performance across sessions and mean standard deviation is written for the three algorithms, as well as the average for each session. CSP was not used in these experiments..

Classifier	**Accuracy**	**StDev**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**	**S17**	**S18**
MRDEM	75,50	3,57	82,81	**77,65**	55,74	56,17	**93,72**	91,83	**76,75**	**73,65**	58,02	52,92	60,84	59,88	**92,97**	**74,82**	**96,15**	**95,04**	**85,20**	**74,93**
MRDRM	**76,22**	3,35	82,81	**77,65**	**58,31**	**57,85**	**93,72**	91,83	**76,75**	**73,65**	**60,05**	**57,17**	**62,34**	**60,74**	**92,97**	**74,82**	**96,15**	**95,04**	**85,20**	**74,93**
Nearest 31	72,25	3,79	**83,13**	70,87	49,45	52,24	92,69	**92,30**	70,18	73,33	58,17	51,67	56,98	54,24	89,05	62,69	90,76	93,72	85,56	73,43

Table 4.7 This experiment is similar to that on table (4.6), but fours spatial filters were used.

Classifier	**Accuracy**	**StDev**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**	**S17**	**S18**
MRDEM	75,64	3,40	80,17	77,29	51,54	62,69	91,01	91,76	71,36	**69,86**	59,27	55,81	63,76	60,80	**92,58**	**78,63**	**94,62**	**95,11**	88,06	**77,21**
MRDRM	76,33	3,17	80,17	77,29	**57,24**	**63,33**	91,01	91,76	71,58	**69,86**	**60,66**	**58,30**	**64,62**	**61,94**	**92,58**	**78,63**	**94,62**	**95,11**	88,06	**77,21**
Closest 31	**76,58**	3,52	**86,27**	**82,78**	53,72	60,56	**94,22**	**93,51**	**71,79**	68,58	59,80	56,53	62,45	61,30	92,40	78,52	94,11	94,40	**91,01**	76,53

[4] The term Riemannian distance is used here loosely to reference the geodesic distance in a Riemanian manifold for SPD matrices, distinguishing it from the classic definition of Euclidean distance.

[5] The iterative algorithm used to approximate the Riemannian mean was based on the algorithm proposed by H. Karcher [12] and the MATLAB function that implements it was provided by the department of Signal Theory and Communications from the University of Seville

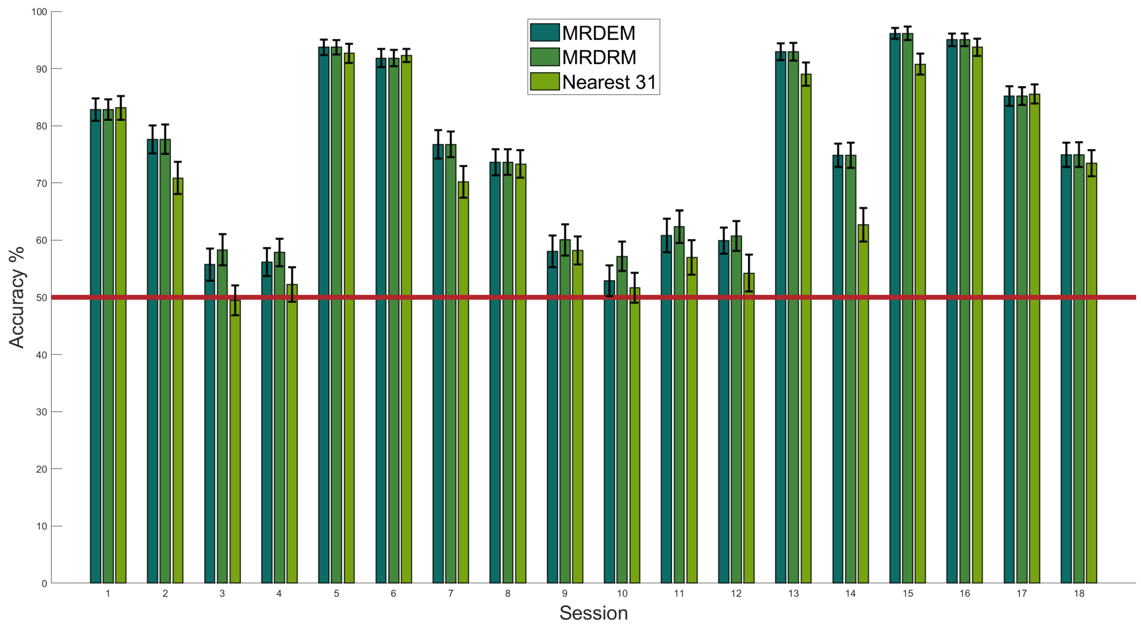

Figure 4.6 The average performance in every session is shown for the three different algorithms, both Minimum Riemmanian distance to Riemannian and Euclidean Mean and the 31 nearest neighbours classifier. CSP was not used in these experiments. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

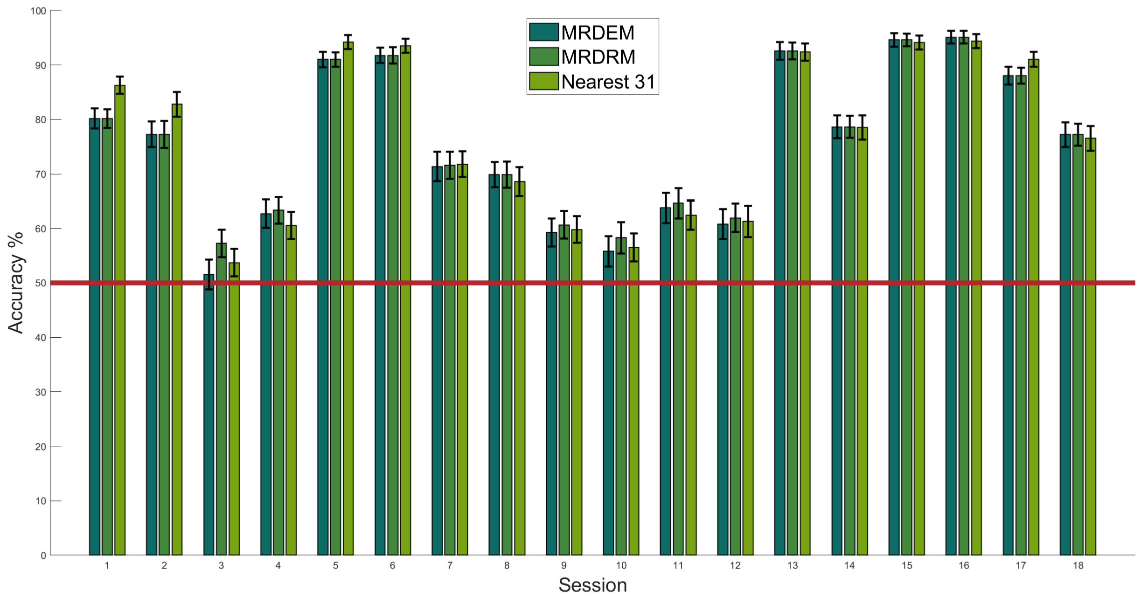

Figure 4.7 This experiment is similar to that on figure (4.6), but four spatial filters were used..

There is not a great difference between using the Euclidean or Riemannian mean in most sessions. Nonetheless, as it was expected, the second method is slightly more accurate. Both of them provide a better performance than the 31 nearest neighbours algorithm without the use of CSP. This is still true for other tested values of N.

On the other hand, when using four spatial filters obtained with CSP, the nearest neighbours classifier performs better than in the last scenario, and the three methods seem viable.

4.2.7 K-means

As K-means is an unsupervised classifier, it is not possible to use CSP. This algorithm makes use of the labels of the samples and employing it would therefore defeat the purpose of having an unsupervised algorithm. However, an alternative, unsupervised version of CSP has been proposed [30] that would solve this problem.

Moreover, the clusters also have to be matched with the corresponding labels. In order to do so, we will use the previously discussed label matching algorithm with three labeled elements from each class. Each category will be matched with the most common label amongst the three closest samples.

First, we will use this algorithm in conjunction with supervised CSP. This is to provide a frame of reference connecting the performance of this classifier to the previously studied supervised classifiers using the same dimensionality reduction technique. Two versions will be tested. For the first one, if the performance is under 50% for a Monte Carlo run, an error in the label matching algorithm is assumed and the result is inverted, effectively automatically matching the categories with the correct labels. The second version uses the proposed label matching algorithm.

The second set of tested classifiers are the same variants in combination with unsupervised CSP.

