UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Paulo Augusto Alves Luz Viana

Blind Source Separation and Classification Methods for Motor Imagery-Based Brain Computer Interfaces

Métodos de Separação Cega de Fontes e Classificação para Interfaces Cérebro-Computador Baseadas em Imagética Motora

Campinas

2024

Paulo Augusto Alves Luz Viana

Blind Source Separation and Classification Methods for Motor Imagery-Based Brain Computer Interfaces

Métodos de Separação Cega de Fontes e Classificação para Interfaces Cérebro-Computador Baseadas em Imagética Motora

Dissertation presented to the School of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master of Electrical Engineering, in the area of Computer Engineering.

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na área de Engenharia de Computação.

Supervisor: Prof. Dr. Romis Ribeiro de Faissol Attux

Co-supervisor: Prof. Dr. Sarah Negreiros de Carvalho Leite

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Paulo Augusto Alves Luz Viana, e orientada pelo Prof. Dr. Romis Ribeiro de Faissol Attux

____

Campinas

2024

Ficha catalográfica
Universidade Estadual de Campinas (UNICAMP)
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

V654b

Viana, Paulo Augusto Alves Luz, 1997-
Blind source separation and classification methods for motor imagery-based brain computer interfaces / Paulo Augusto Alves Luz Viana. – Campinas, SP : [s.n.], 2024.

Orientador: Romis Ribeiro de Faissol Attux.
Coorientador: Sarah Negreiros de Carvalho Leite.
Dissertação (mestrado) – Universidade Estadual de Campinas (UNICAMP), Faculdade de Engenharia Elétrica e de Computação.

1. Interfaces cérebro-computador. 2. Eletroencefalografia. 3. Análise de componentes independentes. 4. Reconhecimento de padrões. I. Attux, Romis Ribeiro de Faissol, 1978-. II. Leite, Sarah Negreiros de Carvalho, 1985-. III. Universidade Estadual de Campinas (UNICAMP). Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Informações Complementares

Título em outro idioma: Métodos de separação cega de fontes e classificação para interfaces cérebro-computador baseadas em imagética motora
Palavras-chave em inglês:
Brain-computer interfaces
Electroencephalography
Independent component analysis
Pattern recognition
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Romis Ribeiro de Faissol Attux [Orientador]
Aline de Oliveira Neves Panazio
Rafael Ferrari
Data de defesa: 31-07-2024
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-4237-2131
- Currículo Lattes do autor: https://lattes.cnpq.br/1171288653699831

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Paulo Augusto Alves Luz Viana RA: 263889

Data da Defesa: 31 de julho de 2024

Título da Tese: Blind source separation and classification methods for motor imagery-based brain computer interfaces

Aos meus pais, Paulo e Alice, pelo amor incondicional e apoio em todas as situações.

Ao meu irmão Gabriel, pelo companheirismo e preocupação.

Aos meus orientadores, Romis e Sarah, por terem me guiado durante todo este processo.

Aos meus amigos e colegas de trabalho, por me acompanharem nesta jornada.

Resumo

Uma Interface Cérebro-Computador (BCI) é um sistema que fornece um caminho alternativo para que a informação do cérebro de um indivíduo possa ser transmitida ao mundo externo. Compreende, entre outros módulos, uma etapa de processamento de sinal que é composta pelo pré-processamento, extração de características, e classificação. Existem muitos algoritmos possíveis que podem ser usados para implementar a BCI. Neste trabalho, avaliamos o desempenho de uma BCI baseada em imagética motora realizando um estudo combinado de oito técnicas de Análise de Componentes Independentes (ICA): *Joint Approximate Diagonalization of Eigenmatrices* (JADE), *Second-Order Blind Identification* (SOBI), Picard, Picard-O, Infomax, *Extended* Infomax, FastICA e *Online Recursive Independent Component Analysis* (ORICA) e sete algoritmos de classificação: Máquina de Vetores de Suporte (SVM), Análise do Discriminante Linear (LDA), *Naïve* bayes, Regressão logística, Perceptron multi-camadas (MLP), Floresta aleatória e EEGNet, um modelo de aprendizagem profunda. Também avaliamos a robustez da BCI e o impacto dos algoritmos de ICA entre sessões, empregando bases de dados coletados em dias diferentes para os mesmos voluntários. Nossos resultados mostram que pode haver diferenças significativas entre o desempenho relativo dos métodos entre indivíduos e entre conjuntos de dados. A BCI usando FastICA e *Extended* Infomax apresentou maior *kappa* dependendo do conjunto de dados, enquanto o classificador de regressão logística se destacou, apresentando as melhores métricas. Com a EEGNet, a ICA ajuda a reduzir o número de épocas de treinamento necessárias, mas com redução do desempenho do BCI. Por fim, vemos que no protocolo de calibração da BCI entre sessões impacta no desempenho, mas a técnica ORICA apresenta um desempenho consistente em ambos os casos.

Palavras-chaves: BCI; EEG; ICA; Reconhecimento de padrões.

Abstract

A Brain-Computer Interface (BCI) is a system that provides an alternative pathway to the information of an individual brain to the external world. It comprises, among other modules, a signal processing stage which is mainly composed of a preprocessing step, a feature extraction step, and a classification step. There are many possible algorithms that can be used in each step, and their choice can greatly impact the BCI. In this work we evaluate the combined use of eight Independent Component Analysis (ICA) algorithms (Joint Approximate Diagonalization of Eigenmatrices (JADE), Second-Order Blind Identification (SOBI), Picard, Picard-O, Infomax, Extended Infomax, FastICA and Online Recursive Independent Component Analysis (ORICA)) and seven classification algorithms (Support Vector Machine (SVM), Linear Discriminant Analysis (LDA), Naïve Bayes, Logistic regression, Multi-layer perceptron (MLP), Random forest and EEGNet, a deep learning model) in a Motor Imagery (MI) BCI. With the goal of also investigating the performance of the ICA methods on different days, we perform another set of experiments to evaluate between-session robustness. Our results show that there can be significant differences between the relative performance of the methods between subjects and between datasets. BCIs using FastICA and Extended Infomax showed the highest classification kappa depending on the dataset, while the Logistic regression classifier stood out with the best metrics. With EEGNet, ICA helps reduce the number of necessary training epochs, but with a reduction to the BCI performance. Finally, we see that in the between-session BCI calibration protocol results could highly vary between the two datasets, but ORICA has consistent performance in both cases.

Keywords: BCI; EEG; ICA; Pattern Recognition.

List of Figures

List of Tables

List of Acronyms

BCI	Brain-Computer Interface
BSS	Blind Source Separation
CNN	Convolutional Neural Network
CNS	Central Nervous System
CSP	Common Spatial Patterns
DL	Deep learning
ECG	Electrocardiography
EEG	Electroencephalography
ERD	Event-Related Desynchronization
ERP	Event-Related Potential
ERS	Event-Related Synchronization
fMRI	Functional Magnetic Resonance Imaging
IC	Independent Component
ICA	Independent Component Analysis
JADE	Joint Approximate Diagonalization of Eigenmatrices
KKT	Karush–Kuhn–Tucker
LDA	Linear Discriminant Analysis
M1	Primary Motor Cortex
MEG	Magnetoencephalography

MI	Motor Imagery
MLP	Multi-layer perceptron
NIRS	Near-Infrared Spectroscopy
NN	Neural Network
ORICA	Online Recursive Independent Component Analysis
PET	Positron Emission Tomography
PM	Premotor Cortex
PSD	Power Spectral Density
QDA	Quadratic Discriminant Analysis
RF	Random Forest
RNN	Recurrent Neural Network
S1	Primary Somatosensory Cortex
SMA	Supplementary Motor Area
SNR	signal-noise ratio
SOBI	Second-Order Blind Identification
SSVEP	Steady-State Visually Evoked Potentials
SVM	Support Vector Machine

Contents

1 Introduction

Brain-Computer Interface (BCI) research has been of increasing interest during the last two decades, going from 95 publications from 2000 to 2002 to 2224 publications from 2018 to 2020 (SAHA *et al.*, 2021). A 2016 study used the *2013 US Paralysis Prevalence & Health Disparities Survey* to estimate that 5.3 million people in the United States lived with paralysis, at the time (ARMOUR *et al.*, 2016). The National Health Survey in Brazil estimated that in 2013 there was more than 2.2 million stroke victims, resulting in 580 thousand people with severe disabilities (BENSENOR *et al.*, 2015). In 2021, the World Health Organization estimated that there was almost 15.4 million people living with spinal cord injury worldwide (World Health Organization, 2024). BCIs applications are commonly focused on amyotrophic lateral sclerosis (ALS) or severe Central Nervous System (CNS) injuries. ALS patients gradually lose motor skills and speech abilities, so they are the ones who can most benefit from this area (CHAUDHARY *et al.*, 2016). The application of such technologies could mean new possibilities for people with impaired mobility or motor control, effectively increasing quality of life. Non-medical applications of BCI also include attention monitoring, mental state monitoring, cognition, mental workload, games. The development of those technologies for both healthy and paralyzed users can help with the social integration of the later group (BLANKERTZ *et al.*, 2010).

A BCI is defined as a system that provides communication between the brain and the external world through non-conventional means, i.e., not using the pathways of muscles and nerves (WOLPAW *et al.*, 2002). However, it still needs these pathways to generate or carry this information, so it can be detected or measured. BCIs have five principal use cases (WOLPAW *et al.*, 2020):

- *Replace* muscle control: A person with a high-level spinal cord injury may be able to control a locomotion device, such as a motorized wheelchair.
- *Restore* muscle control: For instance, it can help a person with spinal cord injury that has been paralyzed to grasp objects, by stimulating muscles using electrodes.
- *Enhance* CNS output: Can be used for monitoring the attention of a person in a attention required task, and alarm them when necessary.

- *Supplement* the CNS output: A person that has healthy control of their both arms can use it to control an additional robotic arm.
- *Improve* the natural CNS output: For a person that suffered stroke or has impaired arm function, the BCI can be used to stimulate the muscles or even control an orthotic device, improving the movement.

It also needs to generate feedback to the user, so they can learn, adapt and improve control over the system through its usage.

Kubler *et al.* (2020) cite that although it exists (BRUNNER *et al.*, 2015), a BCI road-map lacks precise steps for bringing the technology to the final users (end-users at home). As an assistive technology (AT), there are only a small number of applications that can be used by end-users without the presence of researchers or outside AT centers. From the user's perspective, reasons for not using such technologies include discouragement from the environment the user is in, embarrassment or self-consciousness about using the device, discomfort/strain in its use and the perception that the device is obtrusive. User reactions to the same stimuli suffer from high trial-to-trial variability and even moment-to-moment variability in some tasks (BLANKERTZ *et al.*, 2010). Additionally, a non-negligible proportion of the population is unable to control (especially sensory-motor rhythm-based) BCIs, either due to high session-to-session variability or to no significant brain rhythm modulation during the performed tasks (VIDAURRE; BLANKERTZ, 2009).

BCI applications are not yet widely used, also mainly due to their reduced robustness and reliability. For most current BCIs there is still a lot of needed time for calibration (in some cases) and cumbersome setups (LOTTE, 2015). In this sense, sensory-motor rhythm-based BCI offer several advantages over other approaches (for the population that can control BCIs), as the use becomes automatized through extended operation and is less likely to negatively affect the usual mental state of the user (MCFARLAND; WOLPAW, 2011). One example of such BCIs is the so called Motor Imagery (MI) paradigm, in which the user thinks about the execution of a specific movement that is decoded, in the sequel, through the processing of electrical signals of the brain.

In this scenario, we intend to analyze how certain processing techniques, namely Independent Component Analysis (ICA), can be used within a BCI to enhance the performance of such systems. The use of ICA is motivated by some works that have already demonstrated its utility in different stages of the BCI pipeline, and by the neurophysiological interpretation of the technique, as it is theoretically able to uncover brain activity (REJER; GóRSKI, 2017; HAMANEH *et al.*, 2014; WINKLER *et al.*, 2011; KACHENOURA *et al.*, 2008). The measured Electroencephalography (EEG) is a mixture of the activity that is elicited by the MI task and many other sources, such as line noise, background noise, muscle electrical activity, and also other brain-related activities that are not related to the BCI, and the ICA is used to separate those independent signals as better as possible, so we can only process the MI-related activity for better performance.

The choice of the ICA algorithm is one important design choice for the BCI, as many works show that each technique may result in different performances (WU *et al.*, 2020; URIBE, 2018; BRUNNER *et al.*, 2007). Each ICA algorithm method has slightly different prior assumptions, objective functions or solutions to those functions, and naturally one can be more fit to the current processing task than others. Source assumptions such as temporal correlation, gaussianity, the sign of their marginal cumulants, or the characteristics of the additive noise are important information for a correct choice (KACHENOURA *et al.*, 2008). The choice of the classifier can follow the same rationale. The classification algorithm is also an important step of a MI-BCI as it is the one that outputs the decoded imagined movement, and its choice is also important for the BCI pipeline (KHADEMI *et al.*, 2023; ALTAHERI *et al.*, 2021).

The evaluation and comparison of different BCIs using those source extraction and classification methods can yield important insights about how well the assumptions of those methods hold for each protocol, subject and paradigm, and help us understand on how to choose them and maximize the BCI's robustness and correctness. In this work we also evaluate the combined effect of the ICA preprocessing with a deep learning model as the classifier. As this classifier does not directly use frequency features (which were used with the more classical models), we intend to evaluate if using the extracted sources instead of the raw EEG signals actually improve the system performance.

1.1 Objectives

In the context of a MI-based BCI system, we aim to compare the systems performance when using any combination of ICA and classification method from a selected set of techniques, the use of ICA in tandem with a deep learning method, and the robustness of the system to session-to-session calibration.

The general goals of this work are:

- To compare how different ICA methods and classification algorithms impact the performance of a MI BCI system.
- To evaluate how ICA methods as preprocessing for a deep-learning-based classifier affects the convergence speed and accuracy of the MI-based BCI.

The specific goals are:

- To identify, among the tested ICA methods and classifiers, which, on average, result in the highest BCI accuracy for the analyzed dataset.
- For each classifier, to identify the ICA method that maximizes the MI-based BCI accuracy.
- For each ICA, to identify the classifier that maximizes the MI-based BCI accuracy.
- To find the combination of ICA method and classifier that results in the highest MI-based BCI accuracy.
- To analyze which ICA method leads to the highest accuracy with EEGNet.
- To analyze the impact of ICA on the number of iterations required to train EEGNet.

The initial results of this work were presented in Viana *et al.* (2024) at the IX Latin American Congress on Biomedical Engineering and XXVIII Brazilian Congress on Biomedical Engineering (CLAIB & CBEB 2022). The within-session and between-session study was first presented at the 9th BRAINN Congress (VIANA *et al.*, 2023). This work is also in the process of being published as a chapter of the book Advances in Neural Engineering Volume 1.

2 Brain-Computer Interfaces

The beginning of the development of noninvasive BCIs dates back to 1924, when Hans Berger was able to record electrical signals from a human brain (BERGER, 1929; BAMDAD; AUAIS, 2015). In 1969, Kamiya and collaborators showed that healthy users could learn to control alpha waves in their brain if given feedback, such as the rise and fall of a tone (CHAUDHARY *et al.*, 2016; NOWLIS; KAMIYA, 1970), in one of the first BCI experiments with humans. In 1973, the term *BCI* was officially proposed by Jacques Vidal when he presented a system that could translate EEG signals into control signals in a computer (VIDAL, 1977). In this chapter, we present an introduction to BCI paradigms, with emphasis on the MI paradigm, which is the object of study of this work, and a brief presentation of aspects of the CNS physiology that are relevant to it.

2.1 BCI paradigms

The idea of BCI paradigm refers to how a subject interacts with the system and how he/she controls it, and also to the main neurophysiological mechanism being employed (URIBE, 2018). For EEG-based BCIs, for instance, the literature reports the use of sensorymotor rhythms, whose generative and manifestative phenomena will be described later on. The P300 paradigm, on the other hand, is based on an Event-Related Potential (ERP), called P300 wave, which is elicited between 220 to 500 ms after the presentation of a infrequently and awaited event using the "oddball paradigm". For instance, the P300 speller is a very popular application in which a grid of letters is presented to the subject, who must focus on the letter that he/she wants to spell, while each row / column is highlighted (PAN *et al.*, 2022). The detection of the associated wave is used for decoding which letter the subject was expecting to be selected. The Steady-State Visually Evoked Potentials (SSVEP) paradigm relies on the spectral content of the EEG signals from the visual cortex. Neurons in this region tend to fire at the same frequency of flickering lights that are visually presented to the subjects. In the presentation of a set of different flickering lights, the BCI can determine at which one the subject is looking to through the EEG (LIU *et al.*, 2021). A non-exhaustive review of different paradigms is presented at Abiri *et al.* (2019).

The paradigm based on imagination of movement execution in a BCI system is known as Motor Imagery paradigm, in which the BCI discriminates activity related to both the somatosensory and motor cortices. It generally relies on the decoding of movement imagination of left and right hands, feet and tongue movements from measured brain activity (in this case, EEG) (SINGH *et al.*, 2021). During most motor behaviors, we can detect an Event-Related Desynchronization (ERD) (suppression) of the low-*mu* rhythms (8-10 Hz) over the whole somatosensory cortex, while high-*mu* (10-13 Hz) rhythms are topographically located. The *beta*-rhythms (13-25 Hz) also exhibit desynchronization during motor behaviors, succeeded by an Event-Related Synchronization (ERS) (enhancement), which is called the *beta* rebound (WOLPAW; WOLPAW, 2012). Additionally, the *beta* ERS can occur in slightly different bands than the ERD (HALDER *et al.*, 2011; PFURTSCHELLER; NEUPER, 2001).

Wolpaw and Wolpaw (2012) described BCI as "*a system that measures CNS activity and converts it into artificial output that replaces, restores, enhances, supplements, or improves natural CNS output and thereby changes the ongoing interactions between the CNS and its external or internal environment*". The *natural* CNS activity output consists of electrophysiological, neurochemical and metabolic phenomena. In our context, the CNS activity is the measured EEG signals, which is captured and then fed to computational systems with the goal of controlling other devices, i.e., replace and improve the natural output.

Figure 2.1 shows a diagram for a conventional BCI system. The BCI starts and ends at the user. Their CNS activity is acquired, digitized and stored or streamed to a computing system. For instance, in EEG-based BCI, the electrical brain activity is recorded via electrodes placed at the scalp. Commercially available EEG devices also amplify the signals, apply analogical band-pass filters and notch filters (to remove powerline noise), converts them to digital signals and transfers the data to a computational system, such as a microcomputer (SANEI; CHAMBERS, 2007). The signal processing block is responsible for translating raw EEG signals into commands that can be understood by an external device or other computational system. The implementation of this block is highly dependent on the source brain activity being acquired, the BCI paradigm and the command space. It generally includes spatial and temporal filtering procedures, feature or characteristics extraction, and their translation to commands. The target application, device

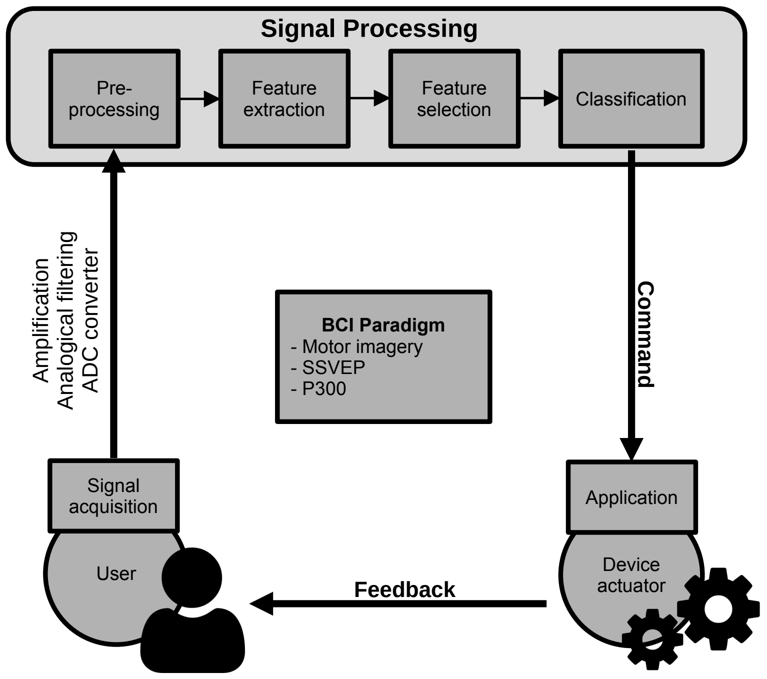


Figure 2.1 – High-level diagram of a BCI system.

or actuator receives those commands, and executes the task as intended by the user. In a BCI, the task may be required to provide a feedback to the user, e.g., an image on a screen indicating which movement they imagined. The final component of the BCI is the operating protocol, which relates to how the user commands the BCI, the commands that the BCI provides to an application, how the BCI converts commands into application actions, and the actions that are produced (SINGH *et al.*, 2021; WOLPAW; WOLPAW, 2012).

The sensitivity of the brain representations to motor imagery tasks makes the paradigm viable for BCI, as it is associated with ERD and ERS of the respective executed movement, and there are many studies that show the similarities between brain activity during motor execution and imagery (WOLPAW; WOLPAW, 2012; MCFARLAND, 2000). As in motor execution, *mu* and *beta* ERDs occur during motor imagery (which is also related to the movement planning phase of the execution), followed by the beta rebound, which has been found to consistently appear in Cz during both foot and hands movement imagination in different regions, but not during tongue motor imagery (PFURTSCHELLER *et al.*, 2005). The activity of those bands/rhythms have been used for the decoding of movement imagination in MI-based BCI (SAIBENE *et al.*, 2023;

URIBE, 2018). Rejer and Górski (2017) used ICA for separating the Independent Components (ICs), followed by extraction of power spectral features, using Lasso regression (TIBSHIRANI, 1996) for selecting features and a linear Support Vector Machine (SVM) as classifier. Gouy-Pailler *et al.* (2010) used a non-stationary filter followed by Common Spatial Patterns (CSP) as signal preprocessing, and calculated spectral power features calibrate a logistic regression classifier. Safitri *et al.* (2020) used time-frequency wavelet analysis, followed by dimensionality reduction with ICA as preprocessing and recurrent neural networks for classification in an MI-BCI.

2.1.1 Feature extraction and selection for motor imagery

ERDs are attenuations in the brain activity during certain tasks, which can be spatially localized depending on the excitatory stimulus. ERDs appear on the sensorimotor cortex (parietal lobe) during motor imagination and execution, hence the possibility of identifying whether a person is imagining the execution of certain tasks using captured EEG activity (PFURTSCHELLER; NEUPER, 2001). This activity is mainly detected in the *mu* (8-13 Hz) and *beta* (13-25 Hz) frequency bands of the EEG signal. To isolate certain frequency bands, we need to perform a frequency analysis of the signals to extract its activity level in order to verify whether an ERD happened. In the context of motor imagery, the most used methods are the Short-Time Fourier Transform and Welch's method (SAIBENE *et al.*, 2023; PRIYATNO *et al.*, 2022; SINGH *et al.*, 2021; AGGARWAL; CHUGH, 2019). We use the latter for the frequency analysis, within a setup based on spectral power estimation by segmenting the signal into many pieces, and using:

$$\hat{S}(\omega) = \frac{1}{KN_HU} \sum_{k=1}^{K} \left| \sum_{n=1}^{N_H} H(n)x(n+kD)e^{-j\omega n} \right| \tag{2.1}$$

to compute an averaged estimate of the Power Spectral Density (PSD). In Equation (2.1), $S(\omega)$ is the power density of frequency ω of a signal $x(n)$. It is the averaged spectral density over small, same-sized windows of the original signal, using K segments of length N_H, each one shifted by D samples from the last segment (WELCH, 1967). We used a Hamming window with $D = 0.25s \times f_s$ and $N_H = 1s \times f_s$, defined by:

$$H(n) = a_0 - (1-a_0)cos\left(\frac{2\pi n}{N_H}\right) \tag{2.2}$$

where f_s is the sampling rate, $a_0 = 0.53836$ and $0 \leq n \leq N_H$ with U defined by:

$$U = \frac{1}{N_H} \sum_{n=1}^{N_H} |H(n)|^2 \tag{2.3}$$

The analysis is typically done to each EEG channel after spatial filtering (LOTTE *et al.*, 2018). In the context of this work, we perform it after source separation for each IC. We calculate PSD over the *mu* and *beta* bands for each IC, yielding 2 times the number of channels values of spectral power that will be used as features for the classifier (feature translator) training and inference. Since not every source is created in the motor cortex, most of the independent components cannot be used for motor classification. This creates the necessity of selecting the features that are specifically discriminative of motor imagery classification and removing features that are not (BEKIRYAZICI *et al.*, 2020), in a process called feature selection. The wrapper method (KOHAVI; JOHN, 1997) greedily builds a feature set that maximizes a determined performance metric score. It uses a hill-climbing algorithm, starting from an empty set and adding features that mostly increase a classifier's score.

The wrapper algorithm employed in this work is outlined as follows:

1. Let $\mathcal{S} = \{\}$ be the selected feature set and $\mathcal{F} = \{a_1, a_2, ..., a_{N_f}\}$ be the complete feature set with N_f features, where each a_i represents a feature (in this case, the PSDs of the extracted ICs).

2. For each feature a_i in $\mathcal{F}$, calculate the score of the classifier via cross-validation using features $\mathcal{S} \cup \{a_i\}$ for the classification task.

3. Select the feature that most increased the score (comparing to using only $\mathcal{S}$) for $\mathcal{S}$ and remove it from $\mathcal{F}$.

4. Repeat steps 2) and 3) until $\mathcal{S}$ reaches a desired cardinality or until there is no improvement to the score.

Using this method, we jointly select the ICs and the respective *mu* and/or *beta* rhythms that are beneficial to the classifier.

The feature translation (or classification) step uses an algorithm that interprets the extracted features and decides which MI task the corresponding EEG signal was captured from. In MI-based BCI, one of the most common algorithms is the Linear Discriminant Analysis (LDA), but there are works that used Support Vector Machines, Random Forests, Hidden Markov Models, Naive Bayes and also adaptive approaches such as adaptive LDA, and online passive-aggressive algorithms (LOTTE *et al.*, 2018). Deep learning methods have also been employed more recently (SANTAMARíA-VáZQUEZ *et al.*, 2020; LAWHERN *et al.*, 2018; SCHIRRMEISTER *et al.*, 2017), with the advantage that they may not require the upstream feature extraction step, learning representations from raw EEG signals. We present those methods in more detail in Chapter 4, including EEGNet, a deep learning method for EEG classification.

ICA is also reported as a relevant tool in the construction of BCIs. ICA methods can be applied to separate information-bearing signals from noise, interference and several kinds of artifacts. Since different ICA methods use different assumptions and signal properties to extract the ICs, they need to be chosen carefully, as well as the classification model. In this work, we investigate the combinations of different methods for both steps, aiming to find how well each ICA method suits with each classifier, and what is the best possible choice for each scenario. We do not manually select the ICs, but use the wrapper method, so that the components are chosen solely based on whether they are capable of increasing BCI accuracy. Additionally, we also evaluate the use of ICA with deep neural networks, training a model to learn the representations from the ICs instead of the raw signals. The different ICA techniques will be presented in Chapter 3, and the classification methods will be presented in Chapter 4.

2.2 Central nervous system

The *central nervous system* CNS comprises the brain and the spinal cord, and its structures are distinguished by their locations, cell types and histology, and their function in processing sensory inputs to motor responses. CNS activity consists of electrical, chemical and metabolic phenomena, and it can be quantified through different methods (WOLPAW; WOLPAW, 2012). The CNS is mainly composed of neurons and glial cells. Neurons are the principal functional components of the brain and are supported by glial cells, which outnumber the neurons by approximately nine to one (THE NATIONAL INSTITUTE OF NEUROLOGICAL DISORDERS AND STROKE, NIH, 2023). Neurons transport information through chemical signals and electrical impulses, allowing sensations, feelings, thoughts, motor commands, limb localization and much more information to be communicated within the brain and from the body to the brain (WOLPAW; WOLPAW, 2012).

The Primary Motor Cortex (M1) resides within the frontal lobe and is related to the activation of muscles. It is organized in such that specific regions are responsible for different muscles, with neighboring body parts typically represented by neighboring areas of the cortex. The responsibilities include control of fingers, wrist, arms, facial movements, tongue movements, chewing, vocalization, and more (GALLEGO *et al.*, 2022; WOLPAW; WOLPAW, 2012). The Premotor Cortex (PM) engages during abstract concepts about the activation of the primary motor cortex, such as direction and target location of the movement. The PM is more active during movement planning, while M1 has higher activity during its execution. The Supplementary Motor Area (SMA) is essential for the realization of complex or combination of movements, such as sequences or multiple movements at the same time (KANDEL *et al.*, 2021). Shima and Tanji (2000) observed that the SMA is activated during a push and pull movement sequence, but is not activated during individual movements. It is important to say that the higher responsibility of each region for each dimension of the movement execution does not mean that the other regions are inactive during them.

The Primary Somatosensory Cortex (S1) conveys sensations of touch, temperature, pains, and limb position, playing a role in the execution of movements. It also conveys the sense of limb movement, which is important for movement planning and guiding, providing feedback to the M1 cortex while the movement is executed. It is located in the parietal lobe. Activity in S1 occurs mostly during the movement execution, in relation to movement planning (KANDEL *et al.*, 2021; WOLPAW; WOLPAW, 2012).

Figure 2.2 shows a depiction of a human brain, with the approximate locations of the mentioned areas.

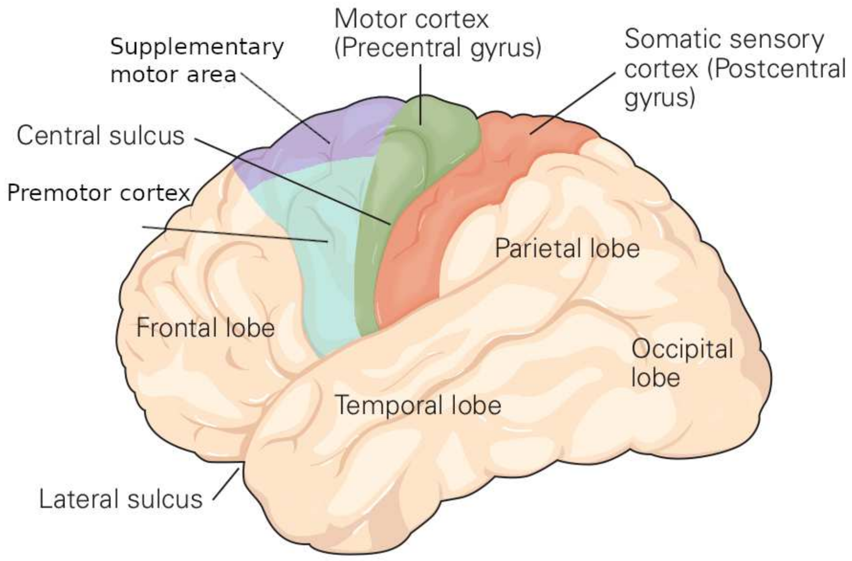


Figure 2.2 – Localization of the motor-related areas and the somatic sensory cortex, adapted from (KANDEL *et al.*, 2021).

Figure 2.3 depicts a sagittal section of the brain. Area A indicates the Premotor cortex, Area B indicates the Motor cortex, and Area C indicates the somatosensory cortex. Those motor-related areas are on the superior (dorsal) part of the brain, which will be important for the goal of recording the electrical activity.

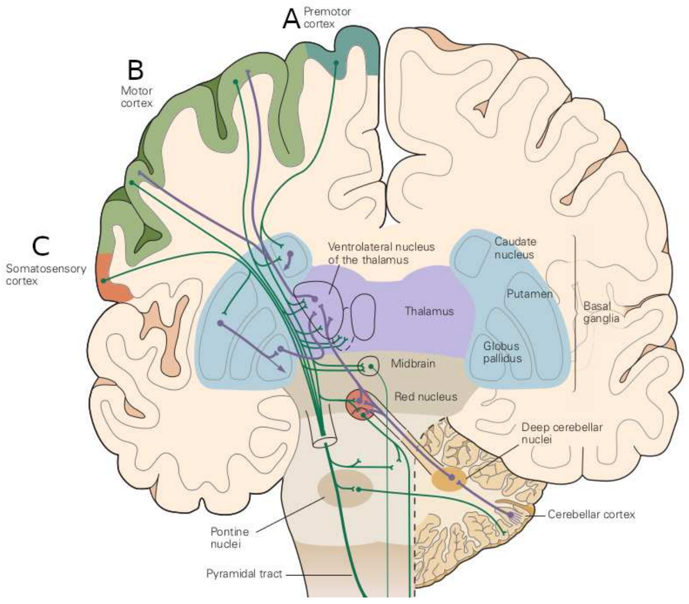


Figure 2.3 – Sagittal section of the human brain, adapted from Kandel *et al.* (2021).

There is some insight on the somatotopic mapping of the motor and somatosensory cortices. Figure 2.4 shows the dedicated areas for each part of the body in both the Motor Cortex (A) and Somatosensory cortex (B). Note that the relative size of the areas dedicated to hand and facial expressions are significantly larger than the others, as the motor control of hand and fingers is very fine and require a higher degree of control, as well as the control and sensory information load of the face (especially the lips) is also relatively rich (KANDEL *et al.*, 2021)

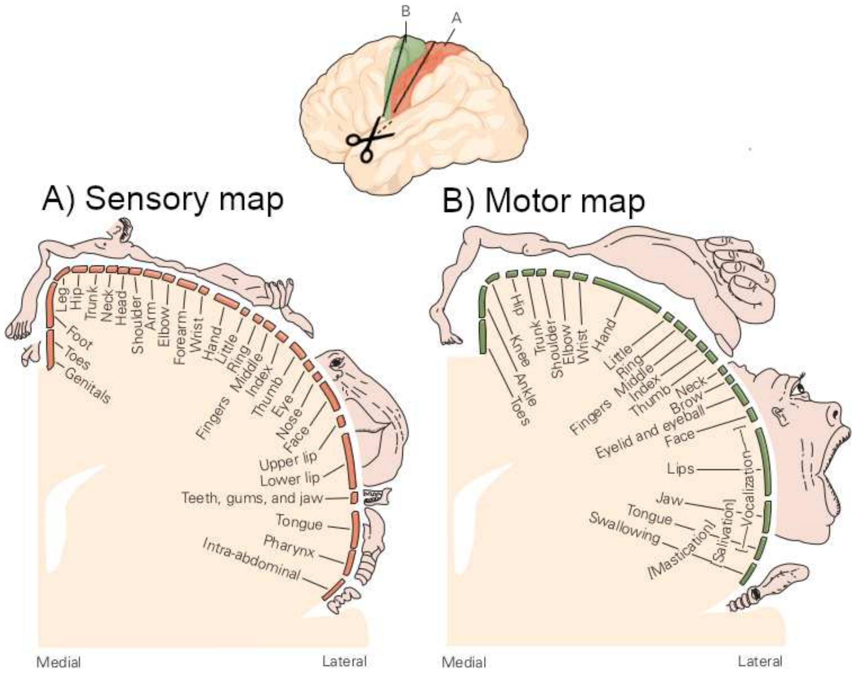


Figure 2.4 – Somatotopic map of the somatosensory cortex (A) and motor cortex (B) in a coronal view, adapted from Kandel *et al.* (2021).

The activity in the CNS is predominantly related to the synaptic current between neuron junctions (called *synapses*). Figure 2.5 shows the component parts of a neuron. The neuron body (5) comprises the cell material and the nucleus (2), which contains the genetic material. The dendrites (1) are terminations that receives impulses from other neurons, and are connected to either dendrites or axons (8) of other cells. Axons (8) are long cylindrical terminations that transmit impulses to other cells. The space between two neurons through axons and dendrites is called, as already mentioned, synapse (10) (SANEI; CHAMBERS, 2007).

