POSITIVE MATRIX FACTORIZATION METHOD FOR IMPROVING EEG MOTOR IMAGERY CLASSIFICATION

By
Raúl A. Huertas Ávila

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
ELECTRICAL ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
May 2017

Approved by:

Vidya Manian, Ph.D. — Date
President, Graduate Committee

Emmanuel Arzuaga, Ph.D. — Date
Member, Graduate Committee

Shawn D. Hunt, Ph.D. — Date
Member, Graduate Committee

Pedro Vasquez Urbano, Ph.D. — Date
Representative of Graduate Studies

José G. Colóm, Ph.D. — Date
Chairperson of the Department

Abstract of Dissertation Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of Master of Science

POSITIVE MATRIX FACTORIZATION METHOD FOR IMPROVING EEG MOTOR IMAGERY CLASSIFICATION

By

Raúl A. Huertas Ávila

May 2017

Chair: Dr. Vidya Manian
Department: Electrical and Computer Engineering Department

Brain Computer Interface (BCI) is a system that is designed to translate a subject's thought into a signal that is interpreted by a device. A BCI provides a communication channel between the human brain and a computer, making possible different applications in the bio-engineering field. The Brain-Computer Interface field has been in constant improvement because of the development of applications for people in need. These systems, BCI systems, need to be user-friendly, manageable, efficient, and suited for people with disabilities or with any physical complication. Thus, this thesis is an effort to seek improvements for those applications, by experimenting Positive Matrix Factorization (PMF) for motor imagery classification. Motor imagery (MI) is a mental process by which a subject mentally simulates a given action. In other words, MI is the process by which a subject is thinking of moving a part of his/her body without moving it physically. Motor imagery classification is the process of classifying a subject's mental simulations. Current methods rely on Common Spatial Pattern (CSP), which can be used for two-class motor imagery classification. The limitations with current methods are the high dimensionality of the EEG data that curtails extraction of discriminatory features for classification. The method presented in this thesis is an essential part of a functioning BCI system; it determines discriminative spectral features using the PMF method. These features are used to train the Support Vector Machine (SVM) classifier. The mentioned classifier is tested using 10-Fold Cross-Validation. Results using different numbers of feature vectors and different number of samples are presented. A complexity analysis of the PMF algorithm is presented.

Resumen de Disertación Presentado a la Escuela Graduada
de la Universidad de Puerto Rico Como Requisito Parcial de los
Requerimientos para el grado de Maestría en Ciencias

MÉTODO DE FACTORIZACIÓN DE MATRICES POSITIVAS PARA MEJORAR LA CLASIFICACIÓN DE IMÁGENES MOTRICES DE SEÑALES EEG

Por

Raúl A. Huertas Ávila

Mayo 2017

Consejera: Dra. Vidya Manian
Departamento: Departamento de Ingeniería Eléctrica y Computadoras

El sistema de Interface Cerebro-Computadora (BCI, por sus siglas en inglés) está diseñado para traducir las intenciones de un sujeto en una señal de control que es reconocida por un dispositivo. Un sistema de BCI provee un canal de comunicación entre el cerebro humano y una computadora, posibilitando diferentes aplicaciones en el campo de la bioingeniería. El campo de BCI está constante mejoramiento debido al desarrollo de aplicaciones para personas necesitadas. Los sistemas de BCI deben ser fáciles de utilizar, manejables, eficientes y apto para personas con discapacidades o cualquier complicación física. Por lo tanto, esta tesis es un esfuerzo para buscar mejoras para esas aplicaciones, mediante la experimentación del método Factorización Matricial Positiva (PMF, por sus siglas en inglés) para mejorar la clasificación de imágenes motrices de señales de electroencefalograma (EEG). La imaginación motriz es un proceso mental en el que un sujeto simula mentalmente una acción. En otras palabras, es el proceso mediante el cual un sujeto está pensando en mover una parte de su cuerpo sin moverla físicamente. La clasificación de imágenes motrices es el proceso de clasificar las simulaciones mentales de un sujeto. Los métodos actuales se basan en el Patrón Espacial Común (CSP, por sus siglas en inglés), el cual puede ser utilizado para clasificar dos clases o grupos de imágenes motoras. Las limitaciones con los métodos actuales son la alta dimensionalidad de los datos que restringe la extracción de rasgos discriminatorios para el proceso de clasificación. El método presentado en esta tesis es una parte

esencial del funcionamiento del sistema de BCI; determina la extracción de características discriminatorias utilizando el método de PMF. Estas características se utilizan para entrenar el clasificador SVM (Support Vector Machine, por sus siglas en inglés). El clasificador mencionado es probado mediante Validación Cruzada de 10 plegados. Se presentan resultados utilizando diferentes números de vectores de características y diferentes números de muestras. Un análisis de complejidad computacional del algoritmo de PMF también es presentado.

This work is dedicated to my parents, my partner, my family, and all those who believe in me.

ACKNOWLEDGMENT

I would like to thank Dr. Vidya Manian and Graduate Committee Members for making possible this investigation. I would also like to thank Jaime Camacho, Orlando Nieves, and Fernando Arias for their constant support and advices. Special thanks to Carlos Fernández and Sandra Montalvo for their moral support and Victor Asencio for the technical support. Thanks to the Brain-Computer Interface Laboratory as well as the Laboratory for Applied Remote Sensing and Image Processing, University of Puerto Rico at Mayagüez for providing the software platforms, computers, and laboratory equipment for data processing.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AgCl - Silver Chloride

A/D - Analog to Digital

BCI - Brain Computer Interface

CMWT - Complex Morlet Wavelet Transform

CSP - Common Spatial Pattern

DFT - Discrete Fourier Transform

EEG - Electroencephalogram

ERD - Event-Related Desynchronization

FPN - Floating Point Numbers

FLOP - Floating Point Operations

GB - Gigabytes

MI - Motor Imagery

MB - Megabytes

NMF - Nonnegative Matrix Factorization

PMF - Positive Matrix Factorization

PTF - Positive Tensor Factorization

RBF - Radial-Basis Function

STFT - Short-Time Fourier Transform

SVM - Support Vector Machine

LIST OF SYMBOLS

Hz - Hertz.

s - Seconds.

δ - Delta Rhythm Frequencies

θ - Theta Rhythm Frequencies

α - Alpha Rhythm Frequencies

β - Beta Rhythm Frequencies

γ - Gamma Rhythm Frequencies

$\circledast$ - Hadamard Product

$\otimes$ **-** Kronecker Product

CHAPTER 1

INTRODUCTION

1.1 Motivation

The human brain is composed of thousands of neurons interconnected by synapses [37]. Each connection transmits one specific signal per second. Those signals are stochastic and non-stationary; therefore, the extraction of features or information becomes difficult because of the amount of signals generated by the human brain. It is very important to have new methods that make the extraction of features from these signals accurate, faster and easier. These methods or tools can help in the development of user-friendly technology that can process thoughts of people with physical disabilities.

Electroencephalography (EEG) is a device used to capture human brain signals over a short period of time. These signals are recorded from multiple electrodes placed on the scalp. Using the Event-Related Desynchronization (ERD) reflected in the motor and sensory cortex when a subject is only imagining movement, feature extraction and classification methods were developed for driving Brain Computer Interfaces (BCIs) with these signals. In this work, Positive Matrix Factorization method is used for the extraction of features and the Support Vector Machine classifier is applied to classify motor imagery signals. In this thesis work, we propose a simpler feature extraction method that reduces the EEG data to be analyzed. Since recorded EEG data is normally large, data reduction is essential in a computational complexity sense. Also, the algorithm should be able to run quickly in order to provide features of real-time BCI applications. Hence, a computational time and space complexity analysis of the algorithm is presented.

1.2 Outline

The outline of the thesis is as follows. Chapter 2 gives a brief description of the methods and algorithms presented in this document. The step by step process of the completion of the objective is explained in Chapter 3. A study of the feature extraction method and classifier applied to EEG signals as well as the comparison between these algorithms in terms of accuracy are given in Chapter 4. Also, a study of computational complexity in terms of Flops and time of execution are presented in Chapter 4. Finally, the conclusions and the direction for further development and improvement are given in Chapter 5.

1.3 Objectives

1.3.1 General Objectives

- Implementation of Positive Matrix Factorization Method for processing EEG signals.
- Analysis of Computational Complexity of PMF algorithm.
- Application to classification of EEG signals.

1.3.2 Specific Objectives

- Implement the most promising algorithm for PMF.
- Make a computational analysis of Positive Matrix Factorization in time and space.
- Use the Positive Matrix Factorization Method for binary motor imagery classification.

1.4 Previous Work

In previous work, a classification of EEG signal has been done by using Nonnegative Tensor Factorization (NTF) to determine discriminative spectral features. Tensor factorization related research called *Nonnegative Tensor Factorization For Continuous EEG Classification* has been done in [24]. In this work, the authors employed the NTF to determine discriminative spectral features and used the Viterbi algorithm to classify multiple mental tasks. Numerical experiments where developed using two data sets in BCI competition: Graz dataset and IDIAP dataset [2]. The Graz dataset involves left/right imagery hand movements and consists of 140-labelled trials for training and 140 unlabeled trials for testing. Each trial has duration of 9 seconds and imagination task is carried out for 6 seconds. It contains EEG acquired from three different channels with sampling frequency of 128Hz. The channels used for this study were C_3 and C_4 because event-related desynchronization (ERD) has contralateral dominance, (it has more discriminant information). On the other hand, the IDIAP dataset contains EEG data recorded from three normal subjects during four non-feedback sessions, which involves three tasks, including the imagination of repetitive self-paced left/right hand movements and the generation of words beginning with the same random letter. The subject performed a given task for about fifteen seconds and then switched randomly to another task at the operator's request. The data were provided in two ways: raw EEG signals (with sampling rate equals to 512Hz) recorded from 32 electrodes, and precomputed features.

Authors applied the method to Graz dataset in BCI competition II with single-trial classification of motor imagery task. The analysis procedure was different according to the dataset and the data structure. In Graz dataset, data form was changed from temporal to spectral by using Complex Morlet Wavelet Transform during the preprocessing. Nonnegative Matrix Factorization (NMF) was enough to extract the meaningful features. Classification results using NMF in [24] was 51.17% without the Viterbi Algorithm and 68.55% using the Viterbi Algorithm. On the other hand, with 4-way tensor that contains the class information is 70.24% and with 3-way tensor is 69.47% using the Viterbi Algorithm. However, BCI Competition III had a result of 68.65% using 2-way tensor, which is lower than the results mentioned above. According to the authors, NTF is more robust than NMF in finding the hidden patterns from noisy training data.

CHAPTER 2

THEORETICAL BACKGROUND

This chapter describes important concepts about EEG signals and the Brain Computer Interface (BCI) system, including the algorithms used for its implementation in this work. We used EEG signals as the input from the users to the system. The input signals are then transformed to time-frequency representation using STFT or CMWT. A spectrogram is used as the input of the Positive Matrix Factorization method to extract features, and the classification stage is implemented using Support Vector Machine (SVM).

2.1 Electroencephalography (EEG)

Electroencephalography (EEG) is an electrical activity of an individual's brain that can be collected using electrodes. It is used to capture the electrical activity of the brain with great temporal resolution. EEG are created by the electrical communication of millions of neural cells. EEG is the most widespread data recording modality, enabling more diverse research in neuroscience and bioengineering. An EEG recording system consists of electrodes, amplifiers, A/D converter, and a recording device. The electrodes, usually made of silver chloride (AgCl), acquire the electrical signal from the scalp. The use of conductive gel is important to reduce the impedance between the skin and the electrodes. The amplifier processes the analog signal from the electrodes by amplifying the amplitude of the EEG signals so that the A/D converter can digitize them in a more accurate way. Finally, the recording device is a computer that stores and displays the data for further processing.

Electroencephalogram is a measurement of the potential difference over time between the active electrode and the reference electrode. A third electrode, known as the ground electrode, is used to measure the differential voltage between the active electrodes and the reference points. The electrical activity is recorded in the order of microvolts. Electroencephalogram signals are stochastic and non-stationary signal, which means its spectrum changes with time, showing oscillations at a variety of frequencies. There are five different frequency bands in which EEG can be divided, called brain rhythms [13]. These are:

- Delta (δ) rhythms: 0.5Hz - 3.5Hz. These are associated with deep sleep and are common in newborns [12, 36].

- Theta (θ) rhythms: 3.5Hz - 7.5Hz. They are most common during sleep. This can be seen in infants and children but high θ rhythms on an awake adult is a sign of a brain disorder.

- Alpha (α) rhythms: 7.5Hz - 12.5Hz. They are normally seen best under mental inactivity and relaxation. Best seen with eyes closed. This will be the main focus frequency range for motor imagery classification.

- Beta (β) rhythms: 12.5Hz - 30.5Hz. This is most evidently seen in the frontal and central lobe area and are associated with mental engagement such as activity, busy and anxious thinking. Beta rhythms are desynchronized during real movement or motor imagery [26].

- Gamma (γ) rhythms: 30.5Hz -100Hz. These bands have very sharp waves, spikes and other non-sinusoidal activity. The presence of gamma waves in the brain activity of a healthy adult is related to certain motor functions or perceptions, such as visual and auditory stimuli [14]. Studies related to this band has been growing recently. Due to the high information transfer rate, it offers higher spatial specificity.