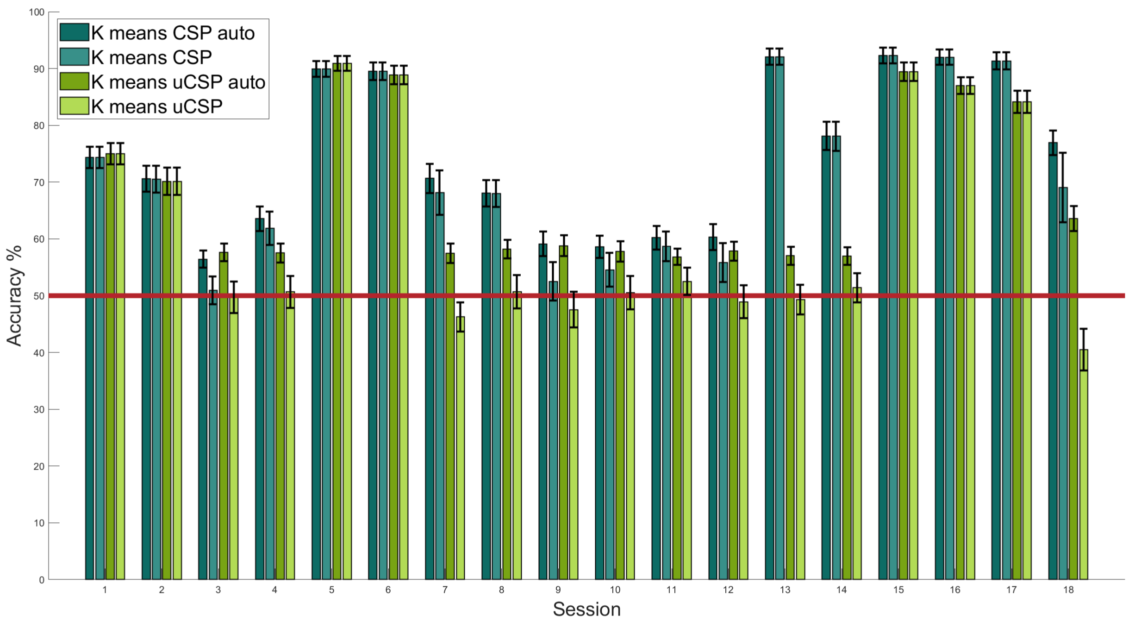

Figure 4.8 The average performance in every session is shown for the K-means algorithm, in conjunction with supervised CSP and unsupervised CSP (uCSP), using the label matching algorithm and automatically matching the labels with the classes. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

Table 4.8 The average performance across sessions and mean standard deviation is written for the K-means algorithm, as well as the average for each session. There are no highlighted cells as these classifiers are not tested on an even ground, and therefore there is no purpose in directly comparing the best performances.

Classifier	Accuracy	StDev	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18
K-means CSP auto	74,67	3,19	74,32	70,59	56,42	63,55	89,94	89,51	70,65	68,05	59,12	58,62	60,20	60,31	92,08	78,13	92,30	91,98	91,34	76,93
K-means CSP	72,76	3,60	74,32	70,51	50,93	61,90	89,94	89,51	68,15	67,98	52,52	54,56	58,70	55,84	92,08	78,06	92,30	91,98	91,34	69,03
K-means uCSP auto	68,07	3,23	75,00	70,12	57,63	57,53	90,90	88,87	57,46	58,20	58,80	57,78	56,85	57,84	57,02	56,99	89,44	87,02	84,14	63,59
K-means uCSP	62,42	4,30	75,00	70,12	49,71	50,67	90,90	88,87	46,26	50,68	47,53	50,50	52,50	48,93	49,30	51,42	89,44	87,02	84,14	40,47

The difference between the use of CSP and unsupervised CSP is existent yet very small for most sessions, which makes this a feasable technique to be implemented in a system. Nonetheless, during some sessions, specially 13 and 14, which belonged to the same person in the data set, uCSP failed to extract the underlying structure of the data while supervised CSP could. This matches the findings in the previously mentioned article where this method was proposed [30].

About our matching algorithm, in most cases where the performance of the classifier itself could be considered to be better than random, it worked almost flawlessly. The main exception is the last session, where for both the supervised and unsupervised CSP versions the algorithm failed and there are large differences between the versions with matched classes and automatically assigned classes. This is also reflected on the large mean standard deviation, as some experiments are located far from the others, under the 50% mark.

For the sessions that are at around 50%, the automatic match solution is fundamentally flawed. If the result is under 50% it is considered to be a missmatch, which is not necessarily true. If the result is far below 50%, this would most likely be the case, however, if it is closer to 50%, the possibility remains that the classifier failed and was categorizing each element into a random class. In this scenario, the small number of test samples would not be large enough for the accuracy estimation to approach the expected value of 50%. This is not a problem in the normal case because during the hundred Monte Carlo runs if the classifier is not working propperly, for each result above 50%, we would expect another one below it, providing a close estimate for the expected value at around the 50% mark.

Nevertheless, as for the automatic match we are correcting every test under 50% to be above, the expected value is misscalculated to always be over 50%. As a result, the performance registered for this method should be considered erroneous when it is near the 50% mark. This problem could be corrected by only considering label missmatches those cases under a lower value than 50%. Be that as it may, there is no clear way of distinguishing in which cases the label matching algorithm failed and in which the classifier failed, and as a result the best available option is to analyse the data with this limitation in mind.

A conclusion that we can extract is that although adding more labeled samples to the label matching algorithm would reduce the overall failure rate, using as little as three samples from each class would work in most scenarios where the classifier itself could be considered reliable. It is possible that a more careful choice of labeled samples would result in a better improvement in the label matching algorithm than the one achivable with more samples. This is something that should be accounted for when dessigning a system, the fact that the few supervised labeled samples should be obtained carefully in a controlled environment.

Another small change to this classifier would be to consider a version similar to that of the N nearest neighbours supervised classifier. In this case, the training is performed in the same way, with the K-means algorithm. During this process, the unlabeled samples are assigned to one of the clusters, generating synthetic labels. When deciding to which class an observation should be assigned to, instead of considering the closest mean, the labels of the nearest N samples are studied and the most repeated value is chosen as the category for this observation. This generates a similar but less linear boundary between the decision regions. In this case, the chosen value of N was 15.

Table 4.9 The average performance across sessions and mean standard deviation is written for the two different methods of assigning observations to clusters, as well as the average for each session.

Classifier	**Accuracy**	**StDev**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**	**S17**	**S18**
K-means	**62,42**	4,30	**75,00**	**70,12**	**49,71**	50,67	**90,90**	88,87	**46,26**	50,68	47,53	**50,50**	**52,50**	48,93	49,30	51,42	**89,44**	**87,02**	**84,14**	40,47
K-means nearest 15	62,21	4,24	74,54	66,80	49,47	**50,85**	90,73	**89,62**	46,05	**50,75**	**48,21**	**50,50**	51,75	**49,28**	**50,07**	**51,50**	89,01	85,91	84,03	**40,69**

This method did not provide any improvement over the original technique used to assign the observations to the clusters.

When starting K-means, an initial state is necessary, which may be found with another algortihm. There are different approaches that might result in a different final state. In our case, the initialization method used to start the algorithm was a random split. The elements used to train were initially divided into two categories and these were used in order to form the first two clusters. As this classifier is known to be sensible to the initialization parameters [14], other more complex methods were tested.

The first approach was to obtain the mean of all the points, calculate the distance from all elements to this global mean and choose one point randomly among the furthest ones to be the first archetype. The second archetype is also selected as one of the points amongst the furthest from the first mean. This guarantees that both starting points are far away from each other.

The second initialization approach is known as the K-means++ algorithm [2]. One point from the data set is randomly chosen in order to be the first archetype. The distance from every point to this sample is computed and the next representative is assigned to a random element with a probability proportional to this distance squared. If we had more classes, the previous steps would be repeated until having as many representatives as desired clusters. The only difference would be that in order to calculate the distance, as we have more than one mean, the distance from each sample to the closest representative would be used. This is

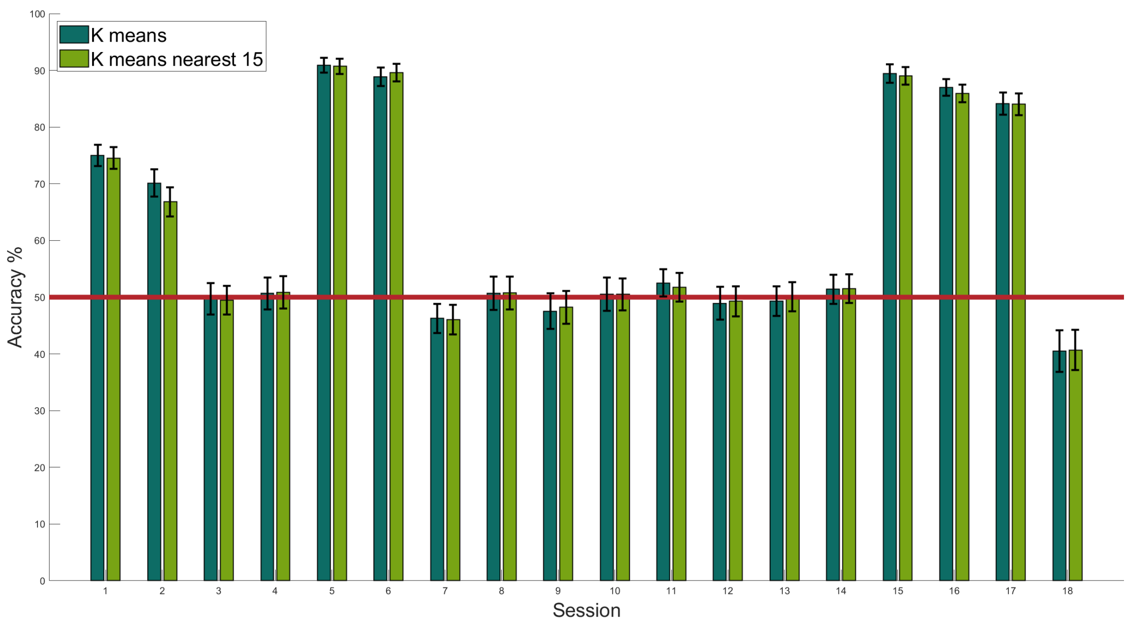

Figure 4.9 The average performance in every session is shown for the K-means algorithm, using the nearest mean and 15 nearest neighbours method to assign the observations to the clusters. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

done because it is more likely to have a starting mean near a group of points that are far away from any other archetype, that is, near a cluster that has no mean nearby to represent it.

As there were only two classes, neither of these methods resulted in an improved accuracy, however, in the multiclass case it is plausible that the initialization method would be more critical and the K-means++ algorithm would be recomended. This is because K-means can optimize the position of the archetypes locally, but it is difficult to realocate the position of these globally [14].