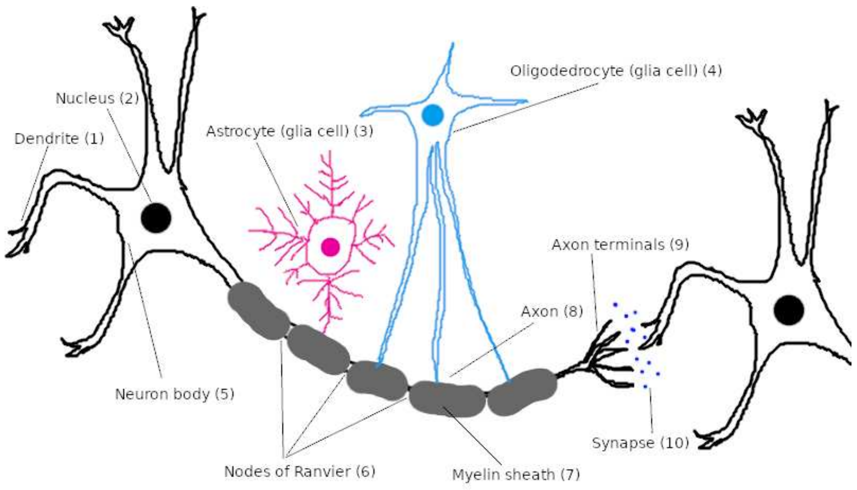
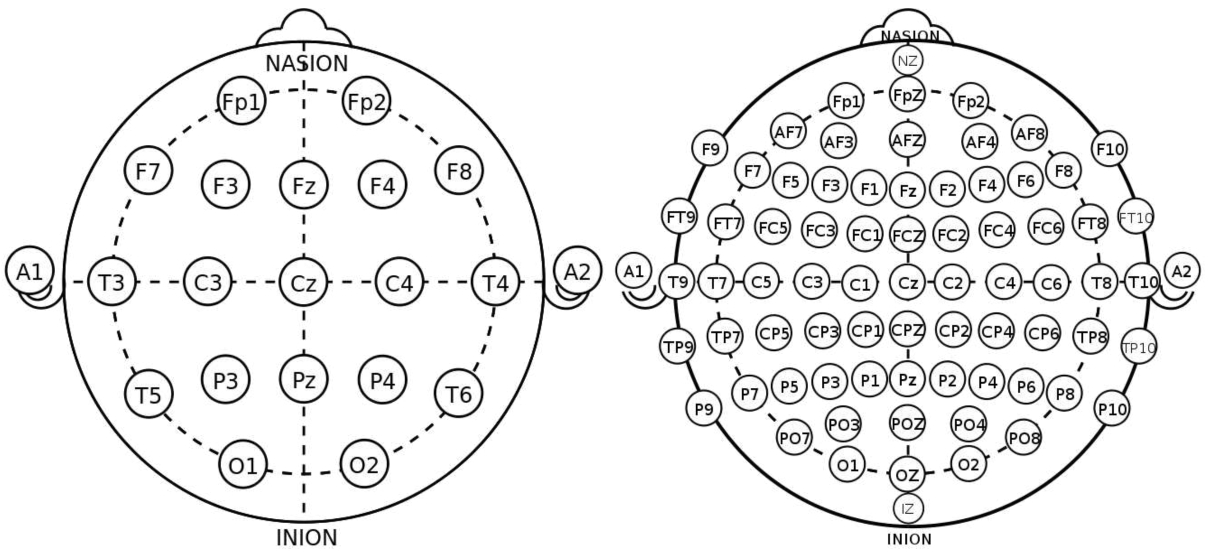


Figure 2.5 – Parts of a neuron, adapted from THE NATIONAL INSTITUTE OF NEUROLOGICAL DISORDERS AND STROKE, NIH (2023).

Initially, the neuron is at a rest state and has an intracellular potential of approximately $-70\ mV$. When the dendrites receive a chemical stimulus, Na+ (sodium cation) channels in the cell membrane open, creating a Na+ influx. This causes the neuron potential to rise to about $+30\ mV$, which is called *depolarization*. The Na+ channels close, and K+ (potassium cation) channels open, leading to a decrease in the cell's potential, which slowly polarizes to its rest state. The repolarization typically overshoots to $-90\ mV$, a process called hyperpolarization, then returns to $-70\ mV$. The whole process lasts between 5 and 10 ms. This information flux is called the action potential and represents the information carried by the neuron (SANEI; CHAMBERS, 2007).

There are approximately 10^{11} neurons in a brain at birth, a number that gradually decreases in the course of life (KANDEL *et al.*, 2021; SANEI; CHAMBERS, 2007). The sum of the action potential of millions of neurons in the brain creates electrical currents that are measurable through one person's scalp, even though this potential is largely attenuated, being measured with an amplitude varying between 10 and 100 μV, even though the neuron internal potential can vary about 100 mV. This method of measuring brain waves is called Electroencephalography. Figures 2.6a and 2.6b show the International 10-20 and 10-10 Systems for EEG setup, which defines some skull landmarks (nasion, preauricular points, inion) and position the EEG electrodes at 10 or 20% the distance of the landmarks (SANEI; CHAMBERS, 2007; TEPLAN, 2002; KLEM *et al.*, 1999). It is important to cite that EEG is actually one of many non-invasive brain signal acquisition techniques, other examples include Near-Infrared Spectroscopy (NIRS);

Magnetoencephalography (MEG); Functional Magnetic Resonance Imaging (fMRI) and Positron Emission Tomography (PET) (YEN *et al.*, 2023; PASZKIEL, 2019; RAMADAN; VASILAKOS, 2017).



(a) 10-20 system. (b) 10-10 system

Figure 2.6 – International 10-20 and 10-10 EEG montages. Electrodes A1 and A2 are placed at the ear lobes.

The motor and sensory cortex becomes stimulated during the execution or imagination of a motor task. This is reflected in the oscillatory rhythms of this region, which can be captured through EEG. Additionally, as the neurons from the motor cortex are crossed in the medulla the imagination or execution of hand or foot movement results in ERD/ERS at the contralateral hemisphere of the brain (PFURTSCHELLER *et al.*, 1997).

3 Independent Component Analysis

Blind Source Separation (BSS) concerns the problem of separating information sources from an observed mixture thereof using a minimum of prior knowledge about their characteristics. A classical example is given by the so-called *cocktail party problem*: consider a room with N_s (number of sources) individuals speaking simultaneously and N_x (number of mixed signals) microphones scattered across the room at different locations. Different microphones will receive all voices at all times, but with distinct magnitudes and time delays - hence, the observed signals can be modeled as a linear mixture of the unseen sources, where the mixing system is unknown. The challenge is to recover the individual speech signals from those mixtures, using, whenever possible, only the hypothesis that the sources are statistically independent. This hypothesis establishes a relationship between source separation and the technique known as ICA. The linear mixing process can be mathematically represented as:

$$x_i = \sum_{j=1}^{N_s} a_{i,j} s_j \tag{3.1}$$

where x_i is the i-th observed source, s_j is the j-th source signal, and $a_{i,j}$ is the weight of the j-th source contribution to the i-th observed signal (HYVARINEN *et al.*, 2001). This can be written in matrix notation as:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{3.2}$$

where $\mathbf{s} = [s_1, \ldots, s_{N_s}]^T$ is the source vector, $\mathbf{x} = [x_1, \ldots, x_{N_x}]^T$ is the observed signal vector and $\mathbf{A}$ is the mixing matrix, composed by the weights $a_{i,j}$. The statistical independence assumption implies that no source conveys information about any other source. These two properties (linear mixing and source independence) are the main assumptions underlying the classical ICA framework, translating successful separation into finding the mixing weights and recovering the sources (HYVARINEN *et al.*, 2001). It will be seen that, in general, the estimated amplitude of the sources and the order in which they are recovered are arbitrary. This is generally done by finding the $N_s \times N_x$ unmixing matrix $\mathbf{W}$ such that

$$\mathbf{y} = \mathbf{W}\mathbf{x} \tag{3.3}$$

where $\mathbf{y}$ vector contains the extracted sources.

ICA has a very rich and interesting history. Hérault and Jutten presented one of the first works that informally used the concept, in 1983, and, a few years later, Comon presented a more rigorous mathematical formulation and a solution to the problem using higher-order cumulants (KACHENOURA *et al.*, 2008; HYVARINEN, 1997). In parallel, Cardoso and Souloumiac (1993) developed the Joint Approximate Diagonalization of Eigenmatrices (JADE) algorithm, which used fourth-order cumulants. Solutions using only second-order statistics were also studied, such as Second-Order Blind Identification (SOBI), by Belouchrani *et al.* (1997), which considers the source time-lagged covariance matrices. The Fast ICA method (HYVARINEN, 1999) differs from all of those approaches by extracting one source at a time from the signal mixture, which is known as deflation, but it has a symmetric variant that allows us to extract all sources at once. Bell and Sejnowski (1995) explored how to directly use the independence assumption to develop the Information Maximization algorithm, which was later improved by Lee *et al.* (1999). Picard, a recently proposed approach, takes into account the real distribution and characteristics of the signals and improves ICA robustness and convergence speed on real data (ABLIN *et al.*, 2018). A simple addition to the technique, which constrains the sources to be orthogonal, was proposed under the name of Picard-O.

ICA is one of many tools that are used for pre-processing and feature extraction in BCIs. Ideally, it may allow information-bearing brain signals to be separated from independent noise, interference and artifacts. This strategy has been applied with success (BAI *et al.*, 2014; WINKLER *et al.*, 2011; DELORME *et al.*, 2007), and the mitigation of Electrocardiography (ECG) artifacts, line noise, blinks and muscular activity is reported in works like Hamaneh *et al.* (2014). There are also works that use it in the contexts of spatial filtering (WANG *et al.*, 2012), source extraction for MI-related activity patterns (WU *et al.*, 2020), and also as a preprocessing step for neural networks (HAN *et al.*, 2022). ICA has shown to be useful for separating the interesting somatosensory activity in the desired bands (*beta* and *mu* rhythms) from other unwanted rhythms and regions, ultimately improving the performance of MI-based BCIs (KACHENOURA *et al.*, 2008).

Some ICA methods require the whitening of the observed signals prior to source extraction. It refers to a linear transformation such that the resulting signals are uncorrelated, effectively removing first and second-order interactions between the signals without losing information. Orthogonalization techniques and Principal Component Analysis also are techniques that remove those interactions and can be used in substitution to achieve the same goal. This step is sometimes mandatory (for instance, in SOBI and JADE), but it is always useful to better pose the data to the source extraction algorithms (HYVARINEN *et al.*, 2001).

In this chapter, we present the characteristics, assumptions, and goals of classical ICA methods.

3.1 Infomax

The Infomax (*Information Maximization*) algorithm was originally designed to maximize the mutual information between the output y and the input x of a neural network (NN). The idea is translated into the ICA domain by assuming that y and x represent the recovered ICA sources and the mixtures, respectively. The mutual information between them is defined as:

$$I(y,x) = H(y) - H(y|x) \tag{3.4}$$

with

$$H(y) = -E[ln(p_y(y))] = -\int_{-\infty}^{\infty} p_y(y) ln(p_y(y)) dy \tag{3.5}$$

where H is the entropy function and $H(y|x)$ is the parcel of the entropy of y that does not come from the input (e.g. a noise parcel). The derivative of $I(y,x)$ w.r.t. to a parameter w of the Neural Network (NN) can be written as:

$$\frac{\partial I(y,x)}{\partial w} = \frac{\partial H(y)}{\partial w}$$

since $H(y|x)$ does not depend on the network.

For a single-input / single-output network, with $y = f(wx + b)$, where w is the network weight, b is the bias, and $f(.)$ is a nonlinearity, $I(y,x)$ is maximized by aligning high density parts of the probability density function (pdf) p_x of x with high varying parts of y, as small changes in the most probable values of the input will cause the highest changes in the output, causing the highest information flow possible. If $f(.)$ is monotonically increasing or decreasing it has a unique inverse and p_y (the pdf of y) can be written as a function of p_x (AMARI, 1998) as:

$$p_y(y) = p_x(x) \left| \frac{\partial y}{\partial x} \right|^{-1} \tag{3.6}$$

Substituting Equation (3.6) into Equation (3.5) yields:

$$H(y) = E\left[ln \left| \frac{\partial y}{\partial x} \right| \right] - E[p_x(x)] \tag{3.7}$$

where the first term of the right-hand side is the average of the logarithm of the derivative of y w.r.t. x. If we assume that we have a dataset of realizations of x that represents the distribution p_x, the derivative of $H(y)$ w.r.t. w can be written as [1]:

$$\frac{\partial H(y)}{\partial w} = \frac{\partial}{\partial w} ln \left(\left| \frac{\partial y}{\partial x} \right| \right) = \frac{\partial y}{\partial x}^{-1} \frac{\partial}{\partial w} \left(\frac{\partial y}{\partial x} \right) \tag{3.8}$$

For a multi-input / multi-output network with output $\mathbf{y} = f(\mathbf{W}^T\mathbf{x} + \mathbf{b})$, similar reasoning can be used. The determinant of the Jacobian of $\mathbf{y}$ represents the volume of the space in $\mathbf{y}$ into which $\mathbf{x}$ is mapped (AMARI, 1998; PAPOULIS, 1991). Equation (3.6) can be rewritten in matrix form as:

$$p_{\mathbf{y}}(\mathbf{y}) = \frac{p_{\mathbf{x}}(\mathbf{x})}{|\nabla \mathbf{y}|} \tag{3.9}$$

where ∇ is the vector differential operator, with $\nabla \mathbf{y} = \left[\frac{\partial y_1}{\partial x}, \ldots, \frac{\partial y_{N_x}}{\partial x} \right]$. By following the same steps as in the one-input one-output case, we arrive at:

$$\frac{\partial H(y)}{\partial \mathbf{W}} = \frac{\partial}{\partial \mathbf{W}} ln\left(|\nabla \mathbf{y}|\right) = \left(\mathbf{W}^T \right)^{-1} + (\mathbf{1} - 2\mathbf{y})\mathbf{x}^T \tag{3.10}$$

where $\mathbf{1}$ is a N_x-sized vector of ones. Differentiating H w.r.t. $\mathbf{b}$ yields

$$\frac{\partial H(y)}{\partial \mathbf{b}} = \frac{\partial}{\partial \mathbf{b}} ln\left(|\nabla \mathbf{y}|\right) = \mathbf{1} - 2\mathbf{y} \tag{3.11}$$

For an individual weight $w_{i,j}$ of $\mathbf{W}$, the rule is equivalent to

[1] This derivative can be calculated using the chain rule $\frac{\partial g}{\partial w} = \frac{\partial g}{\partial u}\frac{\partial u}{\partial t}\frac{\partial t}{\partial w}$, with $g = ln\left|\frac{\partial y}{\partial x}\right|$, $u = ln(x)$ and $t = |x|$.

$$\Delta w_{i,j} \propto \frac{\text{cof } w_{i,j}}{\det \mathbf{W}} + x_j(1 - 2y_i) \tag{3.12}$$

where cof $w_{i,j}$ is the cofactor of $w_{i,j}$, which is $(-1)^{i+j}$ times the determinant of the matrix obtained by removing the i-th row and j-th column from $\mathbf{W}$ (BELL; SEJNOWSKI, 1995). This gradient can then be used for an iterative update of the network weights.

Infomax is one of the most used ICA algorithms due to its implementation in common computational frameworks (DELORME; MAKEIG, 2004), and has been applied as a preprocessing step in BCIs (URIBE, 2018; URIBE *et al.*, 2016). It has been also found to be an effective method in minimizing the mutual information between the recovered sources (DELORME *et al.*, 2012). It is also reportedly, among compared ICA methods, the one that most improves MI-BCI accuracy when used in EEG preprocessing (BRUNNER *et al.*, 2007; NAEEM *et al.*, 2006).

3.2 Extended infomax

The Information Maximization method proposed by Bell and Sejnowski (1995) is shown to fail in extracting sources that have subgaussian distributions (negative kurtosis). This is described in their work and is due to the use of the logistic function as the activation function in the neural network that performs the calculation. As we mentioned, the most probable values of the sources p.d.f. must match the high slope parts of the nonlinearity. For this to happen, the activation function must approximate the cumulative distribution function (c.d.f.) of the source signals (up to a constant value, as in practice, only the derivative of the function is used), i.e., $f(v) \approx \int_{-\infty}^{v} f_v(y)dy$. In the same work, it is proposed to use the generalized sigmoid function as a nonlinearity, first optimizing the parameters of this function and then adjusting the network weights (LEE *et al.*, 1999).

Let $\mathbf{s}(t) = [s_1(t), \ldots, s_{N_s}(t)]$ be independent signal sources, such that the observed data vector is $\mathbf{x}(t) = [x_1(t), \ldots, x_{N_x}(t)] = \mathbf{A}\mathbf{s}(t)$, where $\mathbf{A}$ is a full rank $N_x \times N_s$ scalar matrix. Since the sources are independent, we can write its multivariate p.d.f. as

$$p(\mathbf{s}) = \prod_{i=1}^{M} p_{s_i}(s_i) \tag{3.13}$$

The mutual information of the observed vector can be written as the Kullback-

Leibler (KL) divergence of the multivariate density $p(\mathbf{s})$ from the product of the marginal densities, as

$$I(\mathbf{x}) = \int_{-\infty}^{\infty} ln \frac{p(\mathbf{x})}{\prod_{i=1}^{N} p_{s_i}(x_i)} d\mathbf{x} \tag{3.14}$$

The goal is to find an unmixing matrix $\mathbf{W}$ such that the unmixed signals $\mathbf{y}(t) = \mathbf{W}\mathbf{x}(t)$ are independent. We saw that the p.d.f. of the unmixed signals can be expressed as a function of the p.d.f. of the mixed observations by using Equation (3.9), which yields Equation (3.15), where $p(\mathbf{y}) = \prod_{i=1}^{N} p_{s_i}(y_i)$ is also the distribution of the sources.

$$p(\mathbf{x}) = p(\mathbf{y})|det(\mathbf{W})| \tag{3.15}$$

The original algorithm has a learning rule of the form $\Delta\mathbf{W} \propto \left[(\mathbf{W}^T)^{-1} - \phi(\mathbf{y})\mathbf{x}^T\right]$, where

$$\phi(\mathbf{y}) = -\frac{\frac{\partial p(\mathbf{y})}{\partial \mathbf{y}}}{p(\mathbf{y})} \tag{3.16}$$

is the score function. It is also possible to use the natural gradient (AMARI, 1998) to improve the convergence speed of this algorithm, which results in

$$\Delta\mathbf{W} \propto \frac{\partial H(y)}{\partial \mathbf{W}} \mathbf{W}^T\mathbf{W} = \left[I - \phi(\mathbf{y})\mathbf{y}^T\right]\mathbf{W} \tag{3.17}$$

The natural gradient is employed to better guide the search through the actual landscape of the objective function, taking into account the manifold that is spanned by it (AMARI, 1998).

In the formulation of Equation (3.17), if we choose $f(y) = tanh(y)$ as the network nonlinearity, the learning rule reduces to

$$\Delta\mathbf{W} \propto \frac{\partial H(y)}{\partial \mathbf{W}} \mathbf{W}^T\mathbf{W} = \left[I - 2\tanh(\mathbf{y})\mathbf{y}^T\right]\mathbf{W} \tag{3.18}$$

The Extended Infomax aims to use different nonlinearities for both subgaussian and supergaussian distributions, which is desired since the original Infomax ICA cannot deal with subgaussian sources. We can create a strictly subgaussian distribution by a mixture of normal distributions, as in Equation (3.19), where μ and $-\mu$ are the means and σ^2 is the variance of the mixed distributions:

$$p(y) = \frac{1}{2}\left(\mathcal{N}(\mu,\sigma^2) + \mathcal{N}(-\mu,\sigma^2)\right) \tag{3.19}$$

The kurtosis of this distribution is written in Equation (3.20), varying between -2 and 0 depending on the values of μ and σ^2:

$$\text{kurt}(p(y)) = \frac{-2\mu^4}{(\mu^2+\sigma^2)^2} \tag{3.20}$$

For this choice of $p(y)$, if we further set $\mu = 1$ and $\sigma^2 = 1$, the function $\phi(y)$ takes the form of

$$\phi_{-}(y) = y - \tanh(y) \tag{3.21}$$

where the '$-$' subscript denotes that it is the scoring function for the subgaussian case. Note that this function is strictly for subgaussian sources. To describe supergaussian sources we can model their distribution as:

$$p(y) = p_0(y)\text{sech}^2(y) \tag{3.22}$$

where $sech(.)$ is the hyperbolic secant function and $p_0(y) = \mathcal{N}(0,1)$. The function $\phi(y)$ takes the form of

$$\phi_{+}(y) = y + \tanh(y) \tag{3.23}$$

for this distribution, where '+' denotes the supergaussian case. Those choices of prior assumptions for the distributions of the sources are interesting because they yield the learning rules of Equation (3.24), which only differ by a sign, as follows:

$$\Delta\mathbf{W} \propto \begin{cases} [\mathbf{I} - \tanh(\mathbf{y})\mathbf{y}^T - \mathbf{y}\mathbf{y}^T]\mathbf{W} & \text{if supergaussian assumption} \\ [\mathbf{I} + \tanh(\mathbf{y})\mathbf{y}^T - \mathbf{y}\mathbf{y}^T]\mathbf{W} & \text{if subgaussian assumption} \end{cases} \tag{3.24}$$

The learning rule can be written in a single equation, as:

$$\Delta\mathbf{W} \propto [\mathbf{I} - \mathbf{K}\tanh(\mathbf{y})\mathbf{y}^T - \mathbf{y}\mathbf{y}^T]\mathbf{W} \tag{3.25}$$

where $\mathbf{K}$ is a $N_x \times N_x$ diagonal matrix and $k_i \in \{-1, 1\}$ are the elements of its diagonal (at the i-th row and i-th column).

The number of subgaussian sources can be assumed or approximated. If we assume N_{sub} subgaussian sources, the k_i values are such that $k_i = -1$ for $i \leq N_{\text{sub}}$ and $k_i = 1$ for $i > N_{\text{sub}}$. Lee *et al.* (1999) presented a switching strategy that is guaranteed to be asymptotically stable, where k_i is calculated using

$$k_i = \text{sign}\left(E[\text{sech}^2(y_i)]E[y_i^2] - E[\tanh(y_i)y_i]\right) \tag{3.26}$$

The computational framework MNE (GRAMFORT *et al.*, 2013) uses the following heuristic calculation:

$$k_i = \text{sign}\left(\text{kurt}(y_i)\right) \tag{3.27}$$

as empirical estimates of the recovered sources kurtosis in order to switch the values of k_i. In this equation, *kurt* represents the empirical kurtosis calculation.

3.3 Fast ICA

The Fast ICA is a fixed-point iterative method that uses the projection pursuit approach with the goal of finding an informative projection of the observed EEG signals. The method uses the notion of negentropy of a random vector $\mathbf{x}$, $J(\mathbf{x})$, which is calculated as the difference of the entropy of $\mathbf{x}$ and that of a Gaussian random variable $\mathbf{x}_{gauss}$ with the same covariance matrix as $\mathbf{x}$ (HYVARINEN, 1999), as follows:

$$J(\mathbf{x}) = H(\mathbf{x}_{gauss}) - H(\mathbf{x}) \tag{3.28}$$

The mutual information can be written in terms of negentropy, as in Equation (3.29). It is used as the metric of independence in Fast ICA, so the goal will be to minimize the mutual information of the recovered sources.

$$I(\mathbf{x}) = J(\mathbf{x}) - \sum_{i=1}^{M} J(x_i) \tag{3.29}$$

Hyvarinen (1997) has shown that $J(x_i)$ can be approximated using applications of practically any non-quadratic functions G (subject to some constraints) to x_i as follows:

$$J(x_i) \approx c(E[G(x_i)] - E[G(\nu)])^2 \tag{3.30}$$

where c is a constant and ν is a random Gaussian variable of null mean and unit variance. This can already be used for finding one independent component $y_i = \mathbf{w}^T\mathbf{x}$, where $\mathbf{w}$ is scaled such that $E[(\mathbf{w}^T\mathbf{x})^2] = 1$ (which is not a problem, since ICA admits that the sources be recovered up to a scaling factor).

This approach yields Equation (3.31) as the estimation of the negentropy, and we can use it as the maximization objective function for each $\mathbf{w}_i$ of $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_{N_s}]^T$ by using a deflation strategy:

$$J_G(\mathbf{w}) = (E[G(\mathbf{w}^T\mathbf{x})] - E[G(\nu)])^2 \tag{3.31}$$

Another approach is to jointly calculate all $\mathbf{w}_i$ by enforcing a decorrelation constraint, as follows:

$$\underset{\mathbf{w}_1,\ldots,\mathbf{w}_{N_s}}{\arg\max} \quad \sum_{i=1}^{N_s} J_G(\mathbf{w}_i) \tag{3.32}$$

$$\text{s.t.} \quad E[(\mathbf{w}_j^T\mathbf{x})(\mathbf{w}_k^T\mathbf{x})] = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases} \tag{3.33}$$

Hyvarinen (1997) recommends the use of three different *G(.)* functions, all of which satisfy the conditions of consistency, asymptotic variance and robustness of the estimator of $\mathbf{w}_i$ and $\mathbf{W}$ – they are presented in Equations (3.34), (3.36) and (3.38), with $g_i(u)$ being the derivative of the respective $G_i(u)$:

$$G_1(u) = \frac{1}{a_1} ln(\cosh(a_1 u)) \tag{3.34}$$

$$g_1(u) = tanh(a_1 u) \tag{3.35}$$

$$G_2(u) = -\frac{1}{a_1} e^{-\frac{a_2 u^2}{2}} \tag{3.36}$$

$$g_2(u) = u e^{-\frac{a_2 u^2}{2}} \tag{3.37}$$

$$G_3(u) = \frac{1}{4} u^4 \tag{3.38}$$

$$g_3(u) = u^3 \tag{3.39}$$

with $1 \leq a_1 \leq 2$ and $a_2 \approx 1$ are constants.

3.3.1 Fixed-point algorithm

If we apply the Karush–Kuhn–Tucker (KKT) conditions to Equation (3.31), we will see that its maxima under the constraint of $||\mathbf{w}||^2 = 1$ respect Equation (3.40), where β can be calculated as $\beta = E[\mathbf{w}_0^T \mathbf{x} g(\mathbf{w}_0^T \mathbf{x})]$ and $\mathbf{w}_0$ is the value of $\mathbf{w}$ that minimizes Equation (3.31):

$$F(\mathbf{w}) = E[\mathbf{x} g(\mathbf{w}^T \mathbf{x})] - \beta \mathbf{w} = 0 \tag{3.40}$$

We can then use Newton's method to solve this optimization problem. The Jacobian of F is

$$\nabla F = E[\mathbf{x}\mathbf{x}^T g'(\mathbf{w}^T \mathbf{x})] - \beta \mathbf{I} \tag{3.41}$$

The method requires the calculation of the inverse of the Jacobian, so a few approximations will be made. First, we will assume that $\mathbf{x}$ is sphered, e.g., its covariance matrix is the identity matrix such that $E[\mathbf{x}\mathbf{x}^T g'(\mathbf{w}^T \mathbf{x})] = E[\mathbf{I} g'(\mathbf{w}^T \mathbf{x})] = E[g'(\mathbf{w}^T \mathbf{x})]$. The value of β is also approximated by using the current value of $\mathbf{w}$ instead of $\mathbf{w}_0$.

The Newton's method step is defined as $\mathbf{w}(n+1) = \mathbf{w}(n) - F(\mathbf{w})/F'(\mathbf{w})$, and can be written in two steps as:

$$\begin{aligned} \mathbf{w} &\leftarrow E[\mathbf{x} g(\mathbf{w}^T \mathbf{x})] - E[\mathbf{x} g'(\mathbf{w}^T \mathbf{x})]\mathbf{w} \\ \mathbf{w} &\leftarrow \frac{\mathbf{w}}{||\mathbf{w}||} \end{aligned} \tag{3.42}$$

Convergence proof of this algorithm can be found at Hyvarinen (1999).

For estimating several $\mathbf{w}$ we must prevent each one from converging to the same result. This can be done by applying a Gram-Schmidt decorrelation at each step when estimating the p-th weight vector ($\mathbf{w}_p$) as:

$$\begin{aligned} &\mathbf{w}_p \leftarrow \mathbf{w}_p - \sum_{i=1}^{p-1} \mathbf{w}_p^T \mathbf{w}_i \mathbf{w}_i \\ &\mathbf{w}_p \leftarrow \frac{\mathbf{w}_p}{||\mathbf{w}_p||} \end{aligned} \tag{3.43}$$

or can be done at once for the entire matrix $\mathbf{W}$ after one full iteration of approximating $\mathbf{w}_p$ for all p by using

$$\mathbf{W} \leftarrow (\mathbf{W}\mathbf{W}^T)^{-\frac{1}{2}}\mathbf{W} \tag{3.44}$$

where $(\mathbf{W}\mathbf{W}^T)^{-\frac{1}{2}}$ can be obtained by eigenvalue decomposition.

As it is a fast and reliable ICA option, it is found to be used for different purposes in the BCI pipeline, such as pre-processing (WU *et al.*, 2020; REJER; GóRSKI, 2017), artifact removal (BAI *et al.*, 2014; WINKLER *et al.*, 2011) and feature dimensionality reduction (SAFITRI *et al.*, 2020).

3.4 Second-Order Blind Identification

The character of time series of the signals, which implies that they have a natural time structure, can be exploited for an ICA solution. SOBI uses the time-lagged covariance matrices as a measurement of independence between the recovered sources, aiming to nullify this second-order dependency. This is not perfect, since independence would imply nullifying dependencies of all orders, but is a reasonable approach. For this, we will assume that the source signal $\mathbf{s}(t) = [s_1(t), \ldots, s_{N_s}(t)]^T$ is a deterministic ergodic sequence or a stationary multivariate process, meaning that the signal autocovariance matrix is a diagonal, as follows:

$$\mathbf{R}_\mathbf{s} = E[\mathbf{s}(t+\tau)\mathbf{s}(t)^*] = \text{diag}[\rho_1(\tau), \ldots, \rho_n(\tau)] \tag{3.45}$$

where $\rho_n(\tau) = E[s_n(t+\tau)s_n(t)^*]$. This condition means that a source at time t is not correlated to any other source at any other present or past time (BELOUCHRANI *et al.*, 1997).

The assumed mixture model is as follows:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{n}(t) \tag{3.46}$$

The mixing matrix $\mathbf{A}$ is assumed to be full rank and has dimensions $N_x \times N_x$ and the observed mixtures $\mathbf{x}(t)$ are generated by a linear transformation of the sources $\mathbf{s}(t)$ plus a noise $\mathbf{n}(t) = [n_1(t), \ldots, n_{N_s}(t)]$ which is assumed to be stationary, temporally and spatially white, as described by:

$$E[\mathbf{n}(t+\tau)\mathbf{n}(t)^*] = \sigma^2\delta(\tau)\mathbf{I} \tag{3.47}$$

where σ^2 is the noise's variance and $\delta(\tau)$ is Kronecker's delta.

It directly follows from those assumptions that the autocovariance matrices $\mathbf{R}_\mathbf{x}(\tau) = E[\mathbf{x}(t+\tau)\mathbf{x}(t)^*]$ of the observed signals can be written as:

$$\mathbf{R}_\mathbf{x}(0) = \mathbf{A}\mathbf{R}_\mathbf{s}(0)\mathbf{A}^H + \sigma^2\mathbf{I} \tag{3.48}$$

$$\mathbf{R}_\mathbf{x}(\tau) = \mathbf{A}\mathbf{R}_\mathbf{s}(\tau)\mathbf{A}^H \qquad \tau \neq 0 \tag{3.49}$$

Since the ICA solution has ambiguities concerning the order and the scale of the sources, we conventionally assume that they have unit variance, meaning that $\mathbf{R}_\mathbf{s}(0) = \mathbf{I}$ and $\mathbf{R}_\mathbf{y}(0) = \mathbf{A}\mathbf{A}^H$, where $\mathbf{y}(t) = \mathbf{A}\mathbf{s}(t)$. This leaves the order and phases of the columns in $\mathbf{A}$ undetermined, so the goal will be to estimate a matrix $\mathbf{W}$ such that $\mathbf{W} = \mathbf{P}\mathbf{A}^{-1}$, where $\mathbf{P}$ is a matrix with only one non-zero element per column and row with unitary modulus, meaning that we can determine $\mathbf{A}$ up to permutation and phase shift and that the sources are also recovered up to permutation and phase shift. The existence of such matrix $\mathbf{P}$ can be represented by $\mathbf{W} \doteq \mathbf{A}^{-1}$, which means that $\mathbf{W}$ is *essentially equal* to $\mathbf{A}^{-1}$.

If we have a whitening matrix $\mathbf{B}$ of $\mathbf{y}(t)$ such that

$$E[\mathbf{B}\mathbf{y}(t)\mathbf{y}(t)^*\mathbf{B}^H] = \mathbf{B}\mathbf{R}_\mathbf{y}(0)\mathbf{B}^H = \mathbf{B}\mathbf{A}\mathbf{A}^H\mathbf{B}^H = \mathbf{I}$$

we can conclude that $\mathbf{B}\mathbf{A} = \mathbf{U}$, where $\mathbf{U}$ is an unitary matrix, i.e., $\mathbf{U}\mathbf{U}^H = \mathbf{I}$, and hence $\mathbf{A} = \mathbf{B}^{\#}\mathbf{U}$, where # denotes the Moore-Penrose inverse. The white signal $\mathbf{z}(t)$ can then be defined as:

$$\mathbf{z}(t) = \mathbf{B}\mathbf{x}(t) = \mathbf{B}[\mathbf{A}\mathbf{s}(t) + \mathbf{n}(t)] = \mathbf{U}\mathbf{s}(t) + \mathbf{B}\mathbf{n}(t) \tag{3.50}$$

From Equation (3.49) and considering $\mathbf{A} = \mathbf{B}^{\#}\mathbf{U}$, we can obtain

$$\mathbf{R}_z(\tau) = \mathbf{U}\mathbf{R}_s(\tau)\mathbf{U}^H, \qquad \forall\tau \neq 0 \tag{3.51}$$

which is used in Theorems 3.1 and 3.2 and discussed in Belouchrani *et al.* (1997):

Theorem 3.1 *First Uniqueness Conditions: For a $\tau \neq 0$ and a unitary matrix $\mathbf{V}$ such that*

$$\mathbf{V}^H \mathbf{R}_z(\tau)\mathbf{V} = diag[d_1, \ldots, d_{N_x}] \tag{3.52}$$

$$\textit{with } \rho_i(\tau) \neq \rho_j(\tau) \ \forall\ 1 \leq i \neq j \leq N_x \tag{3.53}$$

hold, the following statements are true:

1. $\mathbf{V} \doteq \mathbf{U}$
2. $[d_1, \ldots, d_{N_x}]$ *is a permutation of* $[\rho_1(\tau), \ldots, \rho_{N_x}(\tau)]$.

Theorem 3.2 *Second Uniqueness Conditions: For a set of K non zero lags $\{\tau_1, \ldots, \tau_K\}$ and $\mathbf{V}$ an unitary matrix such that*

$$\forall\ 1 \leq k \leq K \qquad \mathbf{V}^H \mathbf{R}_z(\tau_k)\mathbf{V} = diag[d_1(k), \ldots, d_{N_x}(k)] \tag{3.54}$$

$$\forall\ 1 \leq i \neq j \leq N_x \qquad \exists k,\ 1 \leq k \leq K\ ,\ d_i(k) \neq d_j(k) \tag{3.55}$$

hold, the following statements are true:

1. $\mathbf{V} \doteq \mathbf{U}$
2. $[d_1(k), \ldots, d_{N_x}(k)]$ *is a permutation of* $[\rho_1(\tau_k), \ldots, \rho_{N_x}(\tau_k)]$.

These two theorems are similar in the sense that Theorem 3.1 states some properties of the diagonalization matrix $\mathbf{V}$ for one time lag τ, and Theorem 3.2 states the same properties but regarding a set of time lags $\{\tau_1, \ldots, \tau_K\}$. For instance, if the source signals have identical normalized spectra, the conditions for Theorem 3.2 fail, while the conditions (3.53) for Theorem 3.1 are "simpler" in the sense that they depend on the source signals and not on finding $\mathbf{V}$. On the other hand, it is not easy to check conditions (3.53) a priori. Note that the existence of $\mathbf{V}$ in conditions (3.52) and (3.54) is guaranteed by Equation (3.51).

The goal of the SOBI algorithm is to jointly diagonalize a set

$$\mathcal{M} = \{\mathbf{M}_1, \ldots, \mathbf{M}_K\}$$

of K covariance matrices (for the set of time lags $\{\tau_1, \ldots, \tau_K\}$), reducing the chance of a bad choice of τ. This also increases statistical efficiency by using a larger set of statistics (BELOUCHRANI *et al.*, 1997).

The "off" measure of an $n \times n$ matrix $\mathbf{M}$ with $m_{i,j}$ values is defined as

$$\text{off}(\mathbf{M}) \triangleq \sum_{1 \leq i \neq j \leq n} |m_{i,j}|^2 \tag{3.56}$$

and the diagonalization of a matrix $\mathbf{M}$ by $\mathbf{V}$ is equivalent to zeroing off$(\mathbf{V}^H\mathbf{M}\mathbf{V})$. Additionally, if $\mathbf{M}$ is in the form of $\mathbf{M} = \mathbf{U}^H\mathbf{D}\mathbf{U}$ with $\mathbf{U}$ an unitary matrix and $\mathbf{D}$ a diagonal matrix, them off$(\mathbf{V}^H\mathbf{M}\mathbf{V}) = 0$ if and only if $\mathbf{U} \doteq \mathbf{V}$.