2.2 Brain-Computer Interface (BCI)

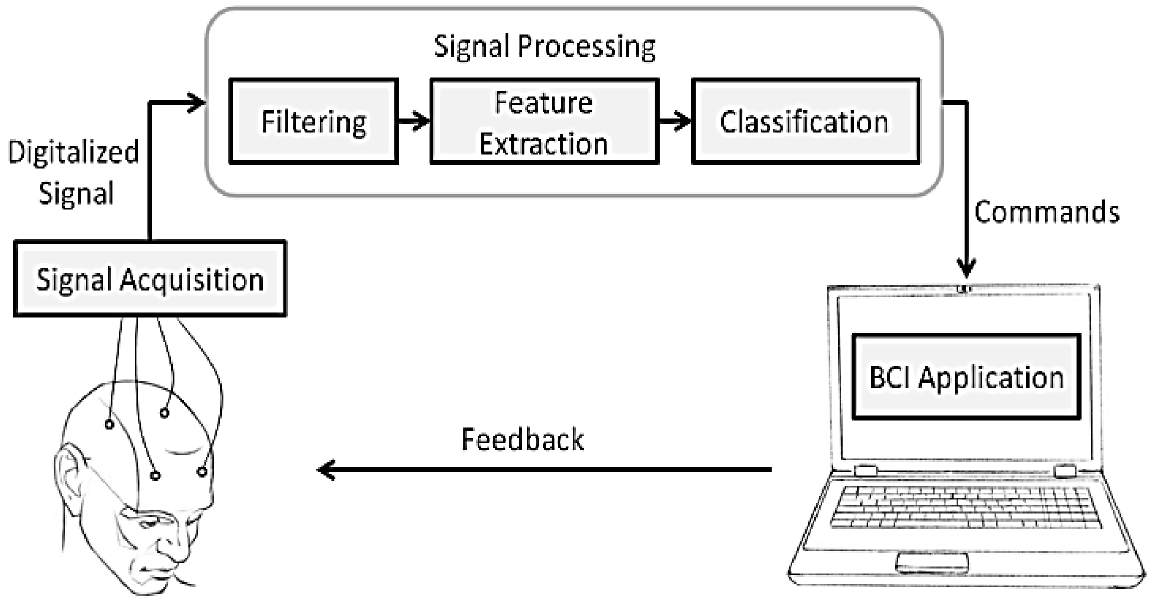

Figure 2.1: A Brain Computer Interface (BCI) System

A Brain Computer Interface (BCI) aims for communication pathways between the computer and device, based on neural activity generated by the brain. It is a system that is designed to translate a subject's intention or mind into a control signal for a device such as a computer [24]. Figure 2.1 shows a BCI where the EEG data acquired from the subject is processed and used to provide feedback through the display to the subject [38]. This way the human subject and the computer interact in performing different tasks. The task for this work is described as the imagination of left and right hand movements. The signal acquisition is done using bipolar EEG channels. The signal processing part has three general tasks: filtering, which is done by using time-frequency representations such as CMWT or STFT, feature extraction and classification, using both PMF and the SVM classifier respectively.

There are four main application areas of BCI systems in bioengineering. Applications for assisting disabled people, clinical monitoring for neurological diseases and sleeping disorders, behavioral neuroscience research with neural signals, and human-machine interaction are the main BCI systems applications. There are some important properties [13] to take in to account in order to design a BCI system.

Properties:

- Noise and Outliers: BCI features are noisy or contain outliers because EEG signals have a poor signal-to-noise ratio.

- High Dimensionality: in BCI systems, feature vectors are often of high dimensionality. Indeed, several features are generally extracted from several channels and from several time segments before being concatenated into a single feature vector.

- Time Information: BCI features should contain time information since brain activity patterns are generally related to specific time variations of EEG.

- Non-Stationarity: BCI features are non-stationary since EEG signals may rapidly vary over time and more specifically over sessions.

- Small Training Sets: The training sets are relatively small; since the training process is time consuming and demanding for the subjects (not have to be true for clinical use).

2.3 Short-Time Fourier Transform (STFT)

The spectral content of the EEG signals is non-stationary, which means that the signal changes over time. The Discrete Fourier Transform (DFT) is a mathematical operation that decomposes a waveform into a sum of sinusoid components, where the coefficients represent the correlation between the signal and the particular frequency sinusoid. But applying the DFT, along the signal does not reveal transitions in the spectra; it shows the frequencies that are present. For that reason, applying the DFT over short periods of time (regular intervals) known as STFT is used. The EEG signal can be considered as stationary. This approach allows the identification of the interval of time at which all frequencies are present in the signal. The discrete STFT is computed using a window function *w* centered at time *n*, given as [5]:

$$y(n,k) = \sum_{m=-\infty}^{\infty} x[m]w[n-m]e^{\frac{-j2\pi km}{N}} \tag{2.1}$$

where $x[m]$ is the signal to be analyzed, $w[n-m]$ is the window function and N is the frequency sampling factor. The resulting STFT is represented as a matrix with time and frequency $\omega = 2\pi k/N$ information. The size of the window has the effect of changing the time-frequency resolution, with a wider window better frequency resolution but lower time resolution and vice versa for a narrow window is obtained.

2.4 Complex Morlet Wavelet Transform (CMWT)

Another method for time-frequency analysis is the Complex Morlet Wavelet Transform, which is very popularly used with speech waveforms. For this study, the Morlet wavelet, a family of complex wavelet transforms will be implemented [5, 6]. Morlet wavelet analysis utilizes a flexible time window length for each frequency, with the largest window applied to the lowest frequencies and the smallest window applied to the highest frequencies resulting in a more accurate time-frequency resolution than STFT, which have fixed window at the cost of computational time. To compute the CMWT the signal is convolved with the mother wavelet $w_f(t,f)$. The Morlet Wavelet is composed of a complex exponential multiplied by a Gaussian window given by:

$$w_f(t,f) = \frac{1}{\sqrt{\pi f_b}} e^{jw_0 t} e^{-t^2/f_b}, \tag{2.2}$$

where $w_0 = 2\pi f_0$, the center frequency is f_0, and f_b is the frequency bandwidth. To satisfy the permissibility condition, w_o should satisfy $w_0 \geq 5$.

2.5 Positive Matrix Factorization (PMF)

Processing large amounts of data such as EEG data has problems with respect to data representation, disambiguation, and dimensionality reduction. There are methods utilized to address the representation of data and its dimensionality reduction. Factor analysis and Principal Component analysis are two of the many classical methods used to accomplish the goal of reducing the number of variables and detecting structures among the variables [6]. The use of low-rank approximations as mentioned earlier facilitates important applications in Bioengineering and Image Processing. Another analysis that is commonly used for data reduction and dimensionality reduction is the Positive Matrix Factorization method.

Positive Matrix Factorization (PMF) is a linear data model which is useful in handling nonnegative data. It determines basis vectors which will reflect meaningful spectral characteristics in motor imagery EEG tasks. The nonnegative data consists of $\boldsymbol{T}$ measurements of $\boldsymbol{N}$ nonnegative scalar variables. A linear approximation of the data is given by

$$a^t \approx \sum_{i=1}^{M} w_i h_i^t = W h^t, \tag{2.3}$$

where $\boldsymbol{a^t}$ $(t = 1, \dots, T)$ are $\boldsymbol{N}$-dimensional measurement vectors (vectors of data), $\boldsymbol{W}$ is a $N \times M$ matrix containing the basis vectors $\boldsymbol{w_i}$ as its columns. Each measurement vector is written in terms of the same basis vectors. The M basis vectors $\boldsymbol{w_i}$ are the building blocks of the data, and the M dimensional coefficient vector $\boldsymbol{h}^t$ describes how strongly each building block is present in the measurement vector $\boldsymbol{a^t}$. The measurements vectors $\boldsymbol{a^t}$ can be arranged into the columns of a $N \times T$ matrix $\boldsymbol{A}$. $\boldsymbol{A}$ is a nonnegative matrix with nonnegative matrix factors $\boldsymbol{W}$ and $\boldsymbol{H}$ such that:

$$\boldsymbol{A} \approx \boldsymbol{W}\boldsymbol{H}, \tag{2.4}$$

where each column of $\boldsymbol{H}$ contains the coefficient vector $\boldsymbol{h}^t$ corresponding to the measurement vector $\boldsymbol{v^t}$. The product $\boldsymbol{WH}$ is called a Positive Matrix Factorization of $\boldsymbol{A}$, although $\boldsymbol{A}$ is not necessarily equal to the product $\boldsymbol{WH}$. Written in this form, it becomes apparent that a linear data representation is simply a factorization of the data matrix. Positive Matrix Factorization requires all entries of $\boldsymbol{W}$ and $\boldsymbol{H}$ to be positive. $\boldsymbol{W}$ is a $N \times R$ nonnegative matrix and $\boldsymbol{H}$ is a $R \times T$ nonnegative matrix as well [15]. Usually, R is chosen to be smaller than N or T, so that $\boldsymbol{W}$ and $\boldsymbol{H}$ are smaller than $\boldsymbol{A}$. Therefore, this results in a compressed version of the original data matrix $\boldsymbol{A}$.

To find the approximate factorization of (2.4), we need to define a cost function that quantifies the quality of the approximation. One useful measure of a cost function is the square of the Euclidean Distance between $\boldsymbol{A}$ and $\boldsymbol{WH}$:

$$\frac{1}{2}\|\boldsymbol{A}-\boldsymbol{WH}\|_F^2 \tag{2.5}$$

Euclidean Distance is lower bounded by zero, so (2.5) vanishes if and only if $\boldsymbol{A}=\boldsymbol{WH}$. Now, Positive Matrix Factorization can be formulated as an optimization problem. The optimization problem to solve is defined as:

Problem: Given a nonnegative matrix $A \in \mathbb{R}^{N \times T}$ and a positive integer $R < \min(N, T)$, find positive matrices $W \in \mathbb{R}^{N \times R}$ and $H \in \mathbb{R}^{R \times T}$ to minimize the function

$$f(\boldsymbol{W},\boldsymbol{H}) = \frac{1}{2}\|\boldsymbol{A}-\boldsymbol{WH}\|_F^2 \tag{2.6}$$

The previous problem is a numerical optimization problem. $\|\boldsymbol{V}-\boldsymbol{WH}\|^2$ is a convex function in $\boldsymbol{W}$ only or in $\boldsymbol{H}$ only, it is not convex in both variables together. There are many numerical methods that minimizes (2.6) to extract underlying features as basis vectors in $\boldsymbol{H}$ such as gradient descent, conjugate gradient, convergence of gradient based, to mention a few, that can be applied to find the solution (local minima) for this problem. Authors explained the gradient descent method in [21], which is the simplest technique to implement but convergence of the solution is slow. This is not appropriate for applications that involve the manipulation of a big dataset. Conjugate gradient on the other hand have a faster convergence but is more complicated to implement than gradient descent.

For this work, two algorithms are used: Multiplicative Update Algorithm and Alternating Least Squares Algorithm. A Multiplicative Update Algorithm is a good compromise between speed and ease of implementation for solving (2.6). This algorithm for iteratively determining a local minimum of the previous problem is described below:

Multiplicative Update Algorithm For PMF:

$\boldsymbol{W} = rand(N, R)$; % $initialize$ $\boldsymbol{W}$ as random dense matrix

$\boldsymbol{H} = rand(R, T)$; % $initialize$ $\boldsymbol{H}$ as random dense matrix

$for\ i = 1{:}\,maxiter$

$$\boldsymbol{H} = \boldsymbol{H}.*\,(\boldsymbol{W}^T\boldsymbol{A})./(\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{H} + 10^{-9}) \tag{2.7}$$

$$\boldsymbol{W} = \boldsymbol{W}.*\,(\boldsymbol{A}\boldsymbol{H}^T)./(\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^{\boldsymbol{T}} + 10^{-9}) \tag{2.8}$$

end

The algorithm above is described using Matlab array operator notation, since Matlab is the software tool utilized to implement this work. Both (2.7) and (2.8) has a 10^{-9} factor added to avoid division by zero. $\boldsymbol{W}$ and $\boldsymbol{H}$ are strictly positive and naturally sparse, producing a "additive parts-based" representation of the data. Something important about the PMF is the initialization of $\boldsymbol{W}$ and $\boldsymbol{H}$ since the convergence of the algorithm mentioned above depends on initial conditions.

The Alternating Least Squares Algorithm on the other hand exploit the fact that, while optimization of equation (2.6) is not convex in both $\boldsymbol{W}$ and $\boldsymbol{H}$, it is convex in either $\boldsymbol{W}$ or $\boldsymbol{H}$. This algorithm is more flexible in a way that it allows the iterative process to escape from poor path. It means that if an element of $\boldsymbol{W}$ or $\boldsymbol{H}$ becomes 0, it must remain 0. The Alternating Least Squares Algorithm is described below:

Alternating Least Squares Algorithm For PMF:

$\boldsymbol{W} = rand(N, R);\% \; initialize \; \boldsymbol{W}$ as random dense matrix

$for \; i = 1\!:\!maxiter$

Solve for $\boldsymbol{H}$ in matrix equation $\boldsymbol{W^T W H} = \boldsymbol{W}^T \boldsymbol{A}$.

Set all negative elements in $\boldsymbol{H}$ to 0.

Solve for $\boldsymbol{W}$ in matrix equation $\boldsymbol{H H^T W^T} = \boldsymbol{H A}^T$.

Set all negative elements in $\boldsymbol{W}$ to 0.

end

PMF algorithms does not have a unique global minimum because of the initial conditions of matrices ***W*** and ***H***. One way to address the challenge of the initial conditions is to use Monte Carlo type approach [11] with different initial conditions to see which one gives the best results. The one with best results are selected for further numerical analysis such as features extraction. The initial conditions problem is an open topic in the research community. Even though, it is a problem for any PMF application, it is not the main goal of this work. A good initialization can improve the speed and accuracy of the algorithms mentioned (Multiplicative Update and Alternating Least Squares algorithms), as it can produce faster convergence to an improved local minimum [4]. For this work, the initial conditions are defined using random numbers with the Matlab tool.

2.6 Positive Matrix Factorization Numerical Example:

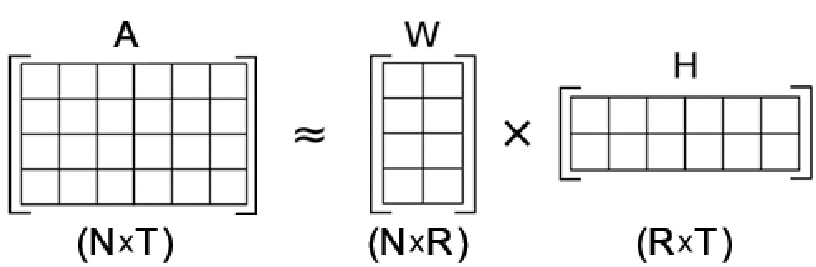

Figure 2.2: Positive Matrix Factorization Illustration

In this section, a basic example of the calculation of the Positive Matrix Factorization is presented. It is important to have a basic knowledge about the method presented in this thesis for understanding its applications to Motor Imagery Classification.

