INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
Laboratorio de Robótica y Mecatrónica

Control de una silla de ruedas en un entorno virtual mediante una interfaz cerebro - computadora

TESIS

Para obtener el grado de:

Maestría en Ciencias en Ingeniería de Cómputo

Presenta:

Ing. Carlos Daniel Virgilio González

Directores de tesis:
Dr. Juan Humberto Sossa Azuela
Dr. Elsa Rubio Espino

Ciudad de México Noviembre 2017

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 12:00 horas del día 22 del mes de noviembre de 2017 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Control de una silla de ruedas en un entorno virtual mediante una interfaz cerebro - computadora"

Presentada por el alumno(a):

Virgilio	**González**	**Carlos Daniel**
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

A	1	6	1	1	6	2

aspirante de: **MAESTRÍA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de Tesis

Dr. Juan Humberto Sossa Azuela

Dra. Elsa Rubio Espino

Dr. Sergio Suárez Guerra

Dr. Herón Molina Lozano

Dr. Jesús Yaljá Montiel Pérez

Dr. Víctor Hugo Ponce Ponce

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Marco Antonio Ramírez Salinas

INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
DIRECCIÓN

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 30 del mes Noviembre del año 2017, el (la) que suscribe Carlos Daniel Virgilio González alumno (a) del Programa de Maestría en Ciencias en Ingeniería de Cómputo con número de registro A161162, adscrito a Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. Juan Humberto Sossa Azuela y Dra. Elsa Rubio Espino y cede los derechos del trabajo intitulado Control de una silla de ruedas en un entorno virtual mediante una interfaz cerebro - computadora, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección danielvg92@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Carlos Daniel Virgilio González

Agradecimientos

A mis padres que siempre han sido mi fuente de apoyo constante e incondicional en toda mi vida y más aún en los duros años de mi carrera profesional, sin su ayuda hubiera sido imposible culminar esta etapa de mi vida, quiero agradecerles por todo, no me alcanzan las palabras para expresar lo bien que me siento por tener una familia tan asombrosa, con quienes he podido aprender cómo luchar y salir victorioso ante las diversas adversidades de la vida.

A mis hermanos por ese respaldo que me dan, así como esa unión que me permite superar cualquier problema que se presenta, se que puedo contar con ellos en todo momento.

Al Dr. Juan Humberto Sossa Azuela y la Dra. Elsa Rubio Espino por su dedicación y esfuerzo en la realización de este proyecto, agradezco sus conocimientos, su orientación, paciencia y motivación las cuales han sido fundamentales para la realización de este trabajo, así como para mi formación académica.

A los miembros del jurado: el Dr. Sergio Suárez Guerra, el Dr. Herón Molina Lozano, el Dr. Jesús Yaljá Montiel Pérez y el Dr. Victor Hugo Ponce Ponce por la atención prestada al realizar la revisión de este trabajo y proporcionar las correcciones pertinentes, gracias a lo cual se pudo mejorar.

A mis compañeros y profesores integrantes del laboratorio de Robótica y Mecatrónica por su apoyo y comentarios durante el desarrollo de este trabajo.

Al Centro de Investigación en Computación por su apoyo y facilidades que contribuyeron para el desarrollo de este trabajo.

Al Consejo Nacional de Ciencia y Tecnología(CONACYT) por haberme otorgado una beca para realizar mis estudios de Maestría.

Al Instituto Politécnico Nacional (IPN) y la Secretaria de Investigación y Posgrado (SIP) por el apoyo económico brindado para llevar a cabo esta investigación. Este trabajo fue apoyado económicamente por SIP-IPN (números 20171548 y 20170693) y CONACYT (número 155014[Investigación Básica] y número 65[Fronteras de la Ciencia]).

Ing. Carlos Daniel Virgilio González

Resumen

En este proyecto se realizó el diseño e implementación de un control de una silla de ruedas. Fue establecido en un entorno basado en realidad virtual. El control se implemento mediante señales cerebrales electroencefalograficas (EEG) aplicadas a un conjunto de algoritmos de machine learning y redes neuronales.

El proceso consistió de cuatro etapas:

1. Organización de las señales EEG.
2. Extracción de rasgos característicos.
3. Aplicación de los algoritmos de clasificación.
4. Prueba de desempeño en el entorno virtual.

Las señales EEG con las que se trabajó fueron basadas en cuatro acciones motoras provenientes de dos diferentes sujetos. Se utilizó el paradigma de visualización motora. Estas señales EEG se obtuvieron de una base de datos proporcionada por la Universidad Tecnológica de Graz.

Para la extracción de rasgos característicos se realizó mediante la aplicación del algoritmo conocido como Common Spatial Pattern (CSP) en conjunto con el cálculo del valor de la media cuadrática (RMS) y la varianza.

Los algoritmos de clasificación utilizados fueron K-Nearest Neighbors (KNN), Perceptrón Multicapa, Maquinas de Soporte Vectorial (MSV) y Redes Neuronales Morfológicas Dendríticas (RNMD) para dos y cuatro clases. Para cuatro clases se utilizó una red de clasificadores binarios.

Se eligieron los clasificadores con mejor desempeño y se implementó el control del modelo de una silla de ruedas virtual. El entorno virtual fue creado en el motor de videojuegos Unity. Se obtuvieron porcentajes de clasificación del 79.44 %(sujeto 1) y del 67.50 %(sujeto 2).

Abstract

In this project the design and implementation of a control of a wheelchair was carried out. It was established in an environment based on virtual reality. The control was implemented by brain electroencephalographic (EEG) signals applied to a set of machine learning algorithms and neural networks.

The process consisted of four stages:

1. Organization of EEG signals.
2. Extraction of features.
3. Application of classification algorithms.
4. Performance test in the virtual environment.

The EEG signals with which we worked were based on four motor actions from two different subjects. The motor visualization paradigm was used. These EEG signals were obtained from a database provided by the Technological University of Graz.

For the extraction of characteristic features, it was carried out by applying the algorithm known as the Common Spatial Pattern (CSP) in conjunction with the calculation of the value of the root mean square (RMS) and the variance.

The classification algorithms used were K-Nearest Neighbors (KNN), Multilayer Perceptron (MLP), Support Vector Machines (SVM) and Dendrite Morphological Neural Networks (DMNN) for two and four classes. For four classes, a network of binary classifiers was used.

The classifiers with the best performance were chosen and the control of the model of a virtual wheelchair was implemented. The virtual environment was created in the Unity video game engine. Classification percentages of 79.44 % (subject 1) and 67.50 % (subject 2) were obtained.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En la actualidad una cantidad considerable de la población mundial sufre de algún tipo de discapacidad que ocasiona dificultades a la hora de desarrollar su rutina diaria. Pueden ir desde la perdida de un sentido (Oído, vista, olfato, etc.) hasta problemas cognitivos, inclusive problemas para desplazarse y según la gravedad pueden requerir que el paciente necesite de otra persona que lo apoye y le ayude a desarrollar todas las tareas cotidianas.