The joint diagonalization of the matrices in $\mathcal{M}$ is done by minimizing the joint diagonality criterion (JD) $C(\mathcal{M}, \mathbf{V})$, given by:

$$C(\mathcal{M}, \mathbf{V}) \triangleq \sum_{k=1}^{K} \text{off}(\mathbf{V}^H\mathbf{M}_k\mathbf{V}) \tag{3.57}$$

A matrix $\mathbf{V}'$ that minimizes the JD criterion is referred to as a joint diagonalizer of $\mathcal{M}$. This is stated in Theorem 3.3, as follows:

Theorem 3.3 *Essential Uniqueness of Joint Diagonalization: Let* $\mathbf{M} = \{\mathbf{M}_1, \ldots, \mathbf{M}_K\}$ *be a set of* K *matrices with* $1 \leq k \leq K$ *and* $\mathbf{M}_k$ *has the form of* $\mathbf{M}_k = \mathbf{U}\mathbf{D}_k\mathbf{U}^H$*, with* $\mathbf{U}$ *unitary and* $\mathbf{D}_k = diag[d_1(k), \ldots, d_n(k)]$ *diagonal. Any joint diagonalizer of* $\mathcal{M}$ *is essentially equal to* $\mathbf{U}$ *if and only if the following condition holds:*

$$\forall\ 1 \leq i \neq j \leq N_x \ \ \exists k,\ 1 \leq k \leq K \ \ d_i(k) \neq d_j(k) \tag{3.58}$$

The Jacobi technique (GOLUB *et al.*, 1989) can be used for the diagonalization of an Hermitian matrix through successive Givens rotations. Belouchrani *et al.* (1997) proposed an extension of the method that can be used for the joint diagonalization of non symmetric matrices, which is similar to the original algorithm but uses the JD criterion given by Equation (3.57).

Algorithm 1 SOBI

```
**Require:** Mixed signals $\mathbf{x}(t)$.
  Estimate $\hat{\mathbf{R}}(0)$ from the samples of $\mathbf{x}(t)$, and calculate its eigenvalues $\{\lambda_1, \dots, \lambda_{N_x}\}$ and corresponding eigenvectors $\{\mathbf{h}_1, \dots, \mathbf{h}_{N_x}\}$.
  Assuming a white noise, its variance can be estimated as $\hat{\sigma}^2$ using the average of the $N_x - N_s$ smallest eigenvalues of $\hat{\mathbf{R}}(0)$. The corresponding whitening matrix $\hat{\mathbf{B}}$ is given by
      $\hat{\mathbf{B}} = [(\lambda_1 - \hat{\sigma}^2)^{-\frac{1}{2}}, \dots, (\lambda_{N_x} - \hat{\sigma}^2)^{-\frac{1}{2}}]^H$   (3.59)
  Calculate the whitened signal $\mathbf{z}(t) = \hat{\mathbf{B}}\mathbf{x}(t)$
  Estimate $\hat{\mathbf{R}}_z(\tau)$ for a set of time lags $\tau \in \{\tau_k | k = 1, \dots, K\}$.
  Find a joint diagonalizer matrix $\hat{\mathbf{U}}$ for the set $\{\hat{\mathbf{R}}_z(\tau_k) | k = 1, \dots, K\}$
  The source signals are estimated as $\hat{\mathbf{s}}(t) = \hat{\mathbf{U}}^H \hat{\mathbf{B}}\mathbf{x}(t)$.
  The mixing matrix is estimated as $\mathbf{W}^H = \hat{\mathbf{B}}^{\#}\hat{\mathbf{U}}$.
  **Output:** $\hat{\mathbf{s}}(t)$, $\mathbf{W}$
```

The complete SOBI algorithm is as follows:

The computational cost of SOBI is relatively small, compared to other methods. It is reported to be one of the least robust ICA methods in terms of signal-noise ratio (SNR) of the input signals (KACHENOURA *et al.*, 2008), and to extract sources that are not necessarily related to ERPs (WU *et al.*, 2020). At the same time, it has been reported to provide an excellent separation when its assumptions are met (SAHONERO; CALDERON, 2017).

3.5 Joint Approximate Diagonalization of Eigenmatrices

The technique of *joint approximate diagonalization of eigenmatrices* (JADE) exploits the fourth-order cumulants of the mixed signals to develop a blind beamforming algorithm, aiming to identify the source vectors without knowledge of the source array configuration. Assuming that the configuration is seen as a linear superposition of the sources, similar to the ICA model, and that the sources are independent, the beamforming problem turns into finding the mixing matrix. Using this approach, the sources can only be estimated up to a complex scalar factor (for a complex signal) and up to a permutation order, i.e., the recovered signals will likely be unordered and with different magnitudes and phase-shifts. This, as already mentioned, is generally the case in ICA, and hence we assume sources with unit variance such that the autocovariance matrices are in the form of:

$$\mathbf{R_s} = E[\mathbf{s}(t)\mathbf{s}(t)^*] = \mathbf{I} \tag{3.60}$$

and

$$\mathbf{R_y} = E[\mathbf{y}(t)\mathbf{y}(t)^*] = E[\mathbf{A}\mathbf{s}(t)(\mathbf{A}\mathbf{s}(t))^H] = \mathbf{A}\mathbf{A}^H \tag{3.61}$$

where $\mathbf{A}$ is the $N_s \times N_s$ mixing matrix, $\mathbf{y}(t) = \mathbf{A}\mathbf{s}(t)$ is the mixed signals vector and $\mathbf{s}(t) = [s_1(t), \dots, s_{N_s}(t)]^T$ are the sources. Here, as in SOBI, the identification of matrix $\mathbf{A}$ (and consequently its inverse) is made by determining a matrix that is essentially equal to $\mathbf{A}$, since ordering and phase indeterminacy are not a problem in our context. We shall also define the observed signals vector $\mathbf{x}(t) = \mathbf{y}(t) + \mathbf{n}(t)$, where $\mathbf{n}(t)$ represents a corrupting additive noise.

The use of cumulants comes from their interesting characteristics as statistical descriptors of probability functions. One important property is that, if all signals $v_i, \dots, v_n$ are independent, it follows that their joint cumulants are zero (PECCATI; TAQQU, 2011) - this is an important result for establishing a connection between JADE and ICA.

The n-th ($n > 1$) order joint cumulant Cum of the random variables (signals) $v_1, \dots, v_n$ can be written as:

$$Cum(v_1, \dots, v_n) = \sum_{\pi} (|\pi| - 1)!(-1)^{|\pi|-1} \prod_{B \in \pi} E\left[\prod_{p \in B} v_p\right] \tag{3.62}$$

where π is the set of all possible *partitions* (non-empty sets whose union create the full set) of $\{1, \dots, n\}$, and B are the sets contained in each partition in π (PECCATI; TAQQU, 2011). For instance, if $n = 3$, then π contains $\{\{1\}, \{2\}, \{3\}\}$, $\{\{1\}, \{2, 3\}\}$, $\{\{2\}, \{1, 3\}\}$, $\{\{3\}, \{1, 2\}\}$ and $\{\{1, 2, 3\}\}$, so:

$$\begin{aligned} Cum(v_1, v_2, v_3) &= E[v_1 v_2 v_3] && (3.63)\\ &- E[v_1 v_2]E[v_3] - E[v_1 v_3]E[v_2] - E[v_2 v_3]E[v_1] && (3.64)\\ &+ 2E[v_1]E[v_2]E[v_3] && (3.65) \end{aligned}$$

Furthermore, let's define a cumulant set $\mathcal{L}_\mathbf{v}$ of a d-dimensional stationary process $\mathbf{v}(t)$ as:

$$\mathcal{L}_{\mathbf{v}} = \{Cum(v_i, v_j^*, v_k, v_l^*)|1 \leq i, j, k, l \leq d\} \tag{3.66}$$

The kurtosis of the p-th complex signal source $s_p(t)$ is defined as the fourth-order cumulant as follows:

$$kurt(s_p(t)) = kurt_p = Cum(s_p, s_p^*, s_p, s_p^*) \tag{3.67}$$

For the next steps, we will have the following set of assumptions for the ICA scenario:

- H_0. The processes $\mathbf{n}(t)$ and $s_1(t), \ldots, s_{N_s}(t)$ are jointly stationary (and hence the time t does not matter for calculating the cumulants).
- H_1. There is at most one source with zero kurtosis.
- H_2. The rows on $\mathbf{A}$ are linearly independent.
- H_3. The processes $s_1(t), \ldots, s_{N_s}(t)$ are statistically independent for each t.
- H_4. There exist consistent estimates for $\mathbf{R_x}$ and $\mathcal{L}_{\mathbf{x}}$ (implying finite variance and kurtosis).
- H_5. The additive noise $\mathbf{n}(t)$ is gaussian and independent of the sources.
- H_6. $\mathbf{n}(t)$ is spatially white with $\mathbf{R_n} = \sigma^2\mathbf{I}$, where σ^2 is the noise's variance.

As in SOBI, we can whiten the process $\mathbf{y}(t)$ by calculating the whitening matrix $\mathbf{B}$. The whitened process

$$\mathbf{z}(t) = \mathbf{Bx}(t) = \mathbf{Us}(t) + \mathbf{Bn}(t) \tag{3.68}$$

still respects the ICA model in Equation (3.46). The matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_{N_s}] = \mathbf{BA}$ will be unitary, since $\mathbf{R}_z = \mathbf{BR}_y\mathbf{B}^H = \mathbf{BAA}^H\mathbf{B}^H = \mathbf{I}$. All of the second order information in $\mathbf{z}(t)$ is exhausted, i.e., multiplying it by any unitary matrix doesn't change its covariance.

Cardoso and Souloumiac (1993) propose two approaches to solve JADE, one based on eigendecomposition and one based on a cumulant criterion.

3.5.1 Eigendecomposition approach

We define a *cumulant matrix* $\mathbf{Q}_z(\mathbf{M})$ associated with a $n \times n$ matrix $\mathbf{M}$, by defining each entry $q_{i,j}$ as:

$$\begin{aligned} q_{i,j} &= \sum_{k=1,l=1}^{n} Cum(z_i, z_j^*, z_k, z_l^*) m_{l,k} \\ 1 \leq i, j &\leq n \end{aligned} \tag{3.69}$$

The (k,l)-th cumulant slice is the matrix whose entry in the i-th column and j-th row is $Cum(z_i, z_j^*, z_k, z_l^*)$, which is equal to $\mathbf{Q}_z(\mathbf{b}_l \mathbf{b}_k^*)$, where $\mathbf{b}_k$ is a $n \times 1$ matrix with zeros in all positions but the k-th. The parallel set $\mathcal{N}^p$ is defined as the set of all parallel slices and can be represented as:

$$\mathcal{N}^p = \{\mathbf{Q}_z(\mathbf{b}_l \mathbf{b}_k^*) | 1 \leq k, l \leq n\} \tag{3.70}$$

Since $\mathbf{z}(t)$ takes the form of Equation (3.68), we can use cumulant properties (additivity, multilinearity, Gaussian rejection) (CARDOSO; SOULOUMIAC, 1993) and write $\mathbf{Q}_z(\mathbf{M})$ as:

$$\begin{aligned} \mathbf{N} = \mathbf{Q}_z(\mathbf{M}) &= \sum_{p=1}^{n} kurt_p \mathbf{u}_p^* \mathbf{M} \mathbf{u}_p \mathbf{u}_p \mathbf{u}_p^* \\ &= \mathbf{U} \mathbf{\Lambda}_{\mathbf{M}} \mathbf{U}^H \\ &\forall\, \mathbf{M} \end{aligned} \tag{3.71}$$

where $\mathbf{\Lambda}_{\mathbf{M}} = \text{diag}(kurt_1 \mathbf{u}_1^* \mathbf{M} \mathbf{u}_1, \ldots, kurt_n \mathbf{u}_n^* \mathbf{M} \mathbf{u}_n)$. From Equation (3.71) we see that $\mathbf{U}$ diagonalizes any cumulant matrix, and since $\mathbf{A} = \mathbf{B}^{\#}\mathbf{U}$ we know that left multiplying the $\mathbf{Q}_z$ eigenvectors $\mathbf{U}$ by $\mathbf{B}^{\#}$ yields the mixing matrix.

3.5.2 Approach based on a cumulant criterion

Let us define $\mathbf{e}(t)$ as:

$$\mathbf{e}(t) = \mathbf{V}^H \mathbf{z}(t) = \mathbf{V}^H \mathbf{U} \mathbf{s}(t) + \mathbf{V}^H \mathbf{B} \mathbf{n}(t) \tag{3.72}$$

where $\mathbf{V}$ is an $n \times n$ unitary matrix. If $\mathbf{V} \doteq \mathbf{U}$, the channels of $\mathbf{e}(t)$ are the possibly permuted and phase-shifted source signals, corrupted by additive Gaussian noise, so their higher-order joint cumulants are zero. It can be shown that $\mathbf{U}$ can be determined as the unitary minimizer of the sum of the squared joint cumulants in $\mathcal{L}_{\mathbf{e}}$. Since the sum of the squared joint cumulants plus the sum of the squared auto-cumulants does not depend on $\mathbf{V}$, this criterion is equivalent to maximizing the sum of the squared auto-cumulants, as follows:

$$c'(\mathbf{V}) = \sum_{i=1}^{n} |Cum(e_i, e_i^*, e_i, e_i^*)|^2 \tag{3.73}$$

Cardoso and Souloumiac (1993) proposed determining $\mathbf{U}$ as the maximizer of:

$$c(\mathbf{V}) = \sum_{i,k,l=1}^{n} |Cum(e_i, e_i^*, e_k, e_l^*)|^2 \tag{3.74}$$

which is equivalent to minimizing the squared joint cumulant with distinct first and second indices. Such criteria can be used in substitution to Equation (3.73) due to an efficient optimization strategy using joint diagonalization. Moreover, Cardoso and Souloumiac (1993) propose that $c(\mathbf{V}) = C(\mathcal{N}^p, \mathbf{V})$ (the same as Equation (3.57)) for any unitary matrix $\mathbf{V}$, meaning that a maximiser of $c(\mathbf{V})$ is a joint diagonalizer of $\mathcal{N}^p$, where $\mathcal{N}^p$ is a K-sized parallel set of matrices. They also propose that, under the JADE prior assumptions, a matrix that jointly diagonalizes $\mathcal{N}^p$ is essentially equal to $\mathbf{U}$. The joint diagonalization of $\mathcal{N}^p$ can be achieved using an extension of the Jacobi technique (CARDOSO; SOULOUMIAC, 1993; GOLUB *et al.*, 1989).

The fourth proposition states that for any d-dimensional complex random vector $\mathbf{v}$ with fourth-order cumulant, there exist d^2 scalars $\lambda_1, \ldots, \lambda_{d^2}$ and d^2 matrices $\mathbf{M}_1, \ldots, \mathbf{M}_{d^2}$, called eigenmatrices, such that:

$$\mathbf{Q}_{\mathbf{v}}(\mathbf{M}_r) = \lambda_r \mathbf{M}_r \tag{3.75}$$

$$Tr(\mathbf{M}_r \mathbf{M}_s^H) = \delta_{r,s} \qquad 1 \leq r, s \leq d^2 \tag{3.76}$$

with $\delta_{r,s} = 1$, if $r = s$, and 0 otherwise. It is possible to represent the relationship $\mathbf{N} = \mathbf{Q}_{\mathbf{v}}(\mathbf{M})$ in vector-matrix form as $\hat{\mathbf{N}} = \hat{\mathbf{Q}}\hat{\mathbf{M}}$, where $\hat{\mathbf{N}}$ and $\hat{\mathbf{M}}$ are created by stacking the $d \times d$ matrices into $d^2 \times 1$ vectors and by mapping $\mathcal{L}_v$ into the $d^2 \times d^2$ matrix $\hat{\mathbf{Q}}$. One way to do this is by defining

$$\hat{n}_a = n_{i,j} \tag{3.77}$$

$$\hat{m}_a = m_{i,j} \tag{3.78}$$

$$\hat{q}_{a,b} = Cum(v_i, v_j^*, v_l, v_k^*) \tag{3.79}$$

with

$$1 \leq a, b \leq d^2$$
$$1 \leq i, j \leq d$$
$$a = i + (j-1)d$$
$$b = k + (l-1)d$$

In Equations (3.77) to (3.79), $\hat{n}_a$ and $\hat{m}_a$ are the a-th value from $\hat{\mathbf{N}}$ and $\hat{\mathbf{M}}$ vectors, respectively, $n_{i,j}$ and $m_{i,j}$ are the values at the i-th column and j-th column of the $\mathbf{N}$ and $\mathbf{M}$ matrices, respectively, and $\hat{q}_{a,b}$ is the value at the a-th row and b-th column of $\hat{\mathbf{Q}}$.

Matrix $\hat{\mathbf{Q}}$ is Hermitian, so it has d^2 eigenvalues and d^2 eigenvectors of size d^2, which can be unstacked to create the eigenmatrices that inherit the orthonormality property from the eigenvectors. The eigenstructure of $\mathcal{L}_z$ is derived from the definition of Q_z in Equation (3.71). The spectrum of $\hat{\mathbf{Q}}$ consists of $n(n-1)$ zero eigenvalues and n non-zero eigenvalues, with magnitude equal to the kurtosis of the sources (CARDOSO; SOULOUMIAC, 1993). Ordering those eigenvalues λ_r by decreasing order of magnitude allow us to define the eigenset of $\mathcal{L}_z$ as:

$$\mathcal{N}^e \equiv \{\lambda_r \mathbf{M}_r | 1 \leq r \leq n\} \tag{3.80}$$

which contains only n non-zero eigenmatrices (with the non-zero eigenvalues λ_r) instead of the n^2 that are in $\mathcal{N}^p$. For the purposes of this method, the set $\mathcal{N}^e$ exhausts the relevant fourth-order information, and Cardoso and Souloumiac (1993) propose that $c(\mathbf{V}) = C(\mathcal{N}^p, \mathbf{V}) = C(\mathcal{N}^e, \mathbf{V})$ for any unitary matrix $\mathbf{V}$. This reduces the number of matrices that need to be jointly diagonalized from n^2 to n.

The JADE algorithm can be summarized by:

- Step 1. Calculate the estimate of $\mathbf{R}_x$ and the whitening matrix $\mathbf{B}$.
- Step 2. Calculate the fourth-order cumulant matrix $\mathcal{L}_z$, and compute the eigenvalues and eigenmatrices $\{\lambda_r, \mathbf{M}_r || 1 \leq r \leq n\}$.
- Step 3. Jointly diagonalize $\mathcal{N}^e = \{\lambda_r \mathbf{M}_r || 1 \leq r \leq n\}$ using an unitary matrix $\mathbf{U}$.
- Step 4. Estimate $\mathbf{A} = \mathbf{B}^{\#}\mathbf{U}$, where # denotes the Moore-Penrose inverse.

Practical and illustrative methods for performing each step are presented in Cardoso and Souloumiac (1993).

JADE is considered one of the classical ICA algorithms (WU *et al.*, 2020) for being one of the first methods that considered higher-order cumulants in its calculations, and because it uses a joint diagonalization strategy, such as SOBI, being more computationally efficient compared to other ICA variants (KACHENOURA *et al.*, 2008). There is literature on it outperforming SOBI and even Infomax in the MI-BCI literature (URIBE *et al.*, 2016).

3.6 Preconditioned ICA for real data

The likelihood function for the ICA model can be written in terms of the unknown $N_s \times N_s$ mixing matrix $\mathbf{A}$ and p_i denoting the p.d.f. of the i-th source signal. Assuming zero-mean, independent and identically distributed sources, we can write the likelihood of $\mathbf{A}$ as:

$$p(\mathbf{X}|\mathbf{A}) = \prod_{t=1}^{T} \frac{1}{|det(\mathbf{A})|} \prod_{i=1}^{N} p_i([\mathbf{A}^{-1}\mathbf{x}]_i(t)) \tag{3.81}$$

where $\mathbf{X}$ is the $N_x \times T$ observed mixed signal matrix, with N_x channels and T time samples for each channel. It is often useful to work with the time averaged negative log-likelihood parameterized by $\mathbf{W} = \mathbf{A}^{-1}$, written as $\mathcal{L}(\mathbf{W}) = -\frac{1}{T} log\left(p(\mathbf{X}|\mathbf{W}^{-1})\right)$. In the case of the ICA model, we have:

$$\mathcal{L}(\mathbf{W}) = -log|det(\mathbf{W})| - \hat{E}\left[\sum_{i=1}^{N} log(p_i(y_i(t)))\right] \tag{3.82}$$

where $\hat{E}$ denotes the empirical mean and with $\mathbf{Y} = \mathbf{WX} = [\mathbf{y}_1, ..., \mathbf{y}_{N_s}]^T$, and $\mathbf{y}_i = [y_i(0), \dots, y_i(T-1)]^T$. The sources and observed signals representations $\mathbf{X}$ and $\mathbf{Y}$ now contain the T realizations of $\mathbf{x}(t)$ and $\mathbf{y}(t)$, so the ICA model representation is slightly changed to

$$\mathbf{Y} = \mathbf{WX}$$

which only serves the purpose of accommodating the additional time dimension, but the underlying physical interpretation is the same.

The Picard algorithm approach is based on learning the curvature of the manifold that $\mathcal{L}(\mathbf{W})$ describes using the available data, making the convergence of the algorithm faster since it will rely more on the data itself making it less prone to optimization problems in cases where the ICA model does not hold exactly. Picard uses the standard Infomax source density model, i.e., $-log(p_i(u)) = 2log(cosh(\frac{u}{2})) + C$, where C is a constant.

To understand the local topology of $\mathcal{L}(W)$ and its changes for small variations $\boldsymbol{\mathcal{E}}$ in $\mathbf{W}$, we will use its Taylor expansion, calculated as:

$$\mathcal{L}((\mathbf{I} + \boldsymbol{\mathcal{E}})\mathbf{W}) = L(\mathbf{W}) + \langle \mathbf{G} | \boldsymbol{\mathcal{E}} \rangle + \frac{1}{2} \langle \boldsymbol{\mathcal{E}} | \mathbf{H} | \boldsymbol{\mathcal{E}} \rangle + \mathcal{O}(||\boldsymbol{\mathcal{E}}||^3) \tag{3.83}$$

The $N_s \times N_s$ matrix $\mathbf{G}$, known as the relative gradient, controls the first-order term while the $N_s \times N_s \times N_s \times N_s$ tensor $\mathbf{H}$, known as the relative Hessian, controls the second-order term. The operation $\langle \mathbf{M} | \mathbf{M}' \rangle = Tr(\mathbf{M}^T \mathbf{M}')$ represents the Frobenius matrix scalar product, where $\mathbf{M}$ and $\mathbf{M}'$ are $N \times N$ matrices. Also, the operation $\langle \mathbf{H} | \mathbf{M} \rangle$, were $\mathbf{H}$ is a $N \times N \times N \times N$ tensor, results in a $N \times N$ matrix $\mathbf{HM}$, with its (i, j)-th value calculated as $[\mathbf{HM}]_{i,j} = \sum_{k,l=1}^{N} \mathbf{H}_{i,j,k,l} \mathbf{M}_{k,l}$. Finally, we have the equality $\langle \mathbf{M}' | \mathbf{H} | \mathbf{M} \rangle = \langle \mathbf{M}' | \mathbf{HM} \rangle$.

Both terms can be obtained by calculating the second-order expansions of $log(|det(\mathbf{I} + \boldsymbol{\mathcal{E}})|)$ and $log(p_i(u + \epsilon))$, resulting in:

$$G(\mathbf{Y}) = \frac{1}{T} \psi(\mathbf{Y}) \mathbf{Y}^T - \mathbf{I} \tag{3.84}$$

$$H_{i,j,k,l} = \delta_{i,l} \delta_{j,k} + \delta_{i,k} \hat{E}[\psi'(y_i) y_j y_l] \tag{3.85}$$

where $\psi(u) = \psi_i(u) = -\frac{p_i'(u)}{p_i(u)}$ (as in Equation (3.16), used in the Infomax method) is the scoring function, $\hat{E}$ is the empirical mean and N is the number of samples. In practice ψ_i is a chosen fixed non-linearity, typically $\psi_i(u) = tanh(u)$.

Using the notations:

$$\begin{aligned} \hat{h}_{i,j,l} &= \hat{E}[\psi_i'(y_i)y_j y_l] \\ \hat{h}_{i,j} &= \hat{E}[\psi_i'(y_i)y_j^2] \\ \hat{h}_i &= \hat{E}[\psi_i'(y_i)] \\ \sigma_i^2 &= \hat{E}[y_i^2] \end{aligned} \tag{3.86}$$

for $1 \leq i, j, l \leq N_s$, we can use two approximations for $H_{i,j,k,l}$. The first approximation, denoted as $\tilde{H}^2_{i,j,k,l}$, is obtained by replacing $\hat{h}_{i,j,l}$ by $\delta_{j,l}\hat{h}_{i,j}$ and is calculated with

$$\tilde{H}^2_{i,j,k,l} = \delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}\hat{h}_{i,j} \tag{3.87}$$

The second approximation, denoted as $\tilde{H}^1_{i,j,k,l}$, extends the first one by also substituting $\hat{h}_{i,j}$ by $\hat{h}_i\hat{\sigma}_j^2$, as follows:

$$\tilde{H}^1_{i,j,k,l} = \begin{cases} \delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}\hat{h}_i\sigma_j^2 & \text{if } i \neq j \\ 1 + \hat{h}_{ii} & \text{for } i = j = k = l \end{cases} \tag{3.88}$$

The superscripts indicate the computational complexity of calculating each approximation, as $H^2_{i,j,k,l}$ scales quadratically with N_s while $H^1_{i,j,k,l}$ scales linearly with N_s. Furthermore, both approximations have a block-diagonal sparse structure that helps in the calculation of the inverse. It can be shown that for $a_{i,j} = \tilde{H}_{i,j,i,j}$ and $i \neq j$

$$[\tilde{H}^{-1}\mathbf{G}]_{i,j} = \frac{a_{j,i}\mathbf{G}_{i,j} - \mathbf{G}_{j,i}}{a_{i,j}a_{j,i} - 1} \tag{3.89}$$

.

For independent signals and in the limit of $t \to \infty$ (infinite data samples), we have the identity $\hat{h}_{i,j,l} = \delta_{j,l}\hat{h}_{i,j} = \delta_{j,l}\hat{h}_i\delta_j^2$, meaning that the two approximations will be equivalent for independent signals. This implies that if the algorithm converges, the Hessian approximation gets very close to the true relative Hessian. If the algorithm is far from convergence, the approximation is not accurate, so it will only be used as a preconditioner for the algorithm.

For the following steps, the Hessian approximations must be positive semi-definite. Since the approximations may not follow this property, a regularization procedure is employed. Since the approximations have a block diagonal structure, the regularization procedure employed by Picard involves shifting the eigenvalue spectrum of the blocks so that their smallest eigenvalue is λ_{min}, using the following algorithm:

- For all (i, j) blocks of $\tilde{\mathbf{H}}$, calculate the smallest eigenvalue with $\lambda_{i,j} = \frac{1}{2}(a_{i,j} + a_{j,i} - \sqrt{(a_{i,j} - a_{j,i})^2 + 4})$, where $a_{i,j} = \tilde{\mathbf{H}}^1_{i,j,i,j}$ or $a_{i,j} = \tilde{\mathbf{H}}^2_{i,j,i,j}$.
- If $\lambda_{i,j} < \lambda_{min}$, we add $(\lambda_{min} - \lambda_{i,j}\mathbf{I})$ to the (i, j) block.

The L-BFGS (LIU; NOCEDAL, 1989) algorithm is a widely used method for unconstrained optimization, due to its flexibility in addressing a variety of problems and its efficient memory usage. It uses an easily invertible approximation of the Hessian, refining it at each iteration, while its inverse is calculated using a recursive algorithm. The main idea of Picard involves using either $\tilde{\mathbf{H}}^1$ or $\tilde{\mathbf{H}}^2$ as the approximation of the Hessian matrix, rather than starting from scratch with the identity matrix. Algorithm 2 shows the L-BFGS algorithm in Picard, while Algorithm 3 shows the recursive calculation of the search direction ($\tilde{\mathbf{H}}^{-1}\mathbf{G}$), with $\gamma_i = G_i - G_{i-1}$, $\xi_i = a_i p_i$ and $\rho_i = \frac{1}{\langle \xi_i | \gamma_i \rangle}$. These algorithms leverage the Hessian approximations and search directions from up to m past iterations.

Algorithm 2 Preconditioned L-BFGS for Picard

```
Require: Mixed signals $\mathbf{X}$, whitening matrix $\mathbf{W}_0$, memory size $m$ (for L-BFGS), number
   of iterations $N_{iter}$
   for $k = 0, \ldots, N_{iter}$ do
       $\mathbf{Y} \leftarrow \mathbf{W}_k\mathbf{X}$
       Calculate $\mathbf{G}_k$ using Equation (3.84)
       Calculate $\tilde{H}_k$ using Equation (3.87) or (3.88).
       Regularize $\tilde{H}_k$ using the regularization procedure (shifting of the eigenspectrum).
       Compute the search direction $p_k \leftarrow -(\tilde{\mathbf{H}}_k^m)^{-1}\mathbf{G}_k$ using Algorithm 3
       Compute the step size $\alpha_k$ using a line search.
       $\mathbf{W}_{k+1} \leftarrow (\mathbf{I} + \alpha_k p_k)\mathbf{W}_k$
   end for
   Output: $\mathbf{Y}$, $\mathbf{W}$
```

L-BFGS works by using both gradient and curvature (the Hessian) information to determine the optimal direction p_k for updating the candidate unmixing matrix $\mathbf{W}_k$. The step size α_k in that direction is determined using a line search algorithm. The search algorithm proposed by Moré and Thuente (1994) is known for its efficiency, but a simpler strategy called backtracking can also be employed. Backtracking involves evaluating the function in the search direction using $\alpha_k = 1$. If the objective function does not decrease,

Algorithm 3 Direction calculation for L-BFGS

```
Require: Current gradient $\mathbf{G}_k$, Hessian approximation $\tilde{\mathbf{H}}_k$, previous values of $\xi_i$, $\gamma_i$, $p_i$
  $\forall i \in \{k-m, \dots, k-1\}$
  $q \leftarrow -\mathbf{G}_k$
  for $i = k-1, \dots, k-m$ do
      $a_i \leftarrow \rho_i \langle \xi_i | q \rangle$
      $q \leftarrow q - a_i \gamma_i$
  end for
  $r \leftarrow \tilde{H}_k^{-1} q$
  for $i = k-m, \dots, k-1$ do
      $\beta \leftarrow \rho_i \langle \gamma_i | r \rangle$
      $r \leftarrow r + \xi_i (a_i - \beta)$
  end for
  $p_k \leftarrow r$
  Output: $p_k$
```

α_k is halved and the process is repeated; if the function does decrease, α_k is retained. Backtracking terminates when α_k becomes too small, indicating minimal weight updates and a potential saddle point. After a certain number of failed backtracking attempts, the current descent direction is considered inefficient and the memory is reset so L-BFGS will use the current relative gradient.

3.6.1 Orthogonal Picard

Many ICA methods also enforce decorrelation of the unmixed signals, such as Fast ICA. This means that $\mathbf{W}$ assumes the form

$$\mathbf{W} = \mathbf{O}\mathbf{W}_0 \tag{3.90}$$

where $\mathbf{W}_0$ is the whitening matrix given by:

$$\mathbf{W}_0 = \left(\frac{1}{N}\mathbf{X}\mathbf{X}^T\right)^{\frac{1}{2}} \tag{3.91}$$

and $\mathbf{O}$ is an orthogonal matrix ($\mathbf{O}\mathbf{O}^T = \mathbf{I}$), which guarantees that the recovered sources $\mathbf{Y} = \mathbf{W}\mathbf{X}$ are orthogonal. Picard-O is an extension of Picard that returns an unmixing matrix as in Equation (3.90), claiming to be faster than Fast ICA as it is designed to deal with real data that often does not accurately follow the ICA model.

The goal is equivalent to minimizing $\mathcal{L}(\mathbf{O}\mathbf{W}_0)$ (from Equation (3.82)) w.r.t. $\mathbf{O}$, and can be achieved by using the iteration $\mathbf{W}_k = \mathbf{O}_k\mathbf{W}_0$, and updating $\mathbf{O}_k$ to another orthogonal matrix $\mathbf{O}_{k+1}$. The matrix $\mathbf{O}_{k+1}$ is parameterized as $\mathbf{O}_{k+1} = e^{\boldsymbol{\mathcal{E}}}\mathbf{O}_k$, where $\boldsymbol{\mathcal{E}}$ is an $N_s \times N_s$ skew-symmetric matrix ($\boldsymbol{\mathcal{E}}^T = -\boldsymbol{\mathcal{E}}$). This approach follows results from differential geometry.

As in the original Picard, we use the second-order Taylor expansion of the likelihood function:

$$\mathcal{L}(e^{\boldsymbol{\mathcal{E}}}\mathbf{W}) = L(\mathbf{W}) + \langle \mathbf{G}|\boldsymbol{\mathcal{E}}\rangle + \frac{1}{2}\langle \boldsymbol{\mathcal{E}}|\mathbf{H}|\boldsymbol{\mathcal{E}}\rangle + \mathcal{O}(||\boldsymbol{\mathcal{E}}||^3) \tag{3.92}$$

Here, the $N_s \times N_s$ relative gradient matrix $\mathbf{G}$ and the $N_s \times N_s \times N_s \times N_s$ relative Hessian tensor $\mathbf{H}$ are calculated using the expansion of $e^{\boldsymbol{\mathcal{E}}} \approx \mathbf{I} + \boldsymbol{\mathcal{E}} + \frac{1}{2}\boldsymbol{\mathcal{E}}^2$, resulting in the following element-wise calculations:

$$G_{i,j} = \hat{E}[\psi_i(y_i)y_j] - \delta_{i,j} \tag{3.93}$$

$$H_{i,j,k,l} = \delta_{i,l}\delta_{j,k}\hat{E}[\psi_i(y_i)y_i] + \delta_{i,k}\hat{E}[\psi_i'(y_i)y_jy_l] \tag{3.94}$$

where $\psi_i = -\frac{p_i'}{p_i}$ is the score function.

Again, computing the Hessian is costly, so we use the approximation $\hat{E}[\psi_i'(y_i)y_jy_l] \approx \delta_{j,l}\hat{E}[\psi_i'(y_i)]\hat{E}(y_i^2)$ by assuming a large sample size and independent signals $\{y_1, \ldots, y_{N_s}\}$, for $i \neq j$. The approximate relative Hessian takes the form:

$$\tilde{H}_{i,j,k,l} = \delta_{il}\delta_{jk}\hat{E}[\psi_i(y_i)y_i] + \delta_{ik}\delta_{jl}\hat{E}[\psi_i'(y_i)] \tag{3.95}$$

considering $\hat{E}[y_i^2] = 1$ for $i \neq j$. By substituting this approximation into Equation (3.92) and performing some arithmetic manipulation, the sum of first-order and second-order terms can be expressed as (ABLIN *et al.*, 2018):

$$\langle \mathbf{G}|\boldsymbol{\mathcal{E}}\rangle + \frac{1}{2}\langle \boldsymbol{\mathcal{E}}|\mathbf{H}|\boldsymbol{\mathcal{E}}\rangle = \sum_{i<j}(G_{i,j} - G_{j,i})\mathcal{E}_{i,j} + \frac{\kappa_i + \kappa_j}{2}\mathcal{E}_{i,j}^2 \tag{3.96}$$

where $\kappa_i = \hat{E}[\psi_i(y_i)y_i] - \hat{E}[\psi_i'(y_i)]$, $G_{i,j}$ and $\mathcal{E}_{i,j}$ are the elements of matrices $\mathbf{G}$ and $\boldsymbol{\mathcal{E}}$, respectively, at their i-th row and j-th column.