Let $A = \begin{bmatrix} 1 & 2 & 3 & 5 \\ 2 & 4 & 8 & 12 \\ 3 & 6 & 7 & 13 \end{bmatrix}$, we want to solve the factorization presented in Fig 3.1. Let $A_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and $A_3 = \begin{bmatrix} 3 \\ 8 \\ 7 \end{bmatrix}$ be the first and the third column of A, respectively. To solve, $A \approx WH$, the following should be made:

$$A_1 = 1 \times A_1 + 0 \times A_3$$

$$A_2 = 2 \times A_1 + 0 \times A_3$$

$$A_3 = 0 \times A_1 + 1 \times A_3$$

$$A_4 = 2 \times A_1 + 1 \times A_3$$

Now, W is composed by A_1 and A_3. In other words, $W = [A_1 \quad A_3] = \begin{bmatrix} 1 & 3 \\ 2 & 8 \\ 3 & 7 \end{bmatrix}$. On the other hand, H is composed by the coefficients of equations A_1, A_2, A_3 and A_4. Hence, $H = \begin{bmatrix} 1 & 2 & 0 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}$.

Thus, the final step of the procedure is the following:

$$A \approx WH = \begin{bmatrix} 1 & 3 \\ 2 & 8 \\ 3 & 7 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 0 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$A \approx WH = \begin{bmatrix} (1 \times 1 + 3 \times 0) & (1 \times 2 + 3 \times 0) & (1 \times 0 + 3 \times 1) & (1 \times 2 + 3 \times 1) \\ (2 \times 1 + 8 \times 0) & (2 \times 2 + 8 \times 0) & (2 \times 0 + 8 \times 1) & (2 \times 2 + 8 \times 1) \\ (3 \times 1 + 7 \times 0) & (3 \times 2 + 7 \times 0) & (3 \times 0 + 7 \times 1) & (3 \times 2 + 7 \times 1) \end{bmatrix}$$

$$A \approx WH = \begin{bmatrix} 1 & 2 & 3 & 5 \\ 2 & 4 & 8 & 12 \\ 3 & 6 & 7 & 13 \end{bmatrix}$$

2.7 Positive Tensor Factorization (PTF)

Positive Tensor Factorization is a multiway extension of Positive Matrix Factorization, where nonnegativity constraints are incorporated. Tensor refers to a multiway data array. To understand the concept of the manipulation of datasets, a vector is a 1-way tensor, a matrix is a 2-way tensor, and a cube is a 3-way tensor, and so on. Spectral EEG data can be represented by a tensor whose coordinates correspond to channel, class, trial, and so on. This is the case of the NTF. In the case of PMF, the data matrix is limited to only two coordinates.

Let $\mathfrak{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ be an *N*-way tensor with *N* indices $(i_1, i_2, \ldots, i_N)$. The elements of tensor $\mathfrak{X}$ are denoted by $x_{i_1, i_2, \ldots, i_N}$ where $1 \leq i_n \leq I_N$. Mode-*n* vectors of an *N*-way tensor $\mathfrak{X}$ are I_n-dimensional vectors obtained from $\mathfrak{X}$ by varying index i_n while keeping the other indices fixed [24]. In matrix, column vectors correspond to mode-1 vectors and row vectors correspond to mode-2 vectors. $\boldsymbol{X}_{(\boldsymbol{n})} \in \mathbb{R}^{I_n \times I_{n+1} \times I_{n+2} \times \ldots \times I_N \times I_1 \times I_2 \times \ldots \times I_{n-1}}$ is the mode-*n* matrix form of the tensor $\boldsymbol{\mathfrak{X}}$. The column vectors of matrix $\boldsymbol{X}_{(\boldsymbol{n})}$ are mode-*n* vectors. The mode-*n* matrix form of $\boldsymbol{\mathfrak{X}}$ is a PARAFAC model.

A PARAFAC model is a decomposition method for multi-way data that seeks the rank-*R* approximation of the tensor $\boldsymbol{\mathfrak{X}}$. The rank of the tensor $\boldsymbol{\mathfrak{X}}$, *R*, is the minimal number of rank-1 tensors that is required to yield $\boldsymbol{\mathfrak{X}}$. The approximation of this model is denoted as:

$$\boldsymbol{X}_{(\boldsymbol{n})} \approx \boldsymbol{W}^{(\boldsymbol{n})} \otimes \boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})}, \qquad (2.9)$$

where $\boldsymbol{W}^{(\boldsymbol{n})} \in \mathbb{R}^{I_n \times R}$, is the mode-n basis matrix and $\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})}$ is the mode-n encoding matrix (contains the features) and $R = rank(\mathfrak{X})$. The approximation described in (2.9) is a column-wise Kronecker product. For more details about the Kronecker product, refer to [32]. As in PMF, PTF has nonnegative constraints of component matrices in the factorization described in (2.9). To find the approximation of (2.9) we need to formulate a Numerical Optimization problem like PMF but using now I-divergence:

$$D[\mathfrak{X}||\hat{\mathfrak{X}}] \approx D\left[\boldsymbol{X}_{(\boldsymbol{n})}||\boldsymbol{W}^{(\boldsymbol{n})}\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})}\right] = \sum_{i_1,i_2,\dots,i_N}\left[\mathfrak{X}_{i_1,i_2,\dots,i_N}\log\frac{\mathfrak{X}_{i_1,i_2,\dots,i_N}}{\hat{\mathfrak{X}}_{i_1,i_2,\dots,i_N}} - \mathfrak{X}_{i_1,i_2,\dots,i_N} + \hat{\mathfrak{X}}_{i_1,i_2,\dots,i_N}\right] \quad (2.10)$$

where $D[\mathfrak{X}||\hat{\mathfrak{X}}]$ is the I-divergence (information for discrimination or the amount of information lost when $\hat{\mathfrak{X}}$ is used to approximate $\mathfrak{X}$) of the tensor $\boldsymbol{\mathfrak{X}}$.

Problem: Minimize $D[\mathfrak{X}||\hat{\mathfrak{X}}]$ with respect to $\hat{\mathfrak{X}}$ subject to the constraint $\hat{\mathfrak{X}} \geq \boldsymbol{0}$

The previous problem is a numerical optimization problem. The I-divergence is a quantity that measures how $\boldsymbol{X}_{(\boldsymbol{n})}$ differs from $\boldsymbol{W}^{(\boldsymbol{n})}\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})}$. In other words, (2.10) represents the amount of information lost when $\boldsymbol{W}^{(\boldsymbol{n})}\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})}$ is used to approximate. It plays the role of Squared Euclidean Distance so it is responsible for the nonnegativity property of the PTF algorithm. There are many numerical optimization algorithms that can be applied to find the solution for this problem. Rules for iteratively determining nonnegative component matrices that minimize the objective function of the previous problem are similar to PMF method. Hence, the Multiplicative Update Algorithm for this optimization problem is as follows:

$$W^{(n)} = W^{(n)} \circledast \frac{\left[\frac{\boldsymbol{X}_{(\boldsymbol{n})}}{\left(\boldsymbol{W}^{(\boldsymbol{n})}\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})}\right)}\right]\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})^T}}{1z^T}, \quad (2.11)$$

where / is the element-size division in the numerator of equation 2.11, $1 \in \mathbb{R}^{I_n \times 1}$ $(a\ column\ vector)$, $z \in \mathbb{R}^{I_n \times 1}$ with $z_i = \sum_j \left[\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})}\right]_{ij}$, and $\circledast$ is a Hadamard product. The updating rule for $\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})}$ is like NMF method and is as follows:

$$\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})} = \boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})} \circledast \frac{\left[\frac{\boldsymbol{X}_{(\boldsymbol{n})}}{\left(\boldsymbol{W}^{(\boldsymbol{n})}\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})}\right)}\right]\boldsymbol{W}^{(\boldsymbol{n})^T}}{1w^T}, \quad (2.12)$$

where / is the element-size division in the numerator of equation 2.12, $1 \in \mathbb{R}^{I_n \times 1}$, $w \in \mathbb{R}^{I_n \times 1}$ with $w_i = \sum_j \left[\boldsymbol{W}^{(\boldsymbol{n})}\right]_{ij}$, and $\circledast$ is a Hadamard product. The updating rule of this algorithm is like (2.7) and (2.8) but using tensor notation. The PTF approach could be used to solve the problem of this thesis but computational complexity would increase.

Multiplicative Update Algorithm For PTF:

$\boldsymbol{W}^{(\boldsymbol{n})} = rand(N, R); \%\ initialize\ \boldsymbol{W}$ as random dense matrix

$\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})} = rand(R, T); \%\ initialize\ \boldsymbol{H}$ as random dense matrix

$for\ i = 1{:}\,maxiter$

$\boldsymbol{X}_{(\boldsymbol{n})} = \boldsymbol{X}_{(\boldsymbol{n})} + \boldsymbol{\alpha\beta}$ **, where β is the gradient of** $\boldsymbol{X}_{(\boldsymbol{n})}$ **and** $\boldsymbol{0 < \alpha < 1}$

$$\boldsymbol{W}^{(\boldsymbol{n})} = \boldsymbol{W}^{(\boldsymbol{n})} \circledast \frac{\left[\boldsymbol{X}_{(\boldsymbol{n})}/\left(\boldsymbol{W}^{(\boldsymbol{n})}\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})}\right)\right]\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})^{\boldsymbol{T}}}}{\boldsymbol{1z^T}}$$

$$\boldsymbol{H}_{\boldsymbol{W}}^{(\boldsymbol{n})} = \boldsymbol{H} \circledast \frac{[\boldsymbol{X}_{(\boldsymbol{n})}/(\boldsymbol{W}^{(\boldsymbol{n})}\boldsymbol{H})]\boldsymbol{W}^{(\boldsymbol{n})^T}}{1w^T}$$

end

2.8 Support-Vector Machine (SVM)

Support Vector Machines (SVM) is a very useful and popular technique for data classification, it is based on supervised learning models with associated learning algorithms that analyze data and recognize patterns. It finds the optimal decision hyperplane that best separates the data into different classes by mapping the input features onto a high-dimensional feature space [8] by solving the following optimization problem:

$$min \frac{1}{2} \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j a_i a_j K(x_i, x_j), \tag{2.13}$$

$$s.t.\ \ \sum_{i=1}^{n} y_i a_i = 0, \quad 0 \le a_i \le C, \tag{2.14}$$

where C is the penalty factor which allow to control the trade-off between the misclassifications and the size of the margin between classes. Because the SVM uses a hyperplane it would only be able to classify classes that can be separated linearly; the problem has to be transformed to a higher dimension for non-linear problems, this is done with a kernel function $K(x_i, x_j)$. The SVM classifier with a radial basis kernel function (RBF) will be used for classification because it can handle non-linear features given by:

$$RBF = K(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2, \gamma > 0), \quad (2.15)$$

where γ is the kernel parameter set by the user [10, 22]. While the kernel function helps with the non-separable classes, SVM is still not a multi-class classifier. In other words, to make the SVM a multi-class classifier several algorithms have been developed: one-versus-one and one-versus-all are the most popular. For this research, the LIBSVM library that is being used uses one-versus-one [10, 22]. Unlike the one-versus-all algorithm that makes n models for n classes, one-versus-one makes model every pair of classes. While one-versus-one has more models, it has been tested to be better with larger problems [23, 24].

2.9 Computational Analysis

The computational analysis that is applied in this work is the time and space complexity. The time complexity is the time taken by an algorithm to run as a function of the length of the input. The time complexity of an algorithm is expressed using Big-O notation. This time is estimated by counting the number of elementary operations performed by the algorithm, where an elementary operation takes a fixed amount of time to perform. Thus, the amount of time taken and the number of operations performed by the algorithm differ by at most a constant factor. For this work, the constant factor is the number of samples.

Performing an accurate calculation of the operation time of a program is very intensive because it depends on the type of computer or processor speed [25]. Complexity, in that sense, is the maximum number of operations that a program may execute. Regular operations are additions, multiplications, assignments, etc. Since, there are a lot of operations, we are only focusing on those operations that are performed the largest number of times. Such operations are called dominant and they depend on the input data. Usually, researchers want to know how the performance time depends on an aspect of the data: the size. Operations includes both scalar and matrix. Some operations are listed below:

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Left Division (\)
- Right Division (/)
- Square Root (√)

The operations mentioned above are called Floating Point Operations (FLOP). FLOP is a method of encoding real numbers within the limits of finite precision available on computers. In other words, FLOP handles Floating Point Numbers (FPN), numbers that includes decimals on it. Counting the number of FLOPS an algorithm requires solving a problem allows us to compare the relative speed of methods. FLOPS is a measurement of the computational efficiency algorithm. It gives the number of basic operations an algorithm performs as a function of its input length. With that said, a function $T \in \mathbb{N}$ can capture the efficiency of an algorithm such that $T(n)$ is equal to the maximum number of basic operations that the algorithm performs on inputs of length n. For instance, an algorithm with $T(n) = O(n)$ is called a linear time algorithm because for large enough input sizes of data, the running time increases linearly with the size of the input. There are different time complexities; some of them are listed in the following table:

Name	**Running Time ($T(n)$)**
Logarithmic Time	$O(\log n)$
Linear Time	$O(n)$
Quadratic Time	$O(n^2)$
Cubic Time	$O(n^3)$
Exponential Time	$2^{O(n)}$
Factorial Time	$O(n!)$

Table 2.1: Table of Common Time Complexities

- **Logarithmic Time:** Algorithms taking logarithmic time are commonly found in operations on binary trees or when using binary search.

- **Linear Time:** Best time complexity in situations where the algorithm must sequentially read its entire input.

- **Quadratic Time:** Algorithms that are comparison-based take this type of complexity. Also, advanced algorithms such as the Shell Sort takes quadratic time.

- **Cubic Time:** Algorithms involving three nested loops take Cubic Time.

- **Exponential Time:** It is used to express the running time of some algorithm that may grow faster than any polynomial.

- **Factorial Time:** Algorithms used to calculate permutations or finding the determinant of a matrix using Laplace expansion has this kind of time complexity.