La discapacidad es un tema importante a nivel mundial, más de mil millones de personas viven en todo el mundo con alguna forma de discapacidad, de ellas, casi 200 millones experimentan dificultados considerables en su funcionamiento. Se piensa que este número aumentará debido a que la población va envejeciendo y va aumentando el riesgo debido a enfermedades crónicas como la diabetes, enfermedades cardiovasculares, cáncer o algún tipo de trastorno en la salud mental.

Las personas con discapacidad sufren de diversas problemáticas día con día, así como un deterioro en su forma de vida, tanto en resultados sanitarios como en tasas de pobreza, una de las causas de esto, además de la dificultad dada por la misma discapacidad, son los obstáculos que están presentes. Aunque las personas sin discapacidad no perciben estos obstáculos con facilidad, estos entorpecen el acceso a servicios como son la salud, educación, empleo, transporte, entre otros, que son fundamentales para toda persona.

Existen diferentes tipos de discapacidad y asi mismo diferentes modos de clasificacion. Para tomar una referencia, se utilizó la forma de clasificacion definida por la OMS [28], la cual establece 6 dominios principales los cuales no son excluyentes entre si y muchas personas pueden tener una discapacidad que abarca más de un dominio, son: Vista, Oído, Movilidad, Funciones cognitivas, Autocuidado, Comunicación.

Este tema sobre discapacidad tiene gran importancia en México, de acuerdo con el Censo de Poblacion y Vivienda del año 2010, en ese año las personas que tenían algun tipo de discapacidad eran 5 millones 739 mil 270, lo que representaba 5.1 % de la poblacion total en ese momento. Predominaba la dificultad para caminar o moverse, estando ésta presente en más de la mitad de las personas con discapacidad. [19].

Para analizar las problemáticas existentes entorno a la discapacidad motora se definirá a grandes rasgos el alcance de ésta. Este tipo de discapacidad se refiere a la dificultad de una

persona para desplazarse o subir escaleras independientemente de la causa, incluyendo a personas que hayan perdido un miembro o parte de éste, o aun teniendo aun ese miembro no pueden moverse de tal forma que necesitan ayuda de terceros o el uso de una silla de ruedas u otro dispositivo de apoyo.

Como se ha dicho la discapacidad tiene un gran impacto en el mundo siendo una de las más importantes la capacidad de moverse, por lo mismo se han llevado a cabo diferentes proyectos con la finalidad de mejorar la calidad de vida de personas con este tipo de situación.

El tipo de solución dependerá de la gravedad que presente el paciente, existirán pacientes que cuenten con sus piernas pero tengan daño a nivel tisular por lo cual no pueda flexionar y por lo tanto no se pueda mover, este tipo de pacientes puede utilizar desde las típicas muletas hasta alguna ortesis asistencial que les permita realizar esos movimientos.

Otro tipo de paciente es el que no cuenta con uno o ambos miembros inferiores o sea tanto el daño que no puedan apoyar su peso en ellos, este tipo de pacientes actualmente utilizan una silla de ruedas y si tienen la capacidad y la fuerza suficiente en los miembros superiores pueden desplazarse por ellos mismos. Lo anterior no ocurre en todos los casos, ya que en ocasiones no pueden desplazarse por si mismos y requieren de una persona que los asista empujándolos.

Actualmente esto último no tiene una solución que le de independencia al paciente. Algunas propuestas para solventar eso es el uso de una silla de ruedas que pueda controlar el paciente o en el mejor de los casos el uso de un exoesqueleto, siendo este último un área de investigación de gran impacto actualmente.

Desde sus orígenes las tecnologías se han desarrollado debido a las necesidades del hombre, por lo mismo el área medica ha sido prioritaria en la búsqueda de mejorar la calidad de vida de los pacientes. Así han surgido diferentes ideas utilizando diferentes métodos para implementar comunicación entre el hombre y su entorno.

Un área de investigación con gran auge es la de las interfaces humano-máquina, las cuales tienen como objetivo generar un enlace con el cual se comuniquen estas dos partes, no necesariamente mediante un joystick o control. Esta línea de investigación pretende utilizar alternativas para llevar a cabo este control, ya sea mediante alguna bioseñal o gestos realizados por alguna parte específica del cuerpo humano.

Una parte específica dentro de las interfaces humano-máquina es en donde se utilizan señales cerebrales para realizar esta comunicación. El cerebro es el que actúa como centro de mando para el control del cuerpo humano por lo cual sería la forma más precisa de transmitir lo que se quiere que haga un dispositivo, a este tipo de campo se le conoce como interfaces cerebro-computadora.

Este método sería el ideal para solucionar los problemas que presentan algún tipo de discapacidad motriz, ya sea desde no poder mover un miembro hasta no poder desplazarse sin la ayuda de un tercero. En este proyecto se buscó controlar algún dispositivo o robot utilizando este método, específicamente una silla de ruedas.

1.1. Justificación

Así como se mencionaron las diferentes soluciones que pueden mejorar la calidad de vida de un paciente con discapacidad motriz se llega a la conclusión que algo que puede ofrecer asistencia a un paciente es el uso de la ya existente silla de ruedas, solo queda solucionar el problema de la independencia que se buscaría darle al paciente cuando la utilizara.

Lo que se busca con una interfaz humano-máquina es establecer comunicación entre éstos, con la finalidad de tener más precisión en los procesos y obtener mayor comodidad al realizar estas acciones.

Una interfaz humano-máquina se puede llevar a cabo por diferentes métodos, como son el movimiento de alguna parte del cuerpo, la respiración, el control de un joystick con la mano, etc.

La problemática que se quiere alcanzar es implementar una forma de control en la que todo paciente pueda usarlo. El método que cumple con esto es el uso de una interfaz cerebro-computadora, ya que ésta emplea las señales cerebrales para llevar a cabo el control de algo, en este caso una silla de ruedas.

La principal razón para utilizar el cerebro humano es que éste es el centro de mando del cuerpo, por lo cual, en un caso extremo donde el paciente sufre tetraplejia seguirá presentando actividad cerebral por lo cual podría operar una interfaz cerebro-computadora.

Este es el caso del tema del proyecto expuesto en este documento, en el cual se busca implementar una interfaz con la cual se pueda ofrecer un método alterno para el manejo de una silla de ruedas, apoyando a pacientes que no tienen la capacidad o tienen dificultad para moverse, empleando una interfaz cerebro-computadora.

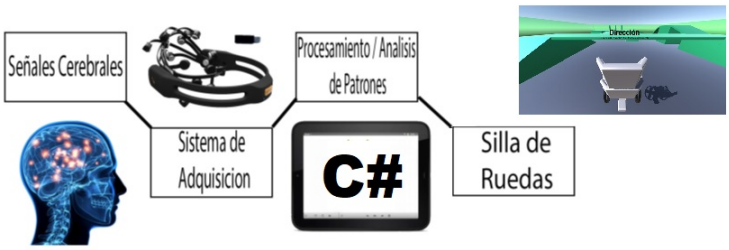


Figura 1.1: Planteamiento de la solución.