4.2.8 Gaussian EM

When studying this unsupervised classifier, we must account for the same difficulties that arised while testing the previous algorithm, and as a result we will make some of the same comparisons. The first of these will be the use of supervised CSP to provide a framework in which to compare the performance of these algorithms to the supervised classifiers when using the same features. The second one is the use of the automatic correction of the experiments to account for any possible missmatch when assigning a label to each cluster. Both this correction and assignment will be carried out in the same manner as it was done in the previous section.

Table 4.10 The average performance across sessions and mean standard deviation is written for the Gaussian EM algorithm, as well as the average for each session. There are no highlighted cells as these classifiers are not tested on an even ground, and therefore there is no purpose in directly comparing the best performances.

Classifier	**Accuracy**	**StDev**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**	**S17**	**S18**
Gaussian EM CSP auto	73,97	3,13	74,79	69,34	54,92	63,30	90,83	89,62	70,36	67,23	58,84	58,33	59,67	61,77	91,26	76,07	89,48	90,44	90,08	75,14
Gaussian EM CSP	71,66	3,58	74,79	69,34	49,72	61,98	90,83	89,62	65,58	67,16	53,81	54,07	57,02	55,24	91,26	76,00	89,48	90,44	90,08	63,46
Gaussian EM uCSP auto	67,55	3,20	74,68	67,99	57,34	57,06	91,62	88,48	57,92	56,20	58,59	58,21	57,06	57,24	56,60	56,78	87,05	85,27	84,64	63,09
Gaussian EM uCSP	62,07	4,24	74,68	67,99	50,22	49,75	91,62	88,48	46,50	49,93	51,10	47,83	52,32	51,65	48,97	50,33	87,05	85,27	84,64	38,94

The conclusions that we can draw for this classifier are also similar to those stated in the previous section. The missmatch correction is flawed near the 50% mark and in general the label matching algorithm works propperly with a very low number of labeled samples.

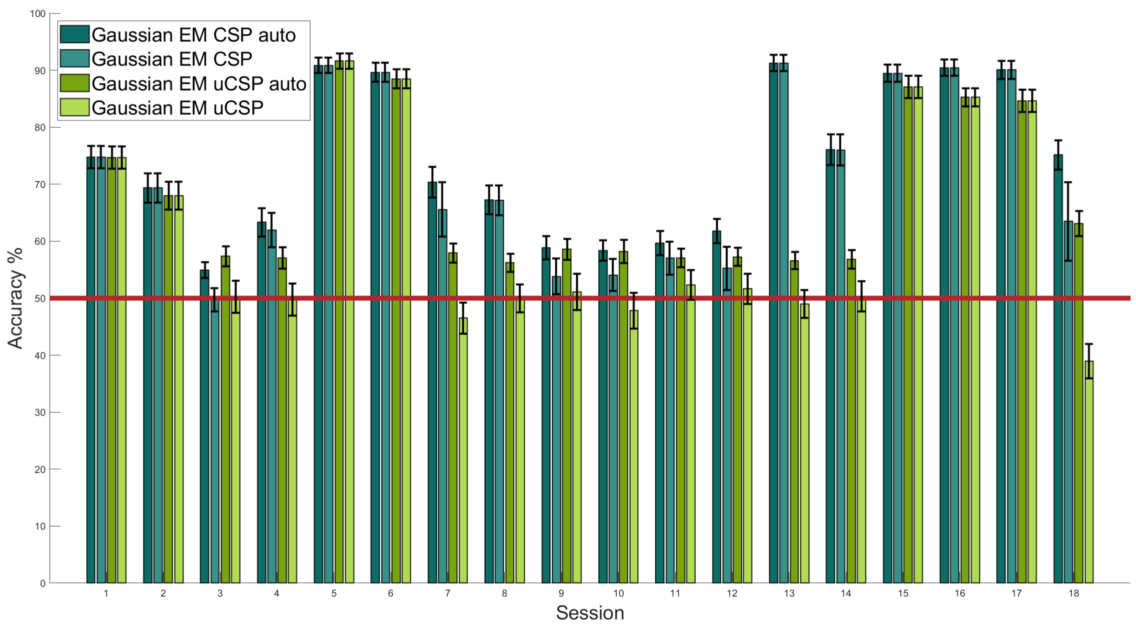

Figure 4.10 The average performance in every session is shown for the Gaussian EM algorithm, in conjunction with supervised CSP and unsupervised CSP (uCSP), using the label matching algorithm and automatically matching the labels with the classes. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

4.2.9 K-means on a Riemannian manifold

The unsupervised classifier K-means is based on geometric concepts like distance and mean, which is the metric that provides the minimum squared distance to a given set of points. We do have equivalent metrics for the natural space of the SPD matrices[6], which is a Riemannian manifold. As a result, we can use these to train a K-means classifier without the need of obtaining the spatial filters using an unsupervised method. Nevertheless, this classifier can be used on the covariance matrices of the signals after filtering with the spatial filters as well. The other change that has to be done in order to make this classifier feasable is to redefine the algorithm used to match the clusters with their corresponding labels. This is because this method was based on the Euclidean distance from the representative of each class to the nearest labeled samples. If we modify this metric to be the geodesic distance in a Riemannian manifold instead, it is now possible to complete this process. The number of labeled samples used is still three from each class.

Table 4.11 The average performance across sessions and mean standard deviation is written for the K-means algorithm in a Riemannian manifold with and without the use of unsupervised CSP, as well as the average for each session.

Classifier	Accuracy	StDev	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18
K-means RM	**62,71**	3,89	66,83	62,55	**51,78**	49,39	77,74	84,74	**46,72**	**52,17**	48,72	**51,92**	**52,07**	47,03	**55,49**	**54,15**	**93,26**	**90,16**	**89,02**	**54,98**
K-means RM uCSP	62,38	4,34	**74,43**	**71,51**	49,96	**50,93**	**90,40**	**88,24**	46,68	50,92	**49,70**	49,07	49,64	**49,15**	49,29	50,57	90,41	87,84	84,35	39,76

The classifier used in conjunction with unsupervised CSP is exactly as it was described earlier, and the label matching algorithm still works consistently.

On the other hand, some corrections had to be done in order to improve the version that used the covariance matrices of the EEG signals directly as it proved to be unstable under some circunstances. During some tests, after a few iterations, one of the clusters would take responsibility for explaining most of the observations if not all of them. This behaviour is undesirable, and in order to correct it, when it is detected the algorithm is reset to a random starting state until convergence to a reasonable state or a maximum number of iterations is reached.

[6] The covariance matrices of the EEG signals are of this form

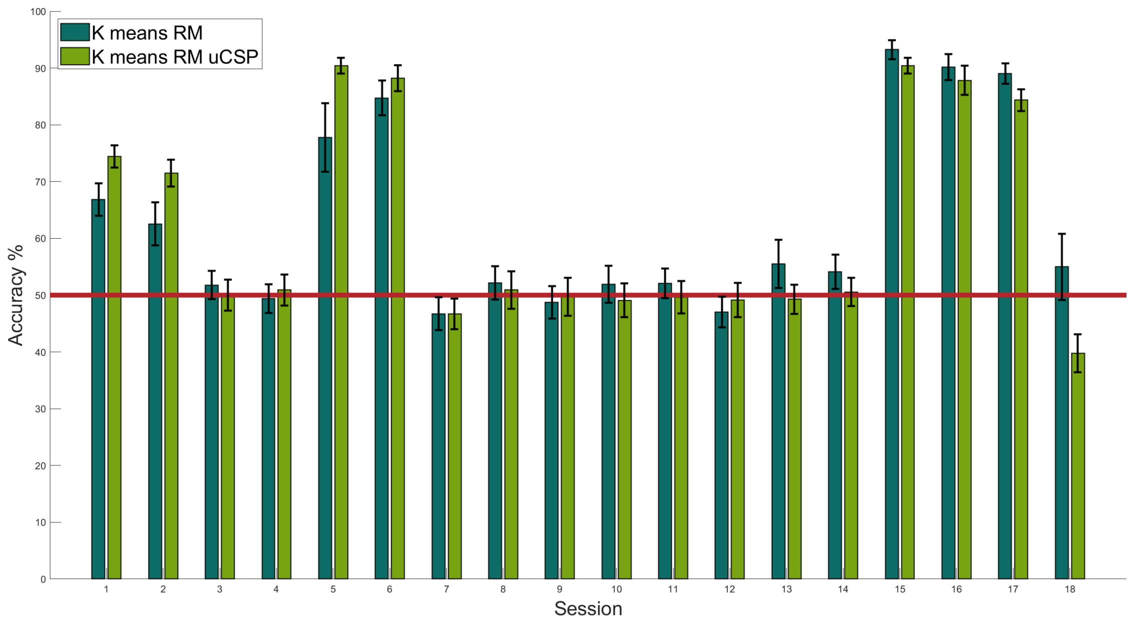

Figure 4.11 The average performance in every session is shown for the K-means algorithm in a Riemannian manifold with both the EEG covariance matrices of the signals before and after filtering with uCSP. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

Both the classifier and the label matching algorithm were more prone to errors during some sessions than for other classifiers as it can be seen by the larger mean standard deviation in many sessions, yet the number of labeled samples was kept at three from each class to facilitate the comparison with other unsupervised algorithms.

There are some possible explanations for the consistent failures of the algorithm. The first one is a well known issue, the high dimensionality of the Riemannian space of large SPD matrices make the classification process more complicated. The second is that the shape of the clusters is not optimal for the use of the K-means algorithm, as this works best with circular[7] clusters of approximately the same size.