Another important aspect is the Space Complexity of an algorithm. Space Complexity is the amount of computer memory required by an algorithm to complete its execution. Space Complexity includes auxiliary space and space used by the input. The auxiliary space is the extra space or temporary space used by the algorithm. To measure the space, natural units are used such as bytes, number of integers used, number of fixed-sized structures, etc. There are up to four aspects of memory usage to consider:

- Memory needed to hold the code for the algorithm
- Memory needed for the input data
- Memory needed for the output data
- Memory needed as working space

It is important to know that both Time and Space Complexity depends on the computer that is used to perform this analysis. Current computers have large amounts of memory such as: cache memory, physical memory, and virtual memory. Cache memory (often static RAM) operates at speeds comparable with the CPU. Physical memory (often dynamic RAM) operates slower than the CPU. Virtual memory (often on disk) gives the illusion of lots of memory and operates thousands of times slower than RAM. An algorithm whose memory needs will fit in cache memory will be much faster than an algorithm which fits in main memory, which in turn will be very much faster than an algorithm which must resort to virtual memory.

CHAPTER 3

METHODOLOGY

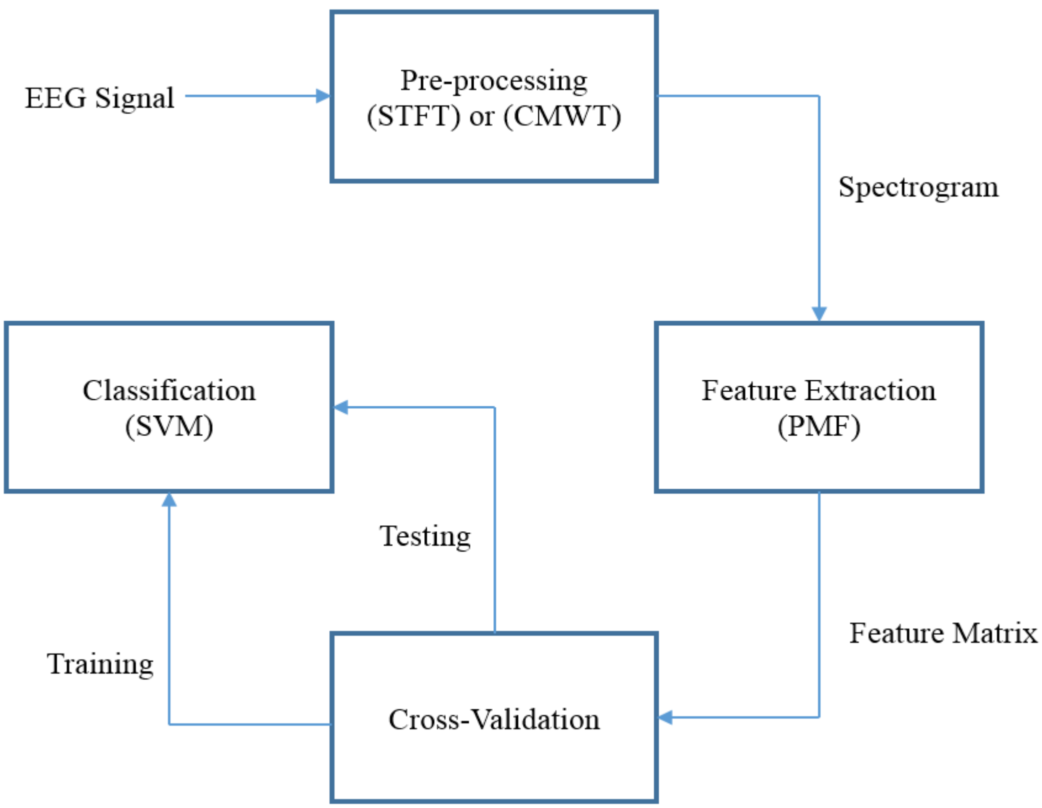

Fig 3.1: Flow Chart of the Methodology

The methodology consists of EEG data acquisition for the motor imagery tasks of thinking movement in two parts of the body such as left hand and right hand. Once sufficient trials of EEG signals for the above tasks are obtained from a subject, the spectrogram is computed on each of these signals by using either STFT or CMWT. The feature extraction stage consists of computing the PMF on the spectrograms. A feature matrix for both testing and training data is constructed by the selection of the most discriminative features. A 10-Fold Cross-Validation of the feature matrices is applied to make the classification process using the SVM classifier more reliable. The work flow for the methodology is given in Figure 3.1.

3.1 EEG Signal Dataset

The EEG signal generated by the cerebral cortex is measured with both frontal and motor imagery electrodes in the scalp surface with AgCl conductive paste applied on the region to provide good conduction. Lower impedance from the region indicates a better conduction. The subject will be asked to imagine left and right hand movement according to an arrow displayed in the BCI application. This process will be repeated until the subject has good performance in the task, with no eye blinking (electrooculography (EOG)), correct imagination of the tasks and no movement of other body parts (electromyography (EMG)).

3.1.1 Binary EEG Motor Imagery Classification Dataset

The dataset was provided by Department of Medical Informatics, Institute for Biomedical Engineering, Graz University of Technology in Austria [27]. This dataset contains EEG data recorded from a normal subject (some 25 years old female). The task was to control a feedback bar by imagining left or right hand movements. The order of left and right cues was random. The experiment consisted of 7 runs of 40 trials each, for a total of 280 trials. The runs of the experiment were conducted on the same day with several minute break in between. Given are 280 trials of 9 second length. One set of 140 trials out of the 280 used was for testing and the other 140 trials for training.

In the first two seconds of the run an acoustic stimulus indicates the beginning of the trial. Then, a cross "+" is displayed for about one second. At $t = 3s$ an arrow (with either left or right direction) was displayed as cue. At the same time, the subject was asked to move a bar into the direction of the cue. The recording was made using a *G.Tec* amplifier and Ag/AgCl electrodes, as previously mentioned. Three EEG channels were measured over C_3, C_z and C_4. The EEG signals are sampled with 128Hz and filtered between 0.5Hz - 30Hz. Figure 3.5 and Figure 3.6 shows the original EEG Signal from channels C_3 and C_4 using only one trial. The trials for training and testing were randomly selected to prevent any systematic effect due to the feedback session of the experiment. A visual representation of the experiment is shown in Figure 3.3. Channels C_3, C_z and C_4 are shown in Figure 3.2:

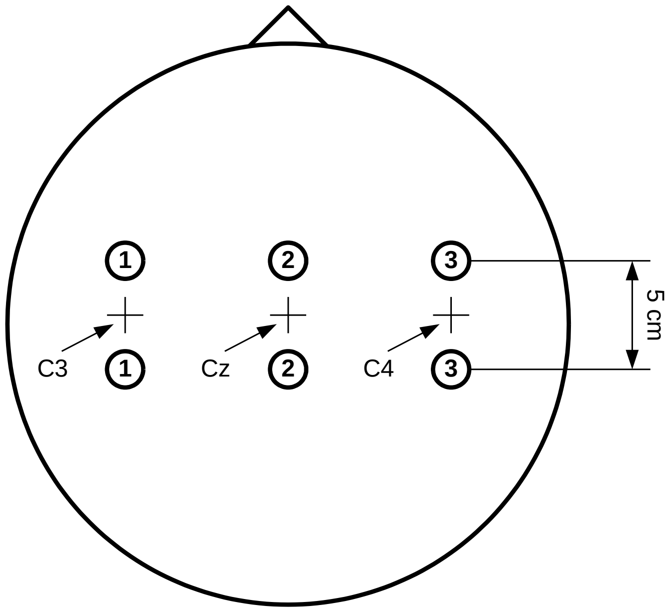

Figure 3.2: Map of Electrode Positions

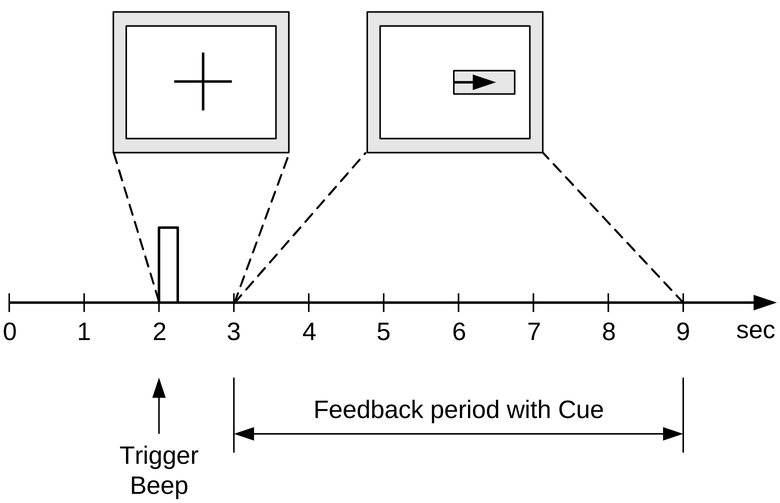

Figure 3.3: Experiment Description: Time Scheme

Electrodes 1, 2 and 3 are bipolar EEG channels. Many BCI systems uses monopolar recordings. In monopolar recordings, one electrode is placed on the scalp and the other one is located away from the interest area [9]. For this work, bipolar recordings are used. In bipolar recordings, the electrodes are placed both on the scalp, as it can be seen in Figure 3.4 (b). According to [35] bipolar channels are more robust to noise. The monopolar channels have the disadvantage of the location of reference electrode which affects greatly the EEG recording. In general, the reference electrode is placed ideally on a point reflecting no brain wave activity.

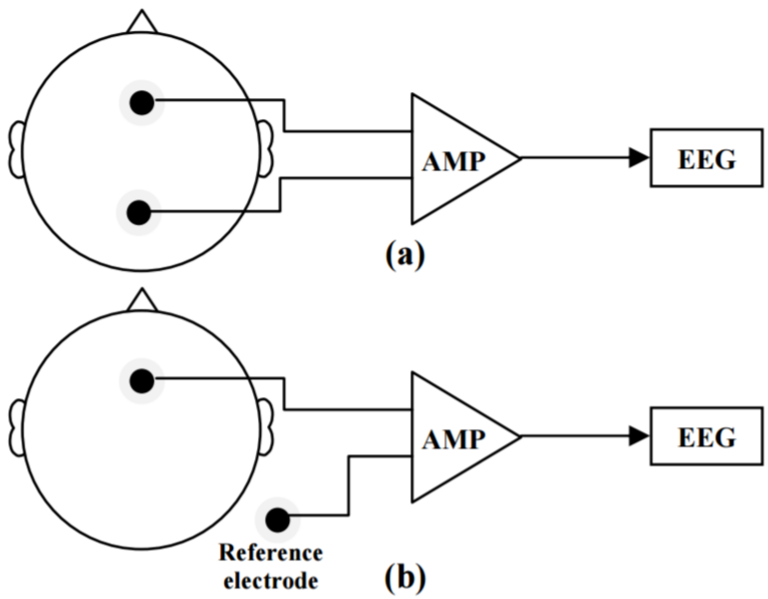

Figure 3.4: EEG placement electrodes method, Bipolar (a) and Monopolar (b)

The format of the data is important. Data is saved in a Matlab file format. The file contains three variables. The first variable (training) contains 3 EEG channels and 140 trials with 9 seconds each. The second variable (testing) contains also the same 3 EEG channels and another 140 trials with 9 seconds each. The last variable contains the class labels “1” and “2”. Label “1” corresponds to left and label “2” corresponds to right. On the other hand, the cue was presented from $t = 3s$ to $t = 9s$, as in Figure 3.3.