It can be shown that if $\kappa_i + \kappa_j > 0$, the value of $\mathcal{E}_{i,j}$ that minimizes Equation (3.96) is given by $\mathcal{E}_{i,j} = -\frac{G_{i,j} - G_{j,i}}{\kappa_i + \kappa_j}$. This leads to the following update rule:

$$\mathbf{W}_{k+1} = e^{\mathbf{D}}\mathbf{W}_k \tag{3.97}$$

with,

$$D_{i,j} = -\frac{2}{\kappa_i + \kappa_j} \frac{G_{i,j} - G_{j,i}}{2} \tag{3.98}$$

where $D_{i,j}$ is the element at the i-row and j-th column of $\mathbf{D}$. In order to enforce $\kappa_i + \kappa_j > 0$, Picard-O uses a non-linearity switching strategy. At each iteration, $\psi_i \leftarrow sign(\kappa_i)\psi$, and when the sign of a source changes, the L-BFGS memory is flushed. Very small values of $\kappa_i + \kappa_j > 0$ may cause numerical instability, so in practice, we use $min(\frac{\kappa_i + \kappa_j}{2}, \kappa_{min})$, where κ_{min} is typically 10^{-2} (ABLIN *et al.*, 2018). Finally, during the L-BFGS iterations, the same backtracking strategy for the line search is used to find the best step size α_k. The algorithm stops when the gradient norm $||\mathbf{G} - \mathbf{G}^T||$ becomes smaller than a small constant threshold ϵ.

All steps of Picard-O are represented in Algorithm 4 and Algorithm 5, with $\mathbf{\Delta}_k = \frac{\mathbf{G}_k - \mathbf{G}_k^T}{2} - \frac{\mathbf{G}_{k-1} - \mathbf{G}_{k-1}^T}{2}$ and $\rho_k = \langle \boldsymbol{\mathcal{E}}_k | \mathbf{\Delta}_k \rangle$. Additionally, although $e^{\boldsymbol{\mathcal{E}}} = \sum_{k=0}^{\infty} \frac{1}{k!} \boldsymbol{\mathcal{E}}^k$, it is approximated up to the second-order term of the expansion. Algorithm 4 describes the adapted L-BFGS algorithm for Picard-O, while Algorithm 5 is used to find the search direction during the optimization.

Algorithm 4 Preconditioned L-BFGS for Picard-O

```
Require: Mixed signals $\mathbf{X}$, memory size $m$ (for L-BFGS), number of iterations $N_{iter}$
  Calculate $\mathbf{W}_0$ with Equation (3.91)
  $Y \leftarrow \mathbf{W}_0\mathbf{X}$
  for $k = 0, \ldots, N_{iter}$ do
      Compute $sign(k_i)$ and flush memory if a signal has changed
      Calculate $\mathbf{G}_k$ using Equation (3.93)
      Find the search direction using Algorithm 5
      Compute the step size $\alpha_k$ using a line search (backtracking method).
      $\mathbf{W}_{k+1} \leftarrow e^{\alpha_k \mathbf{D}_k}\mathbf{W}_k$
      $\mathbf{Y} \leftarrow \mathbf{W}_{k+1}\mathbf{X}$
  end for
  Output: $\mathbf{Y}$, $\mathbf{W}_k$
```

Frank *et al.* (2022) have shown that Picard and Picard-O can be computationally faster than Infomax ICA, while being competitive in terms of reduction of mutual information, making them interesting in scenarios where computation speed is important.

3.7 Online Recursive Independent Component Analysis

Online Recursive Independent Component Analysis (ORICA) is an online ICA method for source separation. It is based on the natural gradient of the Infomax criterion

Algorithm 5 Direction calculation for L-BFGS

```
Require: Current gradient $G_k$, $\kappa_i$, previous values of $\boldsymbol{\mathcal{E}}_l$, $\boldsymbol{\Delta}_l$, $\rho_l$ $\forall l \in \{k-m,\dots,k-1\}$
  $\mathbf{Q} \leftarrow \frac{-(\mathbf{G}_k - \mathbf{G}_k^T)}{2}$
  for $l = k-1,\dots,k-m$ do
      $a_l \leftarrow \rho_l \langle \boldsymbol{\mathcal{E}}_l | \mathbf{Q} \rangle$
      $\mathbf{Q} \leftarrow \mathbf{Q} - a_l \boldsymbol{\Delta}_l$
  end for
  Compute $\mathbf{D}$ with $D_{i,j} = \frac{Q_{i,j}}{max\left(\frac{\kappa_i+\kappa_j}{2}, \kappa_{min}\right)}$
  for $l = k-m,\dots,k-1$ do
      $\beta \leftarrow \rho_l \langle \boldsymbol{\Delta}_l | \mathbf{D} \rangle$
      $\mathbf{D} \leftarrow \mathbf{D} + \boldsymbol{\mathcal{E}}_l (a_i - \beta)$
  end for
  $p_k \leftarrow r$
  Output: Search direction $\mathbf{D}$
```

and has the advantage of adapting to nonstationarity. Online means that the unmixing matrix is iteratively updated during the signal separation phase, which differs from most ICA algorithms where there is a phase for computing the unmixing matrix, which is used afterward. Instead of buffering all the data that the system has ever received to recompute the unmixing matrix from time to time, ORICA uses a Recursive-Least-Squares-like method, where the matrix can be updated at each received sample (or batch of samples).

It is part of the family of incremental algorithms that use the following general weight update:

$$\frac{d\mathbf{W}(t)}{dt} = \mu(t)\left[\boldsymbol{\Lambda} - \boldsymbol{\Gamma}\boldsymbol{\varphi}(\mathbf{y}(t))\boldsymbol{\psi}^T(\mathbf{y}(t))\right]\mathbf{W}(t) \tag{3.99}$$

where $\boldsymbol{\Lambda}$, $\boldsymbol{\Gamma}$ and $\mathbf{W}$ are $N_s \times N_s$ matrices, with $\boldsymbol{\Lambda}$ being positive definite (AKHTAR *et al.*, 2012; CICHOCKI *et al.*, 1994). The functions $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are the element-wise application of a vector of functions, such that $\boldsymbol{\psi}(\mathbf{y}(t)) = [\psi_1(y_1(t)),\dots,\psi_{N_s}(y_{N_s}(t))]^T$ and $\boldsymbol{\varphi}(\mathbf{y}(t)) = [\varphi_1(y_1(t)),\dots,\varphi_{N_s}(y_{N_s}(t))]^T$, with all functions being odd. For instance, the natural gradient update rule of the Infomax ICA shown in Equation (3.17) also falls into this family, with $\boldsymbol{\Lambda} = \boldsymbol{\Gamma} = \mathbf{I}$, $\boldsymbol{\varphi}(\mathbf{y}(t)) = tanh(\mathbf{y}(t))$, $\boldsymbol{\psi}(\mathbf{y}(t)) = \mathbf{y}(t)$, $\mu(t) = \eta$ (constant). The discrete time update rule can then be written as:

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta\left[\mathbf{I} - \boldsymbol{\varphi}(\mathbf{y}(t))\mathbf{y}(t)\right]\mathbf{W}(t) \tag{3.100}$$

In the limit of a small learning rate and iterating over the same data multiple times, the update rule algorithm converges to a fixed-point solution, defined as when the average update $E[\mathbf{W}(t+1) - \mathbf{W}(t)]$ converges to zero. This condition is met when $\mathbf{W} = E[\mathbf{f}(t)\mathbf{x}^T(t)]\mathbf{W}\mathbf{W}^T$, where $\mathbf{f}(t) = \boldsymbol{\varphi}(\mathbf{y}(t))$. Note that a projection of $\mathbf{W}$ on its null space is a valid solution, but not a desired one. Although $\mathbf{f}(t)$ depends on $\mathbf{W}$, we can assume that it is fixed, since a typical $\boldsymbol{\psi}$ will barely be affected by small changes on $\mathbf{W}$. This leads to the full-rank solution:

$$\mathbf{W}^{\#} = \hat{\mathbf{A}} = E[\mathbf{x}(t)\mathbf{f}^T(t)] \tag{3.101}$$

here $\mathbf{W}^{\#}$ represents the Moore-Penrose pseudoinverse of $\mathbf{W}$. The calculation of $\mathbf{W}$ is done by the pseudoinversion of $\hat{\mathbf{A}}$.

In an online scenario, the expected value $\hat{\mathbf{A}}$ will be calculated as a windowed average over recent data. We will use the recursive formula:

$$\hat{\mathbf{A}}(t+1) = \alpha_t \hat{\mathbf{A}}(t) + \beta_t \mathbf{x}(t)\mathbf{f}^T(t) \tag{3.102}$$

for updating the expected value $\hat{\mathbf{A}}$ on iteration t using a sliding windows with exponential decay λ ($\beta_t = 1 - \alpha_t = \lambda$), implying in the assumption of a non-stationary $\mathbf{A}$ (AKHTAR *et al.*, 2012). The non-recursive calculation can also be written as:

$$\hat{\mathbf{A}}(t) = \lambda \sum_{i=1}^{\infty} (1-\lambda)^{n-i} \mathbf{x}(t-i)\mathbf{f}^T(t-i) \tag{3.103}$$

We then use the Sherman-Morrison formula:

$$\left[\mathbf{H} + \mathbf{u}\mathbf{v}^T\right]^{-1} = \mathbf{H}^{-1} - \frac{\mathbf{H}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{H}^{-1}}{1 + \mathbf{v}^T\mathbf{H}^{-1}\mathbf{u}} \tag{3.104}$$

with $\mathbf{v}^T\mathbf{H}^{-1}\mathbf{u} \neq -1$ for the recursive inversion of $\hat{\mathbf{A}}$ without the need for explicit inversion at every sample, reducing its computational complexity.

The next iteration's $\mathbf{W}(t+1)$ can then be written as:

$$\mathbf{W}(t+1) = \frac{1}{1-\lambda_t}\left[\mathbf{W}(t) - \frac{1}{\frac{1-\lambda_t}{\lambda_t} + \lambda_t(\mathbf{f}^T(t)\mathbf{y}(t) - 1)}\mathbf{y}(t)\mathbf{f}^T(t)\mathbf{W}(t)\right] \tag{3.105}$$

which can then be used for the online ICA algorithm (AKHTAR *et al.*, 2012). Note that λ has been substituted by λ_t, meaning that the exponential decay may be changed during the iterations to increase or decrease the weight of past data in the calculation of $\hat{\mathbf{A}}$. As in the Extended Infomax ICA, we can use two nonlinearities for simultaneous extraction of supergaussian and subgaussian sources. The function used for the supergaussian source separation is:

$$\varphi^{+}(y_i) = -2tanh(y_i) \tag{3.106}$$

While the function used for subgaussian source extraction is:

$$\varphi^{-}(y_i) = tanh(y_i) - y_i \tag{3.107}$$

These are similar to the activation functions used in Extended Infomax. Further improvements can be made to this approach to improve ICA convergence (HSU *et al.*, 2014). The RLS whitening algorithm (ZHU, 2004) can be used for online whitening of the observed signals if added as the first step of the pipeline. The algorithm assumes zero mean, unit power, statistically mutually independent sources with at most one normally distributed source, and uses the cost function (CARDOSO; LAHELD, 1996) as follows:

$$J(\mathbf{M}) = \mathrm{tr}(\mathbf{M}\mathbf{R}_{\mathbf{x}}\mathbf{M}^T) - \log(\det(\mathbf{M}\mathbf{R}_{\mathbf{x}}\mathbf{M}^T)) - N_s \tag{3.108}$$

It can be shown that $J(\mathbf{M}) \geq 0$, where the equality holds if and only if $\mathbf{M}\mathbf{R}_{\mathbf{x}}\mathbf{M}^T = \mathbf{I}$, i.e., $\mathbf{M}$ whitens $\mathbf{x}$. To construct the RLS algorithm, we can define the exponentially weighted autocovariance matrix of $\mathbf{x}$ as:

$$\tilde{\mathbf{R}}_{\mathbf{x}} = (1-\lambda)\sum_{i=1}^{t} \lambda^{t-i}\mathbf{x}(i)\mathbf{x}^T(i) \tag{3.109}$$

And substitute it into Equation (3.108), with λ as the forgetting factor. The derivative of $J(\mathbf{M})$ w.r.t. $\mathbf{M}$ can then be calculated as:

$$\nabla J(\mathbf{U}) = 2\mathbf{M}\tilde{\mathbf{R}}_{\mathbf{x}} - 2[\mathbf{M}\tilde{\mathbf{R}}_{\mathbf{x}}\mathbf{M}^T]^{-1}\mathbf{M}\tilde{\mathbf{R}}_{\mathbf{x}} \tag{3.110}$$

If we set $\nabla J(\mathbf{M})$ to 0, we find that the minimum value of J is when $\mathbf{M} = [\tilde{\mathbf{R}}_{\mathbf{x}}\mathbf{M}^T]^{-1}$, giving us the recursive equation $\mathbf{M}(t+1) = [\tilde{\mathbf{R}}_{\mathbf{x}}\mathbf{M}^T(t)]^{-1}$, where $\mathbf{M}(t)$ represents the estimated whitening matrix at the t-th iteration. Substituting Equation (3.109) into it results in:

$$\mathbf{M}(t+1) = [\lambda\mathbf{M}^{-1}(t) + (1-\lambda)\mathbf{x}(t)\mathbf{v}(t)^T]^{-1} \tag{3.111}$$

where $\mathbf{v}(t) = \mathbf{M}(t)\mathbf{x}(t)$. By using the Sherman-Morrison formula, we derive the RLS-like whitening algorithm, as follows:

$$\mathbf{M}(t+1) = \frac{1}{1-\lambda_t}\left[\mathbf{I} - \frac{\mathbf{v}(t)\mathbf{v}^T(t)}{\frac{1-\lambda_t}{\lambda_t} + \mathbf{v}^T(t)\mathbf{v}(t)}\right]\mathbf{M}(t) \tag{3.112}$$

Where we make λ a function of the iteration t, so we can adjust the algorithm's effective window size. It is interesting to note that both source separation and whitening iterative updates have the form of Equation (3.99), showing that the ORICA method can be viewed as a nonlinear online RLS whitening method.

The calculation of λ_t for both $\mathbf{W}$ and $\mathbf{M}$ is heuristic. One possible approach is to use $\lambda_t = \frac{\lambda_0}{n^\gamma}$, where λ_0 is the initial forgetting factor and γ is the decay rate of λ_t.

As in Extended Infomax ICA, we need to set the prior assumption on the number of subgaussian sources N_{sub}. This reflects on the $\boldsymbol{\varphi}$ nonlinearities that will be used, as follows:

$$\varphi_i = \begin{cases} \varphi^- & \text{for } 1 \leq i \leq N_{sub} \\ \varphi^+ & \text{otherwise} \end{cases} \tag{3.113}$$

For high sampling rates, it becomes very costly to perform those computations on every newly recorded sample, so a block-update rule has been proposed (HSU *et al.*, 2014). By assuming that λ_t is small and approximating $\mathbf{y}(t+l)$ as $\mathbf{y}(t)$ for $1 \leq l \leq L$, where L is the block size, we can write the approximate block-rule update as follows:

$$\mathbf{W}(t+L) \approx \left(\prod_{l=t}^{n+L-1} \frac{1}{1-\lambda_l}\right)\left[\mathbf{I} - \sum_{l=t}^{n+L-1} \frac{\mathbf{y}(l)\mathbf{f}^T(l)}{\frac{1-\lambda_l}{\lambda_l} + \mathbf{f}^T(l)\mathbf{y}(l)}\right]\mathbf{W}(t) \tag{3.114}$$

Using big block sizes worsens the approximation accuracy but greatly decreases the computational cost, so L needs to be chosen carefully.

4 Pattern Recognition and Classification

Pattern recognition concerns the use of mathematical algorithms for detecting patterns in data. In the field of machine learning, the starting point is, in general, a labeled data set i.e. a set of pre-collected data representative of the task of interest. The idea is that the algorithm be trained using this dataset, so that, when it is presented with new (non-annotated) data, it is able to detect the general patterns present in the training phase. This is an instance of the approach known as *supervised learning* because the algorithm has access to some sort of desired response or label (BISHOP, 2006). The use of such methods has benefited immensely from the availability, in the last decade, of vast amounts of data, as well as from the remarkable developments in parallel hardware design.

In the context of MI BCI, an essential pattern recognition task is that of classifying a sample of windowed data as one of the intended tasks, assuming that, during the training stage, the algorithm is presented with a representative set of examples from all relevant categories. As presented in Chapter 2, MI tasks engender ERDs and ERSs on the *mu* and *beta* rhythms of the contra-lateral sensory and motor cortices (PFURTSCHELLER; NEUPER, 2001). The nature of those responses reflects how different authors design the feature extraction steps of the BCI, which will have an impact on the performance of the feature translation and classification module.

In MI BCIs, the use of features extracted using the Power Spectral Density (PSD) together with a linear classifier is a well-established option (SAIBENE *et al.*, 2023; BASHASHATI *et al.*, 2007). It is also usual that the feature extraction step be preceded by a Common Spatial Patterns (CSP) step, which creates a smaller subset of channels and amplifies their discrimination power (KHADEMI *et al.*, 2023). LDA is a popular linear classifier option, along with its extensions and variations, such as Quadratic Discriminant Analysis (QDA) and adaptive LDA. SVMs have been used with success and good classification results (KHADEMI *et al.*, 2023; AGGARWAL; CHUGH, 2019; LOTTE *et al.*, 2018). It is important to remark that Random Forest (RF) classifiers are highly non-linear classifiers that are also present in the MI BCI literature (LOTTE *et al.*, 2018), sometimes with higher accuracy than LDA.

The Naive Bayes classifier is a conceptually simple classification algorithm (BISHOP, 2006), but with a fair performance nonetheless; although it is not widely used, it is known for sometimes achieving very good results even when its premises are not strictly respected (ZHANG, 2004). The logistic regression classifier falls into the same case, and, although it is not commonly used for MI, it has been applied with success to other domains that also use EEG signals (SAIBENE *et al.*, 2023; HOSSEINI *et al.*, 2021).

Neural networks form a class of algorithms that are inspired by certain aspects of the functionality of the nervous system. For their great flexibility, they have been applied to numerous classification problems, such as MI. A classical NN is the Multi-layer perceptron (MLP) classifier, which consists of fully connected layers and is endowed with universal approximation capability (LOTTE *et al.*, 2018). More recently, Deep learning (DL) methods are being employed in diverse problems for having the ability to learn the representations and patterns directly on the signal space instead of the feature space. In this context, Convolutional Neural Networks (CNNs) (such as EEGNet) and Recurrent Neural Networks (RNNs) have been applied with success and it has been shown their superiority and practical ability on BCIs (ALTAHERI *et al.*, 2021; LOTTE *et al.*, 2018)

The objective of the classification algorithm in a BCI is to translate the raw, pre-processed, or processed (feature vector) EEG signals into classes, which will be further translated into commands for the digital or physical device. The differences between them lie in their capability to generalize (transfer their knowledge between sessions, or even subjects), their assumptions (of feature distributions, dependence, stationarity), flexibility (linear vs non-linear classifiers, and also the DL approaches (ALTAHERI *et al.*, 2021)), structure, learning capability under a low number of samples, high number of non-discriminant features, complexity (computational cost in during training or inference, and number of learned parameters). In evaluating many classification methods, we intend to compare some of those aspects and ponder how great or poor classification results correlate with those characteristics.

Khan *et al.* (2023) studied the performance of a two-class MI-BCI, where the EEG signals were preprocessed using ICA and they evaluated the use of some classification algorithms like SVM, Naïve Bayes, Logistic Regression, and LDA. The system achieved up to 95.42% for five subjects using Logistic Regression. Mohammadi *et al.* (2022) used the random forest, SVM, and LDA classifiers for calibrating a three-class MI-BCI, in which they concluded that LDA had the highest performance for most of the subjects in all datasets. Sharma *et al.* (2022) compared the use of many classifiers, such as the MLP, SVM, LDA, Logistic Regression, Decision Tree, Random Forest, and Naive Bayes for a three-class Motor imagery application, where the Logistic Regression and the MLP attained accuracies of up to 90%. Tiwari *et al.* (2020) collected a dataset of seven subjects, which performed eight imagination tasks. The logistic regression outperformed other classifiers for only two out of the seven subjects. Wahid and Tafreshi (2021) used, among all classification methods, Random Forest, LDA, and SVM as the classifiers of an ensemble for a two-class MI-BCI, and concluded that the Random Forest had more consistent results than other classifiers, achieving up to 80% accuracy.

4.1 Support Vector Machine

Linear models are paradigmatic in classification in view of their relative simplicity and mathematical tractability. In a D dimensional space, considering that the classifier weight vector is $\mathbf{w} \in \mathbb{R}^D$, and that the bias term is b, the classifier output is:

$$y(\mathbf{v}) = \mathbf{w}^T \phi(\mathbf{v}) + b \tag{4.1}$$

In this formulation, it is considered that an input vector $\mathbf{v} \in \mathbb{R}^{D'}$ is mapped via $\phi(\mathbf{v}) : \mathbb{R}^{D'} \to \mathbb{R}^D$, which denotes a fixed feature-space transformation i.e. a function which is applied to $\mathbf{v}$ before applying the conventional linear model. The dataset is composed of N input vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$, with corresponding target values $\{t_1, \ldots, t_N\}$, where $t_n \in \{-1, 1\}$ and $n = 1, \ldots, N$, representing the two possible classes. In the binary case, we assume that the two classes are linearly separable i.e. that there exist $\mathbf{w}$ and b such that $t_n y(\mathbf{v}_n) > 0$ for any n. This means that, in the feature space, there exists a hyperplane (the decision boundary) capable of perfect separation.

There may be infinite solutions for $\mathbf{w}$ and b that linearly separate the data; a robust solution, in terms of generalization, is to choose the one that maximizes the margin between the decision boundary and any of the samples (which will be the closest to the boundary). The perpendicular distance between the point $\mathbf{v}_n$ and the hyperplane defined by $y(\mathbf{v}_n) = 0$ can be calculated as $\frac{|y(\mathbf{v_n})|}{||\mathbf{w}||}$ (BISHOP, 2006). In the linearly separable case, we know that $t_n y(\mathbf{v}_n) > 0 \; \forall n$. We can then write the equality:

$$\frac{t_n y(\mathbf{v}_n)}{||\mathbf{w}||} = \frac{t_n \left(\mathbf{w}^T \phi(\mathbf{v}_n) + b\right)}{||\mathbf{w}||} \tag{4.2}$$

The margin is defined as the minimum perpendicular distance from the boundary to any vector $\mathbf{v}_n$. So, in order to maximize the margin, we need to solve the following equation:

$$\arg\max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||} \min_n \left[t_n(\mathbf{w}^T \phi(\mathbf{v}_n) + b) \right] \right\} \tag{4.3}$$

This optimization problem can be converted to an equivalent problem that is easier to solve. Firstly, we chose an α_s to re-scale both $\mathbf{w}$ and b such that the margin is equal to 1, satisfying:

$$t_n(\alpha_s \mathbf{w}^T \phi(\mathbf{v}_{n'}) + \alpha_s b) = 1 \tag{4.4}$$

for a $\mathbf{v}_{n'}$ that is the closest to the margin. Let $\mathbf{w} \leftarrow \alpha_s \mathbf{w}$ and $b \leftarrow \alpha_s b$. As a consequence, all vectors $\mathbf{v}_n$ will satisfy:

$$t_n(\mathbf{w}^T \phi(\mathbf{v}_n) + b) \geq 1 \tag{4.5}$$

The data points for which the equality condition holds are called active, while the remainder are said to be inactive. There will be at least one active point, which will be the point with the closest distance to the boundary, and after the margin is maximized there will be at least two active points. So, the optimization problem of Equation (4.3) simplifies to maximizing $\frac{1}{||\mathbf{w}||}$, which is equivalent to minimizing $||\mathbf{w}||^2$, subject to Equation (4.5). This optimization problem is formulated as follows:

$$\arg\min_{\mathbf{w},b} \frac{1}{2} ||\mathbf{w}||^2 \tag{4.6}$$

We can solve this constrained optimization problem using Lagrange multipliers with one multiplier a_n for each constraint in (4.5), given that this problem satisfies the KKT conditions, which is a requirement for using this method (KUHN; TUCKER, 1951). The Lagrangian function for this case is written as:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{n=1}^{N} a_n \left(t_n(\mathbf{w}\phi(\mathbf{v}_n) + b) - 1\right) \tag{4.7}$$

with $\mathbf{a} = \{a_n | n = 1, \ldots, N\}$. Note that we are minimizing w.r.t. $\mathbf{w}$ and b and maximizing w.r.t. $\mathbf{a}$ (hence the minus sign in the Lagrange multiplier term). Calculating the derivatives w.r.t $\mathbf{w}$ and b of the Lagrangian and setting them to zero yields the two following equations:

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{v}_n) \tag{4.8}$$

$$0 = \sum_{n=1}^{N} a_n t_n \tag{4.9}$$

Substituting Equations (4.8) and (4.9) into Equation (4.7) yields the *dual representation* of the maximum margin problem, written as:

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{v}_n, \mathbf{v}_m) \tag{4.10}$$

completely eliminating $\mathbf{w}$ and b from the problem. In this equation $k(\mathbf{v}_i, \mathbf{v}_j) \triangleq \phi(\mathbf{v}_i)^T\phi(\mathbf{v}_j)$, i.e., a inner product in the codomain of ϕ, and this step is known as the *kernel trick* because instead of calculating $\phi(\mathbf{v}_i)$ and $\phi(\mathbf{v}_j)$ we just need to calculate the *kernel function* k between the both. The goal now is to maximize Equation (4.10) w.r.t $\mathbf{a}$, subject to the constraints:

$$a_n \geq 0 \tag{4.11}$$

$$\sum_{n=1}^{N} a_n t_n = 0 \tag{4.12}$$

This characterizes a quadratic programming problem, which can addressed using known algorithms such as the Sequential Minimal Optimization algorithm (PLATT, 1998). Equation (4.8) also enables us to create an equation for classifying new data points, by substituting it in Equation (4.1), which yields the following classification equation:

$$y(\mathbf{v}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{v}, \mathbf{v}_n) + b \tag{4.13}$$

Having found a solution for $\mathbf{a}$, there is the need to find b. The KKT conditions will lead to the following equations:

$$a_n \geq 0 \tag{4.14}$$

$$t_n y(\mathbf{v}_n) - 1 \geq 0 \tag{4.15}$$

$$a_n(t_n y(\mathbf{v}_n) - 1) = 0 \tag{4.16}$$

from which we can conclude that either a_n is 0 or $t_n y(\mathbf{v}_n) = 1$. Note that any data point with $a_n = 0$ will not have any effect on the prediction of new data points by using Eq. (4.13), and for the remaining data points $t_n y(\mathbf{v}_n) = 1$ is true. Those remaining points that are used in the prediction are called the *support vectors*. It is possible to choose any $\mathbf{v}_n$ and substitute it into Eq. (4.13) and calculate b, but generally, it is calculated as the average value of all possible solutions, as follows:

$$b = \frac{1}{|\mathcal{S}|} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{v}_n, \mathbf{v}_m) \right) \tag{4.17}$$

where $\mathcal{S}$ represents the set of indexes of the support vectors (note that the term inside the outermost sum is the solution to b by substituting $\mathbf{v}_n$ into Eq. (4.13)).

In real data, examples from different classes are often not linearly separable, which will ultimately make this solution fail. It is possible to make a small modification to the problem to deal with this, by allowing data points to be on the "wrong side" of the decision boundary margin with the introduction of "slack variables" ξ_n, one for each $\mathbf{v}_n$. Those variables represent a penalty to the optimization objective, and their values are proportional to their distance to the margin boundary, so points that are correctly classified will have $\xi_n = 0$ and all other points will have $\xi_n = |t_n - y(\mathbf{v}_n)|$. A data point in the decision boundary will have $\xi_n = 1$. Equation (4.5) is then replaced by:

$$t_n(\mathbf{w}^T \phi(\mathbf{v}_n) + b) \geq 1 - \xi_n \tag{4.18}$$

It can be seen that points that are correctly classified but are inside the margin will have $0 \leq \xi_n \leq 1$, and data points that are incorrectly classified will have $\xi \geq 1$. Therefore, the goal is to maximize the margin while penalizing points that lie on the wrong side of the margin boundary, leading to the minimization problem:

$$\arg\min_{\mathbf{w},b} C\sum_{n=1}^{N}\xi_n + \frac{1}{2}||\mathbf{w}||^2 \tag{4.19}$$

where $C > 0$ controls the penalty weight. Since $\xi_n > 1$ for misclassified points, $\sum_{n=1}^{N}\xi_n$ is the upper bound to the number of misclassified points, and for $C \to \infty$ the problem degenerates to the case where we assumed linear separability.

Again, the minimization goal in Equation (4.19), subject to Equation (4.18) can be solved using Lagrange multipliers, with the Lagrangian function given by (BISHOP, 2006):

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n\left\{t_n(\mathbf{w}\phi(\mathbf{v}_n) + b) - 1 + \xi_n\right\} - \sum_{n=1}^{N}\mu_n\xi_n \tag{4.20}$$

where $\{a_n \geq 0 | n = 1, \dots, N\}$ and $\{\mu_n \geq 0 | n = 1, \dots, N\}$ are the the Lagrange multipliers. The KKT conditions are given by the following equations:

$$\begin{aligned}
\mu_n &\geq 0 && (4.21)\\
\xi_n &\geq 0 && (4.22)\\
\mu_n\xi_n &= 0 && (4.23)\\
a_n &\geq 0 && (4.24)\\
t_n(\mathbf{w}\phi(\mathbf{v}_n) + b) - 1 + \xi_n &\geq 0 && (4.25)\\
a_n\left(t_n(\mathbf{w}\phi(\mathbf{v}_n) + b) - 1 + \xi_n\right) &= 0 && (4.26)\\
\forall n &= 1, \dots, N
\end{aligned}$$

Similarly to the process done to obtain Equation (4.10), by deriving Equation (4.20) w.r.t $\mathbf{w}$, b and $\{\xi_n \geq 0 | n = 1, \dots, N\}$ and setting the value to 0, we can arrive at a new dual representation of the Lagrangian in the form of:

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{v}_n, \mathbf{v}_m) \tag{4.27}$$

which is equal to Eq. (4.10) but with an additional constraint (from the KKT conditions):

$$a_n = C - \mu_n \tag{4.28}$$

along with the following two constraints:

$$0 \leq a_n \leq C \tag{4.29}$$

$$\sum_{n=1}^{N} a_n t_n = 0 \tag{4.30}$$

Since $\mu \geq 0$, the new optimization problem is to minimize Eq. (4.27) w.r.t $\{a_n \geq 0|n = 1, \ldots, N\}$, subject to Equations (4.29) and (4.30), resulting in a quadratic programming problem.

By substituting Equation (4.8) into Equation (4.1), we get:

$$y(\mathbf{v}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{v}, \mathbf{v_n}) + b \tag{4.31}$$

which is used to classify new data points as a function of each a_n and each support vector.

Note that $a_n < C$ implies $\mu > 0$ and $\xi_n = 0$, so the corresponding data points lie on the margin. On the other hand, $a_n = C$ implies $\mu_n = 0$, so from Equation (4.23) ξ can be either be less or equal than one, for correctly classified points, or greater than one, for misclassified points.

Figure 4.1 shows a linear SVM decision boundary (full line, defined by $\mathbf{w}_1$), the margins (dotted lines) and the data as colored points, where each color represents each class. In this example the vector $\mathbf{v}$ has only two dimensions, represented by $\mathbf{v}[1]$ and $\mathbf{v}[2]$ as the plot axes. The support vectors are marked with black borders. Note that data points located on the correct side of the classification boundary have $\xi_n = 0$, and points crossing the margin have $\xi_n > 0$.

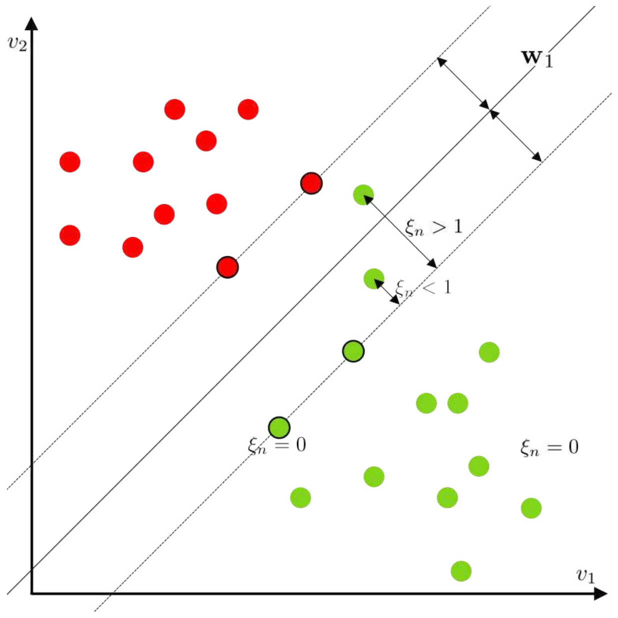


Figure 4.1 – SVM decision boundary.

4.1.1 Kernels

In SVM, kernels are functions that represent generalized inner products. Their importance comes from the so-called *kernel trick* - a kernel function can be used to calculate inner products within a feature (mapped) space without the explicit need to perform the underlying mapping.

There is no need to explicitly know the ϕ mapping or the space in which the kernel describes an inner product, but we need to guarantee that there exists at least one. One necessary and sufficient condition is that the Gram matrix $\mathbf{K}$, whose elements are $k(\mathbf{v}_i, \mathbf{v}_j)$, must be positive semi-definite for all possible choices of $\mathbf{v}_i$ and $\mathbf{v}_j$ in the dataset. Another way is to verify whether k satisfies the Mercer's Theorem (THEODORIDIS; KOUTROUMBAS, 2006):

Theorem 4.1 *Mercer's Theorem. Let* $\mathbf{v} \in \mathbb{R}^{D'}$ *and a mapping* ϕ *with* $\phi(\mathbf{v}) \in \mathbb{H}$*, where* $\mathbb{H}$ *is a Hilbert space. Then, the inner product operation* $\langle \phi(\mathbf{v}), \phi(\mathbf{v}') \rangle = k(\mathbf{v}, \mathbf{v}')$ *where* $\langle \cdot, \cdot \rangle$ *denotes the inner product in* $\mathbb{H}$ *and* $k(\mathbf{v}, \mathbf{v}')$ *is symmetric, continuous and satisfies Equation* (4.32) *where* $g(\mathbf{v})$ *is any function that satisfies Equation* (4.33) *and* C *is a compact subset of* $\mathbb{R}^{D'}$*. The opposite is true: any symmetric, continuous function* $k(\mathbf{v}, \mathbf{v}')$ *that satisfies Equation* (4.32) *and Equation* (4.33) *defines the inner product of a space. Such functions are known as kernels and those spaces are known as Reproducing Kernel Hilbert spaces (RKHS).*

$$\int_C \int_C k(\mathbf{v}, \mathbf{v}')g(\mathbf{v})g(\mathbf{v}')d\mathbf{v}d\mathbf{v}' \geq 0 \tag{4.32}$$

$$\int_C g(\mathbf{v})^2 d\mathbf{v} < +\infty \tag{4.33}$$

For a more detailed discussion about the properties and the applicability of different kernels, see (SHAWE-TAYLOR; CRISTIANINI, 2004) and (HERBRICH, 2001). There are well-known kernel functions that satisfy these properties:

1. **Polynomial**: In Equation (4.34), γ is a scalar multiplier, c_0 is a constant offset, and q is the degree of the kernel. This kernel defines a polynomial decision rule for the SVM.

$$k(\mathbf{v}, \mathbf{v}') = (\gamma \mathbf{v}^T \mathbf{v}' + c_0)^q \qquad q > 0 \tag{4.34}$$

2. **Linear**, which is a particular case of the polynomial kernel when $\gamma = 1$, $c_0 = 0$ and $q = 1$, as shown in Equation (4.35). This kernel defines linear decision rules and boundaries.

$$k(\mathbf{v}, \mathbf{v}') = \mathbf{v}^T \mathbf{v}' \tag{4.35}$$

3. **Radial Basis Function (RBF)**, also known as Gaussian kernel, but here it does not represent a probability density. This kernel is special because the resulting transformation depends only on the euclidean distance between $\mathbf{v}$ and $\mathbf{v}'$. As shown by:

$$k(\mathbf{v}, \mathbf{v}') = exp\left(-\gamma ||\mathbf{v} - \mathbf{v}'||^2\right) \tag{4.36}$$

4. **Sigmoidal**: Shown in Equation (4.37), γ is a scalar multiplier known as the slope and c_0 is the kernel's intercept. This kernel doesn't actually respect Mercer's theorem but has been used in practice with some success since the resulting SVM functionally resembles a neural network (VAPNIK, 2000).

$$k(\mathbf{v}, \mathbf{v}') = tanh(\gamma \mathbf{v}^T \mathbf{v}' + c_0) \tag{4.37}$$

4.2 Logistic regression

The logistic regression algorithm is a method in which the logarithm of the ratio of the posterior probabilities of each class C_k, calculated using Bayes' theorem, is modeled as a linear equation on $\mathbf{v}_n$. Under certain hypotheses, it provides a Bayesian approach to estimating conditional probabilities of events (such as C_k occurring given $\mathbf{v}_n$), which are associated with class labels.