En la figura 4.4 se muestra la solución planteada para el desarrollo de este proyecto. Estará dividida en tres etapas: 1)Sistema de adquisición de señales cerebrales, 2)Procesamiento y clasificación de señales cerebrales y 3)Control de la silla de ruedas virtual. La etapa critica en este proyecto es la del procesamiento y clasificación de las señales cerebrales ya que aquí se implementaran métodos adecuados para la extracción de rasgos característicos, así como la aplicación de modelos para la clasificación de las mismas.

1.2. Objetivos

1.2.1. Objetivo general

Controlar una silla de ruedas en un entorno virtual mediante una interfaz cerebro-computadora utilizando señales basadas en visualización motora.

1.2.2. Objetivos particulares

- Implementar un sistema de procesamiento para el manejo de señales cerebrales.
- Implementar un algoritmo de extracción de rasgos característicos de las señales cerebrales.
- Diseñar e implementar un algoritmo para clasificar cuatro clases de tareas motoras.
- Diseñar e implementar un entorno virtual en el motor de gráficos Unity.
- Implementar el sistema para el control de una silla de ruedas virtual.

1.3. Organización del documento

A continuación se muestra una breve descripción de los temas a tratar en los siguientes capítulos.

Capítulo 2 Estado del arte

Se hablará de los avances en el campo de las interfaces cerebro-computadora que han servido como referencia para la realización del proyecto.

Capítulo 3 Marco teórico

Donde serán explicados los conceptos y herramientas que fueron empleadas para llevar acabo en el proceso de implementación de esta interfaz cerebro-computadora.

Capítulo 4 Metodología

Describe el proceso de diseño e implementación del mismo, como es la adquisición de señales cerebrales, el procesamiento aplicado a éstas, la clasificación de las tareas mentales, el control de la silla, etc.

Capítulo 5 Presentación y discusión de resultados

Se muestran los resultados y se explica a detalle la metodología de los experimentos desarrollados para probar las capacidades del prototipo.

Capítulo 6 Conclusiones

Son explicados los resultados obtenidos en los experimentos y si fueron alcanzados los objetivos propuestos.

Capítulo 7 Recomendaciones y trabajo futuro

Se plantean cuales son las futuras mejoras que se le pudieran hacer al prototipo.

Capítulo 2

Estado del arte

En la actualidad existen diferentes enfoques en el tema de interfaz humano-máquina (Human-Machine Interface) ya que el ser humano cuenta con diferentes sentidos que le permiten comunicarse con su entorno, a partir de los cuales se busca enlazarlo o comunicarlo con una máquina. Con esto se busca gestionar las acciones de la máquina, incluso extender las capacidades del humano mismo en algunos casos.

Por lo anterior han surgido diferentes ideas de implementación de las interfaces, por ejemplo, se puede utilizar el movimiento de los ojos, gestos realizados con alguna parte de nuestro cuerpo, por voz, etc. Una de los métodos con mayor impacto es el utilizar las señales cerebrales para establecer una interfaz, éstas en específico reciben el nombre de interfaz cerebro-computadora (Brain Computer Interface - BCI).

Las BCI tienen como objetivo medir las señales electroencefalografícas, identificar patrones relacionados con ciertos eventos y de cierta forma traducirlas en ordenes a la máquina.

Este tipo de proyectos han surgido a lo largo del tiempo ya que el área de la salud es un campo que tiene impacto en todo ser humano, por lo mismo siempre se buscan soluciones para mejorar la calidad de vida de las personas con este tipo de problemas.

Una de las primeras aplicaciones fue realizada en el año de 1991 [35], en este trabajo implementaron una interfaz cerebro-computadora basada en señales electroencefalograficas (EEG) para controlar el cursor de navegación en una pantalla. Esta aplicación ha sido implementada en gran cantidad de artículos, con diferente enfoque en la clasificación o extracción de características en el manejo de las señales cerebrales.

Una aplicación reciente para el control de navegación en pantalla fue presentado en [15], en el cual tienen como objetivo controlar el *Tobi Dynavox Communicator 5*. Es un software diseñado para ofrecer accesibilidad a la hora de utilizar una computadora, ya que implementa una interfaz basada en iconos, comandos de voz y un cursor controlado por la mirada. El objetivo de en este trabajo fue agregar un controlador basado en el procesamiento de señales cerebrales. Utilizan como base el análisis de la onda P300, la cual es un potencial evocado presentado como una deflexión positiva de voltaje con una latencia de unos 300ms en el EEG.

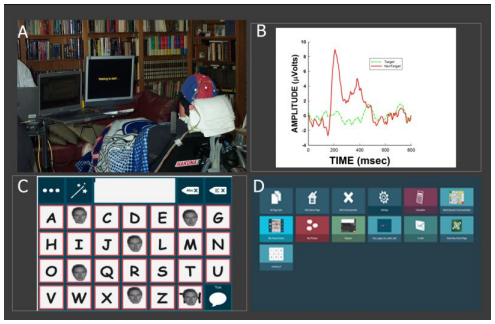


Figura 2.1: Interfaz cerebro-computadora en el software *Tobi Dynavox Communication 5*

Este mismo paradigma basado en la onda P300 es comunmente utilizado en diversas aplicaciones como: sistema de deletreo [12], control de un robot humanoide(NAO) [22], control de un cursor en una pantalla [23], control de la iluminación [20], control de un objeto en un entorno virtual [7],manejo de un teléfono celular [9], incluso el control de una silla de ruedas [26].

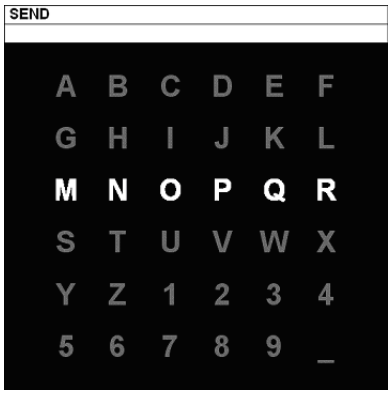


Figura 2.2: Interfaz utilizada en aplicaciones de sistemas de deletreo.

En el tema del **procesamiento de señales** tambien existen diferentes formas de abordar el problema, por ejemplo en [35] utilizan análisis de frecuencia de las señales adquiridas de electrodos colocados en el surco central del cerebro, este método dependía de la amplitud de la señal cerebral. Para amplitudes grandes se mueve el cursor hacia arriba y para las más pequeñas amplitudes se mueve hacia abajo.

Otro método muy utilizado en el procesamiento de señales cerebrales es la transformada wavelet en conjunto con el método de filtrado de patrón común espacial para clasificar patrones de 2 clases de señales de tipo *Motor Imagery*, utilizan modelos de clasificación como son K vecinos cercanos y máquina de soporte vectorial [16].

De igual forma se utilizan otros metodos de extraccion como son el modelo autorregresivo de orden 6 [18], banda de potencia [18],análisis de componentes independientes (ICA),análisis de componentes principales (PCA), entre otros.

El método de extracción de caracteristicas por filtrado de patrón común espacial (Commun

Spatial Pattern) ya ha sido utilizado en la clasificación de señales cerebrales y actualmente tiene gran popularidad ya que mejora el desempeño de la misma [3, 8, 14, 17, 24, 36, 38].