Lastly, a numerical error might have occured, as for sessions 1, 2 and 4, in which the inestability of the algorithm was the greatest, the determinants of the covariance matrices used as features are of the order of 10^{-2} to 10^{-5}. On the other hand, for sessions 15, 16, 17 and 18, which were less problematic, the order of the determinants ranged from 10^3 to 10^5. This algorithm is deppendant on the calculation of a large number of distances. In order to calculate each distance, the inverse of a 22 by 22 matrix has to be computed[8], which may induce cumulative errors resulting in an unstable system. This possible error or the influence it may have on the algorithm has not been measured.

4.3 Comparing classifiers

Now that we have established the performance of each classifier and some of their possible variants, we may compare the different algorithms and study the main differences between them.

4.3.1 Supervised classifiers

In order to keep the comparisons simple, we will divide the supervised classifiers into three groups. The first one is the set of them that are based on probability theory. That includes the Bayesian and LDA classifiers. The second group is formed by the algorithms that rely on geometrical properties in an Euclidean space. This is the nearest archetype, N nearest neighbours and gradient descent classifiers. Finally, we will revise

[7] The concept of circle is refered to a two dimensional Euclidean space. In higher dimensions or a Riemannian manifold this would be the equivalent to a circle in that space. This could be understood as a set of points that are equidistant to another point known as the center of the circle.

[8] The MATLAB operator $A\backslash B$ was used instead of $inv(A) * B$ as this is considered to be more accurate.

the previously analysed classifiers based on Riemannian geometry. Once this study is complete, we may compare the best performance from each group.

Lastly, we may observe the difference between the supervised and unsupervised algorithms in order to analyse how the lack of labels may hinder the performance of the classifiers.

We start by looking at the first group, the Bayesian classifier and linear discriminant analysis algorithms. Both of them will be tested in their basic versions, as none of the modifications to these resulted in a noticeable increase in accuracy.

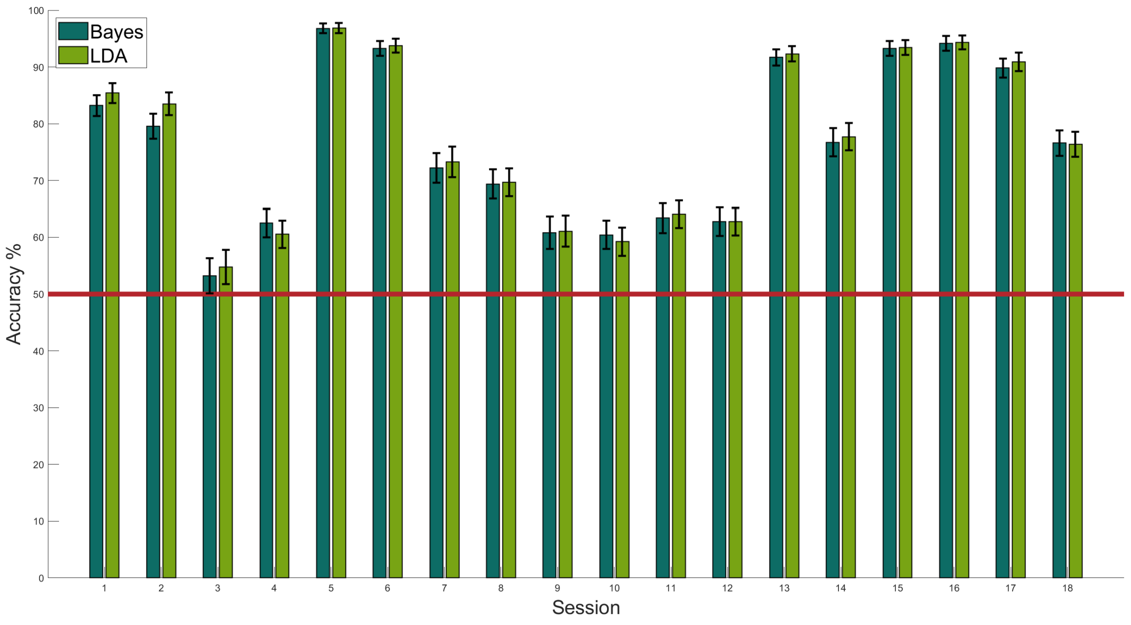

Figure 4.12 The average performance in every session is shown for both classifiers, Bayessian and LDA. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

Table 4.12 The average performance across sessions and mean standard deviation is written for the two different classifiers, as well as the average for each session.

Classifier	Accuracy	StDev	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18
Bayes	76,66	3,34	83,20	79,56	53,21	**62,52**	96,79	93,26	72,22	69,41	**60,81**	**60,41**	63,41	**62,76**	91,69	76,75	93,29	94,19	89,83	**76,60**
LDA	**77,23**	3,41	**85,41**	**83,52**	**54,74**	60,52	**96,86**	**93,80**	**73,29**	**69,69**	61,08	59,23	**64,09**	**62,76**	**92,30**	**77,72**	**93,44**	**94,33**	**90,91**	76,39

In general, for most sessions, the LDA classifier performed slightly better. The reason behind this might be that the boundary between regions in the LDA classifier is linear and as it is a simpler estimate, it makes a better prediction of the model, less prone to overfitting.

The main difference between both algorithms is the estimation of the sample covariance matrices, as if we used a shared covariance matrix between classes for the Bayesian classifier as we do for the LDA algorithm, we would have the same decision regions, even if the technique used to assign observations to categories is different. This confirms that using all the samples to compute a single, more accurate sample covariance matrix estimate shared between classes is not only feasable but actually yields a better result.

As it has been hinted at before, it would be possible to use the Bayesian classifier to obtain the same accuracy using the single covariance matrix estimate for both classes. Nonetheless, this would not be recomendable due to the increased computational cost, at least for the binary classifier.

The next group of classifiers is composed by those based on Euclidean geometry, the N nearest neighbours, nearest archetype and gradient descent algorithms. For the first one, the value of 15 neighbours was previously found to be reliable in our particular situation.

The nearest archetype classifier had two very similar versions, one that used the mean and other that employed the median of the labeled samples. We will use the first one for this comparison.

Lastly, there were two versions of the gradient descent algorithm. We will use the adversarial version, that trained two different classifiers in opposite directions and compared the greatest results. They will be trained for five full cycles as this provided slightly better accuracy.

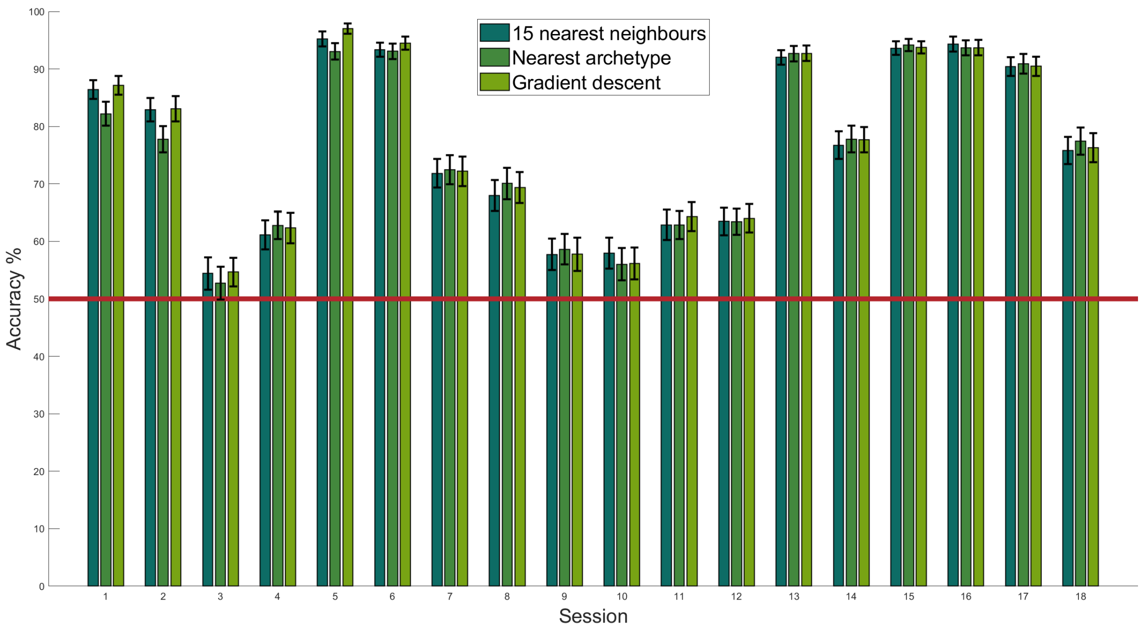

Figure 4.13 The average performance in every session is shown for the three different classifiers. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

Table 4.13 The average performance across sessions and mean standard deviation is written for the three different classifiers, as well as the average for each session.

Classifier	**Accuracy**	**StDev**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**	**S17**	**S18**
15 nearest neighbours	76,57	3,47	86,41	82,92	54,42	61,13	95,22	93,33	71,86	67,98	57,74	**57,96**	62,88	63,48	92,01	76,74	93,61	**94,33**	90,40	75,83
Nearest archetype	76,22	3,41	82,20	77,79	52,75	**62,80**	93,05	93,08	**72,47**	**70,08**	**58,63**	56,03	62,84	63,44	92,69	**77,81**	**94,18**	93,65	**90,91**	**77,46**
Gradient descent	**77,07**	3,51	**87,16**	**83,06**	**54,67**	62,33	**97,04**	**94,51**	72,22	69,37	57,78	56,17	**64,34**	**64,01**	**92,73**	77,70	93,76	93,68	90,48	76,32

Overall, the adversarial gradient descent algorithm performed better. However, in order for this classifier to work properly, two parameters have to be tuned carefully, the target value and learning rate (the first one should be irrelevant, yet it conditions the perfect value of the second parameter). This requirement makes the use of this classifier highly inconvenient and therefore any of the other two classifiers would be preferable.