From a probabilistic point of view, we can model the class-conditional densities $p(\mathbf{v}|C_k)$ and class priors $p(C_k)$, where C_k represents the k-th class and $\mathbf{v}$ is a vector from the dataset. Bayes' theorem is then used to derive the posterior probabilities for C_k (HASTIE *et al.*, 2009). For the case of two classes C_1 and C_2, $p(C_k|\mathbf{v})$ can be written as:

$$p(C_1|\mathbf{v}) = \frac{p(\mathbf{v}|C_1)p(C_1)}{p(\mathbf{v}|C_1)p(C_1) + p(\mathbf{v}|C_2)p(C_2)} = \frac{1}{1 + exp(-a)} = \sigma(a) \tag{4.38}$$

where a is defined as:

$$a = ln\frac{p(\mathbf{v}|C_1)p(C_1)}{p(\mathbf{v}|C_2)p(C_2)} \tag{4.39}$$

and $\sigma(a)$ is called the logistic function, or sigmoidal function. It can be shown that in the case of K classes, we can write the posteriors $p(C_k|\mathbf{v})$ as:

$$p(C_k|\mathbf{v}) = \frac{e^{a_k}}{\sum_{i=1}^{K} e^{a_i}} \tag{4.40}$$

where $a_k = ln(p(\mathbf{v}|C_k)p(C_k))$. In the logistic regression approach, we model a_k as linear function of $\mathbf{v}$, represented by:

$$a_k = ln\left(p(\mathbf{v}|C_k)p(C_k)\right) = \mathbf{w}_k^T\mathbf{v} \tag{4.41}$$

For each class k, we assign a weight vector $\mathbf{w}_k^T$ of the same size as $\mathbf{v}$. One way to find good values for the weight vectors $\mathbf{w}_k$ is by using maximum likelihood estimation (MLE), an estimation criterion that maximizes the joint probability of a set of observations $\mathbf{y}$ given the set of parameters $\boldsymbol{\theta}$. This criterion can be written as:

$$\{\mathbf{w}_1, \ldots, \mathbf{w}_K\} = \underset{\mathbf{w}_1,\ldots,\mathbf{w}_K}{\arg\max}\; p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) \tag{4.42}$$

where $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]^T$, and each $\mathbf{t}_n = \{t_{n,1}, t_{n,2}, \ldots, t_{n,K}\}$ is a K-dimensional vector representing the class of the n-th observation, with $t_{n,k} \in \{0, 1\}$. In $\mathbf{t}_n$, all values are zero except for its k-th position, indicating that it belongs to class C_k.

To model $\mathbf{t}_n$, we may use the multinomial distribution, described by the following equation:

$$p(\mathbf{t}_n|\boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_k^{t_{n,k}} \tag{4.43}$$

where $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_K\}$ are the parameters from the multinomial distribution, with $\sum_{k=1}^{K} \mu_k = 1$ and $\mu_k = p(t_{n,k} = 1) = p(C_k) \geq 0$.

In our model, we assume that $p(C_k|\mathbf{v}_n) = p(t_{n,k} = 1|\mathbf{v}_n)$, such that a sample from the k-th class has $\mathbf{t} = [0, \ldots, 1, \ldots, 0]$, where all values are zero except the k-th value. If we assume N independent observations, we can write the likelihood function as:

$$p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\mathbf{v}_n)^{t_{n,k}} \tag{4.44}$$

It is common to use the negative of the logarithm of the likelihood function as the minimization criteria, so we can arrive at the optimization goal:

$$\{\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_K\} = \underset{\mathbf{w}_1,\ldots,\mathbf{w}_K}{\arg\min} \; -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{n,k} ln(y_{n,k}) \tag{4.45}$$

where $\{\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_K\}$ are the estimated parameters of the logistic regression and $y_{n,k} = \frac{e^{\mathbf{w}_k^T \mathbf{v}_n}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T \mathbf{v}_n}}$. So, the logistic regression estimate of $p(C_k|\mathbf{v}_n) = y_{n,k}$ is calculated as:

$$y_{n,k} = \frac{e^{\hat{\mathbf{w}}_k^T \mathbf{v}_n}}{\sum_{j=1}^{K} e^{\hat{\mathbf{w}}_j^T \mathbf{v}_n}} \tag{4.46}$$

Often, an additional term is added to the objective function of Equation (4.45) to prevent the values in the weight vectors $\{\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_K\}$ from growing large (in absolute value). This term penalizes the magnitude of the weights, keeping them close to zero. It is commonly used as a regularization method aimed at limiting the flexibility of the model, thereby improving generalization. The new optimization goal is written as:

$$\{\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_K\} = \underset{\mathbf{w}_1,\ldots,\mathbf{w}_K}{\arg\min} \; -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{n,k} ln(y_{n,k}) + \lambda h(\mathbf{W}) \tag{4.47}$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_K]$, $h(\mathbf{W})$ is the additional regularization term and λ is its strength.

Two common methods for regularization are the Lasso, or L_1-regularization (represented by h_{L_1}), and the Ridge, or L_2-regularization (represented by h_{L_2}) (HASTIE *et al.*, 2009). The penalty term for the Lasso assumes the following form:

$$h_{L_1}(\mathbf{W}) = \sum_{k=1}^{K} \sum_{i=1}^{D} |w_{k,i}| \tag{4.48}$$

where $w_{k,i}$ is the i-th element of $\mathbf{w}_k$. This penalty, also called $L1$ loss function, tends to yield sparse weight vectors, so it can be used for selecting features by zeroing many weights from $\mathbf{W}$. On the other hand, the Ridge regularization is as follows:

$$h_{L_2}(\mathbf{W}) = \sum_{k=1}^{K} \sum_{i=1}^{D} |w_{k,i}|^2 \tag{4.49}$$

and tends to spread the penalty across all weights instead of zeroing out a few. As a result, the weight vectors will have a smaller norm. This is especially useful for highly correlated features, where there is uncertainty in their values because a very large positive weight can be "canceled out" by a very large negative weight of a correlated feature, thereby potentially increasing the variance of the predictor. h_{L_2} is also referred to as the $L2$ loss function.

4.3 Linear Discriminant Analysis

The term *discriminant* refers to functions that can map an input vector $\mathbf{v}_n$ to a class C_k. Fisher's linear discriminant, named after Sir Ronald Aylmer Fisher (COHEN, 2013), is a method that linearly projects the original feature space into a lower-dimensional space, aiming to maximize the differences between the means of classes while minimizing their variance to reduce overlap. This mapping is represented as:

$$\mathbf{v}' = \mathbf{W}^T \mathbf{v} \tag{4.50}$$

where $\mathbf{v}' \in \mathbb{R}^{D'}$ is the feature vector in the lower dimensional space, $\mathbf{v} \in \mathbb{R}^{D}$ is the feature vector in the original space, and $\mathbf{W} \in \mathbb{R}^{D \times D'}$ is the linear mapping. For the two-class case, the mapping is done in only one dimension using a vector $\mathbf{w}$.

Fisher's criterion (BISHOP, 2006) is expressed as:

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \tag{4.51}$$

which summarizes the two objective functions. The numerator represents the between-class variance, where m_1 and m_2 are the means of the classes in the one-dimensional projected space, and the denominator represents the within-class variance, where s_1 and s_2 are the variances of two classes in this space.

More generally, Equation (4.51) can be written as:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \tag{4.52}$$

where $\mathbf{S}_B$ is the between-class covariance matrix given by:

$$\mathbf{S}_B = (\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1)(\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1)^T \tag{4.53}$$

and $\mathbf{S}_W$ is the within-class covariance matrix given by Equation (4.54). Here, $\bar{\mathbf{v}}_k = \frac{1}{|\mathcal{S}_k|} \sum_{n \in \mathcal{S}_k} \mathbf{v}_n$ is the average vector of class k, and $\mathcal{S}_k$ is the set of indices for feature vectors from the k-th class.

$$\mathbf{S}_W = \sum_{n \in \mathcal{S}_1} (\mathbf{v}_n - \bar{\mathbf{v}}_1)(\mathbf{v}_n - \bar{\mathbf{v}}_1)^T + \sum_{n \in \mathcal{S}_2} (\mathbf{v}_n - \bar{\mathbf{v}}_2)(\mathbf{v}_n - \bar{\mathbf{v}}_2)^T \tag{4.54}$$

To generalize to multiple classes, we return to Equation (4.50) as our mapping. The generalized within-class covariance matrix is calculated as:

$$\mathbf{S}_W = \sum_{k=1}^{K} \mathbf{S}_k \tag{4.55}$$

$$\mathbf{S}_k = \sum_{n=1}^{N} (\mathbf{v}_n - \bar{\mathbf{v}}_k)(\mathbf{v}_n - \bar{\mathbf{v}}_k)^T \tag{4.56}$$

with $\mathbf{S}_k$ as the covariance matrix from the k-th class. We can assume that the total covariance matrix $\mathbf{S}_T$ can be decomposed as a sum of the within-class covariance matrix and the between-class covariance matrix $\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B$ (HASTIE *et al.*, 2009). As $\mathbf{S}_T$ is the total covariance, it can also be calculated as:

$$\mathbf{S}_T = \sum_{n=1}^{N} (\mathbf{v}_n - \bar{\mathbf{v}})(\mathbf{v}_n - \bar{\mathbf{v}})^T \tag{4.57}$$

$$\bar{\mathbf{v}} = \frac{1}{N} \sum_{n=1} \mathbf{v}_n \tag{4.58}$$

where $\bar{\mathbf{v}}$ is the average value of $\mathbf{v}$. With this system of equations, we can calculate $\mathbf{S}_B$ with the following:

$$\mathbf{S}_B = \sum_{k=1}^{K} N_k (\bar{\mathbf{v}}_k - \bar{\mathbf{v}})(\bar{\mathbf{v}}_k - \bar{\mathbf{v}})^T \tag{4.59}$$

To proceed with the LDA solution, we will need to calculate all of these values in the projected space. By denoting the equivalent covariance matrices and average vectors in the new space with an apostrophe, the values are calculated using the following equations:

$$\mathbf{S}'_T = \sum_{n=1}^{N} (\mathbf{v}'_n - \bar{\mathbf{v}}')(\mathbf{v}'_n - \bar{\mathbf{v}}')^T \tag{4.60}$$

$$\mathbf{S}'_T = \mathbf{S}'_B + \mathbf{S}'_W \tag{4.61}$$

$$\mathbf{S}'_W = \sum_{k=1}^{K} \mathbf{S}'_k \tag{4.62}$$

$$\mathbf{S}'_k = \sum_{n=1}^{N} (\mathbf{v}'_n - \bar{\mathbf{v}}'_k)(\mathbf{v}'_n - \bar{\mathbf{v}}'_k)^T \tag{4.63}$$

$$\mathbf{S}'_B = \sum_{k=1}^{k} N_k (\bar{\mathbf{v}}'_k - \bar{\mathbf{v}}')(\bar{\mathbf{v}}'_k - \bar{\mathbf{v}}')^T \tag{4.64}$$

$$\bar{\mathbf{v}}'_k = \frac{1}{|\mathcal{S}_k|} \sum_{n \in \mathcal{S}_k} \mathbf{v}'_n \tag{4.65}$$

$$\bar{\mathbf{v}}' = \frac{1}{N} \sum_{n=1} \mathbf{v}'_n \tag{4.66}$$

where $\mathbf{S}'_T$ represents the total covariance matrix, $\mathbf{S}'_B$ represents the between-class covariance matrix, $\mathbf{S}'_W$ represents the total within-class covariance matrix, $\mathbf{S}'_k$ represents the within-class covariance matrix for the k-th class, $\bar{\mathbf{v}}'$ represents the average dataset vector and $\bar{\mathbf{v}}'_k$ represents the average class vector (all in the projected space).

The idea of LDA is to maximize $\mathbf{S}'_B$ and minimize $\mathbf{S}'_W$, so different classes lay distant from each other and samples from the same class are maximally packed together. The criterion for defining the mapping $\mathbf{W}$ must be a scalar that increases as the between-class covariance increases and the within-class covariance decreases. There are many possible criteria for this, and the classical algorithm uses the following criterion (FUKUNAGA, 1990):

$$J(\mathbf{W}) = tr(\mathbf{S}_W'^{-1}\mathbf{S}_B') = tr\left((\mathbf{W}\mathbf{S}_W\mathbf{W}^T)^{-1}(\mathbf{W}\mathbf{S}_B\mathbf{W}^T)\right) \tag{4.67}$$

where the solution to $\mathbf{W}$ are the D' eigenvectors of $\mathbf{S}_W^{-1}\mathbf{S}_B$ that corresponds to the D' largest eigenvalues.

We know that $\mathbf{S}_B$ is a sum of K matrices, each one of rank 1 because they are outer products of vectors. Also, the rank of $\mathbf{S}_B$ is at most $K-1$ due to Equation (4.59), showing that D' will be at most $K-1$. It also shows that the projection into the D'-dimensional space will not change the value of $J(\mathbf{W})$. Howland *et al.* (2003) proposed a method for optimizing the criterion in Equation (4.67) using auxiliary matrices and Generalized Singular Vector Decomposition (GSVD), with the advantage of not needing to explicitly calculate $\mathbf{S}_W$ and $\mathbf{S}_B$.

Now that the optimal projection for the discrimination criterion is found, we can calculate the logarithm of the probabilities $p(\mathbf{v}|C_k)$ that $\mathbf{v}$ belongs to class C_k for each class. For this, we first assume that the data vectors from class k have multivariate normal distribution $\mathcal{N}(\bar{\mathbf{v}}_k, \mathbf{\Sigma}_k)$, described as:

$$f_k(\mathbf{v}) = \frac{1}{(2\pi)^{\frac{D'}{2}}|\mathbf{\Sigma}_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{v}-\bar{\mathbf{v}}_k)^T\mathbf{\Sigma}_k^{-1}(\mathbf{v}-\bar{\mathbf{v}}_k)} \tag{4.68}$$

LDA assumes that the classes covariance matrices are equal, i.e. $\mathbf{\Sigma}_k = \mathbf{\Sigma}\ \forall k$. The chosen class for $\mathbf{v}$ will be the one that maximizes the probability function. The linear discrimination function is written as:

$$\delta_k(\mathbf{v}) = \mathbf{v}^T\mathbf{\Sigma}^{-1}\bar{\mathbf{v}}_k - \frac{1}{2}\bar{\mathbf{v}}_k^T\mathbf{\Sigma}^{-1}\bar{\mathbf{v}}_k + log\frac{N_k}{N} \tag{4.69}$$

and represents the log-probability of $\mathbf{v}$ being an example from the k-th class, so k can be estimated as $\arg\max\limits_k \delta_k(\mathbf{v})$ (HASTIE *et al.*, 2009). We also need to estimate $\mathbf{\Sigma}$, which can be done using:

$$\hat{\mathbf{\Sigma}} = \frac{1}{N-K}\sum_{k=1}^{K}\sum_{n\in\mathcal{S}_k}(\mathbf{v}_n - \bar{\mathbf{v}}_k)(\mathbf{v}_n - \bar{\mathbf{v}}_k)^T \tag{4.70}$$

The decision boundaries between two classes l and k are defined by $\delta_k(\mathbf{v}) = \delta_l(\mathbf{v})$, which is linear in $\mathbf{v}$.

4.4 Random forest

Random forests (RFs) can be thought of as ensembles i.e. a group of relatively weak classifiers in which the ensemble prediction is a composition of the predictions of each model in the group. There are many ways to build a set of diversified classifiers, such as Bagging or Bootstrap Aggregation, in which the components are trained using random subsets of the whole dataset (BISHOP, 2006). The random subspace method (HO, 1998) trains each weak classifier using only a subset of the characteristic features i.e. not all are used for fitting each predictor. In the case of RFs, each one of those individual models is *Decision Trees*, hence the epithet *Forests*.

4.4.1 Decision tree

Decision trees are hierarchical structures that use a recursive tree of decisions in order to predict outcomes. At each node, one condition is verified and if the condition holds the decision flow continues to one or another branch. Building the optimal tree structure and the best conditions is an infeasible computational problem, so there are many heuristics for it. A particular common framework for building such trees is called *classification and regression trees* (CART) (BREIMAN *et al.*, 1984).

In this framework, a decision tree is a binary tree (graph) where each node corresponds to a decision to continue to the left or right child node. This decision is based on thresholding the value of one of the dimensions of a feature vector $\mathbf{v}_n = [v_{n,1}, \ldots, v_{n,D}]^T$ (with $\mathbf{v}_n$ being the n-th feature vector of the dataset and D the number of dimensions of $\mathbf{v}_n$), and hence its decision boundaries consists of hyper-planes that partition the feature space into M regions $\{\mathcal{R}_1, \ldots, \mathcal{R}_M\}$ defined by the M leaves of the tree. Those boundaries are parallel to the coordinate system axes ($\mathbb{R}^D$) since they are defined by equations of the form $v_{n,l} - \theta_k = 0$, where θ_k are thresholds that are learned during the model training phase.

The tree is grown starting from a root node, represented by $\mathcal{Q}_0 = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$, which is the whole feature vector space. $\mathcal{Q}_0$ is split into two disjoint partitions (called its "children" nodes) $\mathcal{Q}_m^{left}(l,\theta) = \{\mathbf{v}|\mathbf{v} \in \mathcal{Q}_m, v_l \leq \theta\}$ and $\mathcal{Q}_m^{right}(l,\theta) = \{\mathbf{v}|\mathbf{v} \in \mathcal{Q}_m, v_l > \theta\}$ (THEODORIDIS; KOUTROUMBAS, 2006) (with $m = 0$ for the first split) by selecting both a feature l and a threshold θ that minimizes a splitting criterion H':

$$\{l_m, \theta_m\} = \underset{l,\theta}{\arg\min}\ H'(\mathcal{Q}_m, l, \theta) \tag{4.71}$$

where H' is a weighted sum of the *impurity* measure H of each partition of Q_m:

$$H'(\mathcal{Q}_m, l, \theta) = \left(\frac{|\mathcal{Q}_m^{left}(l,\theta)|}{|\mathcal{Q}_m|} H(\mathcal{Q}_m^{left}(l,\theta)) + \frac{|\mathcal{Q}_m^{right}(l,\theta)|}{|\mathcal{Q}_m|} H(\mathcal{Q}_m^{right}(l,\theta))\right) \tag{4.72}$$

The impurity is generally a metric on how clustered feature vectors from the same class are in the partition. One common metric is the Gini index (BREIMAN *et al.*, 1984), defined as:

$$H(\mathcal{Q}_m) = \sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k}) \tag{4.73}$$

where $\hat{p}_{m,k}$ is defined as

$$\hat{p}_{m,k} = \frac{1}{|\mathcal{Q}_m|} \sum_{\mathbf{v}_n \in \mathcal{Q}_m} I(t_n = k) \tag{4.74}$$

To find values of l and θ that minimize Equation (4.71) we can calculate the objective function for all l and iterate θ overall unique values of $v_{n,l}$ for all $\mathbf{v}_n \in \mathcal{Q}_m$. This partitioning is recursively applied to $\mathcal{Q}_m^{left}(l_m,\theta_m)$ and $\mathcal{Q}_m^{right}(l_m,\theta_m)$ until a stop criterion is achieved, which can be a minimum cardinality of $\mathcal{Q}_m$, a minimum cardinality of both children of $\mathcal{Q}_m$, a maximum tree depth, a minimum decrease in impurity (difference from $H(\mathcal{Q}_m)$ and $H'(\mathcal{Q}_m, l_m, \theta_m)$) (THEODORIDIS; KOUTROUMBAS, 2006). The partitions defined by the tree leaves are denoted by $\mathcal{R}_1$ to $\mathcal{R}_M$ and represent a sequence of decisions on $\mathbf{v}_n$. They are called leaves because they don't have children and represent the final decision of the classifier.

The class prediction of $\mathbf{v}_n$ is $y(\mathbf{v}_n) = C_k$ such that $k = \underset{k}{\arg\max}\ \hat{p}_{i,k}$, where $\mathbf{v}_n \in \mathcal{R}_i$ (and we know that there is only one m that satisfies this since $R_i \cap R_j = \{\}\ \forall\{i,j\}, i \neq j$). One can make probabilistic predictions to model the conditional probabilities $p(C_k|\mathbf{v}_n) = \hat{p}_{i,k}$ with $\mathbf{v}_n \in R_i$, since $\sum_{k=1}^{K} \hat{p}_{i,k} = 1$. Note that this means that we need to find which partition $\mathcal{R}_i$ contains $\mathbf{v}_n$, which is done by recursively checking which child partition contains $\mathbf{v}_n$, starting from $\mathcal{Q}_0$ until a *leaf* partition (a partition without children) is achieved.

4.4.2 The ensemble

The set of features and thresholds $\{(l_1, \theta_1), \ldots, (l_M, \theta_M)\}$ defines a decision tree T_i, and $T_i^k(\mathbf{v}_n)$ is its output probability estimate of $\mathbf{v}_n$ being from the k-th class (as in Equation (4.74)). To create the random forest, N_T decision trees $T_i,\ i \in \{1, \ldots N_T\}$ are created using a modified version of Equation (4.71), as follows:

$$\{l_m, \theta_m\} = \underset{l \in \mathcal{F}_i, \theta}{\arg\min}\ H'(\mathcal{Q}_m, l, \theta) \tag{4.75}$$

with $\mathcal{F}_i$ denoting the random subset of possible features l of $\mathbf{v}$ used in the i-th tree T_i, so $|\mathcal{F}_i| < D$. Apart from this, the N_T (an arbitrarily chosen number of trees) trees are constructed, but each with different $\mathcal{F}_i$. The modification yields a set of trees that are created using fewer features than D, so each tree is tackling the classification task with different information making the random forest robust to outliers, noise and not overfit (given a sufficient number of trees) (BREIMAN, 2001).

4.4.2.1 Extremely randomized trees

The **Ext**remely **ra**ndomized Trees (GEURTS *et al.*, 2006) ensemble algorithm goes one step further into the randomization, and uses random thresholds at each split iteration for each tree. At each split a random threshold $\theta_l \in (min(v_{\cdot,l}), max(v_{\cdot,l}))$ is chosen for each feature whose index is in $\mathcal{F}_i$, where $min(v_{\cdot,l})$ and $max(v_{\cdot,l})$ are respectively the minimum and maximum possible values for $v_{n,l}$. To choose the best feature l and threshold θ for the split, a new modification is made in Equation (4.71), resulting in the following optimization criteria:

$$\{l_m, \theta_m\} = \underset{(l,\theta) \in \{(l',\theta') | l' \in \mathcal{F}_i, \theta' \in \Theta_i\}}{\arg\min}\ H'(\mathcal{Q}_m, l, \theta) \tag{4.76}$$

With $\Theta_i = \{\theta_l | l \in F_i\}$ being the set of random thresholds chosen for each feature.

The ensemble prediction is typically done by a majority voting of the set of classifiers. In all of these approaches, we calculate the predicted probability of class C_k by averaging the predictions of each tree, as:

$$p(C_k|\mathbf{v}_n) = \frac{1}{N_T}\sum_{i=1}^{N_T} T_i^k(\mathbf{v}_n) \tag{4.77}$$

and the predicted class is the one with highest probability, i.e., $\underset{C_k}{\arg\max}\ p(C_k|\mathbf{v}_n)$

4.5 Naive Bayes

The Naïve Bayes is a family of classifiers that makes use of Bayes' theorem to estimate $p(C_k|\mathbf{v})$ (BISHOP, 2006):

$$p(C_k|\mathbf{v}) = \frac{p(\mathbf{v}|C_k)p(C_k)}{p(\mathbf{v})} \tag{4.78}$$

In this equation, $p(C_k)$ is called the prior of class C_k and can be estimated as its prevalence in the dataset, and $p(\mathbf{v}|C_k)$ is the conditional probability of observing $\mathbf{v}$ given that we know it is from the k-th class. In the following equations, $\mathbf{v} = [v_1, \ldots, v_D]^T$ represents a random vector of features (with D as its dimensionality), and $\mathbf{v}_n = [v_{n,1}, \ldots, v_{n,D}]^T$ represents the n-th feature vector of the dataset, i.e., a realization of $\mathbf{v}$.

The name comes from one "naïve" hard assumption that all features in $\mathbf{v}$ are conditionally independent, such that $p(\mathbf{v}|C_k) = \prod_{i=1}^{D} p(v_i|C_k)$. Since $p(\mathbf{v})$ is constant given $\mathbf{v}$, to estimate the feature vector class we need to find k that maximizes $p(C_k)\prod_{i=1}^{D} p(v_i|C_k)$.

One of the members of this family of models is the Gaussian Naïve Bayes, which assumes that each feature has normal conditional distribution $p(v_i|C_k) = N(\mu_{i,k}, \sigma^2_{i,k})$, where $\mu_{i,k}$ and $\sigma^2_{i,k}$ are the features' class conditioned means and variances, respectively. Those statistics can be estimated using maximum likelihood estimation (CHAN *et al.*, 1982) as follows:

$$\hat{\mu}_{i,k} = \frac{1}{N_k}\sum_{n\in\{n|t_n=k\}} v_{n,i} \tag{4.79}$$

$$\hat{\sigma}^2_{i,k} = \frac{1}{N_k - 1}\sum_{n\in\{n|t_n=k\}} (v_{n,i} - \hat{\mu}_{i,k})^2 \tag{4.80}$$

where N_k is the number of samples v_n that belongs to the k-th class. Finally, the probabilistic estimator is given by:

$$p(C_k|\mathbf{v}) = \frac{p(C_k)\prod_{i=1}^{D} p(v_i|C_k)}{\sum_{j=1}^{K}\left(p(C_j)\prod_{i=1}^{D} p(v_i|C_j)\right)} \tag{4.81}$$

This equation comes from the conditional independence assumption which, together with Equation (4.78), implies $p(C_k|\mathbf{v}) \propto p(C_k)\prod_{i=1}^{D} p(v_i|C_k)$ and that $\sum_{k=1}^{K} p(C_k|\mathbf{v}) = 1$. The prior $p(C_k)$ can be estimated empirically through the dataset using the following:

$$p(C_k) = \frac{N_k}{N} \tag{4.82}$$

and $p(v_i|C_k)$ is calculated by:

$$p(v_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_{i,k}^2}} exp\left(\frac{(v_i - \mu_{i,k})^2}{2\sigma_{i,k}^2}\right) \tag{4.83}$$

4.6 Multilayer perceptron

The multilayer perceptron (MLP) is a machine learning structure from the family of the fully-connected feedforward neural networks. This model, historically, is an extension of the classical (Rosenblatt's) perceptron towards a multilayer framework in which nonlinear classification tasks can be straightforwardly dealt with (BISHOP, 2006).

The name *neural network* (NN) historically comes from the attempt to represent certain aspects of the brain structure mathematically and computationally. An NN is based on units called neurons, which are connected to other neurons through their synapses. Feedforward networks have a multi-layer structure in which information only goes one way ("forward"), and the "fully-connected" qualifier refers to the fact that all neuron outputs from one layer are connected to all neurons of the next layer.

Following the general outline of the classical perceptron, the inputs of a given neuron are added and transformed by a nonlinear mapping, yielding an output $z_{l,i}$, which corresponds to the i-th neuron of the l-th layer. To simplify the notation, we will use the equality $z_{0,i} = v_{n,i}$, meaning that $z_{0,i}$ is not the output of a neuron but one of the input features from the feature vector. Mathematically, the neuron performs the following operation:

$$z_{l,i} = f_{l,i}\left(\sum_{j=1}^{n_{l-1}} w_{l,i,j} z_{l-1,j} + b_{l,i}\right) \tag{4.84}$$

where n_l is the number of neurons in the l-th layer, $w_{l,i,j}$ is the synaptic weight for the j-th input of the i-th neuron of the l-th layer, $b_{l,i}$ is the bias, and $f_{l,i}$ is the activation function (BISHOP, 2006). The vector $\mathbf{w}_{l,i} = [w_{l,i,1}, \dots, w_{l,i,n_{l-1}}]^T$ is a more concise notation for the synaptic weights, where each $w_{l,i,j}$ is the weight of the neuron's j-th input (or equivalently the weight for the j-th neuron of the previous layer $l-1$). Figure 4.2 illustrates the structure of this neuron model. The sum can be represented by an inner product, leading to:

$$z_{l,i} = f_{l,i}\left(\mathbf{w}_{l,i}^T \mathbf{z}_{l-1} + b_{l,i}\right) \tag{4.85}$$

where $\mathbf{z}_l = [z_{l,1}, \dots, z_{l,n_l}]^T$. Since all neurons at the l-th layer have the same number of inputs, the intermediary values $z_{l,i}$ can be written in vector notation as:

$$\mathbf{z}_l = \begin{bmatrix} z_{l,1} \\ \vdots \\ z_{l,n_l} \end{bmatrix} = \begin{bmatrix} f_{l,1}\left(\mathbf{w}_{l,1}^T \mathbf{z}_{l-1} + b_{l,1}\right) \\ f_{l,2}\left(\mathbf{w}_{l,2}^T \mathbf{z}_{l-1} + b_{l,2}\right) \\ \vdots \\ f_{l,n_l}\left(\mathbf{w}_{l,n_l}^T \mathbf{z}_{l-1} + b_{l,n_l}\right) \end{bmatrix} \tag{4.86}$$

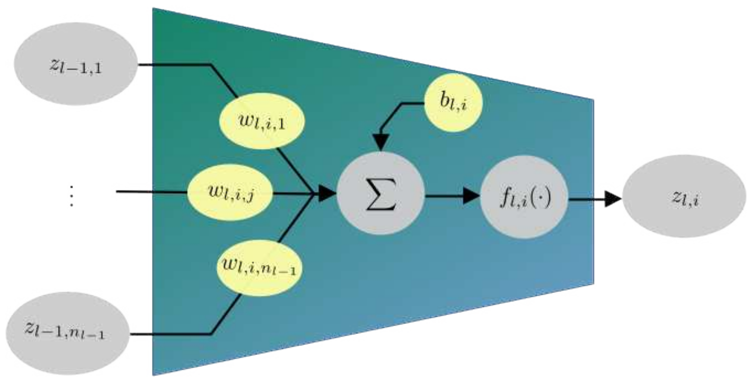


Figure 4.2 – Functional structure of the i-th neuron of the l-th layer.

It is very common to have just one type of activation function for all neurons in the same layer l, i.e., $f_{l,i} = f_l$, so Equation (4.86) can be further simplified using matrix notation as:

$$\mathbf{z}_l = f_l(\mathbf{W}_l^T \mathbf{z}_{l-1} + \mathbf{b}_l) \tag{4.87}$$

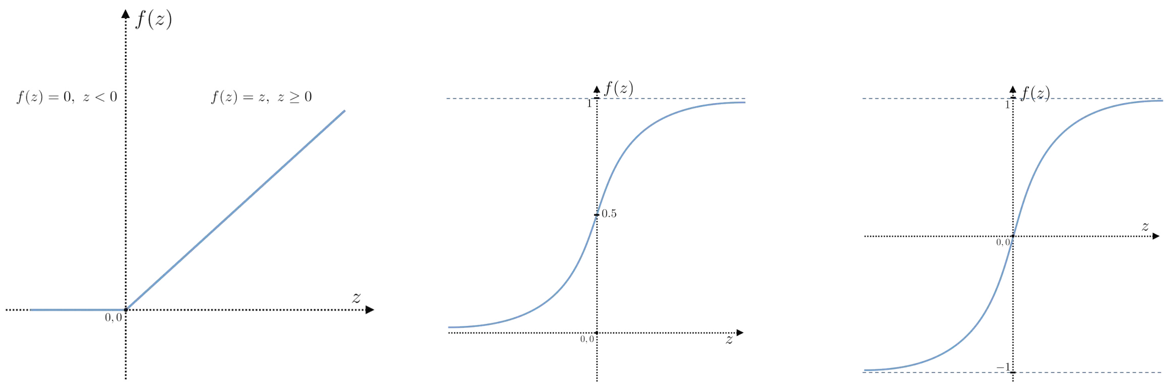
where $\mathbf{b}_l = [b_{l,1}, \ldots, b_{l,n_l}]^T \in \mathbb{R}^{n_l}$ and $f(\mathbf{v}) : \mathbb{R}^{n_l} \rightarrow \mathbb{R}^{n_l} | f(\mathbf{v}) = [f_l(v_1), \ldots, f_l(v_{N_v})]^T$ for $\mathbf{v} = [v_1, \ldots, v_{n_l}]^T$.

Common choices for $f_l(.)$ are the hyperbolic tangent (Equation (4.88), Figure 4.3c), the sigmoid function (Equation (4.89), Figure 4.3b), the rectified linear unit function (ReLU, shown in Equation (4.90), Figure 4.3a). It is also common to use the *softmax* function (Equation (4.91)) as the activation function of the last layer, since the sum of all neuron outputs will be equal to 1, allowing them to be interpreted as the conditional probabilities $p(C_k|\mathbf{v})$.

$$tanh(z) = \frac{e^{2z}+1}{e^{2z}-1} \tag{4.88}$$

$$sigmoid(z) = \frac{1}{1+e^{-z}} \tag{4.89}$$

$$ReLU(z) = \max(0, z) \tag{4.90}$$



(a) ReLU function. (b) Sigmoid function. (c) Hyperbolic tangent function.

Figure 4.3 – Common activation functions for neural networks.

$$f_l(\mathbf{z}_l) = \frac{1}{\sum_{n=1}^{n_l} e^{z_n}} \begin{bmatrix} e^{z_1} \\ \vdots \\ e^{z_{n_l}} \end{bmatrix} \tag{4.91}$$

The *softmax*, shown in (4.91), has the same functional form of Equation (4.40) used for the logistic regression, which establishes an interesting and relevant conceptual link with this linear classifier.

In order to simplify the following calculations, we can redefine some variables. Having the term $b_{l,i}$ at each f_l for calculating $z_{l,i}$ is mathematically equivalent to having an additional weight on $\mathbf{w}_{l,i}$ with value $b_{l,i}$, and an additional value in the $\mathbf{z}_{l-1}$ vector equal to 1. So, let's consider $\mathbf{w}_{l,i} \leftarrow [w_{l,i,1}, \ldots, w_{l,i,n_{l-1}}, b_{l,i}]^T$ and $\mathbf{z}_l \leftarrow [z_{l,1}, \ldots, z_{l,n_l}, 1]$, and also adjust the layer size $n_l \leftarrow n_l + 1$, and define $z_{l,n_l} = 1$ and $w_{l,i,n_l} = b_{l,i}\ \forall\ l, i$ to keep the notation consistent. This allows us to rewrite Equation (4.87) as:

$$\mathbf{z}_l = f_l(\mathbf{W}_l^T \mathbf{z}_{l-1}) \tag{4.92}$$

The learnable parameters of this approach are the redefined $\mathbf{W}_l$ matrices (containing the $b_{l,i}$ biases). The number of layers L and neurons per layer n_l are called hyperparameters and are not learned, because they represent the model's *architecture* and hence are modeling choices. One common objective function for finding the parameters is defined by:

$$\{\mathbf{W}_1, \ldots, \mathbf{W}_L\} = \underset{\{\mathbf{W}_1,\ldots,\mathbf{W}_L\}}{\arg\min} \quad -\sum_{n=1}^{N} J_n \tag{4.93}$$

which uses the cross-entropy function:

$$J_n = J(\mathbf{v}_n, \mathbf{W}_1, \ldots, \mathbf{W}_L) = -\sum_{k=1}^{K} t_{n,k} ln(y_{n,k}) \tag{4.94}$$

where $y_{n,k}$ denotes the output of the k-th neuron in the last layer (which is $z_{L,k}$) when the input is $\mathbf{v}_n$ to the neural network. This is the same objective function as used in the logistic regression (Equation (4.44)) because this is an MLE solution with the only assumption that the targets in $\mathbf{T}$ follow the multinomial distribution. In regression tasks, on the other hand, the mean-squared error (MSE) cost function is the classical choice.

The classical optimization strategy for MLPs is the gradient descent family of algorithms, in which the derivatives of the objective function w.r.t. the parameters are used to find the loss minimization direction in an iterative process. A straightforward approach for updating the parameters is the *stochastic gradient descent* (BISHOP, 2006), defined by:

$$\mathbf{W}_l^{(r+1)} = \mathbf{W}_l^{(r)} - \eta \sum_{n=1}^{N} \frac{\partial J_n}{\partial \mathbf{W}_l} \tag{4.95}$$

where $\mathbf{W}_l^{(r+1)}$ is the value of $\mathbf{W}_l$ at the r-th iteration of the gradient update and η is the gradient update step size.