En el año 2007 Reinhold Scherer et al. [21] realizaron una investigación acerca de la implementación de una BCI, en este trabajo buscaron demostrar por primera vez que las señales cerebrales de pacientes con tetraplejia podían ser utilizados para controlar los movimientos de una silla de ruedas en ambientes virtuales. Obtuvieron buenos resultados para la tarea asignada, la cual era "recorrer" una calle virtual deteniéndose en diferentes puntos. Fue desarrollado en el Laboratorio de interfaces cerebro-computadora de la Universidad Tecnológica de Graz, ubicada en Graz, Austria.

Figura 2.3: Interfaz cerebro-computadora en Universidad Tecnológica de Graz

Como fue mencionado en [26] se implementó una interfaz cerebro-computadora para controlar una silla de ruedas utilizando como base el estímulo evocado de la onda P300. Al usuario le presentan una interfaz con botones y éste debe concentrarse en la posición del botón que desea presionar.

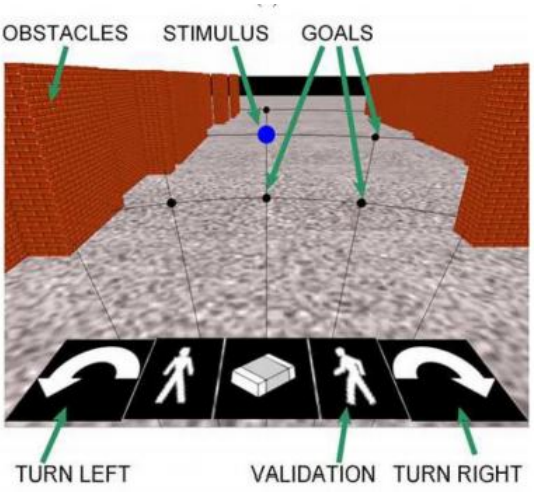


Figura 2.4: Interfaz cerebro-computadora para el control de una silla de ruedas.

Otro ejemplo es el proyecto realizado por Andrzej S. Cichocki et al. [4] en 2009 en el "RIKEN BSI-TOYOTA Collaboration Centre" en Saitama, Japon. En este proyecto se utilizaron señales EEG para el control de la silla de ruedas. El proyecto tuvo la finalidad de apoyar a las personas con limitada movilidad, la única complicación que tenía era que necesitaba de alguien asistiendo al paciente para colocar los instrumentos de medición, ya que estos eran demasiado complejos tanto en su posicionamiento como en su instalación, además de la portabilidad del electroencefalografo teniendo como meta a futuro solucionar esto.

Han surgido otras ideas para implementar el control de una silla de ruedas, como por ejemplo una silla de ruedas controlada por olfateos. Ésta fue desarrollada en el año 2010 por Noam Sobel et al. [5] en el Departamento de Neurobiología en el Instituto Weizman en Rehovolt, Israel. El sistema identifica cambios en la presión del aire que se exhala e inhala en el proceso de respiración y los convierte en señales eléctricas, con las cuales se lleva a cabo el control de los movimientos de la silla de ruedas. Fue probado en pacientes sanos y pacientes con casos de parálisis obteniendo buenos resultados así como observaciones de que el sistema era de fácil manejo.

Además de esto también en el año 2013 Marton Juhasz et al en Hungria desarrolló una silla de ruedas controlada por movimientos realizados con la cabeza, esto también fue enfocado a pacientes con casos de cuadraplejía. La idea consistió en utilizar un sistema de navegación el cual indicara la posición de la cabeza. Actualmente Marton Juhasz se enfocó más al sistema de navegación, el cual ahora es conocido cono GyroSet.

Capítulo 3

Marco teórico

A continuación se explicarán las herramientas que fueron necesarias para la elaboración de este trabajo, entre las que se encuentran los fundamentos teóricos, algoritmos, descripción del hardware y software.

3.1. Interfaz cerebro-computadora (Brain Computer Interface)

Una interfaz cerebro-computadora es un canal de comunicación entre el cerebro y el mundo externo que hace posible el uso de protesis neuronales o dispositivos tecnologicos mediante las señales cerebrales generadas por una persona. [34]

Ésta es un tipo específico de interfaz humano-máquina, que como su nombre lo dice lo hace mediante el cerebro, utilizando las señales generadas por el mismo.

Como se menciona en [34] una interfaz cerebro-computadora consiste de 3 componentes esenciales:

- Estímulo voluntario codificado.
- Algoritmo computacional de control que decodifique el estímulo.
- Retroalimentación de los resultados en tiempo real.

Por lo mismo la parte correspondiente al algoritmo computacional constará de 3 pasos:

- Adquisición de señales cerebrales.
- Procesamiento de señales cerebrales.
- Clasificación de patrones.

3.2. Señales cerebrales

Señales cerebrales se refiere a la actividad eléctrica producida por las células cerebrales, esta actividad es el resultado de la suma de potenciales sinapticos en las neuronas. Al registro de esta actividad se le conoce como *electroencefalograma*(en adelante EEG). [25]

El EEG es el registro de la actividad eléctrica producida por las neuronas, específicamente de la corteza cerebral, esta medición se lleva a cabo mediante electrodos que son colocados en la superficie del cuero cabelludo.

Para captar la señal se utilizan diferentes tipos de electrodos:

- Electrodos superficiales: Se aplican sobre el cuero cabelludo.
- Electrodos basales: Se aplican en la base del cráneo sin necesidad de procedimiento quirúrgico.
- Electrodos quirúrgicos: para su aplicación es precisa la cirugía y pueden ser corticales o intracerebrales.

En la corteza se han podido identificar secciones en las cuales se ha presentado actividad eléctrica que parece estar relacionada a diferentes funciones congnitivas, como por ejemplo una area sensitiva o la area de actividad motora [6]. Como se muestra en la figura 3.1 el área motora se puede identificar en el lóbulo frontal, en la parte más cercana al surco central.

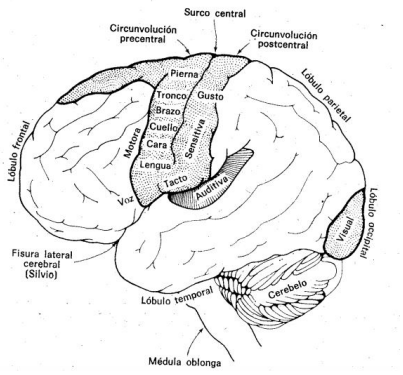


Figura 3.1: Corteza Cerebral

En el análisis del EEG se han observado diferentes ondas o ritmos, reciben el nombre de alfa, beta, theta y delta. Son diferenciadas por la banda de frecuencia en que se presentan, como se muestra en la figura 3.2 [6].