It is true that the N nearest neighbour algorithm has also one parameter that has to be tuned, yet the performance is not affected drastically by small changes on it, and as we studied in a previous section, any value of N in the range between 15 and 31 would work consistently for our case of study. Still, if we used another value of N outside of this range, the average accuracy would decrease slightly but the classifier would still be functional[9].

All in all, one of the two more robust algorithms is recomended despite the slight decrease in accuracy. The N nearest algorithm performed noticeably better during some sessions. Nevertheless, it is also more computationally expensive, and this cost increases with the number of labeled samples, as for each observation that we wish to categorize, the distance to every labeled sample has to be computed and sorted.

The classifiers that work in the natural space of the SPD matrices have been analised before. Without the use of spatial filters, the most successful algorithm was the nearest archetype with the geodesic distance in a Riemannian manifold as the distance metric. The use of the Riemannian mean provided a better performance during some sessions and therefore this will be the classifier used as reference from this group in future comparisons. Nonetheless, using the Euclidean mean also resulted in a similar accuracy for most trials,

[9] This only applies to a reasonable value of N outside of the cited range. Values up to 89 have been tested. The average accuracy drops significatively but there is no total failure, meaning that the algorithm still performs better than a random classifier.

and could be used in a reliable classifier should an algorithm to approximate the Riemannian mean not be available.

Finally, we may compare one of the classifiers from each group.

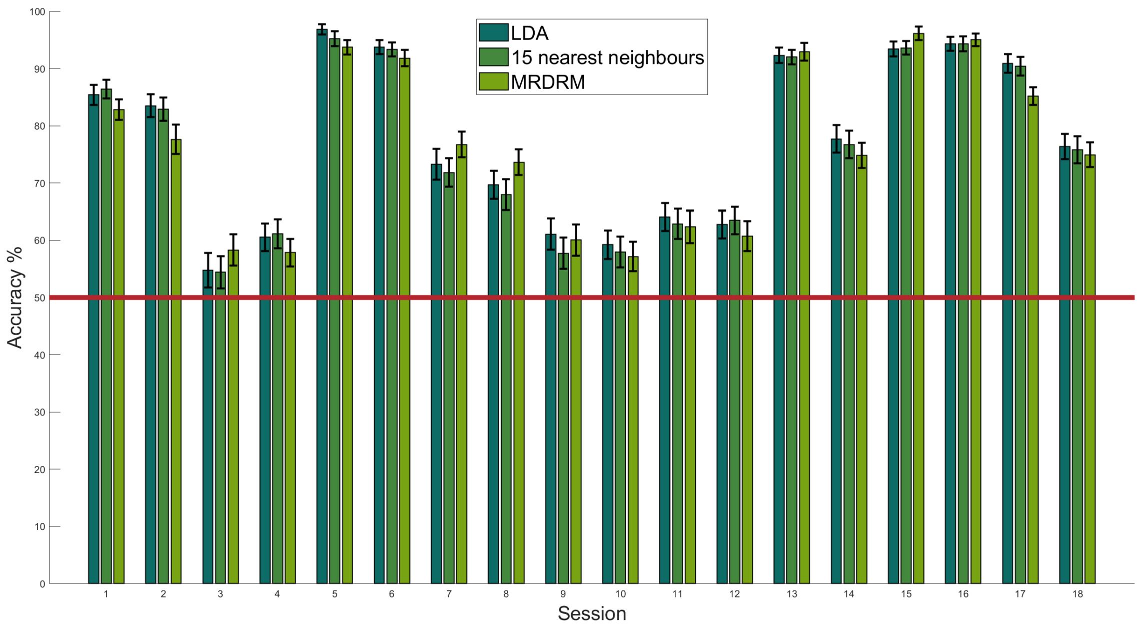

Figure 4.14 The average performance in every session is shown for the three different classifiers. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

Table 4.14 The average performance across sessions and mean standard deviation is written for the three different classifiers, as well as the average for each session.

Classifier	**Accuracy**	**StDev**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**	**S17**	**S18**
LDA	**77,23**	3,41	85,41	**83,52**	54,74	60,52	**96,86**	**93,80**	73,29	69,69	**61,08**	**59,23**	**64,09**	62,76	92,30	**77,72**	93,44	94,33	**90,91**	**76,39**
15 nearest neighbours	76,57	3,47	**86,41**	82,92	54,42	**61,13**	95,22	93,33	71,86	67,98	57,74	57,96	62,88	**63,48**	92,01	76,74	**93,61**	94,33	90,40	75,83
MRDRM	76,22	3,35	82,81	77,65	**58,31**	57,85	93,72	91,83	**76,75**	**73,65**	60,05	57,17	62,34	60,74	**92,97**	74,82	96,15	**95,04**	85,20	74,93

Although the best method to use depended on the session, LDA was the most accurate classifier overall. Despite this fact, the differences are not abysmal and all of the algorithms could be used reliably.

As an addition, we may observe how the use of an unsupervised classifier results in a lower accuracy when employing the same features, obtained with supervised CSP.

Table 4.15 The average performance across sessions and mean standard deviation is written for the two different classifiers, as well as the average for each session. No cells are highlighted as the different nature of the algorithms would make this comparison unfair and missleading..

Classifier	**Accuracy**	**StDev**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**	**S17**	**S18**
LDA	77,23	3,41	85,41	83,52	54,74	60,52	96,86	93,80	73,29	69,69	61,08	59,23	64,09	62,76	92,30	77,72	93,44	94,33	90,91	76,39
K-means	72,76	3,60	74,32	70,51	50,93	61,90	89,94	89,51	68,15	67,98	52,52	54,56	58,70	55,84	92,08	78,06	92,30	91,98	91,34	69,03

During most sessions the performance was affected by the lack of labels during training, and this effect varies from almost being negligible during some sessions to resulting in a loss of accuracy over 10%.

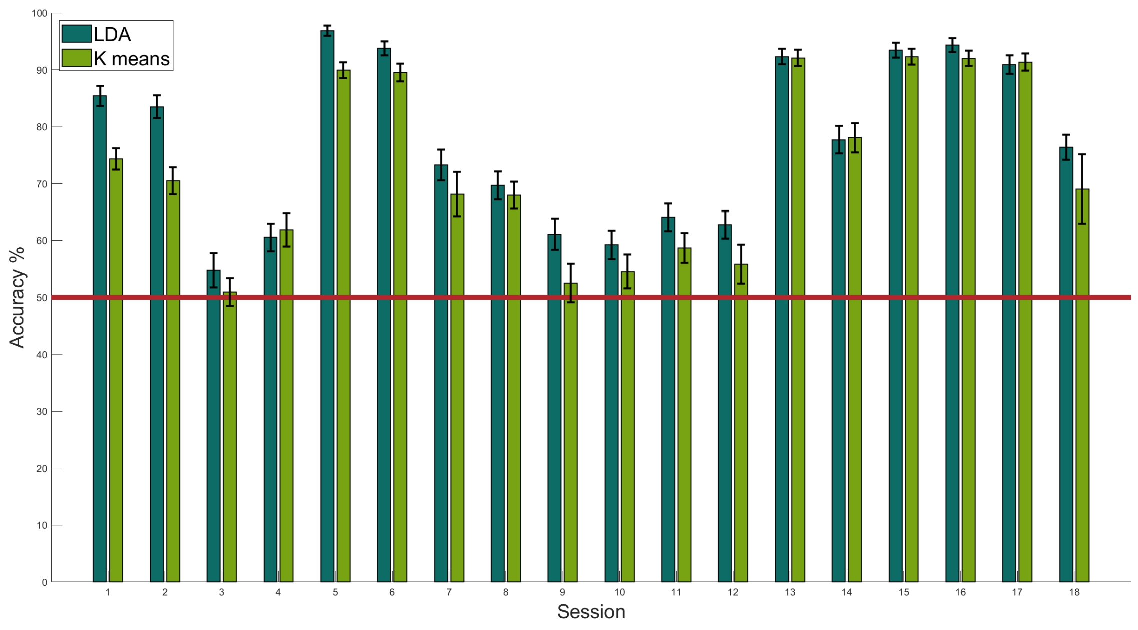

Figure 4.15 The average performance in every session is shown for the two different classifiers. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

4.3.2 Unsupervised classifiers

Now we can compare the performance of the different unsupervised algorithms. As the effects of using unsupervised CSP and the label matching algorithm have already been measured, we may directly compare the classifiers in their fully unsupervised form.

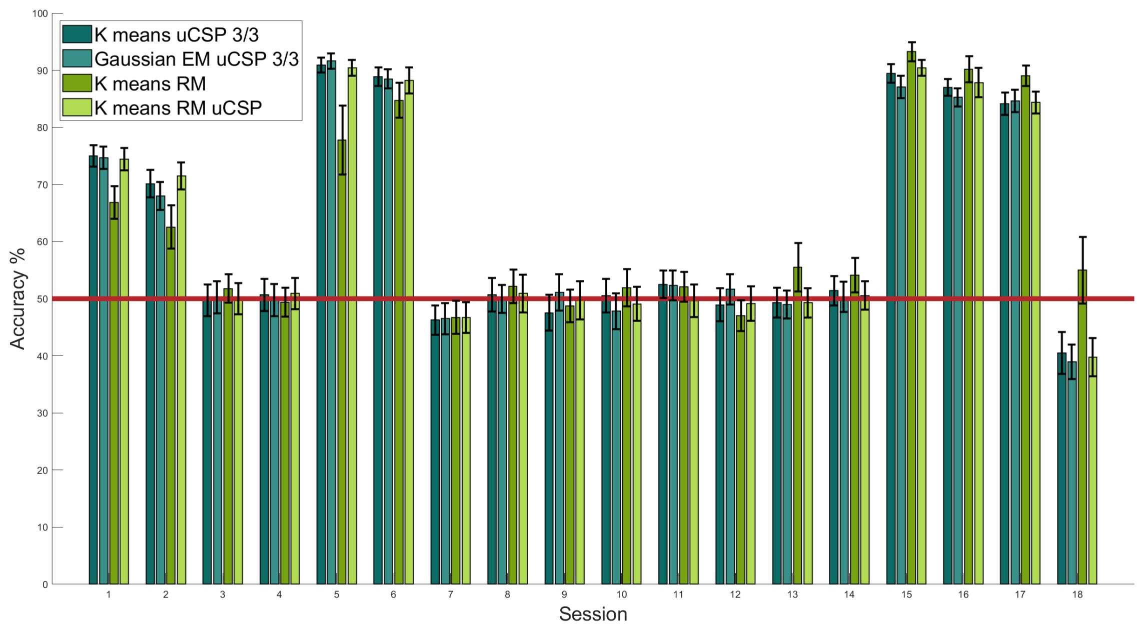

Figure 4.16 The average performance in every session is shown for the four different classifiers. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly.