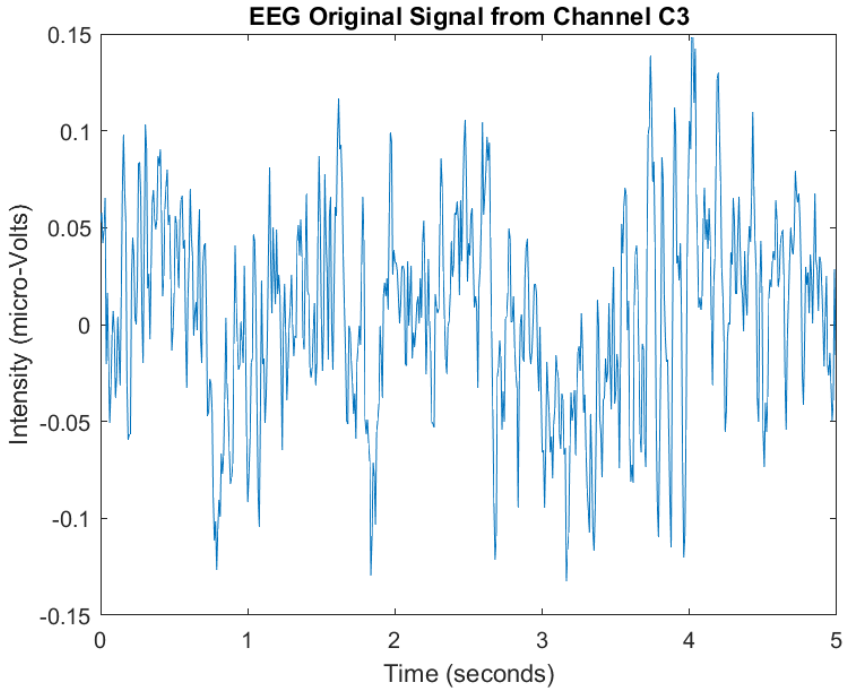

Figure 3.5: Original EEG Signal from Channel C_3

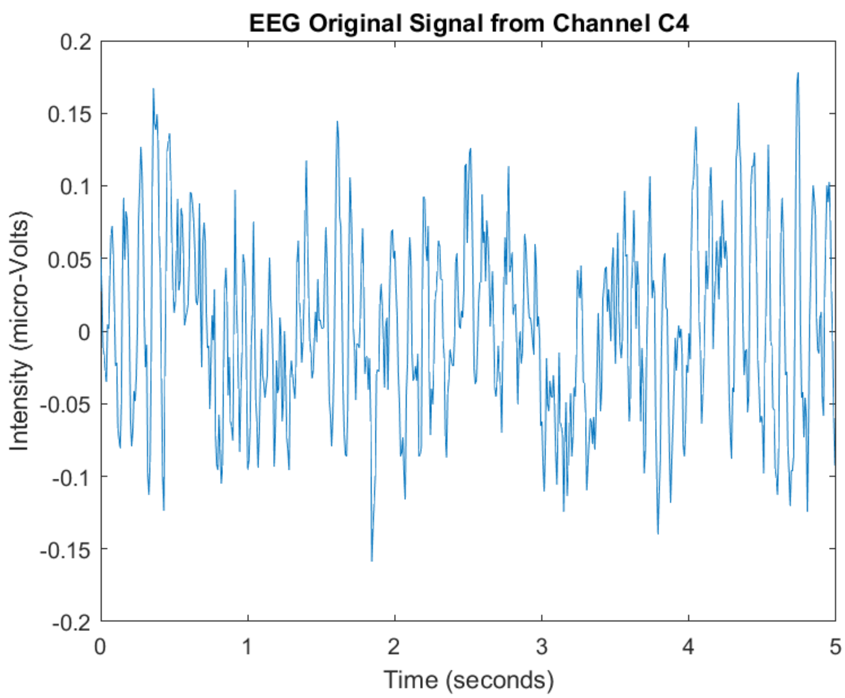

Figure 3.6: Original EEG Signal from Channel C_4

3.2 Time-Frequency Analysis

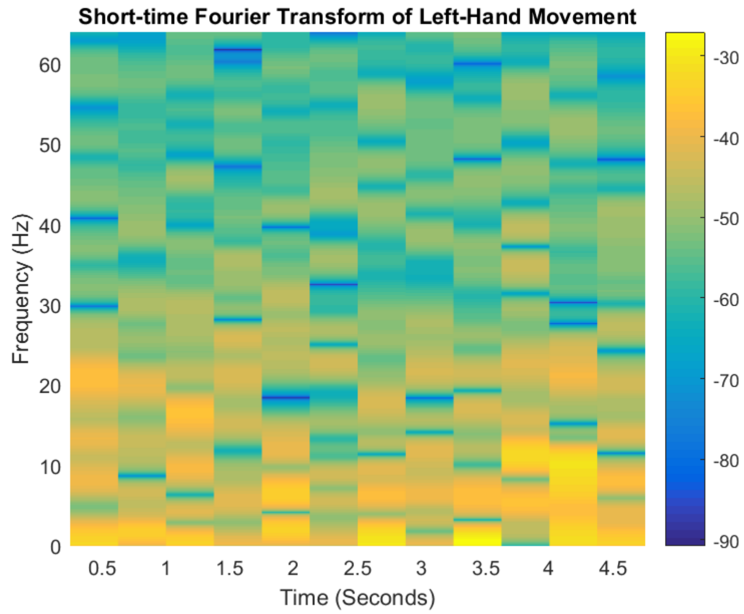

Figure 3.7: Spectrogram of Left-Hand Trial using STFT

Figure 3.8: Spectrogram of Right-Hand Trial using STFT

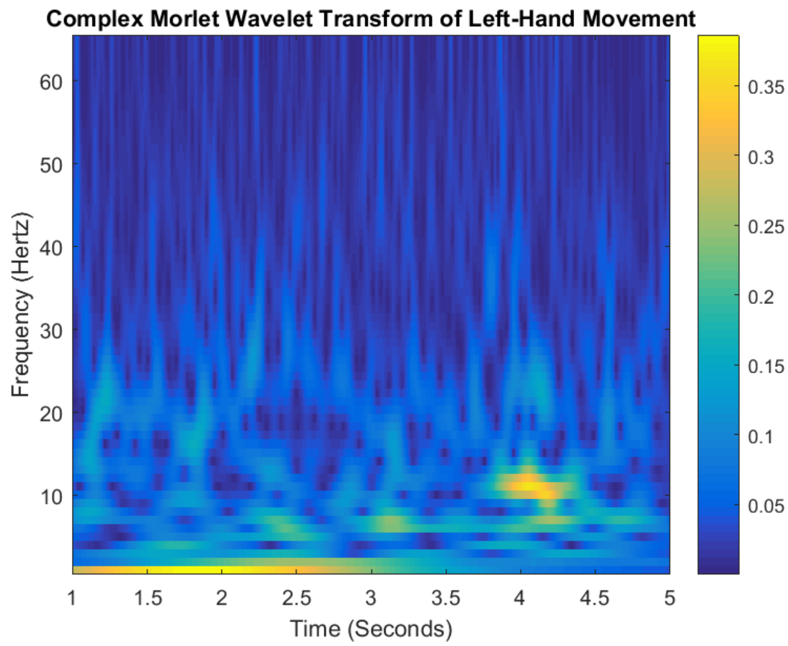

Figure 3.9: Spectrogram of Left-Hand Trial using CMWT

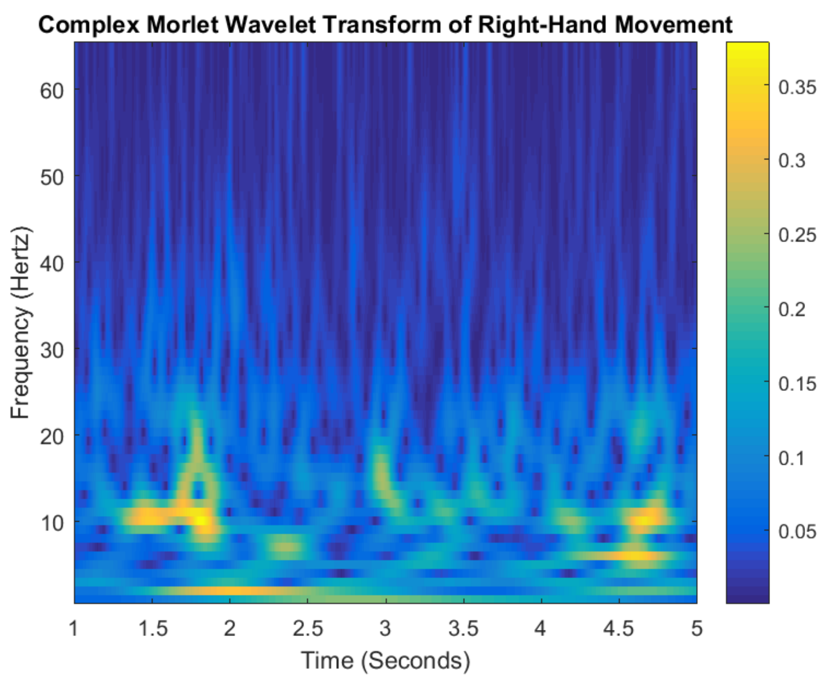

Figure 3.10: Spectrogram of Right-Hand Trial using CMWT

After data is collected and the generic data has been acquired, a time-frequency representation of the data is used to get more information of the data. The time-frequency methods used in this work are the STFT and the CMWT. This work was done using STFT first and then the CMWT to compare results. The STFT of each EEG sample was obtained using the Matlab spectrogram function. STFT is done like [24] where the window function is being multiplied by the Fourier Transform of the EEG data. Figure 3.7 shows the spectrogram (computed with STFT) for imaginary left-hand movement and Figure 3.8 shows the spectrogram (computed with STFT) for imaginary right-hand movement. On the other hand, the CMWT of each EEG signal was obtained using the CMWT function created in the Brain Computer Interface Lab. of the University of Puerto Rico at Mayaguez using Matlab, as well. Figure 3.9 and Figure 3.10 shows the time-frequency representation of an imaginary left and right hand movements using CMWT. The time-frequency representation of EEG data computed by the methods mentioned before are a positive data matrix that is used as the input to the PMF method.

3.4 Feature Extraction

Positive matrices are obtained with either Short Time Fourier Transform or Complex Morlet Wavelet Transform. For each trial, a positive matrix is calculated for a total of 140 positive matrices for testing and another 140 positive matrices for training. Taking STFT matrix or CMWT as input for the PMF method, the output is the product of two matrices W and H in the form of a multiplication. In other words, the multiplication WH is the PMF of either STFT or CMWT matrix. According to [22], W contains basis vectors in its columns and H is the encoding matrix where each row represents a feature vector. Since both STFT and CMWT are different methods that are used to compute the positive matrices, different sizes of W and H are obtained. Figure 3.11 shows the PMF applied to an input matrix obtained with the STFT approach. As mentioned above, feature vectors in Figure 3.11 are the rows of matrix H. For this case, the input matrix is of size $513{\times}13$, matrix W is of size $513{\times}10$ and matrix H is of size $10{\times}13$. Figure 3.12 shows the PMF applied to an input matrix obtained with the CMWT approach. Feature vectors in Figure 3.12 are the rows of matrix H. For this case, the input matrix is of size $64{\times}640$, matrix W is of size $64{\times}10$ and matrix H is of size $10{\times}640$. The number of feature vectors depends on the number of the basis selected for the PMF method. This basis must be a number less than the size of the dimension of the positive matrix used as input for the PMF. In other words, the basis factor gives the number of feature vectors. This basis also guarantees that analysis of less amount of data still gives good results. But, good results also depend on the features. For that reason, a feature selection method is done.

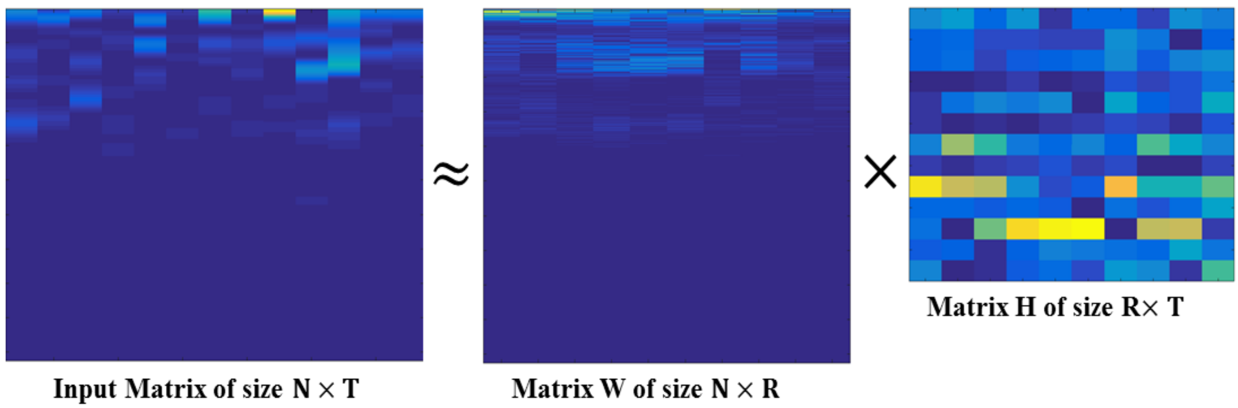

Figure 3.11: PMF method applied to STFT matrix of trial one

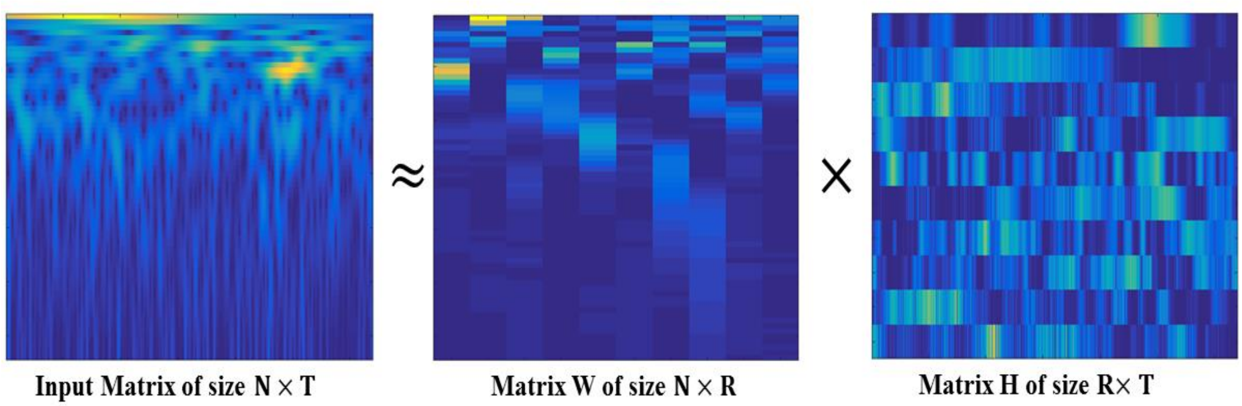

Figure 3.12: PMF method applied to CMWT matrix of trial one

3.5 Feature Selection

To quantify the separation between classes, left and right hand-movement, this research used the same procedure in [30]. Let M be the normalized matrix of the features and $\hat{Y}$ is the feature matrix for each subject. We obtained the sum of the distances:

$$D_L = \sum_{J} \left| \hat{Y}_{i,j} - M_i \right| \tag{3.1}$$

where i is the feature index, L is the number of the features, j is the class index, and J is the total number of matrix imagery class. The standard deviation for each feature was also computed:

$$\sigma_L = \sqrt{\frac{1}{J} \sum_{j=1}^{J} \left(Y_{i,j} - M_i \right)^2} \tag{3.2}$$

Sorting the results of equations (3.1) and (3.2), the first half of features were selected. Features with less distance means that are near the class it corresponds than those features with high distance. The same occurs with the standard deviation. Selected features were used for classification.

3.6 Cross-Validation

After selecting the best features from the selected channels, the data was split using 10-fold cross-validation. Cross-validation is used to get the accuracy for the classifier C. Accuracy is defined as:

$$A_{cc} = P_r(C(v) = y) \tag{3.3}$$

for a randomly selected instance $\langle v, y \rangle \in X$, where the probability distribution over the instance space is the same as the distribution that was used to select the instance for the inducer's training set. This is where the inducer builds a classifier from a given dataset. An inducer is an algorithm that is used to build a classifier using a given dataset [19]. One accuracy does not give many details about the classified data. For that reason, cross-validation is used.

The function used in this experiment is the *k*-fold cross-validation. In *k*-fold cross-validation, the folds $D_1, D_2, \dots, D_k$ are randomly split k subset of approximately equal size of dataset *D*. For each of the k subset, the inducer is trained and tested k times, where each $t \in \{1,2,\dots,k\}$, it is trained on $D \backslash D_t$. Let the instance $x_i = \langle v, y \rangle$ be a test set in $D_{(i)}$ then the cross-validation estimate of accuracy:

$$Acc_{cv} = \frac{1}{n} \sum_{\langle v_i, y_i \rangle \in D} \delta\left(I\left(D \backslash D_{(i)}, v_i\right), y_i\right) \tag{3.4}$$

where $\delta(i,j) = 1$ if $i = j$ and 0 otherwise and *I* is the inducer. Then a complete cross-validation is used where the average of all $\binom{m}{m \backslash k}$ possibilities for choosing $m \backslash k$ instance out of m [18].