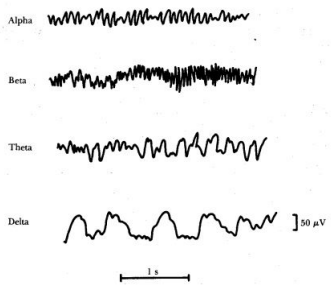


Figura 3.2: Ondas Normales en EEG

Las frecuencias correspondientes a cada onda son:

- Beta : $>$ 13 Hz
- Alfa : 8 - 13 Hz
- Theta : 3.5 - 7.5 Hz
- Delta : $<$ 3 Hz

Además de éstas existe la onda o ritmo Mu correspondiente a las bandas de frecuencia de 8 - 13 Hz y 15 - 25 Hz. [29] Ésta es particularmente importante ya que se genera en la corteza motora y por lo mismo relacionada directamente con la actividad motora, siendo base en el desarrollo del presente proyecto.

3.3. Sistema 10/20

Es un sistema específico y estándar para todos los laboratorios. Se emplea un mínimo de 21 electrodos, conformados por 19 electrodos craneanos y dos referenciales a oreja o mastoides, éstos últimos conocidos como auriculares. [25]

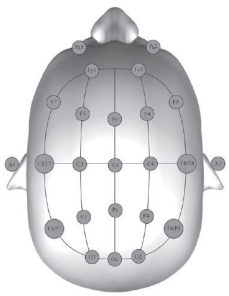


Figura 3.3: Colocación de electrodos utilizando el sistema 10 - 20

Como se observa en la figura 3.3 la colocación de los electrodos cubre la mayor parte de la corteza cerebral, sin embargo este sistema puede tener variaciones según las características del estudio a realizar. Puede aumentar el número de electrodos en una cierta área o incluso solo trabajar con los electrodos de un hemisferio.

3.4. Visualización motora (Motor Imagery)

El ser humano tiene la capacidad de imaginar casi cualquier cosa, si se tiene alguna experiencia relacionada mejora esta capacidad, ya sea una acción, un escenario o algún objeto. Mentalmente se pueden realizar acciones aunque en la realidad no se tenga la capacidad para llevarlas a cabo.

La visualización motora es un proceso cognitivo en el cual el sujeto imagina que está desarrollando una acción motora sin realizar movimiento ni tensar los musculos. Este proceso requiere de la activación de manera consiente de áreas específicas dedicadas a la preparación y ejecución de movimientos. [27]

En la actualidad se utiliza como terapia para la recuperación de pacientes con lesiones neurológicas, asi como en atletas se han observado efectos positivos. En algunos trabajos se ha demostrado que la terapia utilizando visualización motora mejora hasta un 40 % la reducción de dolor. [37]

3.5. Adquisición de señales cerebrales

3.5.1. Base de datos

BCI Competition III : Data set IIIa (4-class EEG data)

La base datos utilizada en este proyecto fue proporcionada por el Laboratorio de interfaces cerebro-computadora de la Universidad Tecnológica de Graz, en el evento llamado "BCI Competition III"(http://www.bbci.de/competition/iii/).

Esta base de datos fue grabada con un amplificador de electroencefalografía con 64 canales de adquisición de la marca Neuroscan, tomando solo 60 en el archivo del formato GDF. Las señales EEG fueron adquiridas con una frecuencia de muestreo de 250 Hz y un filtrado en la banda de frecuencia 1 - 50 Hz.

Cuenta con el registro de tres sujetos de prueba, de cada uno de ellos se tienen 90 patrones de las cuatro diferentes clases de acciones de visualización motora:

- Mano Izquierda (90 patrones)
- Mano Derecha (90 patrones)
- Pie (90 patrones)
- Lengua (90 patrones)

La colocación de los electrodos se muestra en la figura 3.4, como se observa se dispone de dos electrodos colocados detrás de las orejas como referencia y tierra.

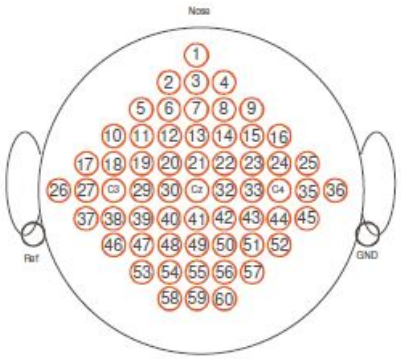


Figura 3.4: Colocación de electrodos para la adquisición de la Base de Datos

Cada patrón tiene una duración de siete segundos, se utilizó una metodología para la adquisición de los patrones como se observa en la figura 3.5.

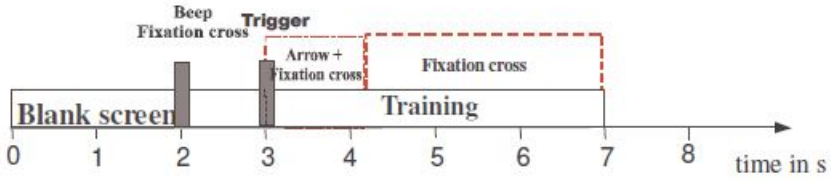
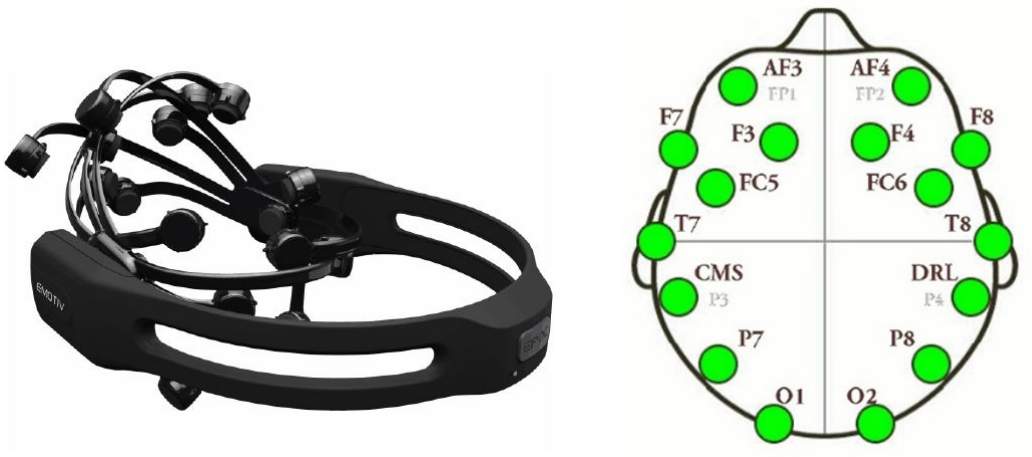


Figura 3.5: Secuencia para la adquisición de patrones.

Los primeros dos segundos la pantalla permanece en blanco, son el tiempo de relajación que tiene el paciente. En $t = 2s$ se escucha un estímulo auditivo que indica el inicio de la prueba, así mismo aparece en la pantalla una cruz para indicar al paciente que debe de estar preparado. En $t = 3s$ en la pantalla aparecerá una flecha a izquierda, derecha, arriba o abajo por un segundo, al mismo tiempo el sujeto debe imaginar el movimiento de la mano izquierda, mano derecha, lengua o pie, respectivamente hasta que la cruz desaparezca en $t = 7s$.