MLPs are universal function approximators i.e. with a sufficient number of layers and/or neurons on each layer, they are able to approximate any function in a defined region of interest (HORNIK *et al.*, 1989), so they are very flexible in terms of adapting to different problems. Another computational advantage is that the J_n gradient w.r.t the model parameters at the l-th layer can be calculated as a function of the gradient of the parameters at the $l+1$-th layer. This means that starting from the L-th layer, we can easily calculate the gradients of the entire network by this *backpropagation* strategy (ROJAS, 1996; RUMELHART *et al.*, 1986). On a note, the stochastic gradient algorithm is often substituted by Adam (**Ada**ptive **M**oment Estimation) (KINGMA; BA, 2017), which uses the notion of "moment" to better guide the gradient direction during the updates.

4.7 EEGNet

Deep learning is a paradigm within neural network theory that advocates the use of a relatively large number of layers as a cascade of feature extractors driven directly by the information present in the dataset. Naturally, the success of this scheme depends on the availability of a sufficiently large dataset and of high-performance parallel hardware. If these conditions are met, deep neural networks can learn which aspects of the data are important and which can be discarded for the task at hand without expert supervision. The deep learning revolution, which took form approximately a decade ago, greatly popularized the use of Convolutional Neural Networks (CNN), which are a class of neural networks that are constructed to deal with arrays of data, like signals, images, or videos. They can be viewed as stacked sets of filters, with less learnable parameters, and are more efficient than the fully connected networks for those kinds of data (LECUN *et al.*, 2015).

EEGNet is a compact CNN designed for EEG signal processing, and can be applied to different paradigms (LAWHERN *et al.*, 2018), consisting of 4 main layers. The first main layer is a conventional 2D convolutional layer with F_1 filters of size $(1, 64)$ (i.e., 500ms). Even though the convolutional operation is 2D, it is equivalent to applying a temporal filter individually to each one of the N_x input EEG channels. The output of this layer is $N_x \times F_1$ feature maps, where each feature map can be seen as a band-filtered version of the respective EEG channel. This shows that the first layer selects interesting bands to be further processed by the network. Let's use the notation that $\mathbf{X}$ is the $N_x \times T$ matrix with the EEG signals, and $\mathbf{X}_k$ is the EEG data after being filtered by the k-th temporal filter, with $1 \leq k \leq F_1$.

The second main layer is a Depthwise Convolution, which is equivalent to learning a spatial filter for each one of the temporal filters from the first layer. This means that after selecting the frequency bands of interest, each $\mathbf{X}_k$ passes through a spatial filter. A parameter D indicates the number of spatial filters that are learned for each temporal filter, meaning that this layer outputs $D * F_1$ frequency and spatially filtered signals.

The third main layer is a Separable Convolution layer, which is implemented using a Depthwise Convolution followed by a Pointwise convolution. Here, the depthwise convolution is equivalent to a set of temporal filters (one for each one of the $D * F_1$ signals from the previous layer), which learns a time-domain summary for each one of the feature maps. The pointwise convolution then learns how to combine those values into F_2 outputs.

The final layer is a traditional fully connected dense layer followed by a softmax operation, which will output the probability estimates of the data pertaining to each one of the K classes (depending on the task and the paradigm).

Those are the main components of the EEGNet, which are interpretable building blocks with signal processing qualities. The actual network has some Batch Normalization layers (IOFFE; SZEGEDY, 2015), Average pooling layers for dimensionality reduction, ELU activation function layers (CLEVERT *et al.*, 2016), and Dropout layers for regularization (SRIVASTAVA *et al.*, 2014) during training. The full architecture is sequentially composed of:

1. Convolutional Layer (F_1 500ms temporal filters)
2. Batch Normalization
3. Dropout (50%)
4. Depthwise Convolution (F_1 spatial filter)
5. Batch Normalization
6. ELU activation function

7. Average Pooling (kernel size (1×4), stride 4)
8. Dropout (50%)
9. Separable Convolution (F_2 500ms filters)
10. Batch Normalization
11. ELU activation function
12. Average Pooling (kernel size (1×8), stride 8)
13. Dropout (50%)
14. Dense Layer (K neurons)
15. Softmax function

The typical values for the parameters are $F_1 = 4$, $D = 2$, and $F_2 = D * F_1$. Additionally, the norm of each spatial filter is constrained to be smaller or equal to 1, and the norm of each weight vector of the final dense layer is constrained to be smaller or equal to 0.25.

The EEG data is pre-processed before being used for the network training. The data is downsampled to 128Hz, then a bandpass Butterworth filter with cut-off frequencies of 4Hz and 38Hz is applied, followed by an electrode-wise exponential moving standardization. For $x_i(t)$ the i-th channel at time t, processed signal x_i' is calculated as:

$$x_i'(t) = \frac{x_i(t) - \mu_t}{\sqrt{\sigma_t^2}} \tag{4.96}$$

$$\mu_i(t) = \lambda x_i(t) + \beta \mu_i(t-1) \tag{4.97}$$

$$\sigma_i^2(t) = \lambda(x_i(t) - \mu_i(t))^2 + \beta \sigma_i^2(t-1) \tag{4.98}$$

where $\mu_i(t)$ is the average of the i-th channel at time t, $\sigma_i(t)$ is its variance. The constants γ and β represents the decay of the moving exponential, with $\lambda = (1 - \beta) = 0.001$. As the formula is recursive, we set $\mu_i(t)$ as the mean of the first 1000 samples of $x_i(t)$, and σ_i^2 as the variance of the first 1000 samples of $x_i(t)$ (SCHIRRMEISTER *et al.*, 2017).

Before each training phase, 30% of the training data is removed to be used as a validation set for the early stopping of the network (YAO *et al.*, 2007), so only 70% of the data is used for training EEGNet. The method consists of tracking the loss function value during training for a separated set (the validation set), and stopping the training when it decreases slower than a threshold or stops increasing. This is done to estimate the generalization error of the network during training, avoiding overfitting and saving time.

In the next Chapter, we will present the materials used in our analyses, the experimental setup used to answer our research questions, the results, and discussions.

5 Methods and Results

In this chapter, we present the experiments that form the core of our contributions and analysis of the obtained results. Two datasets were used in our experiments, in association with different experiments. The first experiment comprises the joint comparison of the use of different ICA and classification methods in a Motor-Imagery BCI. The methods are compared in terms of Cohen's Kappa metric, and we calculate the average performances for each ICA method and classifier. In the following scenario, we replace the feature extraction and classification modules with the EEGNet (LAWHERN *et al.*, 2018), a neural network designed for the processing and feature extraction of EEG, originally proposed for classifying signals in MI-BCI. Each ICA method is evaluated in terms of accuracy, in comparison with not using ICA, and in convergence speed of the network during training. An additional experiment is done to compare the different ICA methods in two scenarios: a within-session calibration of the BCI, and a between-session calibration. With this, we can evaluate the methods that better generalize between different acquisition sessions or days.

5.1 Datasets

The BCI Competition IV dataset 2a (dataset D1) (BRUNNER *et al.*, 2004) is composed of data from 9 subjects, with each one performing four motor imagery tasks during two days, with two sessions on each day. At each session, the subjects performed 36 trials of each motor imagination task: left hand, right hand, foot and tongue movement (4 classes). The task cues were presented on a monitor screen. The trial protocol starts with a beep to indicate its start, and after two seconds a visual cue is displayed on the monitor during 1.25s, signaling the subject to imagine the respective task for 4s. The study followed the 10-20 international montage system with 22 electrodes at 250Hz sampling rate, and the signal was bandpass filtered between 0.5Hz and 100Hz, and also by a notch filter at 50Hz. We used only the 4s of imagination in our experiments.

The OpenBMI dataset (dataset D2) (LEE *et al.*, 2019) is also composed of two data collection days, with two sessions on each day. On each session, each one of the 54 subjects performed 50 trials of hand motor imagination for left and right hands (2 classes), totaling 400 trials over the four sessions. Each trial starts with a fixation cross that appears on a monitor screen for 3 s, followed by the image of an arrow that indicates which hand to imagine the movement. The imagination lasts for 4 s, and is succeeded by 6 $s \pm 1.5$ s of a blank screen indicating rest. The montage followed the 10-20 international system, with 62 electrodes at 1000Hz. EEG was bandpass-filtered using a 5th-order IIR butterworth filter at 8Hz and 30Hz. We cropped the trial windows from 1 s to 3.5 s to reproduce the methodology in LEE *et al.*.

5.2 ICA and classifier comparison

In this section, we will present results from the benchmarking of the set of ICA algorithms and classification methods that were discussed. The BCI will be evaluated in terms of Cohen's kappa, which is calculated as follows:

$$\kappa = 1 - \frac{1 - p_o}{1 - p_e} \tag{5.1}$$

where p_o is the relative number of concordances between prediction and true class values (prediction is correct) and p_e is the expected accuracy if predictions are made at random. It is a measure of agreement between predicted and expected values and is proportional to a balanced accuracy metric, but it is normalized so that random-guess classifiers have a score of 0, while perfect classifiers still have a score of 1. Classifiers with predictions that are on average worse than a random guess have a negative kappa.

The experiments of this section were performed using the BCI Competition IV dataset 2a (dataset D1) and the OpenBMI dataset (dataset D2), at different times. As both datasets have sessions performed on two different days, we use the first day as training (calibration) set and the second as testing data, as shown in Figure 5.1. For each subject of each dataset, the first and second sessions are selected. Then, for each ICA algorithm, the ICA model is calibrated, the sources are extracted and the features are calculated using Welch's spectrogram. The ICA method denoted as "None" represents a baseline scenario where no actual source extraction is done (the raw EEG channels are used), and is used to assess how ICA preprocessing compares with not using ICA. The features are scaled using a Z-score transformation, as follows:

$$v'_i = \frac{v_i - \bar{v}_i}{\sigma_i} \tag{5.2}$$

where v_i is the i-th feature, v'_i is the transformed i-th feature, $\bar{v}_i$ is the average value of v_i and σ_i is it's standard deviation. A smaller number of features are selected using the Wrapper technique, and each classifier is used in a grid search for the best hyperparameters, using a K-Fold cross-validation with 4 folds.

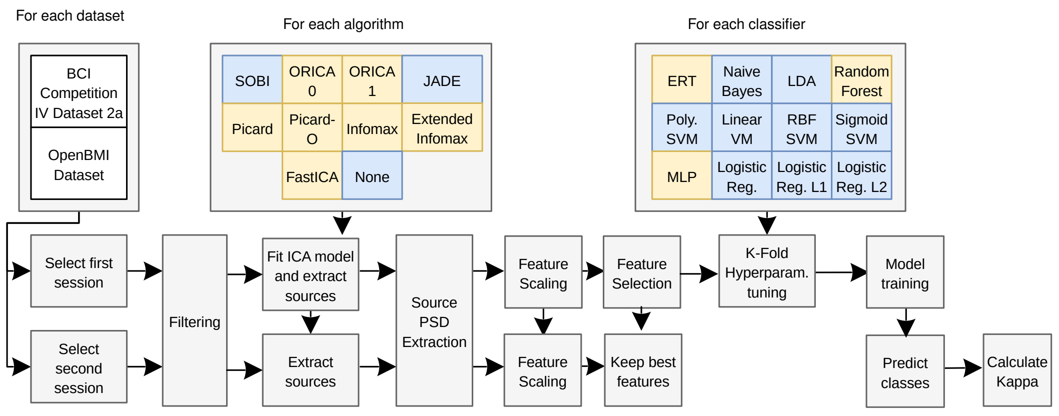


Figure 5.1 – Pipeline of the experiment.

The parameter space is shown in Table 5.1, representing the possible values that each parameter may assume during the optimization. During the hyperparameter search, all combinations between all parameters are tested, for each classifier. Then, the parameter combination that yields the highest average kappa between the 4 validation sets is chosen, and the model is retrained using all training data with this fixed parameter set.

Classifier	**Parameter**	**Space**
LDA	-	-
Logistic Regression (L1)	regularization λ	20 values in the [0.01,1000] interval, equally log-spaced
Logistic Regression (L2)	regularization λ	20 values in the [0.01,1000] interval, equally log-spaced
Logistic Regression (unregularized)	-	-
MLP	hidden layer sizes	(-), (10), (6), (4), (10, 6), (6, 4), (4, 4)
	validation fraction	25%
	optimizer	Adam
	initial learning rate η	1e-3
	patience	50
	nonlinearity	ReLU, Sigmoid
Gaussian Naive Bayes	-	-
SVC - RBF Kernel	regularization C	6 values in the [0.1,1000] interval, equally log-spaced
	kernel coefficient γ	5 values in the [0.001,0.1] interval, equally log-spaced
SVC - Linear Kernel	regularization C	10 values in the [0.001,1000] interval, equally log-spaced
SVC - Poly. Kernel	regularization	6 values in the [0.001,0.1] interval, equally log-spaced
	kernel coefficient γ	4 values in the [0.001,0.1] interval, equally log-spaced
	degree q	2, 3
SVC - Sig. Kernel	regularization C	6 values in the [0.1,1000] interval, equally log-spaced
	kernel coefficient γ	6 values in the [0.001,0.1] interval, equally log-spaced
Random Forest	N. of trees N_T	10, 15, 25, 40
	number of features per tree	sqrt(D), $log_2(D)$
	maximum depth	2, 3
Extra Trees	N. of trees N_T	10, 20, 30, 40
	maximum depth	2, 3, 5

Table 5.1 – Hyperparameter search space

The MLP has at least two layers, the D-sized input layer and K-sized output layer, where D and K are the dimensionality of the feature vector and the number of classes (except for 2 classes, where K is 1). The hidden layer parameter controls the layers in between those two. For instance, "(-)" means no hidden layer, "(16, 4)" means one 16-neuron layer followed by a 4-neuron layer. The MLP training uses an early stopping technique to avoid overfitting: a fraction of the training data (validation fraction) is taken, and the model is trained on the remaining data. On each epoch, the model performance is evaluated on the validation set. If the metric does not improve after a number of training epochs (patience parameter), the training stops. This is done both to terminate the training when no more improvements are being achieved and to avoid overfitting by limiting the model adjustment to the training data.

Note that only training data is used to learn the separation matrix associated with ICA (the same unmixing matrix for all MI classes), to calculate the mean and standard deviation for the z-score transform, to select the features using the Wrapper, to find the classifier best parameters in the grid search and to fit the classifier. The following subsections show the results of the mentioned experiments for each one of the two datasets.

5.2.1 BCI Competition IV 2a

In this subsection, we present the results of experiments where we followed the described methodology for dataset D1. We will evaluate the best (in terms of Cohen's Kappa) ICA and classifier combination per subject, the best ICA for each classifier, the best classifier for each ICA, and the best overall classifier and ICA methods. Here, "best" means the highest in the kappa metric for the four MI classes classification. Results with no standard deviation indicate that the classifier and the ICA methods are deterministic (initial conditions do not affect the results).

5.2.1.1 The best combinations per subject

Table 5.2 shows the top-3 combinations of classifier and ICA for each subject in terms of kappa (average and std. over the 10 runs for each combination). We see that ORICA was in the top-3 for 4 out of 9 subjects, and logistic regression appears 7 times in the top-3 positions. ORICA had the highest kappa for 3 out of 9 subjects. Subject 3 had the highest kappa among subjects, and both linear and non-linear classifiers were tied for the highest accuracy. Results with zero standard deviation mean that both the classifier and ICA method are deterministic (initial conditions do not affect the results).

Table 5.2 – Top 3 classifier and ICA combinations, per subject in dataset D1.

Subject	Classifier	ICA	Kappa	
			mean	std
1	Log. Reg. (L1)	SOBI	0.574	-
	MLP	SOBI	0.545	0.017
	Log. Reg. (L2)	SOBI	0.542	-
2	Random Forest	Infomax	0.247	0.035
	Random Forest	Picard	0.239	0.029
	Random Forest	Picard-O	0.237	0.040
3	Log. Reg.	ORICA (1)	0.676	-
	Log. Reg. (L1)	ORICA (1)	0.676	-
	SVM (RBF)	ORICA (1)	0.676	-
4	SVM (Lin.)	Picard-O	0.284	0.013
	LDA	Picard-O	0.279	0.014
	Log. Reg. (L2)	Picard-O	0.279	0.009
5	Naïve Bayes	ORICA (1)	0.102	-
	Random Forest	JADE	0.094	0.021
	Random Forest	Infomax	0.094	0.040
6	Log. Reg. (L1)	SOBI	0.278	-
	Log. Reg. (L2)	SOBI	0.273	-
	Log. Reg.	SOBI	0.269	-
7	Log. Reg. (L1)	ORICA (0)	0.611	-
	Log. Reg. (L2)	ORICA (0)	0.602	-
	Log. Reg.	ORICA (0)	0.593	-
8	SVM (RBF)	Picard	0.626	0.014
	SVM (Lin.)	Picard	0.625	0.015
	Log. Reg.	Picard	0.624	0.003
9	SVM (Lin.)	SOBI	0.662	-
	SVM (RBF)	SOBI	0.662	-
	Log. Reg. (L2)	ORICA (1)	0.644	-

5.2.1.2 The best ICA per subject

Table 5.3 shows the average kappa for the top-3 best-performing ICA methods, averaged over all classifiers and runs, for each subject. ORICA had the top-3 kappa for 4 out of 9 subjects, while SOBI and Infomax had the top-1 kappa for 2 out of 9 subjects each. FastICA got at most to the 3rd best ICA for any subject. Note that in all cases the use of any of the ICA methods is better than using raw signals.

Table 5.3 – Top 3 ICAs per subject in dataset D1.

Subject	ICA	Kappa	
		mean	std
1	SOBI	0.521	0.030
	JADE	0.469	0.033
	FastICA	0.468	0.053
2	Infomax	0.185	0.053
	ORICA (0)	0.184	0.029
	Picard	0.183	0.038
3	ORICA (1)	0.643	0.035
	Picard-O	0.589	0.028
	FastICA	0.585	0.052
4	Picard	0.259	0.036
	Picard-O	0.258	0.032
	JADE	0.258	0.020
5	Infomax	0.071	0.040
	JADE	0.070	0.020
	Picard	0.070	0.030
6	SOBI	0.231	0.052
	Picard	0.184	0.032
	FastICA	0.164	0.058
7	ORICA (0)	0.527	0.078
	Picard	0.487	0.074
	ORICA (1)	0.485	0.084
8	Infomax	0.584	0.035
	Picard	0.581	0.058
	FastICA	0.575	0.038
9	ORICA (1)	0.613	0.040
	SOBI	0.601	0.063
	ORICA (0)	0.553	0.065

5.2.1.3 The best classifier per subject

Table 5.4 shows the average kappa for the top-3 best-performing classifiers, averaged over all ICA methods and runs, for each subject. Results may be a little negatively biased because the baseline source extraction is taken into consideration here. The logistic regression models are ranked as top-1 for 6 out of the 9 subjects, and always appear in the top-3. The tree-based ensembles only appear on top-3 for 1 subject, showing that linear classifiers are almost always the most suitable option.

Table 5.4 – Top 3 classifiers per subject.

Subject	Classifier	Kappa	
		mean	std
1	Log. Reg. (L2)	0.474	0.054
	Log. Reg.	0.473	0.045
	Log. Reg. (L1)	0.467	0.067
2	Random Forest	0.183	0.084
	SVM (RBF)	0.167	0.045
	Extra Trees	0.164	0.066
3	Log. Reg.	0.561	0.085
	Log. Reg. (L2)	0.561	0.082
	Log. Reg. (L1)	0.556	0.092
4	Log. Reg.	0.222	0.073
	Log. Reg. (L2)	0.221	0.076
	Log. Reg. (L1)	0.221	0.069
5	Log. Reg.	0.057	0.039
	LDA	0.056	0.038
	Log. Reg. (L2)	0.055	0.042
6	SVM (Sig.)	0.142	0.056
	LDA	0.137	0.063
	Log. Reg. (L2)	0.136	0.066
7	Log. Reg. (L1)	0.433	0.162
	Log. Reg.	0.425	0.147
	Log. Reg. (L2)	0.423	0.152
8	SVM (Lin.)	0.535	0.092
	Log. Reg.	0.531	0.085
	Log. Reg. (L2)	0.529	0.091
9	Log. Reg. (L2)	0.518	0.114
	Log. Reg.	0.518	0.108
	Log. Reg. (L1)	0.516	0.113

5.2.1.4 The best classifier per ICA - average over the dataset

Table 5.5 shows the classifier with the highest average kappa, for each ICA. ORICA (1) was the only method that had the best performance with a non-linear classifier. Apart from that, for no ICA method a SVM variant had the highest average kappa. Since the statistics are calculated among all subjects, those values can be interpreted as the expected results independently of the subject. JADE and SOBI methods were the ones that performed the worst, even for their best-case classifiers. This is interesting because Table 5.3 shows SOBI as a good choice for several subjects, showing that it has more unfavorable scenarios, which is corroborated by the fact that its median kappa is considerably smaller than its mean kappa.

Table 5.5 – Best classifier per ICA in dataset D1.

ICA	Classifier	Kappa		
		median	mean	std
Picard	Log. Reg. (L2)	0.498	0.380	0.199
FastICA	Log. Reg. (L2)	0.464	0.379	0.209
Picard-O	Log. Reg. (L1)	0.480	0.373	0.219
ORICA (1)	Random Forest	0.438	0.373	0.231
Infomax	Log. Reg. (L2)	0.438	0.356	0.197
Ext. Infomax	Log. Reg. (L1)	0.439	0.355	0.211
ORICA (0)	Log. Reg. (L1)	0.472	0.344	0.260
SOBI	Log. Reg. (L1)	0.278	0.344	0.229
JADE	Log. Reg.	0.278	0.291	0.147
None	LDA	0.259	0.248	0.177

Figure 5.2 extends the results from Table 5.5, by showing the results from all classifiers. The red asterisk below each bar indicates that the result is significantly smaller than the best classifier (Wilcoxon's one-sided test), for each ICA method. The logistic regression classifiers consistently appear in the top-3 positions, and it is interesting that for the ORICA (1) method, the random forest was statistically tied with many other classifiers.

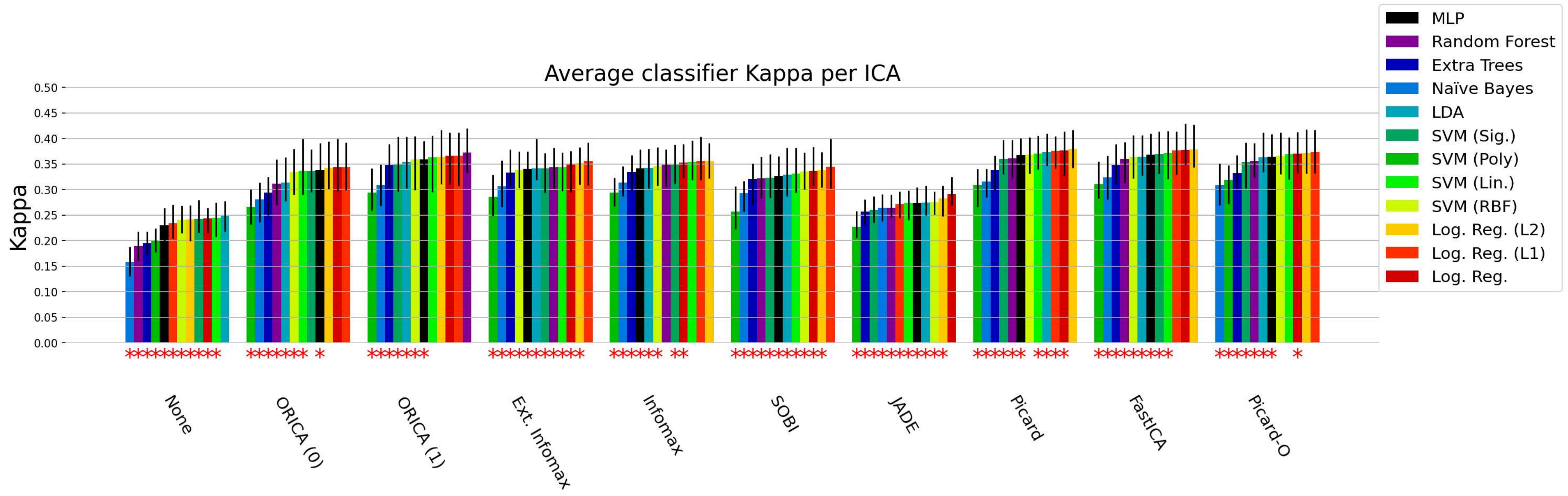


Figure 5.2 – Average kappa values for each classifier, per ICA method in dataset D1.

5.2.1.5 The best ICA per classifier - average over the dataset

Table 5.6 shows the ICA with the highest average kappa for each classifier. FastICA and Picard stood out having the highest kappa for 6 and 3 out of 12 classifiers, respectively. This is interesting, because, in Table 5.6, most of the cases present a mean value considerably lower than the median value, indicating a prevalence of scenarios where the kappa is higher than the mean value. Those values can be interpreted as the expected kappa value for the respective ICA and classifier combination independently of the subject, as in Table 5.5. Even though FastICA and Picard rarely appeared as top-1 in Table 5.3, here they appear as the best ICA for most of the classifiers, showing that even though they are not optimal for all classifiers on average, there are many scenarios they can be optimal.

Table 5.6 – Best ICA per classifier in dataset D1.

Classifier	ICA	Kappa		
		median	mean	std
Log. Reg. (L2)	Picard	0.498	0.380	0.199
Log. Reg.	FastICA	0.468	0.378	0.207
Log. Reg. (L1)	FastICA	0.486	0.377	0.207
LDA	Picard	0.477	0.373	0.187
Random Forest	ORICA (1)	0.438	0.373	0.231
SVM (Lin.)	FastICA	0.452	0.371	0.203
SVM (Sig.)	FastICA	0.460	0.369	0.205
SVM (RBF)	Picard	0.435	0.368	0.190
MLP	FastICA	0.448	0.368	0.198
Extra Trees	ORICA (1)	0.439	0.348	0.235
Naïve Bayes	FastICA	0.345	0.324	0.200
SVM (Poly)	Picard-O	0.377	0.319	0.194

Figure 5.3 extends the results from Table 5.6, by showing the results from all ICAs. The red asterisk below each bar indicates that the result is significantly smaller than the best classifier, for each classifier method. Picard, FastICA, and Picard-O consistently appear with the top-3 kappa values. Here we see that the tree-based methods perform well with ORICA, as they are the top-1 ICAs with those classifiers (even though they are statistically tied with some other source extraction methods).

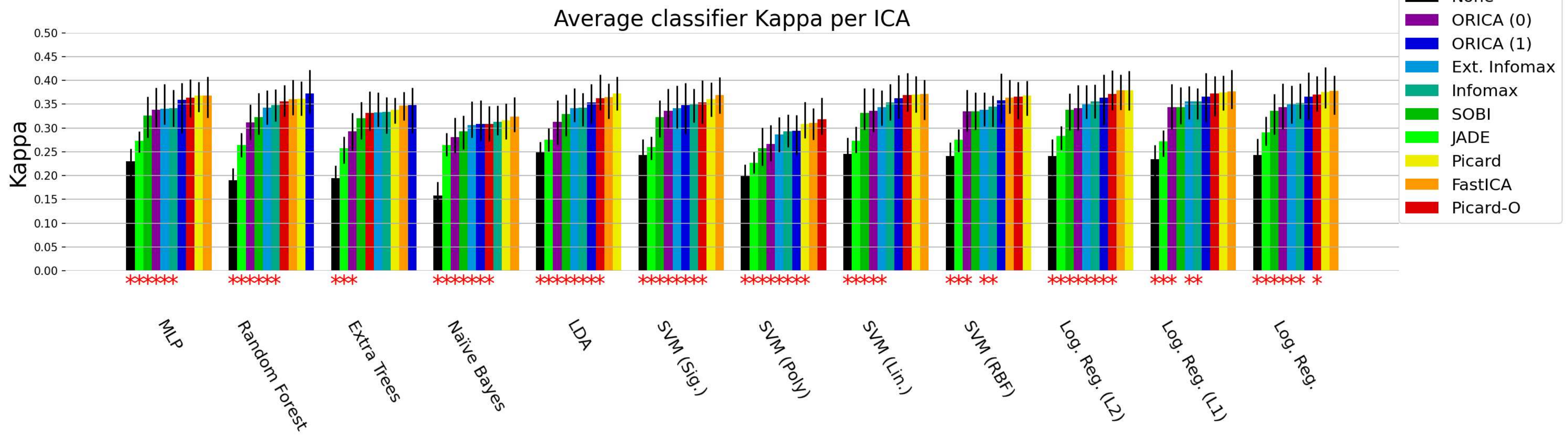


Figure 5.3 – Average kappa values for each ICA, per classifier method.

5.2.1.6 The best ICA method and the best classifier - average over the dataset

Figures 5.4 and 5.5 show the average kappa for each ICA method and classifier, respectively. The results are ordered from lowest to highest mean kappa and are averaged over all runs, subjects, and classifiers or ICA methods. Here, the red asterisk indicates that the one-sided Wilcoxon's signed-rank test between each bar and the highest (rightmost) bar had a p-value lower than 0.05, meaning that its value is significantly lower than the highest bar, showing that Picard and FastICA are tied for the best-performing ICA. The logistic regression classifiers had the highest average kappa among classifiers, with significant difference. In Uribe *et al.* (2016), JADE has shown great performance for a 2-class MI-BCI system, with better classification accuracy than SOBI and Infomax, contrasting with the results here, where JADE has been consistently a low accuracy ICA method. This could be due to differences in dataset or subject characteristics.

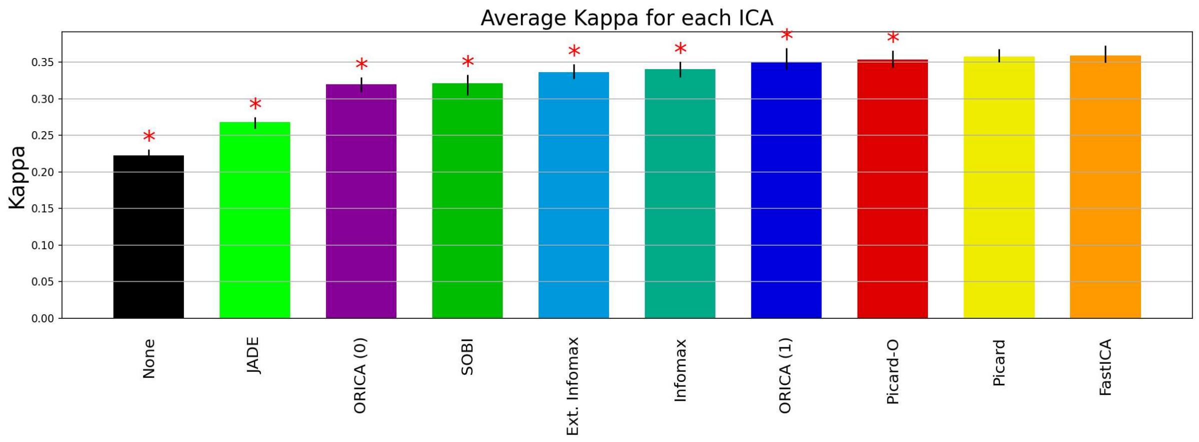


Figure 5.4 – Average kappa values for each ICA in dataset D1.

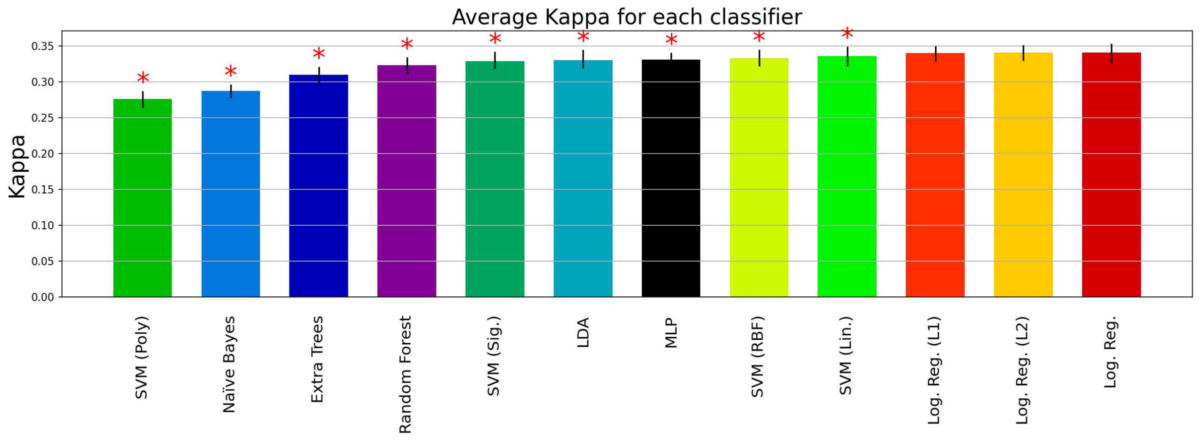


Figure 5.5 – Average kappa values for each classifier in dataset D1.

Even though LDA is commonly used as a classifier in BCI systems, we can note that the logistic regression model and its regularized variations are significantly better in terms of classification accuracy, especially in the case of using power spectral density features. It is also possible to conclude that, when using those features, it is always better to obtain the power spectral density of the ICs instead of raw EEG. This is expected as the known underlying phenomena for MI (the ERD/ERS) is known to be spatially and spectrally localized, and while the localization of the event may be difficult to extract, it is facilitated by the ICA filtering. Out of the many classifiers in our experiments, there was no significant gain when using non-linear approaches, as logistic regression, LDA and linear SVM were among the top average kappa values.

5.2.2 OpenBMI

Tables 5.7 and 5.8 show the best ICA and classifier combination for each subject, and the respective average kappa and its standard deviation. Column "N" shows the number of ICA and classifier combinations that yielded an average kappa with no significant difference from the best, as tested by Wilcoxon's one-sided test. This shows that even for some subjects that have high average kappa, there are few (and sometimes many) scenarios where the kappa value is statistically similar to the best scenario. Subject 29, for instance, has one of the best BCI controls even without source extraction, with 3 other combinations that would be statistically similar to it. Subject 21 has 35 ICA and classifier combinations that achieved similar performance than the best strategy, showing that for this subject the BCI calibration is more easily done, independently of the chosen source extraction and classifier.

5.2.2.1 The best classifier per ICA method - average over the dataset

Table 5.9 shows the classifier with the highest average kappa for each ICA. The results were averaged over all runs, for each ICA and classifier, and then the one with the highest kappa was selected. We see that the top-5 ICA have almost the same average kappa and are shown to have the logistic regression as the best option for the downstream classifier. Despite of that, FastICA shows a higher median value, indicating that most of the time it can yield higher kappa values than the other ICA methods, when we choose the right classifier. Overall, in all cases, we see a very high standard deviation, and this

Table 5.7 – Best ICA and classifier combination for each subject and number of combination with no significant difference, first part (dataset D2).

Subject	N	Classifier	ICA	Kappa mean	Kappa std
1	6	Log. Reg. (L2)	Ext. Infomax	0.385	0.034
2	3	Log. Reg. (L1)	None	0.220	-
3	25	Log. Reg. (L1)	FastICA	0.718	0.158
4	1	Naïve Bayes	ORICA (1)	0.190	-
5	1	MLP	None	0.482	0.024
6	20	Log. Reg.	Picard	0.634	0.041
7	52	Log. Reg.	Infomax	0.300	0.095
8	1	SVM (Lin.)	None	0.100	-
9	11	MLP	None	0.383	0.059
10	5	SVM (RBF)	None	0.150	-
11	3	SVM (Lin.)	ORICA (1)	0.070	-
12	1	SVM (Poly)	SOBI	0.180	-
13	1	SVM (RBF)	None	0.170	-
14	3	SVM (Lin.)	ORICA (1)	0.160	-
15	2	Naïve Bayes	ORICA (0)	0.100	-
16	1	LDA	ORICA (0)	0.250	-
17	4	Random Forest	FastICA	0.291	0.057
18	10	SVM (Poly)	SOBI	0.600	-
19	1	SVM (Sig.)	SOBI	0.340	-
20	23	Extra Trees	SOBI	0.133	0.042
21	35	Log. Reg. (L1)	Ext. Infomax	0.954	0.020
22	2	SVM (Sig.)	SOBI	0.450	-
23	31	Log. Reg.	Infomax	0.333	0.063
24	2	SVM (Sig.)	ORICA (0)	0.090	-
25	1	Log. Reg.	None	0.340	-
26	13	Extra Trees	ORICA (1)	0.110	0.071
27	1	SVM (Poly)	SOBI	0.150	-

is due to the high heterogeneity of the subjects in the dataset, as seen in Tables 5.7 and 5.8.