There are small differences between the K-means algorithm and EM algorithm, as they are very similar

Table 4.16 The average performance across sessions and mean standard deviation is written for the three different classifiers, as well as the average for each session.

Classifier	Accuracy	StDev	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18
K-means uCSP 3/3	62,42	4,30	**75,00**	70,12	49,71	50,67	90,90	**88,87**	46,26	50,68	47,53	50,50	**52,50**	48,93	49,30	51,42	89,44	87,02	84,14	40,47
Gaussian EM uCSP 3/3	62,07	4,24	74,68	67,99	50,22	49,75	**91,62**	88,48	46,50	49,93	**51,10**	47,83	52,32	**51,65**	48,97	50,33	87,05	85,27	84,64	38,94
K-means RM	**62,71**	3,89	66,83	62,55	**51,78**	49,39	77,74	84,74	46,72	52,17	48,72	**51,92**	52,07	47,03	**55,49**	**54,15**	**93,26**	**90,16**	**89,02**	**54,98**
K-means RM uCSP	62,38	4,34	74,43	**71,51**	49,96	**50,93**	90,40	88,24	46,68	50,92	49,70	49,07	49,64	49,15	49,29	50,57	90,41	87,84	84,35	39,76

and one is used in order to initialize the other. Despite the Gaussian EM algorithm being a more sophisticated approach from the statistical point of view, it did not perform better than the Kmeans classifier. Using one of the improved covariance estimators [42] sugested during the definition of the LDA algoritm may result in an increased accuracy.

Using the covariance matrices of the filtered EEG signals as features instead of the variance resulted in a similar average during most sessions.

The major difference with the other classifiers comes when analyzing the version of K-means that uses the covariance matrix of the EEG signals directly. Despite being capable of approximating the clusters in the same sessions in which K-means did, it seemed to be less reliable, failing more often and requiring several resets before converging during some sessions.

4.4 Summary

In this chapter we have tested several classifiers on a Motor Imagery Brain Computer Interface data set in order to distinguish left hand from right hand movement. There were nine users and for each one of them two sessions were recorded.

The tested classifiers were those described in the previous chapter as well as some of their variants. Amongst the unsupervised classifiers, two of them were based on probability theory, three of them in the Euclidean geometry of the feature space and one in the Riemannian geometry of the natural space of the covariance matrices of the EEG signals. The results obtained with all of them were very similar, yet the best average performance across users was obtained with the LDA algorithm.

Two main unsupervised classifiers were tested in conjunction with an algorithm to assign labels to the classes. The first classifier was a clustering algorithm that tried to minimize the total square distance from a group of representatives to a set of training samples. The second classifier refined the parameters that define the shape of the clusters by accounting for the normal distribution of the samples. Even if the first one was more successful in our tests, improved covariance estimators [42] could convert the second classifier in the best option.

An effort was made to translate the K-means algorithm to the natural space of SPD matrices. In conjunction with CSP, despite not being better than normal K-means, this technique does work as intended. Although using the covariance matrices of the signals without filtering seemed promissing at first, and it yielded better results for some sessions, it proved to be an unreliable method prone to failure.

The algorithm that matches labels with the clusters worked propperly with a very low number of samples from each class when the classifier itself was operating correctly during most of the sessions.

Furthermore, the performance of supervised classifiers has been compared against that of unsupervised classifiers. The use of an unsupervised algorithm instead of a supervised algorithm resulted in a different level of degradation of the performance for each session, ranging from over a 10% to being almost negligible. Moreover, the introduction of unsupervised CSP resulted in a small disminution of the results in most cases. For one user, however, the change from supervised to unsupervised CSP had a significant impact.

5 Conclusions

Computers are good at following instructions, but not at reading your mind.

DONALD E. KNUTH

The focus of this work has been to analyze the performance of different classifiers in a MI-BCI task. Both supervised and unsupervised algorithms have been studied, the first group being more accurate as it would be expected.

The analysed data set proposes a great challenge, as the low signal to noise ratio of the EEG signals has to be compensated for with advanced signal processing techniques. We also have attempted to solve the most complicated task available in the data set, left hand versus right hand movement. Furthermore, we had the added difficulty of dividing the data available for each user into two independent sessions, and even in these adversarial conditions the different proposed algorithms have resulted in a partial success.

The classifiers themselves have proven to be sustained on solid theory and performed as expected, however the low separability of the classes and low number of available samples have limited the maximum possible accuracy of the algorithms for each session.

The average result across users in the case of the supervised algorithms was over 70%, which is far from the 50% that would indicate that the classifiers are not discriminating between categories. Nevertheless, this would not be enough for a real system, as if it failed almost one of every three times, it would be frustrating for the user [8]. A similar observation could be made about the unsupervised classifiers, with the added concern that in this case the results are less promising.

However, we should not draw a negative conclusion from this experiment. Even if on average the results are far from ideal, during some sessions corresponding to the same users the results were above the 90% whereas during others they were bellow 60%. This difference between users could be pointing to the fact that the performance of this system depends on the ability of the user to control it, and therefore these skills can be trained.

This concept is intuitive, as with every new interface, the proficiency of the user depends on the previous experiences of the person with similar systems. As an slightly outdated, yet rather simple example, a person that uses a keyboard for the first time might be slow, inaccurate and have trouble finding the right keys which are perhaps located in an unusual place, whereas a mechanographist that has been using a typewriter for years will encounter the new interface simple and intuitive. The first user may take a mechanography course, which implements a well known teaching method that will help the student reach peak performance with the interface in the shortest ammount of time. As this MI-BCI is not similar to other systems we may have encountered in the past, it is reasonable that the adaptation process would be more difficult and that a proven teaching method would significantly improve the usability of the interface. Although to our understanding up to this date there is no definite answer on which is the best approach to learning the correct use of these interfaces or how much performance can be improved by user training, research has been conducted on this subject [21].

Furthermore, CSP was considered as the main dimensionality reduction technique in this work because of it being relatively simple and common in other related MI-BCI investigations, however, newer feature extraction methods that provide better results do exist [3, 8, 37, 11].

Finally, there are some improvements to be made upon the studied classifiers, such as the previously mentioned improved covariance estimators for LDA [42], which could be used in conjunction with Gaussian EM. A large number of other classification algorithms do exist, and some of them might offer slight improvements in accuracy over the ones analysed here. However, due to the small differences obtained when testing distinct classifiers, it seems unlikely that a different classification algorithm would provide a much better outcome using the same set of features and signal preprocessing techniques.

5.1 Future lines of research

Several methods that would improve the performance of the system have been previously discussed, such as better user training, more advanced feature extraction algorithms or more accurate parameter estimations for the current classifiers. Still, other promising approaches remain to be explored.

The first proposed technique was mentioned in chapter 2, and it refers to the use of functional Near Infrarred Spectroscopy (fNIRS) in conjunction with Electroencelography (EEG), as both of these are compatible and share similar characteristics such as price range, portability and setup. By combining the information obtained from both systems, it is possible that a better result could be achieved.

Another possibility would be to develop a robust classifier using information from all the available users instead of training a new classifier from scratch for each session. As each user is different and even multiple sessions from the same user could not be similar due to the placement of the electrodes, a transfer of information would be necessary to translate every experiment into a common space in which every session shows a similar behaviour. This would allow for an homogeneous classification of the trials across users.

5.2 Closing statement

Although many difficulties have to be overcome yet before a complete MI-BCI system is ready for real world applications outside of a test environment, in our experiments at least two of the nine subjects achieved a success rate of well over 90% with both supervised and unsupervised processes, which makes us think that a MI-BCI system based on EEG recordings is plausible and part of a not so distant future.

All in all, this work has provided an overview of the classic workflow used to approach the problem of MI-BCI, as well as some insight into other more advanced or recent techniques that improve the studied system. Despite not all of our experiments being completely successful, we may look forward hopefully on this subject, as the conclusions that we may extract from them provide a bright prospect.

Beyond the current applicability of MI-BCI, the general problem of pattern recognition has been analysed and different classification algorithms have been studied. The usability of these trascend the scope of the proposed task and may be employed in a wide range of fields, providing engineering solutions that improve people's quality of life.