3.5.2. Emotiv EPOC+

Este es dispositivo para la adquisición de EEG (figura 3.6), cuenta con 14 canales además de dos electrodos de referencia. Digitaliza a una frecuencia de 128 muestras por segundo con 14 bits de resolución, cuenta con un filtrado en la banda de 0.2 - 45 Hz, además de filtros digitales del tipo notch para las frecuencias de 50 Hz y 60 Hz [13].



(a) Emotiv EPOC+ (b) Colocación de Electrodos

Figura 3.6: Dispositivo de adquisición de EEG.

Además cuenta con conectividad Bluetooth® y utiliza electrodos de tipo superficial a los cuales se les aplica solución salina para mejorar la conductividad.

Así mismo se realizaron grabaciones de patrones utilizando una metodología similar a la mostrada anteriormente: los primeros tres segundos la pantalla permanece en blanco, son el tiempo de relajación que tiene el paciente. En $t = 4s$ se escucha un estímulo auditivo que indica el inicio de la prueba y aparece en la pantalla una cruz para indicar al paciente que debe de estar preparado. En $t = 5s$ en la pantalla aparecerá una flecha a izquierda, derecha, arriba o abajo por 2 segundos, al mismo tiempo el sujeto debe imaginar el movimiento de la mano izquierda, mano derecha, lengua o pie, respectivamente hasta que la cruz desaparezca en $t = 10s$ (figura 3.7).

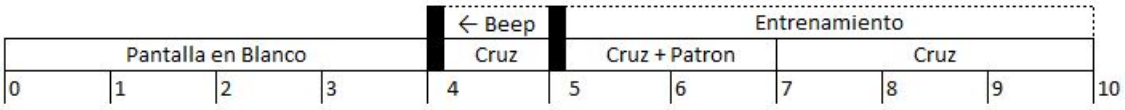


Figura 3.7: Secuencia para la adquisición de patrones.

3.6. Extracción de rasgos característicos

3.6.1. Filtrado de patrón común espacial (Common Spatial Pattern Filter)

En este proyecto se utilizó el método de extracción de rasgos característicos conocido como Common Spatial Pattern, CSP en adelante. Es un algoritmo basado en la descomposición

de señales EEG para generar rasgos característicos espaciales de dos diferentes clases. Este método es usado principalmente en imágenes para detección de rostros u objetos, tambien para la detección de anomalías en señales EEG. Fue aplicado por primera vez en interfaces cerebro-computadora por Ramoser [31], actualmente se ha seguido aplicando satisfactoriamente en este tipo de interfaces. Es utilizado para el diseño de filtros espaciales mediante la diagonalización simultanea de dos matrices de covarianza**??**.

El método se aplica de la siguiente forma: primero se calcula un kernel correspondiente al filtro espacial, siendo una matriz de tamaño NxN (N:número de canales) que permitirá extraer los rasgos característicos de las señales cerebrales correspondientes a cada una de las tareas mentales, el proceso a continuación es analizar la señal de prueba realizando un filtrado por medio de este kernel previamente calculado, con lo que se obtendrá un conjunto de rasgos característicos que permitirá discriminar entre las dos clases en el conjunto de datos de prueba. Una de estas clases se caracteriza por tener la máxima varianza y la otra por tener la mínima varianza, haciendo una clara diferencia entre estos dos grupos como se muestra en la figura 3.8. La idea principal de CSP es encontrar una matriz de proyección óptima.

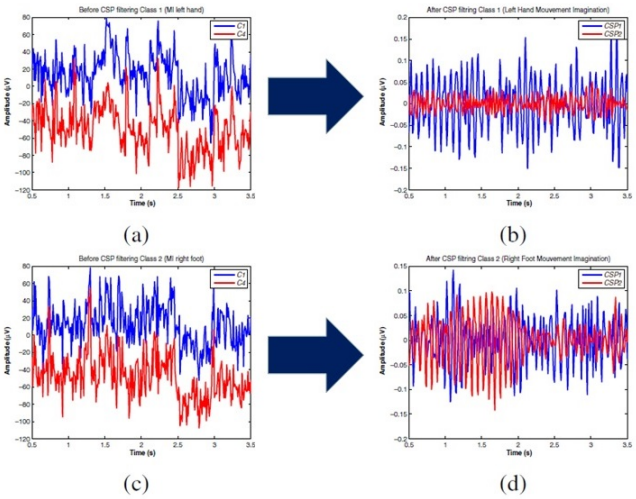


Figura 3.8: Efecto del filtrado con CSP**??**.

En la figura 3.8 se muestra el efecto del filtrado espacial. En las subfiguras a y c se muestra las señales EEG provenientes de dos canales (C1 y C4), siendo la subfigura a la tarea mental de imaginar el movimiento de la mano izquierda y la subfigura c el movimiento del pie derecho. Como primer enfoque se observa que ambas tareas mentales presentan señales parecidas por lo que no se puede realizar una discriminación adecuada. Al realizar el filtrado espacial se obtendrán nuevos canales, los cuales se observan en las subfiguras b y d. Como se observa ya existe una diferencia significante en las nuevas señales del canal CSP2 (de color rojo), para la tarea mental del movimiento de la mano izquierda el canal CSP2 presenta una señal con poca varianza, mientras que el movimiento del pie derecho presenta una gran varianza en el mismo canal. Con esta diferencia en las señales se puede realizar una discriminación adecuada.

A continuación se explica el método a detalle:

La señal EEG original de cada prueba es definida como E_{N*T}, donde N es el número de canales y T es el número de muestras. El proceso para el cálculo del CSP es el siguiente: Paso 1: Cálculo de la matriz de covarianza espacial de cada prueba de acuerdo a la ecuación 3.1

$$C = \frac{EE^T}{traza(EE^T)} \tag{3.1}$$

La traza(X) es la traza de la matriz X, lo que significa que es la suma de los elementos de la diagonal. Por lo cual el promedio de las matrices de covarianza para todas las pruebas de una clase (siendo C_1 las pruebas correspondientes a la clase 1 y C_2 las pruebas correspondientes a la clase 2) se calcula de acuerdo a las Ecuaciones 3.2 y3.3.

$$C_1 = \frac{\sum_{i=1}^{n} C_{1,i}}{n} \tag{3.2}$$

$$C_2 = \frac{\sum_{i=1}^{n} C_{2,i}}{n} \tag{3.3}$$

Obteniendo la covarianza del espacio de muestra C_c

$$C_c = C_1 + C_2 \tag{3.4}$$

Paso 2: Descomposición de la matriz de covarianza C_c

$$C_c = U_c \Lambda_c U_C^T \tag{3.5}$$

Donde Λ_c es una matriz diagonal de valores propios generalizados y U_c es una matriz cuyas columnas corresponden a los vectores propios.