Figure 5.6 shows the average kappa over all runs, then over each ICA for each classifier. The asterisks indicate whether each result is significantly different than the highest kappa for the same classifier. We see that JADE and the ORICA methods consistently underperform in relation to the other ICA methods. We also see that the logistic regression methods are almost always on the top-3 classifiers. Differently from dataset D1, Random Forest appears on the top 3 classifiers 6 out of 12 times. Using no ICA, the best kappa value is similar to the kappa values of the least performing classifiers that were combined with ICA (except JADE and ORICA). We see that the difference between the highest and lowest kappas for each ICA method is of about 0.05.

Table 5.8 – Best ICA and classifier combination for each subject and number of combination with no significant difference, second part (dataset D2).

Subject	N	Classifier	ICA	Average kappa	std
28	6	Log. Reg. (L1)	SOBI	0.760	-
29	3	Log. Reg.	None	0.970	-
30	4	MLP	None	0.198	0.029
31	35	SVM (RBF)	Ext. Infomax	0.111	0.075
32	12	Random Forest	SOBI	0.751	0.029
33	11	SVM (Poly)	SOBI	0.850	-
34	1	SVM (RBF)	ORICA (1)	0.150	-
35	46	MLP	Ext. Infomax	0.371	0.218
36	11	Log. Reg.	Ext. Infomax	0.951	0.007
37	1	SVM (Lin.)	None	0.840	-
38	2	Naïve Bayes	ORICA (1)	0.090	-
39	24	MLP	Infomax	0.476	0.127
40	1	SVM (Sig.)	ORICA (1)	0.100	-
41	28	LDA	Picard	0.063	0.043
42	41	Random Forest	FastICA	0.173	0.077
43	1	Random Forest	SOBI	0.311	0.014
44	1	Log. Reg.	SOBI	0.960	-
45	24	Random Forest	Infomax	0.804	0.030
46	2	Log. Reg.	SOBI	0.450	-
47	3	SVM (RBF)	None	0.110	-
48	3	Random Forest	SOBI	0.198	0.085
49	1	SVM (Sig.)	None	0.340	-
50	1	SVM (Poly)	ORICA (0)	0.140	-
51	28	SVM (Sig.)	FastICA	0.283	0.024
52	29	Naïve Bayes	SOBI	0.330	-
53	3	Naïve Bayes	None	0.100	-
54	1	SVM (Poly)	ORICA (0)	0.180	-

Table 5.9 – Best classifier per ICA in dataset D2.

ICA	Classifier	Kappa		
		median	mean	std
Infomax	Log. Reg.	0.108	0.238	0.295
Picard	Log. Reg. (L1)	0.106	0.238	0.292
Picard-O	Log. Reg. (L1)	0.106	0.238	0.292
Ext. Infomax	Log. Reg. (L1)	0.091	0.238	0.292
FastICA	Log. Reg. (L1)	0.136	0.237	0.294
SOBI	Random Forest	0.091	0.231	0.294
None	Log. Reg. (L1)	0.085	0.209	0.278
ORICA (0)	SVM (Sig.)	0.040	0.095	0.224
ORICA (1)	Random Forest	0.036	0.077	0.195
JADE	MLP	0.013	0.064	0.149

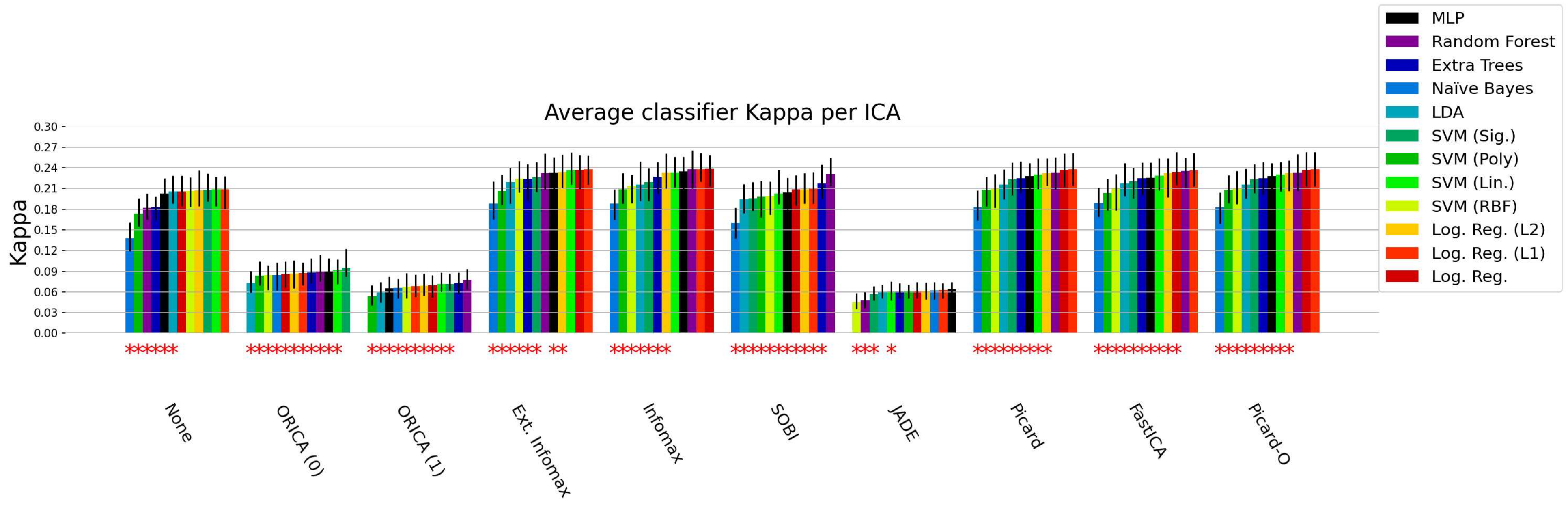


Figure 5.6 – Average and confidence interval over 10 runs of the kappa values for each classifier, for each ICA method in dataset D2.

Table 5.10 – Best ICA per classifier in dataset D2.

Classifier	ICA	Kappa median	Kappa mean	Kappa std
Log. Reg.	Infomax	0.108	0.238	0.295
Log. Reg. (L1)	Infomax	0.104	0.238	0.291
Random Forest	Infomax	0.135	0.238	0.291
SVM (Lin.)	Ext. Infomax	0.088	0.236	0.288
MLP	Infomax	0.088	0.235	0.289
Log. Reg. (L2)	Ext. Infomax	0.094	0.234	0.289
Extra Trees	Infomax	0.096	0.227	0.290
SVM (Sig.)	Ext. Infomax	0.099	0.226	0.284
SVM (RBF)	Ext. Infomax	0.075	0.224	0.283
LDA	Ext. Infomax	0.081	0.220	0.270
SVM (Poly)	Infomax	0.070	0.209	0.272
Naïve Bayes	FastICA	0.061	0.189	0.266

5.2.2.2 The best ICA per classifier - average over the dataset

Table 5.10 shows the ICA with the highest kappa, on average, for each classifier. It is noticeable that aggregating by the classifier, Infomax and its extended version perform better than the other ICA algorithms. Also, the average kappa values are much more homogeneous, so we can conclude that different classifiers have less influence on the final metric than the different ICA possibilities. The Random Forest classifier with Infomax ICA combination had a higher median kappa than others, but with no statistical difference when comparing to the first and second best combinations of the table (using Mood's median test, with 95% confidence level).

Figure 5.7 shows the average kappa over all runs, then over each classifier for each ICA method. The asterisks indicate whether each result is significantly different than the highest kappa for the same classifier. We see that most of the time, the Extended Infomax, Infomax, Picard-O, and Picard are almost tied in the top positions, indicating that independently of the downstream classifier, using any of those methods is almost similar. This was not so evident in dataset D1, as shown in Figure 5.3. It is also very interesting to see that, differently from dataset D1, the online ICA methods heavily underperform, as well as JADE.

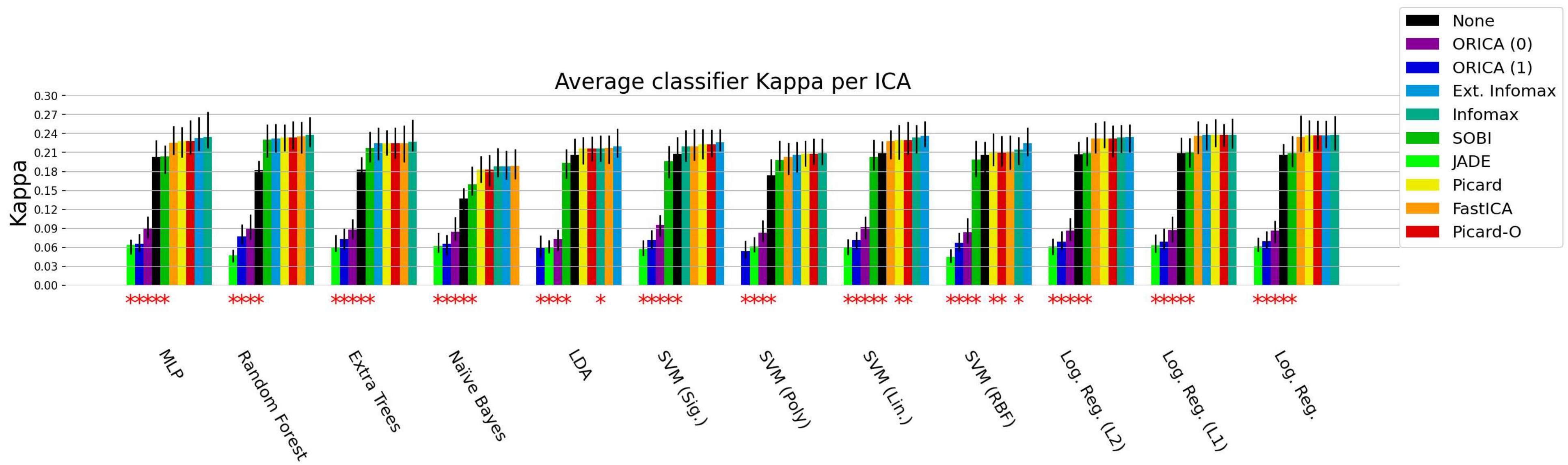


Figure 5.7 – Average and confidence interval over 10 runs of the kappa values for each ICA method, for each classifier.

5.2.2.3 The best ICA and best classifier - average over the dataset

Figure 5.8 presents the average kappa value for each ICA method. The kappa is averaged over all runs for each subject, classifier, and ICA, then over all classifiers and subjects, for each ICA. The red asterisk indicates statistically significant differences compared to the rightmost (highest) bar using Wilcoxon's one-sided signed-rank test. Extended Infomax is shown with the highest kappa, being significantly higher than all other ICA methods, while JADE appears in the last position with approximately one fourth of the kappa value of the first place ICA. Also, three ICA methods had the average kappa value lower than the baseline, which did not happen in dataset D1.

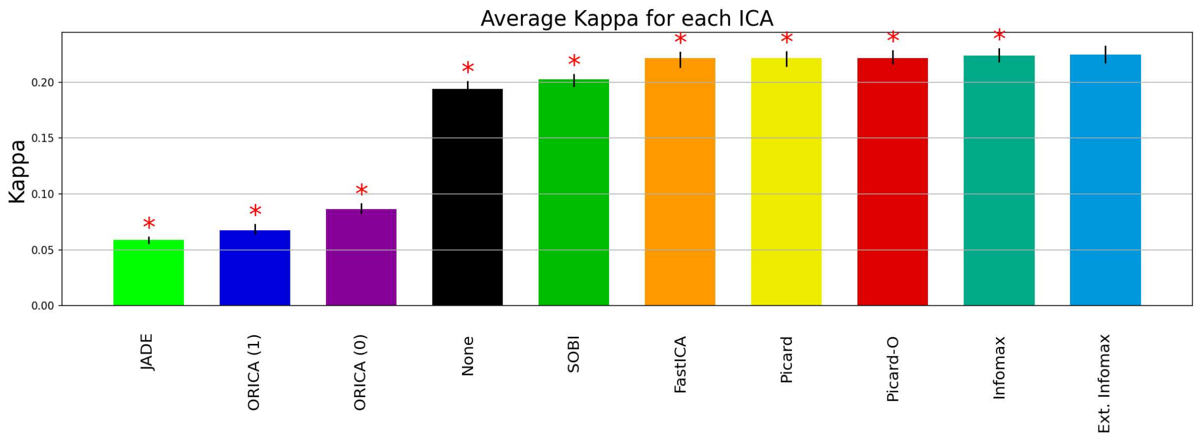


Figure 5.8 – Average kappa value for each ICA method, over all classifiers and subjects in dataset D2.

Figure 5.9 presents the averaged kappa values for all classifiers. Here, the kappa is averaged over all runs for each subject, classifier, and ICA, then over all ICAs and subjects, for each classifier. The red asterisk also indicates the classifiers with significant differences from the first one. The L1-regularized version of logistic regression significantly outperformed all other classifiers, but we see that the difference between the highest and lowest kappa classifiers (in this case, Naïve Bayes) is much lower than what happened with the ICAs. This indicates that the classifier choice is less important to the kappa value than the choice of the ICA method.

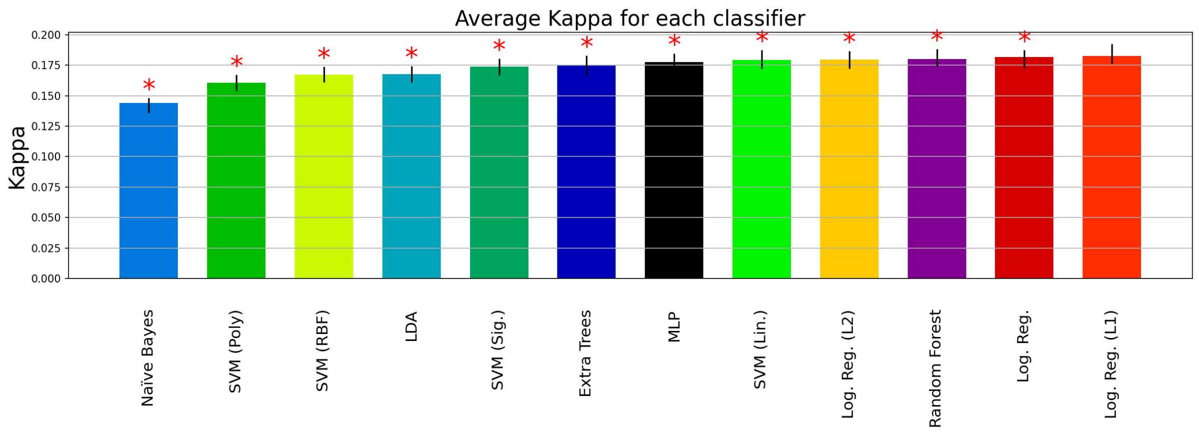


Figure 5.9 – Average kappa value for each ICA method, over all classifiers and subjects in dataset D2.

The results contrast with the ones obtained using dataset D1, as for many subjects the baseline ("none") preprocessing was the best. As for the classifier, the random forest performed better than most of the other algorithms, which was not seen in dataset D1. On the other side, the RBF kernel SVM went from the 5th position to the 9th position, from dataset D1 to D2 regarding average kappa. Interestingly, LDA stood at the 8th place. The naïve bayes classifier was the one with the lowest kappa, which could be expected since it is one of the classifiers with more broad assumptions (feature independence).

The ORICA algorithm degraded in performance compared to dataset D1, on average. Its kappa value is lower than the baseline, indicating that the method may not be suitable for the average subject, or that another preprocessing method is needed before applying the algorithm. Works by Brunner *et al.* (2007) and Naeem *et al.* (2006) compared SOBI, FastICA, and Infomax regarding classification accuracy in a 4-class MI task. In the testing phase, Infomax had the best accuracy, followed by FastICA and SOBI, which was validated by our results in D2. JADE rarely outperformed other methods, similar to the findings of Wu *et al.* (2020), where Infomax also had the top-2 classification accuracy among ICA methods.

5.3 EEGNet

EEGNet is a network that was designed for learning temporal filters, spatial filters, feature extraction, and classification in just one structure. This joint learning is generally better since all steps are optimized to minimize the same loss function in a supervised manner. It was designed to deal with EEG signals in the electrode space instead of the source space, which was considered in the last section. In this section, we feed signals in source space to this network and evaluate kappa and also training convergence speed. We assume that some sources extracted using ICA have a lot more information about the classification task we are training the BCI system to identify, so the network learning phase should be easier because we expect that at least the spatial filters will not need to be as complex as if we were using raw signals. The experiments were carried out using only the BCI Competition dataset due to the dataset size of D2. During EEGNet training we also used the early stopping strategy (as we use for the MLP), with a patience of 100 epochs.

5.3.1 The best ICA method per subject

Table 5.11 shows the average kappa for the top 3 best-performing ICA algorithms for each subject. The results are the average and standard deviation over 10 runs with different EEGNet weights initialization, while Figure 5.10 shows the kappa average and standard deviation for all ICAs, for each subject.

Figure 5.10 extends the results from Table 5.11, showing the average kappa value, along with the 95% confidence interval for each ICA algorithm and each subject. The red asterisk indicates that the respective ICA algorithm kappa value is significantly lower than the best ICA for the respective subject (one-sided Wilcoxon's signed-rank test). For Subjects 3, 6, 7, 8 and 9 using no ICA is always better than using any ICA, but with no statistical significant difference for subjects 7 and 8. For subjects 1, 2, 4, and 5 using ICA is statistically different from not using ICA.

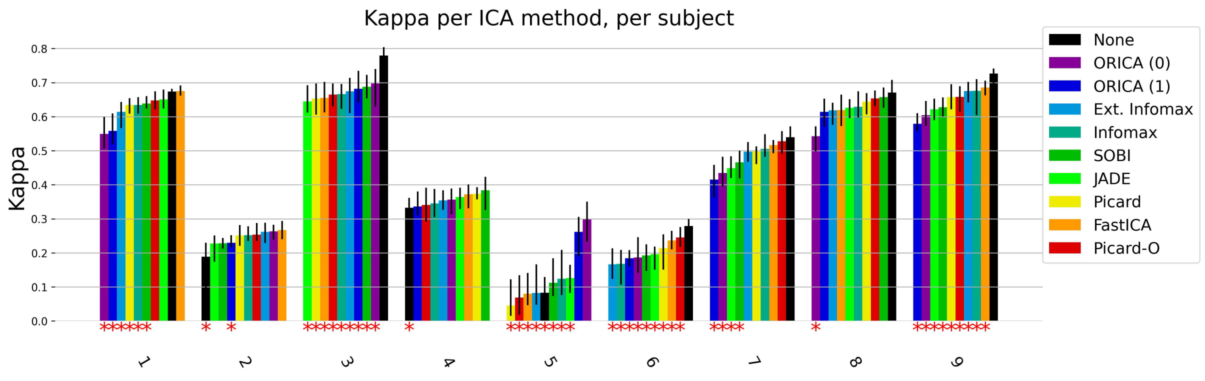


Figure 5.10 – Average kappa value for each ICA method, for each subject in dataset D2.

Table 5.11 – Top 3 ICA classifiers regarding average kappa value, for each subject in dataset D1.

Subject	ICA	Kappa	
		mean	std
1	FastICA	0.675	0.028
	None	0.674	0.021
	JADE	0.650	0.049
2	FastICA	0.267	0.048
	ORICA (0)	0.263	0.035
	Ext. Infomax	0.262	0.050
3	None	0.779	0.049
	ORICA (0)	0.697	0.087
	SOBI	0.688	0.057
4	SOBI	0.384	0.082
	Picard	0.374	0.038
	FastICA	0.372	0.065
5	ORICA (0)	0.300	0.098
	ORICA (1)	0.262	0.106
	JADE	0.126	0.082
6	None	0.279	0.041
	Picard-O	0.245	0.049
	FastICA	0.236	0.047
7	None	0.540	0.055
	Picard-O	0.527	0.059
	FastICA	0.517	0.032
8	None	0.670	0.060
	SOBI	0.657	0.057
	Picard-O	0.654	0.036
9	None	0.726	0.042
	FastICA	0.686	0.047
	Infomax	0.676	0.086

5.3.2 Training convergence speed per ICA

Table 5.12 shows the comparison between the best performing ICA, for each subject, with the baseline (no ICA). For subjects 2, 4 and 5 the kappa values are significantly greater than the baseline, and for subjects 2 and 4 there was a significant reduction on the number of needed training epochs. Overall, only for one subject the average number of epochs increased, and since it was an increase of only 6% it is not significant. Subjects with high baseline kappas had worsened results. The significance was measured using Wilcoxon's test at 5% confidence interval, with alternative hypothesis that ICA's kappa or number of epochs is greater.

Table 5.12 – Comparison of best ICA method and baseline average kappa and number of training epochs.

Subject	ICA	Baseline epochs mean	ICA epochs mean	Baseline kappa mean	Baseline kappa std	ICA kappa mean	ICA kappa std
1	FastICA	656.0	464.5	0.674	0.021	0.675	0.028
2	FastICA	368.0	214.0	0.189	0.087	0.267	0.048
3	ORICA (0)	924.0	607.5	0.779	0.049	0.697	0.087
4	SOBI	290.5	206.5	0.332	0.054	0.384	0.082
5	ORICA (0)	341.0	339.5	0.083	0.117	0.300	0.098
6	Picard-O	300.5	239.5	0.279	0.041	0.245	0.049
7	Picard-O	501.5	300.0	0.540	0.055	0.527	0.059
8	SOBI	605.5	559.5	0.670	0.060	0.657	0.057
9	FastICA	721.0	755.5	0.726	0.042	0.686	0.047

Table 5.13 shows the average kappa and number of training epochs over all subjects. We see an interesting correlation between number of epochs and average kappa. As we performed the training early stopping using a validation set, it is possible that the lower number of epochs harmed the learning phase for the ICA methods. This could be due to the additional need to learn the unmixing matrix and apply it to the unseen validation set, to which it is not generalizing well to, so the training phase is stopped. All ICA kappa results are significantly lower than the baseline, showing that, on average, using ICA harms EEGNet performance. FastICA was the algorithm that least decreased the average kappa, with a reduction of 4% but with a reduction in the necessary number of training epochs by 25.2%.

Table 5.13 – Average ICA method kappa and number of epochs until training convergence.

ICA	N. epochs mean	N. epochs std	Kappa mean	Kappa std
None	523.1	262.5	0.475	0.252
Picard	408.3	253.0	0.441	0.226
Ext. Infomax	399.4	232.9	0.438	0.226
FastICA	391.8	242.2	0.456	0.221
Infomax	391.2	221.7	0.444	0.224
SOBI	390.4	216.0	0.444	0.221
Picard-O	388.3	226.5	0.451	0.225
ORICA (0)	375.9	203.2	0.437	0.179
JADE	354.6	207.1	0.434	0.211
ORICA (1)	333.8	191.1	0.429	0.187

As we used early stopping for the EEGNet training, it is possible that training with more epochs would yield better kappa. Since the validation set is removed from training data, the potential results from EEGNet are not directly comparable to the experiments using the traditional classifiers, which used the entire training set.

We did not use the validation set for estimating the ICA unmixing matrix to avoid biasing the validation set metric estimation. If we view the ICA and the EEGNet as independent structures, it can be argued that we could have used both train and validation sets, since the early stopping is only required for the training of the EEGNet and not for the source separation matrix design.

5.4 Within- and between-session calibration of a BCI

In this study, we perform a comparative analysis of a Motor Imagery (MI) Brain-Computer Interface (BCI) calibrated with within-session data and between-session data (session-to-session transfer). We aim to quantify the impact of the latter in the accuracy of the BCI in comparison with the prior, and also compare the effect of different ICA techniques. This analysis is based on two freely accessible datasets, the BCI Competition IV Dataset 2A and the OpenBMI dataset.

We used the left and right-hand movement imagination tasks from the BCI Competition IV and OpenBMI datasets. Although both datasets differ in number of EEG channels, we used the same processing and classification steps for both. Each session data, on each day, was split into two. For the within-session experiment, the first half of the first session data is used for calibration and the second half is used for testing. In the between-session experiment, the same first half of the first day is used for calibration, and half of the other day is used for testing. This is done so the results are comparable, i.e., the model is calibrated on the same data, and only the test set changes. This data splitting is done for both datasets. The pipeline is run 20 times to account for ICA randomness, when necessary. In this experiment, we used all of the mentioned ICA methods but only used LDA as the classifier.

Figure 5.11 shows the within-session against the between-session experiment accuracy. The diagonal line is the identity line. Accuracies were averaged over all runs, then all subjects. Points below the identity line mean that the average accuracy for the within-session protocol is higher.

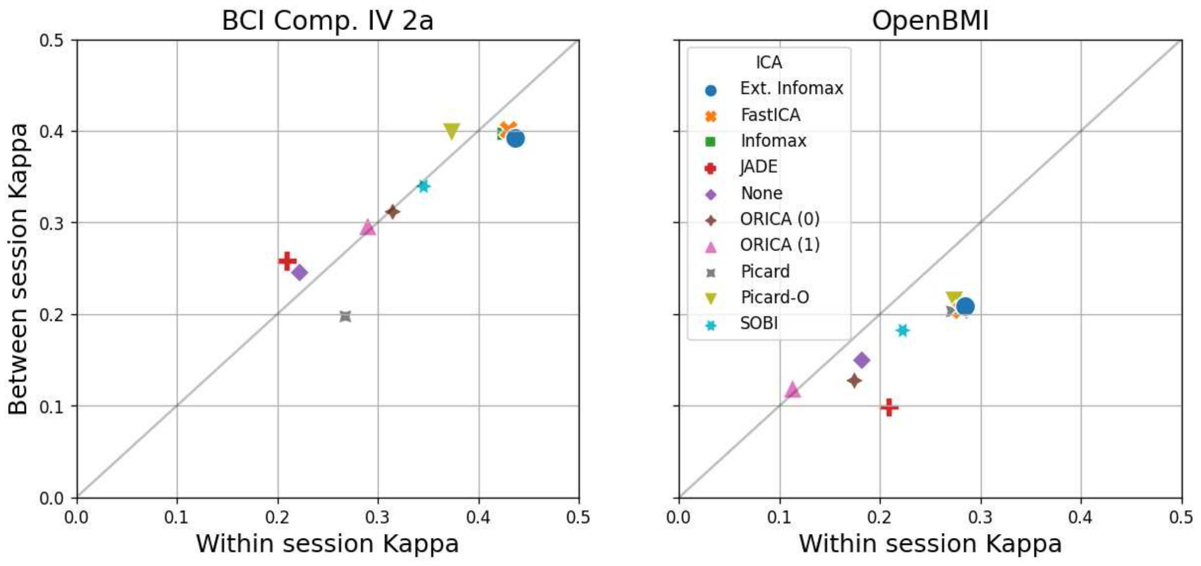


Figure 5.11 – Within-session vs between-session kappa results in a BCI calibration.

For both datasets, the Extended Infomax, Infomax and FastICA algorithms yielded very similar results. Those algorithms also seem to suffer performance loss in the between-session experiment. Additionally, JADE and the baseline preprocessing also present differences between both protocols, but for dataset D1 the kappa increases in the between-session experiment. In dataset D1, Picard, FastICA, Infomax, and Extended Infomax had a significant decrease in performance in the between-session case, while in dataset D2 all algorithms but ORICA (1) had a significant decrease. The significance test was done using Wilcoxon's signed-rank test, with the alternative hypothesis that the between-session kappa was lower (tested for all subjects and runs).

Figures 5.12 and 5.13 show boxplots of the average (over the 20 runs) subject performance for each ICA algorithm, at each protocol, for datasets D1 and D2, respectively. "None" refers to the baseline case, without source extraction. We see that for dataset D1 the distribution of the kappa values is similar between protocols, while for dataset D2 the difference is more pronounced. The kappa distribution for the baseline method is similar between the two protocols, but the kappa is smaller when compared to most of the other processing strategies.

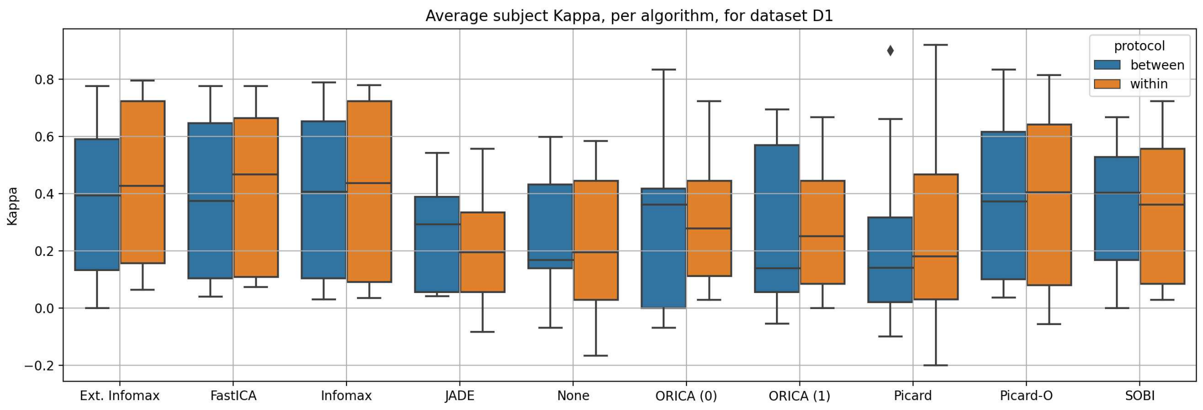


Figure 5.12 – Box-plot of subjects kappa results for each ICA, for dataset D1.

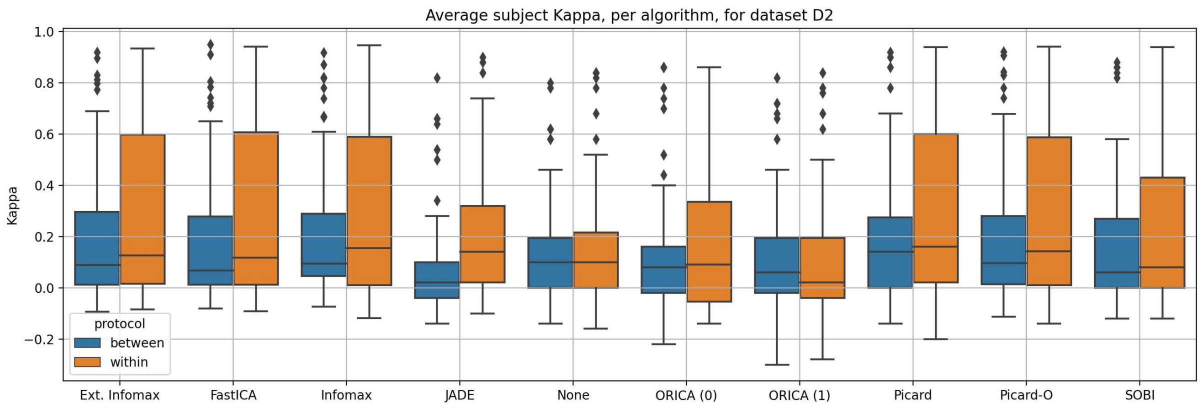


Figure 5.13 – Box-plot of subjects kappa results for each ICA, for dataset D2.

6 Conclusion

In this work, we presented a comparative analysis of ICA and classification methods in the context of MI-based BCI. The study had the aim of providing a systematic view of two key aspects of brain signal classification.

On subject specific combinations of ICA and classifiers for dataset D1, we can point logistic regression and Online Recursive Independent Component Analysis (ORICA) as the combination that appeared more times, and we note that ORICA appears on at least the top-3 combinations for 4 subjects. For dataset D2, we presented the top-1 classifier and ICA combination, where SOBI appeared at the top for 14 subjects, no ICA for 13 subjects, and the ORICA methods for 12 subjects. In dataset D1 the logistic regression was on a top-3 combination for 6 out of the 9 subjects. For dataset D2, SVM s achieved the top-1 combination performance for 21 out of 54 subjects (where the sigmoidal and polynomial kernels were at top-1 for 6 subjects each), followed by logistic regression (for 13 subjects, where the non-regularized version performed better for 8 subjects). Interestingly, LDA was outperformed by the other classification methods.

For dataset D1, when evaluating ICAs averaged over all classifiers and subjects we see that Picard appeared on the top-3 ICA methods for 6 subjects, followed by FastICA and ORICA, which appeared on the top-3 ICA methods for 4 subjects. Logistic regression models appeared in top-3 for 7 out of 9 subjects, followed by SVM (for 4 subjects). Overall, the highest average and median kappa combination was Picard with the L2-regularized logistic regression.

When averaging all results for each ICA in dataset D2, Extended Infomax was significantly better than all other methods, followed by Infomax and Picard-O. As for the classifiers, logistic Regression with L1 reg. significantly outperformed all other classifiers, followed by the non-regularized version and random forest. The combination that got the highest average kappa was the logistic regression with Infomax ICA, although with no significative difference to the L2-regularized version and to random forest, both also with Infomax algorithm. The performance for the top 5 best-case classifiers for each ICA is almost tied. The same happens for approximately the top 3 best-case ICA for each classifier. Interestingly, the random forest classifier achieves a much higher median kappa value than the other methods.

The EEGNet was originally proposed to deal with the EEG signals in the electrode space, i.e., with no source extraction. The learning of the spatial filters could theoretically be sped up since the source extraction possibly has separated the signals into interesting sources, non-interesting sources, and noise. If for some reason the source space does not have discriminatory signals, learning could be hampered. Finally, we note that the average kappa is higher when not using ICA, and the result matches the accuracy reported by Lawhern *et al.* (2018). The ICA method that achieved the highest average kappa value was FastICA, with a 5% kappa loss from the baseline scenario.

In the within-session and between-session comparison, datasets D1 and D2 yielded different between-session behaviors. In D1, the difference between both protocols was much more less pronounced, while for D2 most of the algorithms had a higher kappa decrease from within to the between-session scenario. Albeit ORICA (1) had no performance decrease from the within to the between session experiment, it also had one of the lowest kappas.

Overall, when using PSD features in an MI-BCI there is no ICA nor classification method that greatly stands out for all of the subjects. On dataset D2 SVMs and logistic regression models were selected the most times as the highest scoring classifiers, while SOBI and ORICA were in the top-1 combinations most of the times. For some subjects not using ICA was the best choice.

As a perspective for future work, we may indicate the extension of this analysis to other BCI paradigms (e.g. SSVEP and P300), considering, if possible, also real-time operation.

References

SAHA, S.; MAMUN, K. A.; AHMED, K.; MOSTAFA, R.; NAIK, G. R.; DARVISHI, S.; KHANDOKER, A. H.; BAUMERT, M. Progress in brain computer interface: Challenges and opportunities. *Frontiers in Systems Neuroscience*, Frontiers Media SA, v. 15, fev. 2021. ISSN 1662-5137. Available at: <http://dx.doi.org/10.3389/fnsys.2021.578875>. Cited in page 14.

ARMOUR, B. S.; COURTNEY-LONG, E. A.; FOX, M. H.; FREDINE, H.; CAHILL, A. Prevalence and causes of paralysis—united states, 2013. *American Journal of Public Health*, American Public Health Association, v. 106, n. 10, p. 1855–1857, out. 2016. ISSN 1541-0048. Available at: <http://dx.doi.org/10.2105/AJPH.2016.303270>. Cited in page 14.

BENSENOR, I. M.; GOULART, A. C.; SZWARCWALD, C. L.; VIEIRA, M. L. F. P.; MALTA, D. C.; LOTUFO, P. A. Prevalence of stroke and associated disability in brazil: National health survey - 2013. *Arquivos de Neuro-Psiquiatria*, FapUNIFESP (SciELO), v. 73, n. 9, p. 746–750, set. 2015. ISSN 0004-282X. Available at: <http://dx.doi.org/10.1590/0004-282X20150115>. Cited in page 14.

World Health Organization. *Spinal cord injury.* 2024. <https://www.who.int/news-room/fact-sheets/detail/spinal-cord-injury>. [Accessed 11-05-2024]. Cited in page 14.

CHAUDHARY, U.; BIRBAUMER, N.; RAMOS-MURGUIALDAY, A. Brain–computer interfaces for communication and rehabilitation. *Nature Reviews Neurology*, Springer Science and Business Media LLC, v. 12, n. 9, p. 513–525, ago. 2016. ISSN 1759-4766. Available at: <http://dx.doi.org/10.1038/nrneurol.2016.113>. Cited 2 times in pages 14 and 18.

BLANKERTZ, B.; TANGERMANN, M.; VIDAURRE, C.; FAZLI, S.; SANNELLI, C.; HAUFE, S.; MAEDER, C.; RAMSEY, L.; STURM, I.; CURIO, G.; MüLLER, K.-R. The berlin brain–computer interface: Non-medical uses of bci technology. *Frontiers in Neuroscience*, Frontiers Media SA, v. 4, 2010. ISSN 1662-4548. Available at: <http://dx.doi.org/10.3389/fnins.2010.00198>. Cited 2 times in pages 14 and 15.