3.7 Classification

After using 10-fold cross-validation to make the partition to divide the data into training and testing sets, the data was run through the Support Vector Machine (SVM) classifier. As discussed in section 2.8, this experiment used LIBSVM Matlab code for the classification process using the SVM approach. The kernel that was used is the Radial Basis Function (RBF) because it generates a non-linear mapping by transforming the features into a different space, where a linear discrimination between classes is done. The RBF was determined using the highest accuracy achieved in cross-validation. The SVM uses a one-versus-one system which makes a model for each class and compares them with each one, giving a point to the class that won. In other words, the class with the most data points wins and that class is assigned a label.

3.8 Computational Complexity

As mentioned before, BCI systems must be accurate and efficient, therefore efficiency should be considered as much as accuracy and the computational time, as well. Most BCI systems must be as reliable as possible because of the real-world applications. Hence, this thesis considered that by obtaining the computational time required to obtain the Positive Matrix Factorization giving a specific number of samples as input. An important fact about the PMF method and time complexity is that with higher number of basis in the PMF, the computational time also increases. This is because with bigger set of data, the number of FLOPS or operations during the PMF method increases. Since, we focused only in the PMF method for this time complexity analysis, the number of FLOPS was obtained using only the PMF function. In other words, the number of Floating Point Operations per second and the time of execution was calculated using only the PMF function. The computer that was used for this work has an Intel Xeon 2.6 Ghz Dual Core and a 32 GB RAM, which was a good computer for handling the data processing and all the computations of this work.

CHAPTER 4

RESULTS

In this chapter, classification accuracy for motor imagery data from [1] is presented. The accuracy includes the use of two channels and Support Vector Machine (SVM) classifier. The data was divided, and $1/10$ of the trials were selected for testing. In other words, 14 trials were randomly selected out of the 140 trials, ten times. After the feature extraction and the feature selection, the data was left with a feature matrix. The size of this matrix depends on PMF and the number of basis utilized. In the following tables, Table 4.1 and Table 4.2, different sizes of feature matrices are shown. The results of these instances are recorded to compute the accuracy for the prediction stage. These results are from pre-recorded data and not real-time data.

4.1 Feature Matrices

	Basis 6	**Basis 8**	**Basis 10**
Feature Matrix	140×78	140×104	140×130

Table 4.1: Feature Matrix size using Short Time Fourier Transform

	Basis 6	**Basis 8**	**Basis 10**
Feature Matrix	140×3840	140×5120	140×6400

Table 4.2: Feature Matrix size using Complex Morlet Wavelet Transform

Table 4.1 and Table 4.2 shows the sizes of features matrices obtained using both STFT and CMWT. The number of basis is given by the user and it represents two things: the number of basis vectors of the PMF method and the number of feature vectors. It is very important to know the fact that the higher the number of observations in the feature matrix, the accuracy for the prediction could increase or decrease. The reason is that noise is present in these observations and it could be difficult to extract all of them without leaving all the important data in the dataset. According to [27] the data set was filtered from $[0.5Hz - 30Hz]$. Of course, there are method to filter unnecessary noise such as Adaptive Filtering, Linear Regression, or Data Decomposition [31] but it is beyond this work.

4.2 Cross-Validation Accuracy

A training model was created using the LIBSVM to study good possibilities in terms of number of samples and basis. Table 4.3 shows different percentages of accuracies according to number of samples and number of basis. As mentioned above, the number of basis is a factor by which the PMF will compute the factorization. This basis is also the number of feature vectors because the rows of matrix H were taken as feature vectors. The normal EEG dataset contains 1152 samples for each trial, for a total of 140 trials. Depending on the subject, this dataset could have more or less noise but all dataset has noise. Therefore, it can be seen that accuracy percentage is higher with less number of samples as in Table 4.3. Table 4.4 also shows cross-validation accuracies using Complex Morlet Wavelet Transform. CMWT gives matrices with greater dimensions than STFT so the number of features and the amount of noise also increased. The same effect as Table 4.3 can be seen; accuracy percentages are in some cases higher with less samples than with a big number of samples. To facilitate lector's interpretation, accuracies are in percentage.

Number of Samples	Accuracy with $Basis$ 2	Accuracy with $Basis$ 4	Accuracy with $Basis$ 6	Accuracy with $Basis$ 8	Accuracy with $Basis$ 10
384	62.14	69.29	65.00	64.29	66.43
512	64.29	70.00	72.14	70.00	70.71
640	68.57	70.71	72.14	72.14	76.00

Table 4.3: Cross-Validation Accuracies using Short Time Fourier Transform

Number of Samples	Accuracy with $Basis$ 2	Accuracy with $Basis$ 4	Accuracy with $Basis$ 6	Accuracy with $Basis$ 8	Accuracy with $Basis$ 10
384	62.86	52.86	54.29	57.86	60.00
512	57.85	58.57	62.86	62.14	63.57
640	53.57	60.71	67.14	70.00	72.14

Table 4.4: Cross-Validation Accuracies using Complex Morlet Wavelet Transform

4.3 Prediction Accuracies

Table 4.5 and Table 4.6 shows Classification Accuracies for different number of samples, different basis and different methods: STFT and CMWT. It can be seen in Table 4.5 that the higher accuracy is 75%, using a basis of 10 and 640 samples. For a basis 10 and 512 samples, a 71.43% was obtained and a 72.86% was obtained using a basis 8 and 640 samples. Even though we had the highest result using the highest amount of both basis and samples, a good accuracy can also be obtained using less basis (72.86%) and less number of samples (71.43%). This means that the PMF method is good for data reduction, which is a big challenge in the data processing perspective. Obtaining good results with less data than the original dataset also reduces computational time and it is cost effective. Short Time Fourier Transform was used to get the results of Table 4.5, which gives also positives matrices smaller than the positive matrices computed using the Complex Morlet Wavelet Transform.

CMWT was used to compute positive matrices for the PMF method. Table 4.6 shows that the highest accuracy was 70% using number of basis of 10 and 640 samples. A 62.14% was obtained with a basis of ten and a number of samples of 512. Also, a 60.00% was obtained using a basis 8 and 640 samples. If we compare the numbers computed using STFT and the numbers computed using CMWT, it can be seen that STFT is a better method for features representation than CMWT. CMWT increases resolution of spectrogram but also increase the computational time. Because it increases the resolution, the noise also increases, making the prediction process a challenge. That is why STFT has better results than CMWT. To facilitate lector's interpretation, accuracies of the tables are in percentage.

Number of Samples	**Accuracy with $\boldsymbol{Base = 6}$**	**Accuracy with $\boldsymbol{Base = 8}$**	**Accuracy with $\boldsymbol{Base = 10}$**
384	61.43	55.71	60.00
512	65.00	63.57	71.43
640	66.43	72.86	75.00

Table 4.5: Classification Accuracies using Short Time Fourier Transform for Testing

Number of Samples	**Accuracy with $\boldsymbol{Base = 6}$**	**Accuracy with $\boldsymbol{Base = 8}$**	**Accuracy with $\boldsymbol{Base = 10}$**
384	57.14	57.86	62.14
512	61.43	55.71	62.14
640	62.86	60.00	70.00

Table 4.6: Classification Accuracies using Complex Morlet Wavelet Transform for Testing

Alternating Least Squares (ALS) and Multiplicative Update (MU) are the two algorithms to be analyzed fort this work. ALS is a good approach when dealing with big amount of data and its convergence is relatively fast but with the PMF method it is not as good as the MU algorithm. The reason is that ALS initialize only one of the two factorization matrices and since PMF does not have a unique local solution, the convergence will not get good results in terms of accuracy.

MU on the other hand initializes both factorization matrices W and H, making faster the convergence in the PMF method. Something important to mention is the fact that the PMF method can get local minima for W or H but not with both at the same time. In other words, the initial conditions in this algorithm plays an important role. To obtain results of Tables 4.5 and 4.6, the PMF was run few times. Researchers suggest running the PMF many times and the one that gives the highest results will be used the Matrix Factorization of the input. This is because of the nature of the initial conditions of W and H. In other words, the algorithm does not have a unique solution due to its initial conditions.

4.4 Confusion Matrices

Confusions matrices that are presented below are those whose accuracies are above $70\ \%$. They show the results for the subject studied by demonstrating how many of the observations were correctly classified. The accuracy percentage is obtained by taking the average of the diagonal of the confusion matrix.

Figure 4.1 shows the confusion matrix obtained using CMWT, 10 basis vectors and 640 samples. As we can see, the SVM classifier predicted 71.43% as left-hand movement and 28.57% as right hand movement while the actual class to be predicted was a left-hand movement. 31.43% was predicted as right-hand movement and 68.57% was predicted as right-hand movement while the actual class to be predicted was a right-hand movement. The diagonal of the confusion matrix of Figure 4.1 is used to calculate the final accuracy. This accuracy is obtained as the average of the diagonal of the confusion matrix in the following manner:

$$Accuracy_1 = \frac{71.43 + 68.57}{2} = 70\%$$

where $Accuracy_1$ is the highest accuracy shown in Table 4.6.

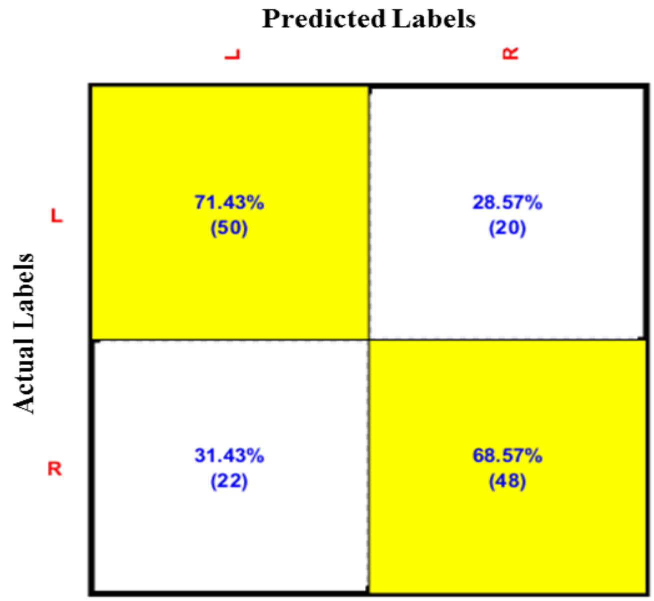

Figure 4.1: Confusion Matrix using CMWT (70%)

Figure 4.2 shows the confusion matrix obtained using STFT, 10 basis vectors and 512 samples. As we can see, the SVM classifier predicted 77.14% as left-hand movement and 22.86% as right hand movement while the actual class to be predicted was a left-hand movement. 34.29% was predicted as right-hand movement and 65.71% was predicted as right-hand movement while the actual class to be predicted was a right-hand movement. The diagonal of the confusion matrix of Figure 4.2 is used to calculate the final accuracy. This accuracy is obtained as the average of the diagonal of the confusion matrix in the following manner:

$$Accuracy_2 = \frac{77.14 + 65.71}{2} = 71.43\%$$

where $Accuracy_2$ is the highest accuracy shown in Table 4.5.

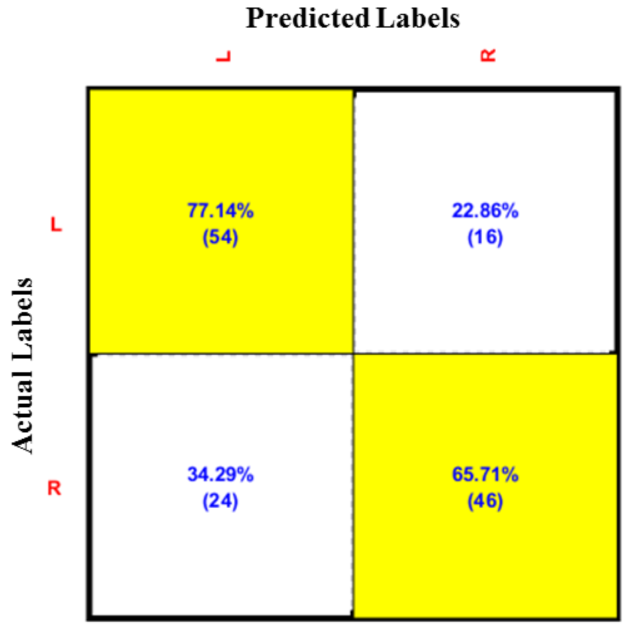

Figure 4.2: Confusion Matrix using STFT (71.43%)

Figure 4.3 shows the confusion matrix obtained using STFT, 8 basis vectors and 640 samples. As we can see, the SVM classifier predicted 74.29% as left-hand movement and 25.71% as right hand movement while the actual class to be predicted was a left-hand movement. 28.57% was predicted as right-hand movement and 71.43% was predicted as right-hand movement while the actual class to be predicted was a right-hand movement. The diagonal of the confusion matrix of Figure 4.3 is used to calculate the final accuracy. This accuracy is obtained as the average of the diagonal of the confusion matrix in the following manner:

$$Accuracy_3 = \frac{74.29 + 71.43}{2} = 72.86\%$$

where $Accuracy_3$ is the highest accuracy shown in Table 4.5.