Paso 3: Construir la matriz de transformación P y transformar la matriz de covarianza. Primero se construye la matriz de transformación P y la matriz de coeficientes espaciales S.

$$P = \Lambda_c^{-\frac{1}{2}} U_C^T \tag{3.6}$$

Luego se transforma C_1 y C_2 como sigue:

$$S_1 = PC_1P^T \tag{3.7}$$

$$S_2 = PC_2P^T \tag{3.8}$$

Finalmente se descomponen los valores propios con S_1 y S_2

$$S_1 = B\Lambda_1 B^T \tag{3.9}$$

$$S_2 = B\Lambda_2 B^T \tag{3.10}$$

Donde B es la matriz cuyas columnas son correspondidas por los vectores propios. Entonces se obtiene una matriz de proyección CSP $N * N$ llamada W de la siguiente manera:

$$W = (B^T P)^T \tag{3.11}$$

Se puede llamar a cada fila de la matriz W como un filtro espacial. Comúnmente solo algunos filtros espaciales son seleccionados, definiendo cierto valor m, reservando la primer m y la ultima m fila de la matriz W y removiendo las $N - 2m$ filas restantes del medio. A estas filas seleccionadas de la matriz W, se le conocerá como kernel del filtro espacial, por lo mismo ésta se puede aplicar a la matriz original de señales EEG.

$$Z_{2m*T} = W_{2m*N} E_{N*T} \tag{3.12}$$

Finalmente se obtiene Z_{2m*T} donde la señal EEG original es filtrada por la matriz W_{2m*N}.

3.6.2. Aproximación a 4 clases *Pair-Wise*

El metodo de CSP tiene una gran limitante, solo trabaja con dos clases a la vez. Para aplicar este metodo de extraccion en cuatro clases se realiza una aproximación conocida como Pair-Wise. El propósito de esta aproximación es el cálculo de rasgos CSP discriminando entre pares de clases [11].

$$\begin{array}{l} Clase[1,2] \text{ vs } Clases[3,4] \\ Clase[1,3] \text{ vs } Clases[2,4] \\ Clase[1,4] \text{ vs } Clases[2,3] \end{array} \tag{3.13}$$

Realizando la clasificación en 3 partes:

1. Clasificación de Clases 1-2 VS Clases 3-4
2. Clasificación de Clase 1 VS Clase 2
3. Clasificación de Clase 3 VS Clase 4

3.6.3. Media cuadrática

Después del filtrado espacial de las señales EEG originales se tienen nuevos vectores de datos a los cuales se les aplica el método estadístico del cálculo de la media cuadrática de la siguiente forma:

$$X_{2*m} = \sqrt{\frac{1}{N} \sum_{i=1}^{T} Z_{2m*i}^2} \tag{3.14}$$

Obteniendo como resultado un vector de tamaño $2 * m$.

3.6.4. Varianza

Así mismo a las señales EEG ya filtradas se les aplica el método estadístico del cálculo de la varianza.

Ésta es una medida de dispersión definida como la esperanza del cuadrado de la desviación de dicha variable respecto a su media.

$$V_{2*m} = \frac{1}{n} \sum_{n}^{i=1} (X_i - \overline{X})^2 \tag{3.15}$$

Obteniendo como resultado un vector de tamaño $2 * m$.

3.7. Clasificadores

3.7.1. K Vecinos Cercanos (K-Nearest Neighbors KNN)

El algoritmo k vecinos cercanos es un típico método de clasificación debido a su fácil implementación y a su buen desempeño.

El principio de funcionamiento es el comparar un nuevo patrón con los ya existentes en el conjunto de entrenamiento y tomar las etiquetas de los más parecidos.

Aquí es donde entra el valor k (número de vecinos), con este dato se sabrá cuantos patrones del conjunto de entrenamiento serán considerados y se definirá la etiqueta del nuevo patrón realizando una votación entre los patrones más parecidos.

Algoritmo 1: K Vecinos Cercanos
Para cada punto del conjunto de datos: Calcular la distancia entre inX y el punto actual. Organizar las distancias en orden creciente. Tomar los k puntos con la mejor distancia a inX. Encontrar la clase mayoritaria entre estos elementos. Regresar la clase mayoritaria como la predicción de clase para inX.

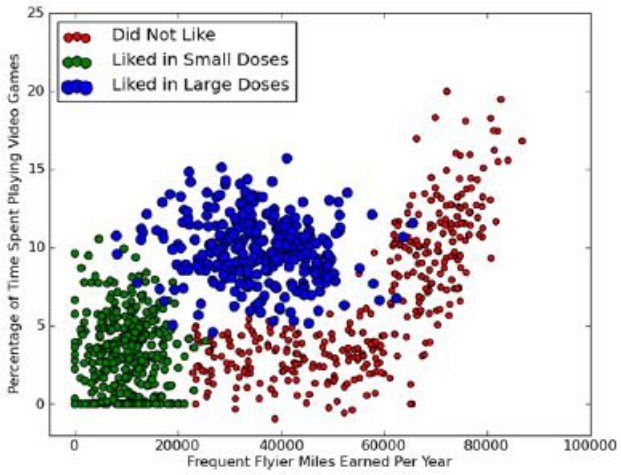


Figura 3.9: Ejemplo de un conjunto de datos para el algoritmo KNN [33].

3.7.2. Máquina de soporte vectorial (Support Vector Machine SVM)

Las máquinas de soporte vectorial fueron creadas por Vapnik basadas en una teoría de aprendizaje estadístico y pueden resolver problemas relacionados con espacio de muestra pequeño, relaciones no lineales y multiple clasificación [10].

El principio de la máquina de soporte vectorial es la construcción de un hiperplano como superficie de decisión para identificar las diferentes clases, así que maximiza el espacio entre ellas, como se muestra en la figura 3.10.

Para resolver los problemas de clasificación no lineal la MSV utiliza funciones kernel para convertir los problemas no lineales en lineales mediante el incremento de su dimensión.

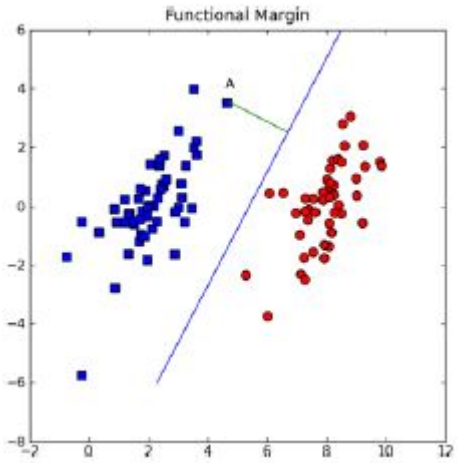


Figura 3.10: Ejemplo del funcionamiento de MSV [33].

SMO: Optimizador mínimo secuencial.

En 1996 John Pratt publicó un algoritmo llamado Optimizador Mínimo Secuencial para entrenar maquinas de soporte vectorial, la esencia de este método radica en tomar el gran problema de optimización y dividirlo en muchos problemas pequeños [30].

Los pequeños problemas pueden ser resueltos más fácilmente y atacando estos de manera secuencial se encontrará la misma respuesta que se obtiene al intentar resolver todo junto. Además de esto el tiempo de procesamiento se reduce considerablemente.

El algoritmo SMO trabaja encontrando un conjunto de alfas y variables b, una vez que se tiene este conjunto se puede realizar el cálculo de los pesos w y a continuación realizar el cálculo del hiperplano de separación.