List of Figures

1.1 Computer image of a brain, with some of the principal regions marked in different colours. Image extracted from [28] 2

1.2 Matrix used in a P300 BCI. Instead of letters, commands such as brush shape, transparency, colour or zoom allow the user to paint. The instructions are highlighted by flashing the face of a famous character, Albert Einstein in this case, as this triggers a stronger response. The actual image was ommited in the original article due to copyright reasons. Image extracted from [20] 3

1.3 This figure shows a setup for a hybrid EEG-fNIRS system. Image extracted from [26] 5

1.4 This figure shows the timeline of a single trial. Extracted from [7] 6

1.5 This figure shows the positioning of the EEG and EOG electrodes. Extracted from [7] 7

2.1 By comparing these two representations of classifiers using synthetic data, it is clear that the separability of the classes limits the performance of the classification algorithm, and therefore selecting appropriate features is a crucial step in the process 10

2.2 EEG signals filtered with four different spatial filters are shown. The first two filters minimize the variance of the right hand movements (in green) and the second two minimize the variance of the left hand movements (in red). Image extracted from [5] 15

2.3 This figure represents the estimated gain applied to each source depending on its origin. The first filter maximizes the variance of the right hand movements while minimizing the variance for the left hand movement. The second filter does the opposite. The minimums correspond to ERD, the origin of the Motor Imagery. It can be observed that left hand movements are produced on the right side of the brain and vice versa. Modified image, original extracted from [5] 16

2.4 This is a representation of a LDA classifier in an R^2 space, using two CSP filters. The data is from the first sesion of the first user in the Data Set 2a. The trials corresponding to both classes, left hand movements (LHM) and right hand movements (RHM) are marked in different colors and different shapes are used for those predictions that would be correct or incorrect. This is not significant as the data used to train the classifier is used to plot the graph, which could cause overfitting as it will be discussed later 17

3.1 $p(x|w_o)$ represents the probability of finding an orange that weights x grams, and $p(x|w_l)$ is the equivalent for lemons. $p(x)$ is the probability of finding either an orange or a lemon of weight x. In the case of $p(x)_1$, finding a lemon was twice as likely as finding an orange (this is represented by the prior probabilities, $P(w_l) = 2P(w_o)$, which are also constrained by the condition $P(w_l) + P(w_o) = 1$), whereas the opposite is true for $p(x)_2$ 20

3.2 For any given value of x, the class choice that maximizes the success rate is the category with the highest posterior probability. This allows us to define the optimal decision regions. The sum of the posterior probabilities of all classes for any particular value of x ammounts to one. In this case, the graph was plotted using the prior probabilities $P(w_o) = \frac{2}{3}$ and $P(w_l) = \frac{1}{3}$ and the likelihood functions from 3.1 21

3.3 $P(w_k|x)p(x)$ is equivalent to $p(x|w_k)P(w_k)$, that is the likelihood functions scaled by the prior probabilities. The integral that calculates the error probability is represented by the solid coloured area, and it depends on the likelihood functions, prior probabilities and decision regions. As it is not possible to control the first two (at least in our system), the error probability can only be minimized by optimizing the decision regions. The graph was plotted using the prior probabilities $P(w_o) = \frac{2}{3}$ and $P(w_l) = \frac{1}{3}$ and the likelihood functions from figure 3.1 22

3.4 This figure shows two different sets of normally shaped distributions (scaled by the prior probabilities) and the decision regions associated to the MAP criterion. This is equivalent to the multivariate case where instead of the variances we would compare the covariance matrices. Provided that the covariance matrix is the same for both distributions, the decision regions are connected and the boundaries are linear. This is not necessarily true for the general case [10] 24

3.5 This graph shows the effect that the value of the parameter N has on the decission regions of the N nearest neighbours classifier, shown in different colors. In order to obtain these, 75 random elements were drawn from each of two bivariate normal distributions with different means and variances. A relevant portion of the $\mathbb{R}^2$ space is shown 26

3.6 This figure represents the decission regions of a N nearest neighbours polichotomizer capable of separating four different categories. The plot was obtained like the ones in figure 3.5, however, this time there are four distributions instead of two 27

3.7 The small colored dots represent the labeled trials from each class, whereas the bigger dots are the archetypes obtained for each category. The boundary represented by the line goes through the midpoint of both means and is perpendicular to the segment that joins them. The data set used was extracted randomly from two bivariate normal distributions 28

3.8 This figure shows how after a number of iterations the linear boundary starts to separate the labeled elements from both classes, represented in different colors. The means for each class are also highlighted with a bigger marker in order to facilitate the comparaison with other classifiers, although this value is not used to train this dichotomizer. As the number of labeled elements from each class, extracted from two normal distributions, was limited, the same elements are reused several times in order to train the classifier 30

3.9 After a few iterations, the K-means algorithm finds two possible representatives, separating the training samples into two distinct clusters where all the observations are close together and far away from the other class 33

3.10 The K-means algorithm is used to initialize the model. After that, the classifier finds the most probable likelihood functions and mixing coefficients. The received unlabeled observations are represented by a sampled probability density function and the shown likelihood functions are already pondered by the mixing coefficients. The log likelihood function improves on every iteration until converging to a local maximum 37

4.1 The average performance in every session is shown for both versions of the Bayesian classifier. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 40

4.2 The average performance in every session is shown for the different versions of the LDA classifier. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 41

4.3 The average performance in every session is shown for different values of N. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 42

4.4 The average performance in every session is shown for the different versions of the nearest archetype classifier. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 43

4.5 The average performance in every session is shown for the gradient descent algorithm and for the adversarial version, where two classifiers are tested against the other. The test is carried out for one and five training cycles of each version. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 44

4.6 The average performance in every session is shown for the three different algorithms, both Minimum Riemmanian distance to Riemannian and Euclidean Mean and the 31 nearest neighbours classifier. CSP was not used in these experiments. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 45

4.7 This experiment is similar to that on figure (4.6), but four spatial filters were used. 45

4.8 The average performance in every session is shown for the K-means algorithm, in conjunction with supervised CSP and unsupervised CSP (uCSP), using the label matching algorithm and automatically matching the labels with the classes. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 46

4.9 The average performance in every session is shown for the K-means algorithm, using the nearest mean and 15 nearest neighbours method to assign the observations to the clusters. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 48

4.10 The average performance in every session is shown for the Gaussian EM algorithm, in conjunction with supervised CSP and unsupervised CSP (uCSP), using the label matching algorithm and automatically matching the labels with the classes. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 49

4.11 The average performance in every session is shown for the K-means algorithm in a Riemannian manifold with both the EEG covariance matrices of the signals before and after filtering with uCSP. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 50

4.12 The average performance in every session is shown for both classifiers, Bayessian and LDA. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 51
4.13 The average performance in every session is shown for the three different classifiers. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 52
4.14 The average performance in every session is shown for the three different classifiers. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 53
4.15 The average performance in every session is shown for the two different classifiers. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 54
4.16 The average performance in every session is shown for the four different classifiers. The represented error is three times the mean standard deviation, the region in which it is more likely to find the actual average. The red line represents the 50% mark. This is the expected success rate for a classifier that labeled the test trials randomly 54

List of Tables

4.1 The average performance across sessions and mean standard deviation is written for both versions of the Bayesian classifier, as well as the average for each session 40

4.2 The average performance across sessions and mean standard deviation is written for the different versions of the LDA classifier, as well as the average for each session 41

4.3 The average performance across sessions and mean standard deviation is written for the different values of N, as well as the average for each session 42

4.4 The average performance across sessions and mean standard deviation is written for the two different methods of calculating the archetype, the mean and median of the labeled trials. The average for each session is also registered for each method 43

4.5 The average performance across sessions and mean standard deviation is written for the two different methods and for one and five training cycles, as well as the average for each session 43

4.6 The average performance across sessions and mean standard deviation is written for the three algorithms, as well as the average for each session. CSP was not used in these experiments. 44

4.7 This experiment is similar to that on table (4.6), but fours spatial filters were used 44

4.8 The average performance across sessions and mean standard deviation is written for the K-means algorithm, as well as the average for each session. There are no highlighted cells as these classifiers are not tested on an even ground, and therefore there is no purpose in directly comparing the best performances 46

4.9 The average performance across sessions and mean standard deviation is written for the two different methods of assigning observations to clusters, as well as the average for each session 47

4.10 The average performance across sessions and mean standard deviation is written for the Gaussian EM algorithm, as well as the average for each session. There are no highlighted cells as these classifiers are not tested on an even ground, and therefore there is no purpose in directly comparing the best performances 48

4.11 The average performance across sessions and mean standard deviation is written for the K-means algorithm in a Riemannian manifold with and without the use of unsupervised CSP, as well as the average for each session 49

4.12 The average performance across sessions and mean standard deviation is written for the two different classifiers, as well as the average for each session 51

4.13 The average performance across sessions and mean standard deviation is written for the three different classifiers, as well as the average for each session 52

4.14 The average performance across sessions and mean standard deviation is written for the three different classifiers, as well as the average for each session 53

4.15 The average performance across sessions and mean standard deviation is written for the two different classifiers, as well as the average for each session. No cells are highlighted as the different nature of the algorithms would make this comparison unfair and missleading. 53

4.16 The average performance across sessions and mean standard deviation is written for the three different classifiers, as well as the average for each session 55

Bibliography

[1] E. D. Adrian and B. H. Matthews, *The interpretation of potential waves in the cortex*, The Journal of physiology **81** (1934), no. 4, 440–471.

[2] David Arthur and Sergei Vassilvitskii, *k-means++: The advantages of careful seeding*, Technical Report 2006-13, Stanford InfoLab, June 2006.

[3] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten, *Multiclass brain–computer interface classification by riemannian geometry*, IEEE Transactions on Biomedical Engineering **59** (2012), no. 4, 920–928.

[4] Christopher M. Bishop, *Pattern recognition and machine learning*, 2 ed., Springer Verlag, 2009.

[5] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K. Muller, *Optimizing spatial filters for robust eeg single-trial analysis*, IEEE Signal Processing Magazine **25** (2008), no. 1, 41–56.

[6] Benjamin Blankertz, *Bci competition iv*, 2008.