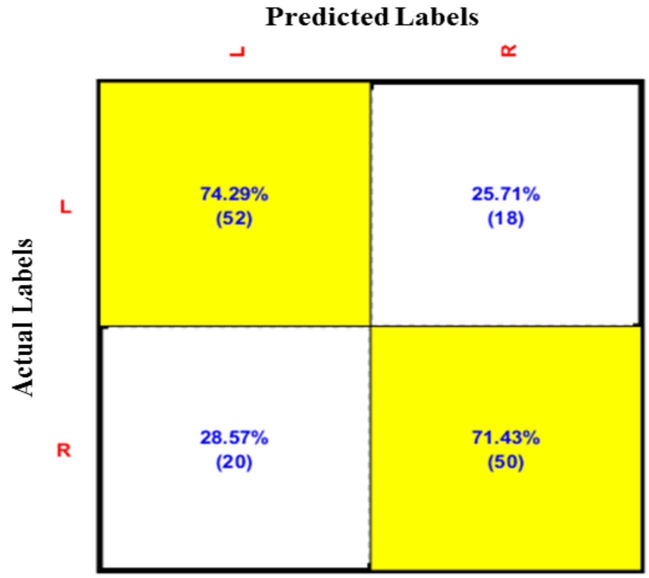

Figure 4.3: Confusion Matrix using STFT (72.86%)

Figure 4.4 shows the confusion matrix obtained using STFT, 10 basis vectors and 640 samples. As we can see, the SVM classifier predicted 57.14% as left-hand movement and 42.86% as right hand movement while the actual class to be predicted was a left-hand movement. 7.14% was predicted as right-hand movement and 92.86% was predicted as right-hand movement while the actual class to be predicted was a right-hand movement. The diagonal of the confusion matrix of Figure 4.1 is used to calculate the final accuracy. This accuracy is obtained as the average of the diagonal of the confusion matrix in the following manner:

$$Accuracy_4 = \frac{57.14 + 92.86}{2} = 75\%$$

where $Accuracy_4$ is the highest accuracy shown in Table 4.5.

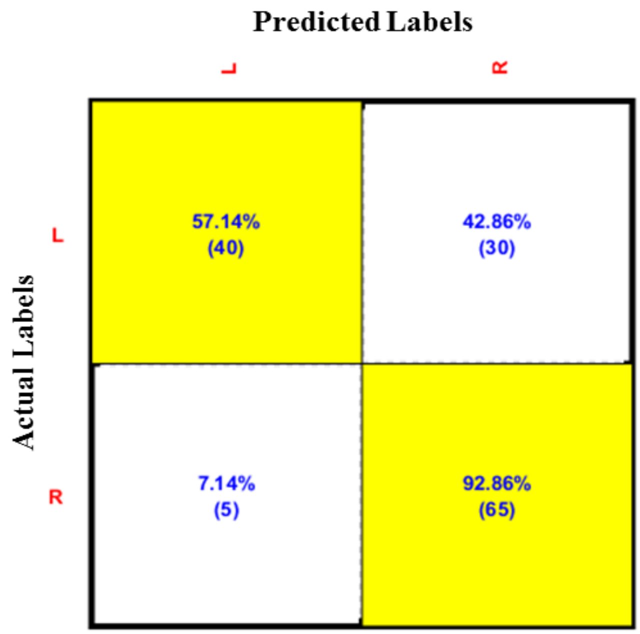

Figure 4.4: Confusion Matrix using STFT (75%)

4.5 Computational Complexity

A time complexity study was performed for this work. Table 4.7 shows the time complexity, in FLOPS, for the Positive Matrix Factorization function using different sets of samples as input. The biggest set of samples is 640 and the smaller is 128 samples. Analyzing the table mentioned above, the PMF method using the STFT to calculate the positive matrices is quite fast. This time depends on many factors such as the computer capacity and the memory compromised during the calculation of this time. As mentioned in Section 3.8, has the computer has an Intel Xeon 2.6 Ghz Dual Core and a 32 GB RAM, which is a good computer for handling the data processing and all the computations of this work. The computational time also depends on how the RAM and the processor of the computer are compromised during the time computation. On the other hand, in Table 4.8 the same time complexity analysis was performed using CMWT method for the calculation of the positive matrices. Comparing both tables, Table 4.7 and Table 4.8, we can see that the number of floating point operations using STFT are less than the number of floating point operations using CMWT. This means that STFT is faster than CMWT. It is true that CMWT improves the resolution time-frequency representation of the dataset, but it also increases the resolution of noise.

Number of Samples	FLOPS
128	8344
256	16664
384	24984
512	33304
640	41624

Table 4.7: Time Complexity of PMF function using base 10 and STFT

Number of Samples	FLOPS
128	36065
256	71905
384	107745
512	143585
640	179425

Table 4.8: Time Complexity of PMF function using base 10 and CMWT

A better visualization of Table 4.7 and Table 4.8 is shown in the following graphs. Figure 4.5 is a graph of Time Complexity using STFT. According to [38], the number of loops in an algorithm along with the operation determines the time complexity. Because of the representation of this plot, we can say that the time complexity of this graph is a linear time complexity. Figure 4.6 is a graph of Time Complexity using CMWT. Because of the representation of this plot and the Big-O notation discussed in 2.9, we can say that the time complexity representation of this graph is Linear Time or in Big-o notation: $T(n) = O(n)$.

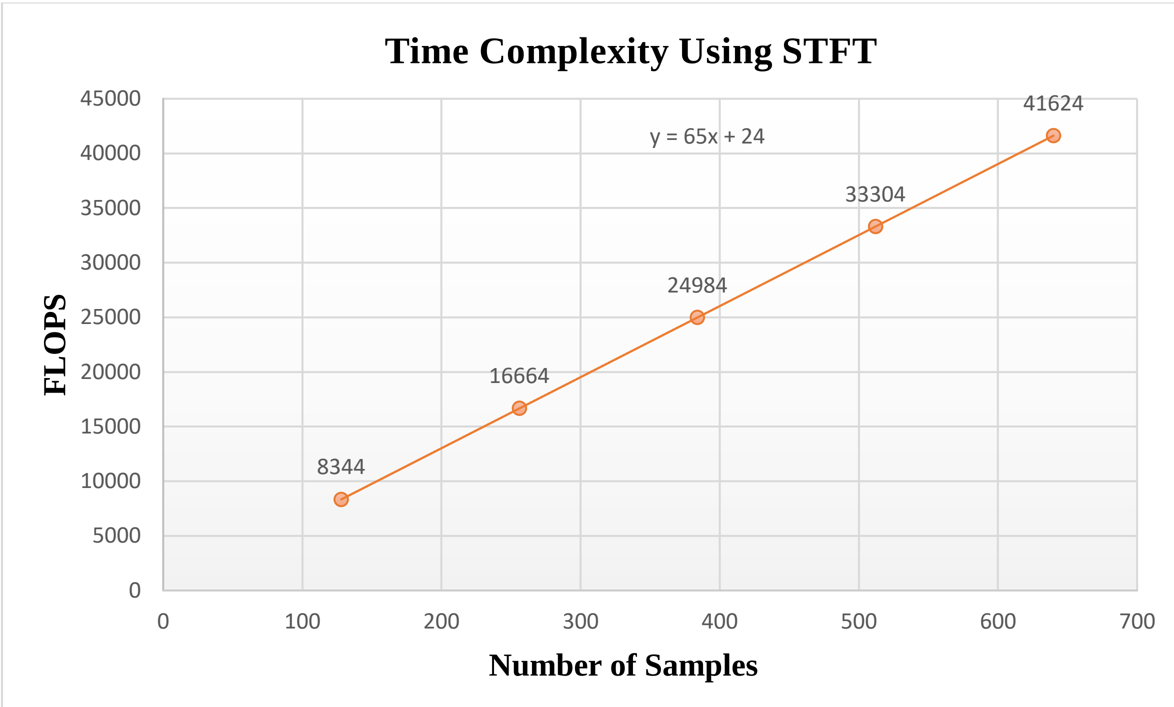

Figure 4.5: Time Complexity using STFT

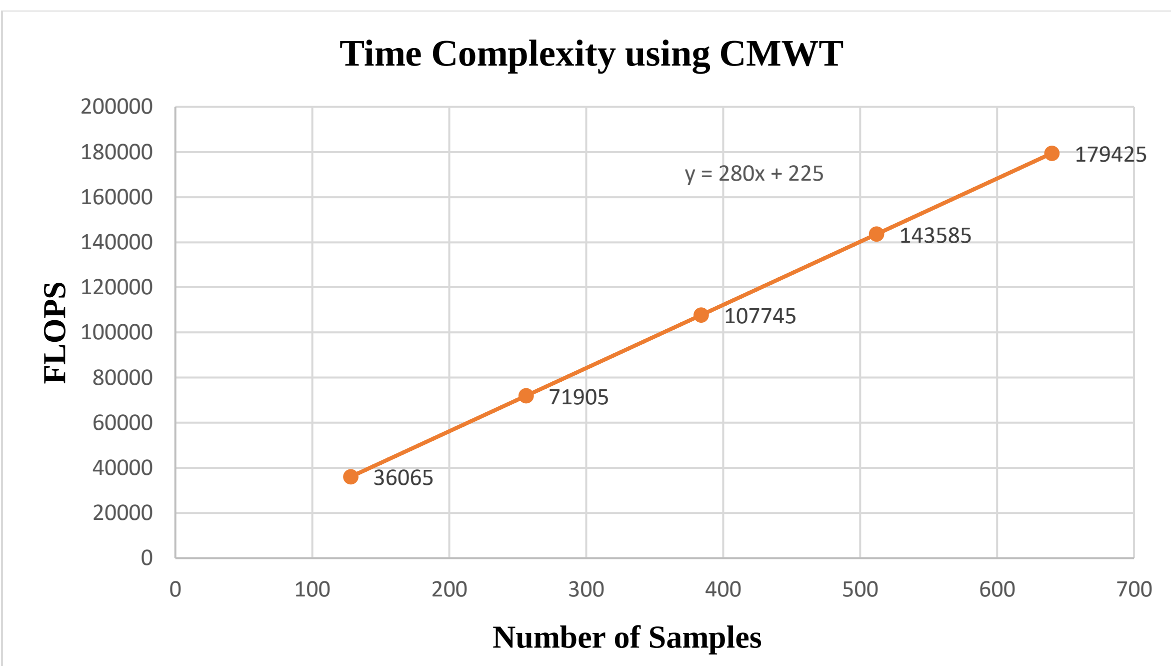

Figure 4.6: Time Complexity using CMWT

FLOPS were obtained using both STFT and CMWT to study how many operations the PMF performs given a number of samples as inputs so the time complexity could be obtained. CMWT generates a bigger matrix than STFT because it has better resolution. In other words, data partition of the CMWT is bigger, so the matrices will be in fact, bigger than the matrix generated using STFT.

On the other hand, Table 4.9 shows the Space Complexity of the PMF method given different sizes of samples and STFT. From Table 4.9 we can see that the number of megabytes is proportional to number of samples. In other words, if the number of samples increases, the number of bytes or space in memory will also increase. FLOPS increase because the PMF method must perform more computations if the number of samples increases. That is, given a big input for the PMF, it means the input matrix will increase in rows or columns, so the computations have more rows or columns to process. Table 4.10, shows the Space Complexity of the PMF method given different sizes of samples and CMWT. As in Table 4.9, the bytes are proportional to number of samples. In this case, the number of operations are more than the number of operations obtained using the STFT. This is because the CMWT gives as output a matrix with a bigger size than the matrix generated with the STFT approach.

Number of Samples	**Megabytes**
128	17.3
256	48.5
384	77.4
512	110.3
640	142.6

Table 4.9: Space Complexity of PMF function using base 10 and STFT

Number of Samples	**Megabytes**
128	360
256	674.5
384	983.3
512	1294.6
640	1621.3

Table 4.10: Space Complexity of PMF function using base 10 and CMWT

A better visualization of Table 4.9 and Table 4.10 is shown in the following graphs. Figure 4.7 is a graph of Space Complexity using STFT. Because of obtained results, we can say that the computational complexity of this graph is linear, so $T(n) = O(n)$. Figure 4.8 is a graph of Space Complexity using CMWT. Because of the results obtained, we can say that the computational complexity representation of this graph is also linear, so $T(n) = O(n)$. Even though the difference in space complexity of STFT and CMWT is not much, STFT still gives better results in the computation of operations. In other words, STFT method is a good choice to calculate the positive matrices of each trial because it requires less memory space to compute the operations.