WOLPAW, J.; BIRBAUMER, N.; MCFARLAND, D.; PFURTSCHELLER, G.; VAUGHAN, T. Brain–computer interfaces for communication and control. *Clinical Neurophysiology*, v. 113, n. 6, p. 767–791, 2002. ISSN 1388-2457. Cited in page 14.

WOLPAW, J. R.; MILLáN, J. del R.; RAMSEY, N. F. Chapter 2 - brain-computer interfaces: Definitions and principles. In: RAMSEY, N. F.; MILLáN, J. del R. (Ed.). *Brain-Computer Interfaces*. Elsevier, 2020, (Handbook of Clinical Neurology, v. 168). p. 15–23. Available at: <https://www.sciencedirect.com/science/article/pii/B9780444639349000020>. Cited in page 14.

KUBLER, A.; NIJBOER, F.; KLEIH, S. Hearing the needs of clinical users. In: _____. *Handbook of Clinical Neurology*. Elsevier, 2020. p. 353–368. Available at: <http://dx.doi.org/10.1016/B978-0-444-63934-9.00026-3>. Cited in page 15.

BRUNNER, C.; BIRBAUMER, N.; BLANKERTZ, B.; GUGER, C.; KüBLER, A.; MATTIA, D.; MILLáN, J. d. R.; MIRALLES, F.; NIJHOLT, A.; OPISSO, E.; RAMSEY, N.; SALOMON, P.; MüLLER-PUTZ, G. R. Bnci horizon 2020: towards a roadmap for the bci community. *Brain-Computer Interfaces*, Informa UK Limited, v. 2, n. 1, p. 1–10, jan. 2015. ISSN 2326-2621. Available at: <http://dx.doi.org/10.1080/2326263X.2015.1008956>. Cited in page 15.

VIDAURRE, C.; BLANKERTZ, B. Towards a cure for bci illiteracy. *Brain Topography*, Springer Science and Business Media LLC, v. 23, n. 2, p. 194–198, nov. 2009. ISSN 1573-6792. Available at: <http://dx.doi.org/10.1007/s10548-009-0121-6>. Cited in page 15.

LOTTE, F. Signal processing approaches to minimize or suppress calibration time in oscillatory activity-based brain–computer interfaces. *Proceedings of the IEEE*, v. 103, n. 6, p. 871–890, 2015. Cited in page 15.

MCFARLAND, D. J.; WOLPAW, J. R. Brain-computer interfaces for communication and control. *Communications of the ACM*, Association for Computing Machinery (ACM), v. 54, n. 5, p. 60–66, maio 2011. ISSN 1557-7317. Available at: <http://dx.doi.org/10.1145/1941487.1941506>. Cited in page 15.

REJER, I.; GóRSKI, P. Independent component analysis in a motor imagery brain computer interface. In: *IEEE EUROCON 2017 -17th International Conference on Smart Technologies*. [S.l.: s.n.], 2017. p. 126–131. Cited 3 times in pages 16, 21, and 39.

HAMANEH, M. B.; CHITRAVAS, N.; KAIBORIBOON, K.; LHATOO, S. D.; LOPARO, K. A. Automated removal of ekg artifact from eeg data using independent component analysis and continuous wavelet transformation. *IEEE Transactions on Biomedical Engineering*, v. 61, n. 6, p. 1634–1641, 2014. Cited 2 times in pages 16 and 30.

WINKLER, I.; HAUFE, S.; TANGERMANN, M. Automatic classification of artifactual ica-components for artifact removal in eeg signals. *Behavioral and Brain Functions*, Springer Science and Business Media LLC, v. 7, n. 1, p. 30, 2011. ISSN 1744-9081. Available at: <http://dx.doi.org/10.1186/1744-9081-7-30>. Cited 3 times in pages 16, 30, and 39.

KACHENOURA, A.; ALBERA, L.; SENHADJI, L.; COMON, P. Ica: a potential tool for bci systems. *IEEE Signal Processing Magazine*, v. 25, n. 1, p. 57–68, 2008. Cited 4 times in pages 16, 30, 43, and 49.

WU, X.; ZHOU, B.; LV, Z.; ZHANG, C. To explore the potentials of independent component analysis in brain-computer interface of motor imagery. *IEEE Journal of Biomedical and Health Informatics*, Institute of Electrical and Electronics Engineers (IEEE), v. 24, n. 3, p. 775–787, mar. 2020. ISSN 2168-2208. Available at: <http://dx.doi.org/10.1109/JBHI.2019.2922976>. Cited 6 times in pages 16, 30, 39, 43, 49, and 108.

URIBE, L. F. S. *New strategies for pre-processing, feature extraction and classification in BCI systems*. Tese (Doutorado) — Universidade Estadual de Campinas, 2018. Available at: <http://dx.doi.org/10.47749/T/UNICAMP.2018.1090526>. Cited 5 times in pages 16, 18, 20, 21, and 33.

BRUNNER, C.; NAEEM, M.; LEEB, R.; GRAIMANN, B.; PFURTSCHELLER, G. Spatial filtering and selection of optimized components in four class motor imagery eeg data using independent components analysis. *Pattern Recognition Letters*, v. 28, n. 8, p. 957–964, 2007. ISSN 0167-8655. Available at: <https://www.sciencedirect.com/science/article/pii/S0167865507000116>. Cited 3 times in pages 16, 33, and 108.

KHADEMI, Z.; EBRAHIMI, F.; KORDY, H. M. A review of critical challenges in mi-bci: From conventional to deep learning methods. *Journal of Neuroscience Methods*, Elsevier BV, v. 383, p. 109736, jan. 2023. ISSN 0165-0270. Available at: <http://dx.doi.org/10.1016/j.jneumeth.2022.109736>. Cited 2 times in pages 16 and 60.

ALTAHERI, H.; MUHAMMAD, G.; ALSULAIMAN, M.; AMIN, S. U.; ALTUWAIJRI, G. A.; ABDUL, W.; BENCHERIF, M. A.; FAISAL, M. Deep learning techniques for classification of electroencephalogram (eeg) motor imagery (mi) signals: a review. *Neural Computing and Applications*, Springer Science and Business Media LLC, v. 35, n. 20, p. 14681–14722, ago. 2021. ISSN 1433-3058. Available at: <http://dx.doi.org/10.1007/s00521-021-06352-5>. Cited 2 times in pages 16 and 61.

VIANA, P.; ATTUX, R.; CARVALHO, S. N. Comparison between online and offline independent component analysis in the context of motor imagery-based brain-computer interface. In: MARQUES, J. L. B.; RODRIGUES, C. R.; SUZUKI, D. O. H.; NETO, J. M.; OJEDA, R. G. (Ed.). *IX Latin American Congress on Biomedical Engineering and XXVIII Brazilian Congress on Biomedical Engineering*. [S.l.]: Springer Nature Switzerland, 2024. p. 302–312. ISBN 978-3-031-49404-8. Cited in page 17.

VIANA, P.; ATTUX, R.; CARVALHO, S. N. Analysis of between-session and within-session calibration of a mi-bci sytem. *Journal of Epilepsy and Clinical Neurophysiology*, v. 28, n. 1, p. 6–7, abr. 2023. ISSN 1676-2649. Cited in page 17.

BERGER, H. Über das elektrenkephalogramm des menschen. *Archiv für Psychiatrie und Nervenkrankheiten*, Springer Science and Business Media LLC, v. 87, n. 1, p. 527–570, dez. 1929. ISSN 1433-8491. Available at: <http://dx.doi.org/10.1007/BF01797193>. Cited in page 18.

BAMDAD, H. Z. M.; AUAIS, M. A. Application of bci systems in neurorehabilitation: a scoping review. *Disability and Rehabilitation: Assistive Technology*, Taylor Francis, v. 10, n. 5, p. 355–364, 2015. PMID: 25560222. Available at: <https://doi.org/10.3109/17483107.2014.961569>. Cited in page 18.

NOWLIS, D. P.; KAMIYA, J. The control of electroencephalographic alpha rhythms through auditory feedback and the associated mental activity. *Psychophysiology*, Wiley, v. 6, n. 4, p. 476–484, jan. 1970. ISSN 1469-8986. Available at: <http://dx.doi.org/10.1111/j.1469-8986.1970.tb01756.x>. Cited in page 18.

VIDAL, J. Real-time detection of brain events in eeg. *Proceedings of the IEEE*, Institute of Electrical and Electronics Engineers (IEEE), v. 65, n. 5, p. 633–641, 1977. ISSN 0018-9219. Available at: <http://dx.doi.org/10.1109/PROC.1977.10542>. Cited in page 18.

PAN, J.; CHEN, X.; BAN, N.; HE, J.; CHEN, J.; HUANG, H. Advances in p300 brain–computer interface spellers: toward paradigm design and performance evaluation. *Frontiers in Human Neuroscience*, Frontiers Media SA, v. 16, dez. 2022. ISSN 1662-5161. Available at: <http://dx.doi.org/10.3389/fnhum.2022.1077717>. Cited in page 18.

LIU, B.; CHEN, X.; SHI, N.; WANG, Y.; GAO, S.; GAO, X. Improving the performance of individually calibrated ssvep-bci by task- discriminant component analysis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, v. 29, p. 1998–2007, 2021. Cited in page 18.

ABIRI, R.; BORHANI, S.; SELLERS, E. W.; JIANG, Y.; ZHAO, X. A comprehensive review of eeg-based brain–computer interface paradigms. *Journal of Neural Engineering*, IOP Publishing, v. 16, n. 1, p. 011001, jan. 2019. ISSN 1741-2552. Available at: <http://dx.doi.org/10.1088/1741-2552/aaf12e>. Cited in page 18.

SINGH, A.; HUSSAIN, A.; LAL, S.; GUESGEN, H. A comprehensive review on critical issues and possible solutions of motor imagery based electroencephalography brain-computer interface. *Sensors*, MDPI AG, v. 21, n. 6, p. 2173, mar. 2021. Available at: <https://doi.org/10.3390/s21062173>. Cited 3 times in pages 19, 20, and 21.

WOLPAW, J.; WOLPAW, E. W. *Brain–Computer Interfaces: Principles and Practice*. Oxford University Press, 2012. ISBN 9780195388855. Available at: <http://dx.doi.org/10.1093/acprof:oso/9780195388855.001.0001>. Cited 4 times in pages 19, 20, 23, and 24.

HALDER, S.; AGORASTOS, D.; VEIT, R.; HAMMER, E.; LEE, S.; VARKUTI, B.; BOGDAN, M.; ROSENSTIEL, W.; BIRBAUMER, N.; KüBLER, A. Neural mechanisms of brain–computer interface control. *NeuroImage*, Elsevier BV, v. 55, n. 4, p. 1779–1790, abr. 2011. ISSN 1053-8119. Available at: <http://dx.doi.org/10.1016/j.neuroimage.2011.01.021>. Cited in page 19.

PFURTSCHELLER, G.; NEUPER, C. Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, v. 89, n. 7, p. 1123–1134, 2001. Cited 3 times in pages 19, 21, and 60.

SANEI, S.; CHAMBERS, J. A. *EEG Signal Processing*. Hoboken, NJ: Wiley-Blackwell, 2007. Cited 3 times in pages 19, 26, and 27.

MCFARLAND, D. J. *Brain Topography*, Springer Science and Business Media LLC, v. 12, n. 3, p. 177–186, 2000. ISSN 0896-0267. Available at: <http://dx.doi.org/10.1023/a:1023437823106>. Cited in page 20.

PFURTSCHELLER, G.; NEUPER, C.; BRUNNER, C.; SILVA, F. L. da. Beta rebound after different types of motor imagery in man. *Neuroscience Letters*, Elsevier BV, v. 378, n. 3, p. 156–159, abr. 2005. ISSN 0304-3940. Available at: <http://dx.doi.org/10.1016/j.neulet.2004.12.034>. Cited in page 20.

SAIBENE, A.; CAGLIONI, M.; CORCHS, S.; GASPARINI, F. Eeg-based bcis on motor imagery paradigm using wearable technologies: A systematic review. *Sensors*, MDPI AG, v. 23, n. 5, p. 2798, mar. 2023. ISSN 1424-8220. Available at: <http://dx.doi.org/10.3390/s23052798>. Cited 4 times in pages 20, 21, 60, and 61.

TIBSHIRANI, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, [Royal Statistical Society, Wiley], v. 58, n. 1, p. 267–288, 1996. ISSN 00359246. Available at: <http://www.jstor.org/stable/2346178>. Cited in page 21.

GOUY-PAILLER, C.; CONGEDO, M.; BRUNNER, C.; JUTTEN, C.; PFURTSCHELLER, G. Nonstationary brain source separation for multiclass motor imagery. *IEEE Transactions on Biomedical Engineering*, Institute of Electrical and Electronics Engineers (IEEE), v. 57, n. 2, p. 469–478, fev. 2010. ISSN 1558-2531. Available at: <http://dx.doi.org/10.1109/TBME.2009.2032162>. Cited in page 21.

SAFITRI, A.; DJAMAL, E. C.; NUGRAHA, F. Brain-computer interface of motor imagery using ICA and recurrent neural networks. In: *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*. [S.l.: s.n.], 2020. p. 118–122. Cited 2 times in pages 21 and 39.

PRIYATNO, S. B.; PRAKOSO, T.; RIYADI, M. A. Classification of motor imagery brain wave for bionic hand movement using multilayer perceptron. *SINERGI*, Universitas Mercu Buana, v. 26, n. 1, p. 57, fev. 2022. ISSN 1410-2331. Available at: <http://dx.doi.org/10.22441/sinergi.2022.1.008>. Cited in page 21.

AGGARWAL, S.; CHUGH, N. Signal processing techniques for motor imagery brain computer interface: A review. *Array*, v. 1-2, p. 100003, 2019. ISSN 2590-0056. Available at: <https://www.sciencedirect.com/science/article/pii/S2590005619300037>. Cited 2 times in pages 21 and 60.

WELCH, P. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, Institute of Electrical and Electronics Engineers (IEEE), v. 15, n. 2, p. 70–73, jun. 1967. ISSN 0018-9278. Available at: <http://dx.doi.org/10.1109/TAU.1967.1161901>. Cited in page 21.

LOTTE, F.; BOUGRAIN, L.; CICHOCKI, A.; CLERC, M.; CONGEDO, M.; RAKOTOMAMONJY, A.; YGER, F. A review of classification algorithms for eeg-based brain–computer interfaces: a 10 year update. *Journal of Neural Engineering*, IOP Publishing, v. 15, n. 3, p. 031005, abr. 2018. ISSN 1741-2552. Available at: <http://dx.doi.org/10.1088/1741-2552/aab2f2>. Cited 4 times in pages 22, 23, 60, and 61.

BEKIRYAZICI, ; DEMIR, A.; YILMAZ, G. Feature selection and analysis eeg signals with sequential forward selection algorithm and different classifiers. In: *2020 28th Signal Processing and Communications Applications Conference (SIU)*. [S.l.: s.n.], 2020. p. 1–4. Cited in page 22.

KOHAVI, R.; JOHN, H. Wrappers for feature subset selection. *Artificial Intelligence*, Elsevier BV, v. 97, n. 1-2, p. 273–324, dez. 1997. Available at: <https://doi.org/10.1016/s0004-3702(97)00043-x>. Cited in page 22.

SANTAMARíA-VáZQUEZ, E.; MARTíNEZ-CAGIGAL, V.; VAQUERIZO-VILLAR, F.; HORNERO, R. Eeg-inception: A novel deep convolutional neural network for assistive erp-based brain-computer interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, v. 28, n. 12, p. 2773–2782, 2020. Cited in page 23.

LAWHERN, V. J.; SOLON, A. J.; WAYTOWICH, N. R.; GORDON, S. M.; HUNG, C. P.; LANCE, B. J. Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces. *Journal of Neural Engineering*, IOP Publishing, v. 15, n. 5, p. 056013, jul. 2018. ISSN 1741-2552. Available at: <http://dx.doi.org/10.1088/1741-2552/aace8c>. Cited 4 times in pages 23, 84, 88, and 116.

SCHIRRMEISTER, R. T.; SPRINGENBERG, J. T.; FIEDERER, L. D. J.; GLASSTETTER, M.; EGGENSPERGER, K.; TANGERMANN, M.; HUTTER, F.; BURGARD, W.; BALL, T. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human Brain Mapping*, Wiley, v. 38, n. 11, p. 5391–5420, ago. 2017. ISSN 1097-0193. Available at: <http://dx.doi.org/10.1002/hbm.23730>. Cited 2 times in pages 23 and 86.

THE NATIONAL INSTITUTE OF NEUROLOGICAL DISORDERS AND STROKE, NIH. *Brain Basics: The Life and Death of a Neuron*. 2023. Accessed: 22-02-2024. Available at: <https://www.ninds.nih.gov/health-information/public-education/brain-basics/brain-basics-life-and-death-neuron>. Cited 2 times in pages 23 and 27.

GALLEGO, J. A.; MAKIN, T. R.; MCDOUGLE, S. D. Going beyond primary motor cortex to improve brain–computer interfaces. *Trends in Neurosciences*, v. 45, n. 3, p. 176–183, 2022. ISSN 0166-2236. Available at: <https://www.sciencedirect.com/science/article/pii/S016622362100254X>. Cited in page 24.

KANDEL, E.; KOESTER, J. D.; MACK, S. H.; SIEGELBAUM, S. *Principles of Neural Science*. 6. ed. [S.l.]: McGraw-Hill Education/Medical, 2021. Cited 4 times in pages 24, 25, 26, and 27.

SHIMA, K.; TANJI, J. Neuronal activity in the supplementary and presupplementary motor areas for temporal organization of multiple movements. *Journal of Neurophysiology*, American Physiological Society, v. 84, n. 4, p. 2148–2160, out. 2000. ISSN 1522-1598. Available at: <http://dx.doi.org/10.1152/jn.2000.84.4.2148>. Cited in page 24.

TEPLAN, M. Fundamental of EEGnxvide measurement. *MEASUREMENT SCIENCE REVIEW*, v. 2, 01 2002. Cited in page 27.

KLEM, G. H.; LÜDERS, H.; JASPER, H. H.; ELGER, C. E. The ten-twenty electrode system of the international federation. the international federation of clinical neurophysiology. *Electroencephalography and clinical neurophysiology. Supplement*, v. 52, p. 3–6, 1999. Cited in page 27.

YEN, C.; LIN, C.-L.; CHIANG, M.-C. Exploring the frontiers of neuroimaging: A review of recent advances in understanding brain functioning and disorders. *Life*, MDPI AG, v. 13, n. 7, p. 1472, jun. 2023. ISSN 2075-1729. Available at: <http://dx.doi.org/10.3390/life13071472>. Cited in page 28.

PASZKIEL, S. Data acquisition methods for human brain activity. In: _____. *Studies in Computational Intelligence*. Springer International Publishing, 2019. p. 3–9. ISBN 9783030305819. Available at: <http://dx.doi.org/10.1007/978-3-030-30581-9_2>. Cited in page 28.

RAMADAN, R. A.; VASILAKOS, A. V. Brain computer interface: control signals review. *Neurocomputing*, Elsevier BV, v. 223, p. 26–44, fev. 2017. ISSN 0925-2312. Available at: <http://dx.doi.org/10.1016/j.neucom.2016.10.024>. Cited in page 28.

PFURTSCHELLER, G.; NEUPER, C.; FLOTZINGER, D.; PREGENZER, M. EEG-based discrimination between imagination of right and left hand movement. *Electroencephalography and Clinical Neurophysiology*, Elsevier BV, v. 103, n. 6, p. 642–651, dez. 1997. Available at: <https://doi.org/10.1016/s0013-4694(97)00080-1>. Cited in page 28.

HYVARINEN, A.; KARHUNEN, J.; OJA, E. *Independent Component Analysis*. John Wiley & Sons, Inc., 2001. Available at: <https://doi.org/10.1002/0471221317>. Cited 2 times in pages 29 and 31.

HYVARINEN, A. New approximations of differential entropy for independent component analysis and projection pursuit. In: *Neural Information Processing Systems*. [S.l.: s.n.], 1997. Cited 2 times in pages 30 and 37.

CARDOSO, J.-F.; SOULOUMIAC, A. Blind beamforming for non-gaussian signals. *IEE Proceedings F Radar and Signal Processing*, Institution of Engineering and Technology (IET), v. 140, n. 6, p. 362, 1993. Available at: <https://doi.org/10.1049/ip-f-2.1993.0054>. Cited 6 times in pages 30, 45, 46, 47, 48, and 49.

BELOUCHRANI, A.; ABED-MERAIM, K.; CARDOSO, J.-F.; MOULINES, E. A blind source separation technique using second-order statistics. *IEEE Transactions on Signal Processing*, v. 45, n. 2, p. 434–444, 1997. Cited 4 times in pages 30, 39, 41, and 42.

HYVARINEN, A. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, Institute of Electrical and Electronics Engineers (IEEE), v. 10, n. 3, p. 626–634, maio 1999. Available at: <https://doi.org/10.1109/72.761722>. Cited 3 times in pages 30, 36, and 38.

BELL, A.; SEJNOWSKI, J. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, MIT Press - Journals, v. 7, n. 6, p. 1129–1159, nov. 1995. Available at: <https://doi.org/10.1162/neco.1995.7.6.1129>. Cited 2 times in pages 30 and 33.

LEE, T.-W.; GIROLAMI, M.; SEJNOWSKI, J. Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources. *Neural Computation*, MIT Press - Journals, v. 11, n. 2, p. 417–441, fev. 1999. Available at: <https://doi.org/10.1162/089976699300016719>. Cited 3 times in pages 30, 33, and 36.

ABLIN, P.; CARDOSO, J.-F.; GRAMFORT, A. Faster independent component analysis by preconditioning with hessian approximations. *IEEE Transactions on Signal Processing*, Institute of Electrical and Electronics Engineers (IEEE), v. 66, n. 15, p. 4040–4049, ago. 2018. Available at: <https://doi.org/10.1109/tsp.2018.2844203>. Cited 3 times in pages 30, 54, and 55.

BAI, X.; WANG, X.; ZHENG, S.; YU, M. The offline feature extraction of four-class motor imagery eeg based on ica and wavelet-csp. In: *Proceedings of the 33rd Chinese Control Conference*. [S.l.: s.n.], 2014. p. 7189–7194. Cited 2 times in pages 30 and 39.

DELORME, A.; SEJNOWSKI, T.; MAKEIG, S. Enhanced detection of artifacts in eeg data using higher-order statistics and independent component analysis. *NeuroImage*, Elsevier BV, v. 34, n. 4, p. 1443–1449, fev. 2007. ISSN 1053-8119. Available at: <http://dx.doi.org/10.1016/j.neuroimage.2006.11.004>. Cited in page 30.

WANG, Y.; WANG, Y.-T.; JUNG, T.-P. Translation of eeg spatial filters from resting to motor imagery using independent component analysis. *PLoS ONE*, Public Library of Science (PLoS), v. 7, n. 5, p. e37665, maio 2012. ISSN 1932-6203. Available at: <http://dx.doi.org/10.1371/journal.pone.0037665>. Cited in page 30.

HAN, Y.; WANG, B.; LUO, J.; LI, L.; LI, X. A classification method for eeg motor imagery signals based on parallel convolutional neural network. *Biomedical Signal Processing and Control*, v. 71, p. 103190, 2022. ISSN 1746-8094. Available at: <https://www.sciencedirect.com/science/article/pii/S1746809421007874>. Cited in page 30.

AMARI, S.-i. Natural gradient works efficiently in learning. *Neural Computation*, v. 10, n. 2, p. 251–276, 1998. Cited 2 times in pages 32 and 34.

PAPOULIS, A. *Probability, Random Variables, and Stochastic Processes*. Third. Boston: [s.n.], 1991. Cited in page 32.

DELORME, A.; MAKEIG, S. Eeglab: an open source toolbox for analysis of single-trial eeg dynamics including independent component analysis. *Journal of Neuroscience Methods*, Elsevier BV, v. 134, n. 1, p. 9–21, mar. 2004. ISSN 0165-0270. Available at: <http://dx.doi.org/10.1016/j.jneumeth.2003.10.009>. Cited in page 33.

URIBE, L. S.; COSTA, T.; CARVALHO, S.; SORIANO, D.; SUYAMA, R.; ATTUX, R. Two-class motor imagery bci based on the combined use of ica and feature selection. In: . [S.l.: s.n.], 2016. Cited 3 times in pages 33, 49, and 100.

DELORME, A.; PALMER, J.; ONTON, J.; OOSTENVELD, R.; MAKEIG, S. Independent eeg sources are dipolar. *PLoS ONE*, Public Library of Science (PLoS), v. 7, n. 2, p. e30135, fev. 2012. ISSN 1932-6203. Available at: <http://dx.doi.org/10.1371/journal.pone.0030135>. Cited in page 33.

NAEEM, M.; BRUNNER, C.; LEEB, R.; GRAIMANN, B.; PFURTSCHELLER, G. Seperability of four-class motor imagery data using independent components analysis. *Journal of Neural Engineering*, IOP Publishing, v. 3, n. 3, p. 208–216, jun. 2006. Available at: <https://doi.org/10.1088/1741-2560/3/3/003>. Cited 2 times in pages 33 and 108.

GRAMFORT, A.; LUESSI, M.; LARSON, E.; ENGEMANN, D. A.; STROHMEIER, D.; BRODBECK, C.; GOJ, R.; JAS, M.; BROOKS, T.; PARKKONEN, L.; HÄMÄLÄINEN, M. S. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, v. 7, n. 267, p. 1–13, 2013. Cited in page 36.

GOLUB, G. H.; LOAN, C. F. V.; LOAN, C. V. *Matrix Computations*. 2. ed. Baltimore, MD: Johns Hopkins University Press, 1989. Cited 2 times in pages 42 and 47.

SAHONERO, G.; CALDERON, H. A comparison of sobi, fastica, jade and infomax algorithms. In: . [S.l.: s.n.], 2017. Cited in page 43.

PECCATI, G.; TAQQU, M. S. *Wiener Chaos: Moments, Cumulants and Diagrams.* Springer Milan, 2011. ISSN 2039-1471. ISBN 9788847016798. Available at: <http://dx.doi.org/10.1007/978-88-470-1679-8>. Cited in page 44.

LIU, D. C.; NOCEDAL, J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, Springer Science and Business Media LLC, v. 45, n. 1–3, p. 503–528, ago. 1989. ISSN 1436-4646. Available at: <http://dx.doi.org/10.1007/BF01589116>. Cited in page 52.

MORé, J. J.; THUENTE, D. J. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software*, Association for Computing Machinery (ACM), v. 20, n. 3, p. 286–307, set. 1994. ISSN 1557-7295. Available at: <http://dx.doi.org/10.1145/192115.192132>. Cited in page 52.

FRANK, G.; MAKEIG, S.; DELORME, A. A framework to evaluate independent component analysis applied to eeg signal: testing on the picard algorithm. In: *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM).* [S.l.: s.n.], 2022. p. 2009–2016. Cited in page 55.

AKHTAR, M.; JUNG, T.-P.; MAKEIG, S.; CAUWENBERGHS, G. Recursive independent component analysis for online blind source separation. In: *2012 IEEE International Symposium on Circuits and Systems (ISCAS).* [S.l.: s.n.], 2012. p. 2813–2816. Cited 2 times in pages 56 and 57.

CICHOCKI, A.; UNBEHAUEN, R.; RUMMERT, E. Robust learning algorithm for blind separation of signals. *Electronics Letters*, v. 30, p. 1386–1387, 1994. Available at: <https://api.semanticscholar.org/CorpusID:62303138>. Cited in page 56.

HSU, S.; MULLEN, T.; JUNG, T.; CAUWENBERGHS, G. Online recursive independent component analysis for real-time source separation of high-density EEG. In: *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society.* IEEE, 2014. Available at: <https://doi.org/10.1109/embc.2014.6944462>. Cited 2 times in pages 58 and 59.

ZHU, X. Natural gradient-based recursive least-squares algorithm for adaptive blind source separation. *Science in China Series F*, Science China Press., Co. Ltd., v. 47, n. 1, p. 55, 2004. Available at: <https://doi.org/10.1360/02yf0242>. Cited in page 58.

CARDOSO, J.-F.; LAHELD, B. Equivariant adaptive source separation. *IEEE Transactions on Signal Processing*, v. 44, n. 12, p. 3017–3030, 1996. Cited in page 58.

BISHOP, C. M. *Pattern Recognition and Machine Learning.* [s.n.], 2006. v. 4. Available at: <https://doi.org/10.1117/1.2819119>. Cited 10 times in pages 60, 61, 62, 66, 73, 76, 79, 80, 81, and 83.

BASHASHATI, A.; FATOURECHI, M.; WARD, R. K.; BIRCH, G. E. A survey of signal processing algorithms in brain–computer interfaces based on electrical brain signals. *Journal of Neural Engineering*, IOP Publishing, v. 4, n. 2, p. R32–R57, mar. 2007. ISSN 1741-2552. Available at: <http://dx.doi.org/10.1088/1741-2560/4/2/R03>. Cited in page 60.

ZHANG, H. The optimality of naive bayes. In: *The Florida AI Research Society.* [S.l.: s.n.], 2004. Cited in page 61.

HOSSEINI, M.-P.; HOSSEINI, A.; AHI, K. A review on machine learning for eeg signal processing in bioengineering. *IEEE Reviews in Biomedical Engineering*, Institute of Electrical and Electronics Engineers (IEEE), v. 14, p. 204–218, 2021. ISSN 1941-1189. Available at: <http://dx.doi.org/10.1109/RBME.2020.2969915>. Cited in page 61.

KHAN, R. A.; RASHID, N.; SHAHZAIB, M.; MALIK, U. F.; ARIF, A.; IQBAL, J.; SALEEM, M.; KHAN, U. S.; TIWANA, M. A novel framework for classification of two-class motor imagery eeg signals using logistic regression classification algorithm. *PLOS ONE*, Public Library of Science (PLoS), v. 18, n. 9, p. e0276133, set. 2023. ISSN 1932-6203. Available at: <http://dx.doi.org/10.1371/journal.pone.0276133>. Cited in page 61.

MOHAMMADI, E.; DANESHMAND, P. G.; KHORZOOGHI, S. M. S. M. Electroencephalography-based brain–computer interface motor imagery classification. *Journal of Medical Signals amp; Sensors*, Medknow, v. 12, n. 1, p. 40, 2022. ISSN 2228-7477. Available at: <http://dx.doi.org/10.4103/jmss.JMSS_74_20>. Cited in page 61.

SHARMA, R.; KIM, M.; GUPTA, A. Motor imagery classification in brain-machine interface with machine learning algorithms: Classical approach to multi-layer perceptron model. *Biomedical Signal Processing and Control*, v. 71, p. 103101, 2022. ISSN 1746-8094. Available at: <https://www.sciencedirect.com/science/article/pii/S1746809421006984>. Cited in page 62.

TIWARI, S.; GOEL, S.; BHARDWAJ, A. Machine learning approach for the classification of eeg signals of multiple imagery tasks. In: *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. [S.l.: s.n.], 2020. p. 1–7. Cited in page 62.

WAHID, M. F.; TAFRESHI, R. Improved motor imagery classification using regularized common spatial pattern with majority voting strategy. *IFAC-PapersOnLine*, v. 54, n. 20, p. 226–231, 2021. ISSN 2405-8963. Modeling, Estimation and Control Conference MECC 2021. Available at: <https://www.sciencedirect.com/science/article/pii/S2405896321022230>. Cited in page 62.

KUHN, H. W.; TUCKER, A. W. Nonlinear programming. In: *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*. Berkeley and Los Angeles: University of California Press, 1951. p. 481–492. Cited in page 63.

PLATT, J. Sequential minimal optimization : A fast algorithm for training support vector machines. *Microsoft Research Technical Report*, 1998. Available at: <https://api.semanticscholar.org/CorpusID:577580>. Cited in page 64.

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern Recognition*. 3. ed. San Diego, CA: Academic Press, 2006. Cited 2 times in pages 68 and 77.

SHAWE-TAYLOR, J.; CRISTIANINI, N. *Kernel Methods for Pattern Analysis*. [S.l.]: Cambridge University Press, 2004. Cited in page 69.

HERBRICH, R. *Learning Kernel Classifiers: Theory and Algorithms*. Cambridge, MA, USA: MIT Press, 2001. ISBN 026208306X. Cited in page 69.

VAPNIK, V. N. *The Nature of Statistical Learning Theory*. Springer New York, 2000. Available at: <https://doi.org/10.1007/978-1-4757-3264-1>. Cited in page 69.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning*. Springer New York, 2009. ISSN 2197-568X. ISBN 9780387848587. Available at: <http://dx.doi.org/10.1007/978-0-387-84858-7>. Cited 4 times in pages 70, 72, 73, and 75.

COHEN. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Routledge, 2013. ISBN 9780203774441. Available at: <http://dx.doi.org/10.4324/9780203774441>. Cited in page 72.

FUKUNAGA, K. Introduction to statistical pattern recognition-second edition. In: . [s.n.], 1990. Available at: <https://api.semanticscholar.org/CorpusID:59916814>. Cited in page 75.

HOWLAND, P.; JEON, M.; PARK, H. Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, v. 25, n. 1, p. 165–179, 2003. Cited in page 75.

HO, T. K. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, n. 8, p. 832–844, 1998. Cited in page 76.

BREIMAN, L.; FRIEDMAN, J. H.; OLSHEN, R. A.; STONE, C. J. *Classification and Regression Trees*. [S.l.]: Wadsworth, 1984. ISBN 0-534-98053-8. Cited 2 times in pages 76 and 77.

BREIMAN, L. *Machine Learning*, Springer Science and Business Media LLC, v. 45, n. 1, p. 5–32, 2001. ISSN 0885-6125. Available at: <http://dx.doi.org/10.1023/A:1010933404324>. Cited in page 78.

GEURTS, P.; ERNST, D.; WEHENKEL, L. Extremely randomized trees. *Machine Learning*, Springer Science and Business Media LLC, v. 63, n. 1, p. 3–42, mar. 2006. ISSN 1573-0565. Available at: <http://dx.doi.org/10.1007/s10994-006-6226-1>. Cited in page 78.

CHAN, T. F.; GOLUB, G. H.; LEVEQUE, R. J. Updating formulae and a pairwise algorithm for computing sample variances. In: *COMPSTAT 1982 5th Symposium held at Toulouse 1982*. Physica-Verlag HD, 1982. p. 30–41. ISBN 9783642514616. Available at: <http://dx.doi.org/10.1007/978-3-642-51461-6_3>. Cited in page 79.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, v. 2, n. 5, p. 359–366, 1989. ISSN 0893-6080. Available at: <https://www.sciencedirect.com/science/article/pii/0893608089900208>. Cited in page 84.

ROJAS, R. The backpropagation algorithm. In: _____. *Neural Networks*. Springer Berlin Heidelberg, 1996. p. 149–182. ISBN 9783642610684. Available at: <http://dx.doi.org/10.1007/978-3-642-61068-4_7>. Cited in page 84.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, Springer Science and Business Media LLC,

v. 323, n. 6088, p. 533–536, out. 1986. ISSN 1476-4687. Available at: <http://dx.doi.org/10.1038/323533a0>. Cited in page 84.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017. Cited in page 84.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Springer Science and Business Media LLC, v. 521, n. 7553, p. 436–444, maio 2015. ISSN 1476-4687. Available at: <http://dx.doi.org/10.1038/nature14539>. Cited in page 84.

IOFFE, S.; SZEGEDY, C. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. Cited in page 85.

CLEVERT, D.-A.; UNTERTHINER, T.; HOCHREITER, S. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. 2016. Cited in page 85.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 56, p. 1929–1958, 2014. Available at: <http://jmlr.org/papers/v15/srivastava14a.html>. Cited in page 85.

YAO, Y.; ROSASCO, L.; CAPONNETTO, A. On early stopping in gradient descent learning. *Constructive Approximation*, Springer Science and Business Media LLC, v. 26, n. 2, p. 289–315, abr. 2007. ISSN 1432-0940. Available at: <http://dx.doi.org/10.1007/s00365-006-0663-2>. Cited in page 86.

BRUNNER, C.; LEEB, R.; MULLER-PUTZ, G.; SCHLOGL, A.; PFURTSCHELLER, G. *BCI Competition IV dataset IIa at <https://www.bbci.de/competition/iv/desc_2a.pdf>*. 2004. Cited in page 88.

LEE, M.-H.; KWON, O.-Y.; KIM, Y.-J.; KIM, H.-K.; LEE, Y.-E.; WILLIAMSON, J.; FAZLI, S.; LEE, S.-W. EEG dataset and OpenBMI toolbox for three BCI paradigms: an investigation into BCI illiteracy. *GigaScience*, v. 8, n. 5, 01 2019. ISSN 2047-217X. Giz002. Available at: <https://doi.org/10.1093/gigascience/giz002>. Cited 2 times in pages 88 and 89.