[7] C. Brunner1, R. Leeb1, G. R. Müller-Putz1, A. Schlög, and G.Pfurtscheller, *Bci competition 2008 – graz data set a*, 2008.

[8] Daniel Rodríguez Cassolá, Sergio Antonio Cruces Álvarez, and Francisco Javier Olías Sánchez, *Medición, procesado y clasificación de señales electroencefalográficas*, Master's thesis, Universidad de Sevilla, 2019.

[9] Sergio Cruces, *Internal report, universidad de sevilla, 2020.*

[10] Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern classification*, 2 ed., John Wiley and Sons, 2001.

[11] G. Feng, L. Hao, and G. Nuo, *Feature extraction algorithm based on csp and wavelet packet for motor imagery eeg signals*, 2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP), 2019, pp. 798–802.

[12] P. T. Fletcher and S. Joshi, *Principal geodesic analysis on symmetric spaces: Statistics of diffusion tensors*, Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis, Springer, Berlin, Heidelberg, 2004, pp. 173–204.

[13] Jan Flusser, Barbara Zitova, and T. Suk, *Moments and moment invariants in pattern recognition*, J. Wiley, 2009.

[14] Pasi Fränti and Sami Sieranoja, *How much can k-means be improved by using better initialization and repeats?*, Pattern Recognition **93** (2019), 95 – 112.

[15] Wolfgang Förstner and Boudewijn Moonen, *Geodesy-the challenge of the 3rd millennium*, ch. A Metric for Covariance Matrices, Springer, 2003.

[16] U. Ghafoor, A. Zafar, M. A. Yaqub, and K. Hong, *Enhancement in classification accuracy of motor imagery signals with visual aid: An fnirs-bci study*, 2019 19th International Conference on Control, Automation and Systems (ICCAS), 2019, pp. 1201–1206.

[17] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley, *Eigenvalue and generalized eigenvalue problems: Tutorial*, 2019.

[18] M. Grosse-Wentrup and M. Buss, *Multiclass common spatial patterns and information theoretic feature extraction*, IEEE Transactions on Biomedical Engineering **55** (2008), no. 8, 1991–2000.

[19] Edwin Hancock, *Pattern recognition*, Aims and scope.

[20] E.M Holz, L. Botrel, and T. Kaufmann, *Long-term independent brain-computer interface home use improves quality of life of a patient in the locked-in state: a case study.*, Archives of physical medicine and rehabilitation **96** (2015), no. 3, S16 – 26.

[21] Camille Jeunet, *Understanding & Improving Mental-Imagery Based Brain-Computer Interface (Mi-Bci) User-Training : towards A New Generation Of Reliable, Efficient & Accessible Brain- Computer Interfaces*, Theses, Université de Bordeaux, December 2016.

[22] George H. Klem, H. Lüders, Herbert H. Jasper, and Christian Elger, *The ten-twenty electrode system of the international federation. the international federation of clinical neurophysiology.*, Electroencephalography and clinical neurophysiology. Supplement **52** (1999), 3–6.

[23] Marek Kuczma and Attila Gilányi, *An introduction to the theory of functional equations and inequalities cauchy's equation and jensen's inequality*, 2 ed., Birkhäuser Basel, 2009.

[24] Magdalena Kuzma-Kozakiewicz, Peter M. Andersen, Katarzyna Ciecwierska, Cynthia Vázquez, Olga Helczyk, Markus Loose, Ingo Uttner, Albert C. Ludolph, and Dorothée Lulé, *An observational study on quality of life and preferences to sustain life in locked-in state*, Neurology **93** (2019), no. 10, e938–e945.

[25] Andrea Kübler and Donatella Mattia, *Chapter 14 - brain–computer interface based solutions for end-users with severe communication disorders*, The Neurology of Conciousness (Second Edition) (Steven Laureys, Olivia Gosseries, and Giulio Tononi, eds.), Academic Press, San Diego, second edition ed., 2016, pp. 217 – 240.

[26] Rihui Li, Thomas Potter, Weitian Huang, and Yingchun Zhang, *Enhancing performance of a hybrid eeg-fnirs system using channel selection and early temporal features*, Frontiers in Human Neuroscience **11** (2017), 462.

[27] D. Liu, C. Liu, J. Chen, D. Zhang, and B. Hong, *Doubling the speed of n200 speller via dual-directional visual motion encoding*, IEEE Transactions on Biomedical Engineering (2020), 1–1.

[28] Jinbiao Liu, Yixuan Sheng, and Honghai Liu, *Corticomuscular coherence and its applications: A review*, Frontiers in Human Neuroscience **13** (2019), 100.

[29] J N Mak, Y Arbel, J W Minett, L M McCane, B Yuksel, D Ryan, D Thompson, L Bianchi, and D Erdogmus, *Optimizing the p300-based brain–computer interface: current status, limitations and future directions*, Journal of Neural Engineering **8** (2011), no. 2, 025003.

[30] R. Martín-Clemente, J. Olias, S. Cruces, and V. Zarzoso, *Unsupervised common spatial patterns*, IEEE Transactions on Neural Systems and Rehabilitation Engineering **27** (2019), no. 10, 2135–2144.

[31] C. S. Mestais, G. Charvet, F. Sauter-Starace, M. Foerster, D. Ratel, and A. L. Benabid, *Wimagine: Wireless 64-channel ecog recording implant for long term clinical applications*, IEEE Transactions on Neural Systems and Rehabilitation Engineering **23** (2015), no. 1, 10–21.

[32] E. S. Nurse, S. E. John, D. R. Freestone, T. J. Oxley, H. Ung, S. F. Berkovic, T. J. O'Brien, M. J. Cook, and D. B. Grayden, *Consistency of long-term subdural electrocorticography in humans*, IEEE Transactions on Biomedical Engineering **65** (2018), no. 2, 344–352.

[33] J. Olias, R. Martín-Clemente, M. A. Sarmiento-Vega, and S. Cruces, *Eeg signal processing in mi-bci applications with improved covariance matrix estimators*, IEEE Transactions on Neural Systems and Rehabilitation Engineering **27** (2019), no. 5, 895–904.

[34] Andreas Pinegger, Hannah Hiebel, Selina C. Wriessnegger, and Gernot R. Müller-Putz, *Composing only by thought: Novel application of the p300 brain-computer interface*, PloS one **12** (2017), no. 9, e0181584 (eng).

[35] L. Puggini and S. McLoone, *Forward selection component analysis: Algorithms and applications*, IEEE Transactions on Pattern Analysis and Machine Intelligence **39** (2017), no. 12, 2395–2408.

[36] Z. Qin, X. Chen, H. Jin, and J. Li, *Voltage and current bi-mode stimulation circuit of low-power deep brain stimulator*, 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2018, pp. 1–4.

[37] P. K. Saha, M. A. Rahman, and M. N. Mollah, *Frequency domain approach in csp based feature extraction for eeg signal classification*, 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2019, pp. 1–6.

[38] A. Schlögl, C. Keinrath, D. Zimmermann, R. Scherer, R. Leeb, and G. Pfurtscheller, *A fully automated correction method of eog artifacts in eeg recordings*, Clinical Neurophysiology **118** (2007), no. 1, 98 – 104.

[39] Ramalingam Shanmugam and Rajan Chattamvelli, *Statistics for scientists and engineers*, John Wiley & Sons, Incorporated, 2015.

[40] P. Stawicki, A. Rezeika, A. Saboor, and I. Volosyak, *Investigating flicker-free steady-state motion stimuli for vep-based bcis*, 2019 E-Health and Bioengineering Conference (EHB), 2019, pp. 1–4.

[41] Francisco Javier Olías Sánchez, *Estudio del método common spatial patterns y sus variantes en interfaces cerebro-ordenador*, Master's thesis, Universidad de Sevilla, 2016.

[42] ———, *Eeg signal processing in motor imagery brain computer interfaces with improved covariance estimators*, Ph.D. thesis, Universidad de Sevilla, 2020.

[43] L.E.H. van Dokkum, T. Ward, and I. Laffont, *Brain computer interfaces for neurorehabilitation – its current status as a rehabilitation strategy post-stroke*, Annals of Physical and Rehabilitation Medicine **58** (2015), no. 3, 3 – 8, Brain Computer Interfaces (BCIs) / Coordinated by Jacques Luauté and Isabelle Laffont.

[44] P. Verma, A. Heilinger, P. Reitner, J. Grünwald, C. Guger, and D. Franklin, *Performance investigation of brain-computer interfaces that combine eeg and fnirs for motor imagery tasks*, 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019, pp. 259–263.

[45] S. Yan, Q. Yu, and H. Wang, *Meg classification based on band power and statistical characteristics*, 2015 12th Web Information System and Application Conference (WISA), 2015, pp. 255–258.

Glossary

BCI Brain Computer Interface. III, V, 1–5, 7, 15, 16, 57, 58

CSP Common Spatial Patterns. 12–15, 17, 23, 31, 39, 40, 44–50, 53–55, 57

ECoG Electrocorticogram. 4, 5

EEG Electroencefalogram. III, V, 5–7, 11–17, 44, 49, 50, 55, 58

EOG Electrooculogram. 6, 7

ERD Event Related Desynchronization. 4, 12, 13, 15, 16

ERS Event Related Synchronization. 4, 12, 13, 15

fMRI functional Magnetic Resonance Imaging. 4

fNIRS functional Near-Infrared Spectroscopy. 5, 58

LDA Linear Discriminant Analysis. III, V, 17, 23, 25, 27, 28, 36, 58

MI Motor Imagery. III, V, 4, 5, 15, 16, 57, 58

PCA Principal Component Analysis. 13

SMR Sensorymotor Rhythms. 4, 12, 15

SSVEP Steady State Visual Evoked Potential. 3