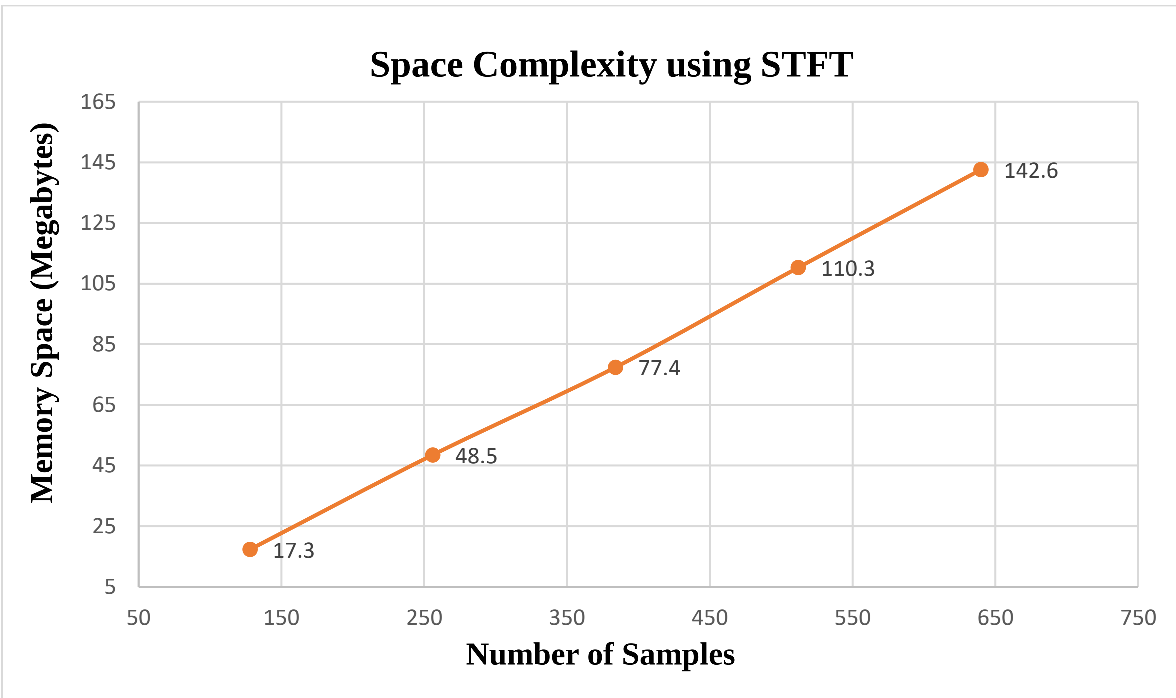

Figure 4.7: Space Complexity using STFT

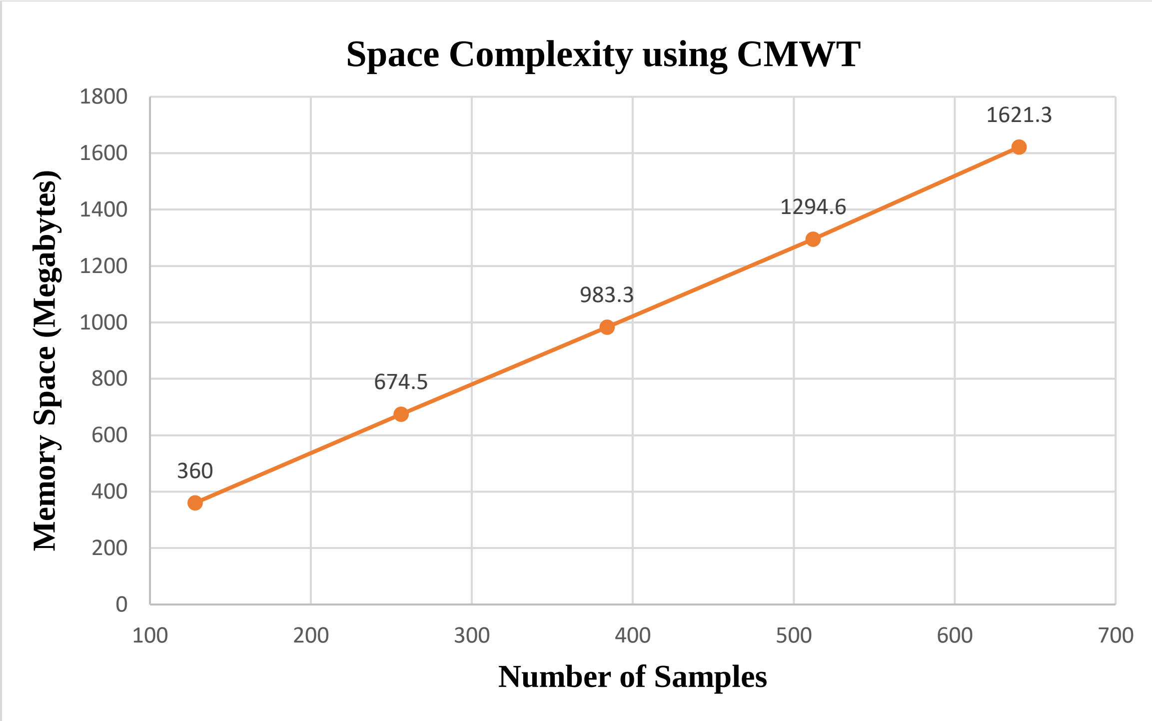

Figure 4.8: Space Complexity using CMWT

4.6 Re-testing the PMF Algorithm

Table 4.11 shows the accuracy (in percentage) using a new dataset [2]. This dataset was collected using channels C_3 and C_4. It also contains 280 trials; 224 trials used for training and 56 trials used for testing. Results from Table 4.11 demonstrate that the PMF algorithm combined with STFT does not change over time. The highest accuracy in Table 4.11 is similar to the highest accuracy of Table 4.5, considering that a basis of 5 was used for the PMF algorithm, in this case.

Number of Samples	Accuracy with $\boldsymbol{Base = 4}$	Accuracy with $\boldsymbol{Base = 5}$
350	69.64	71.42

Table 4.11: Classification Accuracies using a different dataset and STFT

4.7 Discussion

Comparing all accuracies, it is seen that the best result overall using channels C_3 and C_4 is 75%. This was using Short Time Fourier Transform to traduce the EEG data into a time-frequency representation that was used as the input positive matrix for the PMF method. The SVM classifier classified the features relatively good because it maps the features to a higher dimensional space and fits linear decision boundaries in the high dimensional space. Also, SVM is good in performance when the dataset is for binary motor imagery classification. In other words, SVM works better for dataset that has two classes. An aspect to consider is that if a subject has lower accuracy for authentication with the SVM classifier or any other classifier, this implies that the subject needs more training in performing the motor imagery task. This can be seen in the prediction percentages in the Confusion Matrices presented above. A better dataset can be obtained if a subject is well trained.

Also, the overall performance in terms of time complexity using STFT or CMWT is about 0.1 seconds for 640 samples taken from the EEG data as input. This means that PMF works fast in terms of computational time. Of course, computational time depends on the computer performance and capabilities but PMF is a good method for dimensionality data reduction. In terms of space complexity, STFT combined with PMF gave better results than CMWT combined with PMF. This is because the CMWT increases resolution and this increases matrix dimension. This is important for applications where storage of data is a big limitation. A suggestion for people developing applications is to use STFT instead of CMWT.

Storage of data and computational time are big limitations when handling big datasets. For this work, a dataset with size of $7{,}561\ KB$ was used. Only data from 3 EEG channels are available in the dataset. Since the analysis of this works takes several computations, a big number of size is obtained in the results. For the results, $19.1\ MB$ of space was occupied using STFT, a basis of 8 and 640 samples. $16.9\ MB$ of space was occupied using STFT, a basis of 10 and 512 samples. $21.1\ MB$ of space was occupied using STFT, a basis of 10 and 640 samples. On the other hand, an amount of $1.34\ GB$ of space was occupied using CMWT with a basis of 10 and 640 samples. If we sum each space occupied by the results, we get a total of $1.3971\ GB$ of space occupied. In terms of CPU time, the calculated time of the whole algorithm using the STFT method is 6.7188 seconds. A total of 162.09 seconds of computational time was obtained using the CMWT method.

CHAPTER 5

CONCLUSIONS

The objectives of this work were achieved. The implementation of the Positive Matrix Factorization method was done by studying two algorithms, Multiplicative Update and Alternating Least Squares and implementing the most promising of these two. The Multiplicative Update Algorithm worked better than Alternating Least Squares because it initializes both matrices W and H, finding a local solution faster. The ALS algorithm, on the other hand, only initialize one of the two matrices, making the task of finding a local solution a challenge. Also, a computational analysis was done for the PMF method. The space complexity as well as the computational time was obtained while increasing the number of samples. Space Complexity measures were obtained using computer tool named task manager, which it calculates the memory space used by an application. This analysis is important in a computer science perspective because in these days, huge amount of data must be managed in a short period of time to make a robust and reliable BCI system. The use of the PMF method for binary motor imagery classification was done, giving an accuracy of 75%.

5.1 Contributions

The STFT or CMWT (spectrograms), was used to translate EEG data into a time-frequency array that was the input of the PMF method discussed in this work. The PMF was used to extract features from EEG signals for experiments based on motor imagery. The STFT and PMF method gave better results compared to results compared to results in previous work using PMF. Also, only 2 channels (out of 3) were used for motor imagery classification, while usually 8 channels are used. The Support Vector Machine was used for the classification stage of this work because is more reliable; it does not have the overfitting problem and its performance is good for binary EEG data.

5.2 Future Work

To improve the presented method to classify motor imagery data better, some suggestions are given below:

- Study bipolar channels for EEG recordings.
- Another option to improve accuracies is to explore other methods of PMF or the Spectro-temporal Pursuit Algorithm for feature extraction.
- Also, the implementation of an Adaptive Filter to extract undesirable noise in the EEG data could also improve accuracies.

BIBLIOGRAPHY

[1] "BCI Competition II." *BCI Competition II. N.P., N.D.* Web. 23 Mar. 2017. <http://www.bbci.de/competition/ii/>.

[2] "BCI Competition III." *BCI Competition III. N.P., N.D.* Web. 23 Mar. 2017. <http://www.bbci.de/competition/iii/desc_V.html>.

[3] A. Schlögl, K. Lugger and G. Pfurtscheller (1997) Using Adaptive Autoregressive Parameters for a Brain-Computer-Interface Experiment, Proceedings of the 19th Annual International Conference if the IEEE Engineering in Medicine and Biology Society ,vol 19, pp.1533-1535.

[4] Age Smilde, Rasmus Bro, and Paul Geladi. Multi-way Analysis. Wiley, West Sussex, England, 2004.

[5] Audone, Bruno, et al. "The short time Fourier transform and the spectrograms to characterize EMI emissions." *Electromagnetic Compatibility-EMC EUROPE, 2016 International Symposium on*. IEEE, 2016.

[6] Berry, M. W., Browne, M., Langville, A. N., Pauca, V. P., & Plemmons, R. J. (2007). Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, *52*(1), 155-173.

[7] C. Neuper, A. Schlögl, G. Pfurtscheller (1999) Enhancement of left-right sensorimotor EEG differences during feedback-regulated motor imagery. *J Clin Neurophysiol*. 16(4):373-82.

[8] Chang, C. C., and C. J. Lin. "LIBSVM: A library for support vector machines. 2001.[cited in 2001]." Software Available from: http://www. csie. ntu. edu. tw/cjlin/libsvm.

[9] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[10] Diez, P. F., Mut, V., Laciar, E., & Avila, E. (2010, August). A comparison of monopolar and bipolar EEG recordings for SSVEP detection. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE* (pp. 5803-5806). IEEE.

[11] Drineas, P., Kannan, R., & Mahoney, M. W. (2006). Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, *36*(1), 132-157.

[12] Duffy, Frank H., Vasudera G. Iyer, and Walter W. Surwillo. Clinical electroencephalography and topographic brain mapping: Technology and practice. Springer-Verlag Publishing, 1989.

[13] Fabien Lotte, Marco Congedo, Anatole Lecuyer, Fabrice Lamarche. “A review of classification algorithms for EEG-based brain–computer interfaces”. Journal of Neural Engineering, Institute of Physics: Hybrid Open Access, 2007, 4.

[14] Gordon E. Synchronous Gamma activity: A review and contribution to an integrative neuroscience model of schizophrenia. Brain Res. Rev. 2003; 41:57–78.

[15] Hoyer, Patrik O. "Non-negative matrix factorization with sparseness constraints." *The Journal of Machine Learning Research* 5 (2004): 1457-1469.

[16] Hsu, Chih-Wei, and Chih-Jen Lin. “A comparison of methods for multiclass support vector machines.” Neural Networks, IEEE Transactions on 13.2 (2002): 415-425.

[17] Huttenlocher, Peter R. "Morphometric study of human cerebral cortex development." *Neuropsychologia* 28.6 (1990): 517-527.

[18] Kohavi, R. (1995, August). A study of cross-validation and bootstrap for accuracy estimation and model selection. In Ijcai (Vol. 14, No. 2, pp. 1137-1145).

[19] Kohavi, Ron. "A study of cross-validation and bootstrap for accuracy estimation and model selection." *Ijcai*. Vol. 14. No. 2. 1995.

[20] L. Qin and B. He, “A wavelet-based time–frequency analysis approach for classification of motor imagery for brain–computer interface applications,” J. Neural Eng. Vol. 2 no. 65, 2005.

[21] Lee, Daniel D., and H. Sebastian Seung. "Algorithms for non-negative matrix factorization." *Advances in neural information processing systems*. 2001.

[22] Lee, H., Cichocki, A., & Choi, S. (2006, September). Nonnegative matrix factorization for motor imagery EEG classification. In *International Conference on Artificial Neural Networks* (pp. 250-259). Springer Berlin Heidelberg.

[23] Lee, Hyekyoung, and Seungjin Choi. "Group nonnegative matrix factorization for EEG classification." *International Conference on Artificial Intelligence and Statistics*. 2009.

[24] Lee, Hyekyoung, et al. "Nonnegative tensor factorization for continuous EEG classification." *International journal of neural systems* 17.04 (2007): 305-317.

[25] Lekshmy, V. M., Rekha, P., & Ramesh, M. V. (2016, September). Impact of algorithm complexity on energy utilization of wireless sensor nodes. In *Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on* (pp. 2592-2596). IEEE.

[26] McFarland, D. J., Miner, L. A., Vaughan, T. M., & Wolpaw, J. R. (2000). “Mu and beta rhythm topographies during motor imagery and actual movements”. Brain topography, 12(3), 177-186.

[27] Mustafa, M., Taib, M. N., Murat, Z. H., Sulaiman, N., and Aris, S. A. M. (2011, March). The analysis of eeg spectrogram image for brainwave balancing application using ann. In Computer

Modelling and Simulation (UKSim), 2011 UkSim 13th International Conference on (pp. 64-68). IEEE.

[28] N. Robinson, A. P. Vinod, K. K. Ang, K. P. Tee and C. T. Guan, “EEG based classification of fast and slow hand movements using wavelet-CSP algorithm,” IEEE trans. Biomedical engineering, vol. 60, n0. 8, Aug. 2013.

[29] Pfurtscheller G, Neuper C, Schlögl A, Lugger K. (1998) Separability of EEG signals recorded during right and left motor imagery using adaptive autoregressive parameters. *IEEE Trans Rehabil Eng.* 6(3):316-25.

[30] Qiao, Xiaoyan, et al. ”Feature extraction and classifier evaluation of EEG for imaginary hand movements.” Natural Computation (ICNC), 2010 Sixth International Conference on. Vol. 4. IEEE, 2010.

[31] Repovs, Grega. "Dealing with noise in EEG recording and data analysis." *Informatica Medica Slovenica*. Vol. 15. No. 1. 2010.

[32] Schacke, Kathrin. "On the kronecker product." *Master's thesis, University of Waterloo* (2004).

[33] Schlögl A., Neuper C. Pfurtscheller G. (2002) Estimating the mutual information of an EEG-based Brain-Computer-Interface, *Biomedizinische Technik* 47(1-2): 3-8.

[34] T. Guernine. (2011,). New fuzzy multi-class method to train SVM. Presented at Third International Conference on Advances in Databases, Knowledge, and Data Application.

[35] Yang, Y., Chevallier, S., Wiart, J., & Bloch, I. (2012, August). Time-frequency selection in two bipolar channels for improving the classification of motor imagery EEG. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE* (pp. 2744-2747). IEEE.

[36] Zhong J.; Shuren Q. "Detection of EEG basic rhythms feature by using band relative intensity ratio (brir). ICASSP, VI:429432, Apr 2003.

[37] Camacho, Jaime, and Vidya Manian. "Real-time single channel EEG motor imagery based Brain Computer Interface." *World Automation Congress (WAC), 2016*. IEEE, 2016.

[38] Cormen, T. H., et al. "Introduction to algorithms 2nd edition. chapter 3: Growth of Functions."