Algoritmo 2: Optimizador Mínimo Secuencial

```
Crear un vector de alfas lleno de ceros
Mientras el número de iteraciones es menor a las iteraciones máximas:
    Para cada vector de datos en el conjunto de entrenamiento:
        Si el vector de datos puede ser optimizado:
            Seleccionar otro vector de datos aleatoriamente.
            Optimizar los dos vectores juntos.
            Si los vectores no pueden ser optimizados: break
    Si no hay más vectores que optimizar:
        incrementar el número del contador de iteraciones.
```

3.7.3. Perceptrón multicapa (Multilayer Perceptron MLP)

El perceptrón multicapa es una red neuronal artificial formada por múltiples capas, esto le permite resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón, puede ser totalmente o localmente conectado.

Se considera una red neuronal que consiste de L capas, sin contar la capa de entrada, al tomar una capa aleatoria llamada ℓ la cual tiene N_ℓ neuronas $X_1^{(\ell)}$, $X_2^{(\ell)}$, ..., $X_{N_\ell}^{(\ell)}$, cada una de ellas con una función de transferencia llamada $f^{(\ell)}$.

Esta función de transferencia puede ser diferente en cada capa y al utilizar como base la regla delta esta función debe ser diferenciable sin necesidad de ser lineal.

Estas neuronas reciben señales de las neuronas de la capa anterior $(\ell - 1)$. Por ejemplo, la neurona $X_j^{(\ell)}$ recibe señales de $X_i^{(\ell-1)}$ con un factor de peso de $w_{ij}^{(\ell)}$. Por lo tanto se tiene $N_{\ell-1}$ para N_ℓ con una matriz de pesos, $\mathbf{W}^{(\ell)}$, en la cual sus elementos están dados por $W_{ij}^{(\ell)}$, para $i = 1, 2, \ldots, N_{\ell-1}$ y $j = 1, 2, \ldots, N_\ell$. La neurona $X_j^{(\ell)}$ también tiene un bias dado por $b_j^{(\ell)}$ y su activación es $a_j^{(\ell)}$.

Para simplificar la notación se utilizará $n_j^{(\ell)}$ para mostrar la entrada de la red en la neurona

$X_j^{(\ell)}$. La cual está dada por:

$$n_j^{(\ell)} = \sum_{i=1}^{N_{\ell-1}} a_i^{(\ell-1)} w_{ij}^{(\ell)} + b_j^{(\ell)}, \qquad j = 1, 2, \ldots, N_\ell.$$

Así como la activación de la neurona $X_j^{(\ell)}$ es:

$$a_j^{(\ell)} = f^{(\ell)}(n_j^{(\ell)}) = f^{(\ell)}(\sum_{i=1}^{N_{\ell-1}} a_i^{(\ell-1)} w_{ij}^{(\ell)} + b_j^{(\ell)}).$$

Se puede considerar la capa cero como la capa de entrada si un vector de entrada $\mathbf{x}$ tiene N componentes, entonces $N_0 = N$ y las neuronas en la capa de entrada tienen activaciones $a_i^{(0)} = x_i, i = 1, 2, \ldots, N_0$.

La capa L de la red neuronal es la capa de salida, asumiendo que el vector de salida $\mathbf{y}$ tiene M componentes, entonces se tendrá $N_L = M$. Estos componentes están dados por $y_j = a_j^{(L)}, j = 1, 2, \ldots, M$.

Para cualquier vector de entrada dado, las ecuaciones de arriba pueden ser utilizadas con el fin de encontrar la activación en cada neurona para cualquier conjunto de pesos dado. Lo que resta es saber como entrenar a la red para establecer el conjunto de pesos que desempeñe una tarea en específico.

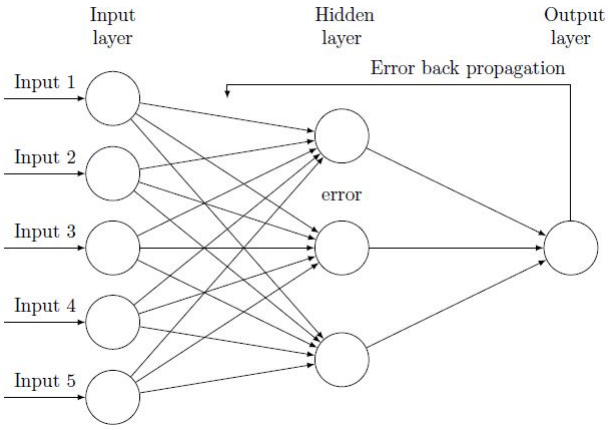


Figura 3.11: Arquitectura de un perceptrón multicapa.

Algoritmo de entrenamiento: Propagación hacia atrás.

El método que se utiliza para entrenar el perceptrón multicapa es el algoritmo de propagación hacia atrás, este es un algoritmo supervisado por lo cual se asume que el conjunto de patrones está etiquetado según su clase.

El vector de entrenamiento $\mathbf{s}^{(q)}$ tiene N componentes,

$$\mathbf{s}^{(q)} = \begin{bmatrix} s_1^{(q)} & s_2^{(q)} & \ldots & s_N^{(q)} \end{bmatrix},$$

y sus etiquetas $\mathbf{t}^{(q)}$ tienen M componentes,

$$\mathbf{t}^{(q)} = \left[\begin{array}{cccc} t_1^{(q)} & t_2^{(q)} & \dots & t_M^{(q)} \end{array} \right].$$

Como en la regla delta, cada patrón del conjunto entrenamiento es presentado uno a la vez a la red durante el proceso de entrenamiento, suponiendo que en el tiempo t de este proceso, un patrón de entrenamiento $\mathbf{s}^{(q)}$ de una q en particular es presentado como entrada, $\mathbf{x}(t)$, a la red. La señal de entrada puede ser propagada hacia adelante a través de la red utilizando las ecuaciones de activación propias de la red y con el conjunto de pesos obtener la salida correspondiente $\mathbf{y}(t)$. Los pesos son ajustados utilizando el algoritmo de descenso por gradiente para minimizar el error cuadrático para este patrón de entrenamiento:

$$E = \|\mathbf{y}(t) - \mathbf{t}(t)\|^2,$$

donde $\mathbf{t}(t) = \mathbf{t}^{(q)}$ es la etiqueta correspondiente para el patrón de entrenamiento $\mathbf{s}^{(q)}$.

Este error cuadrático E está en función de todos los pesos de la red, por lo tanto $\mathbf{y}(t)$ depende de ellos. Se necesita encontrar un conjunto de reglas de ajuste basadas en el algoritmo de descenso por gradiente:

$$w_{ij}^{(\ell)}(t+1) = w_{ij}^{(\ell)}(t) - \alpha \, \frac{\partial E}{\partial w_{ij}^{(\ell)}(t)}$$

$$b_j^{(\ell)}(t+1) = b_j^{(\ell)}(t) - \alpha \, \frac{\partial E}{\partial b_j^{(\ell)}(t)},$$

donde $\alpha(> 0)$ es el coeficiente de aprendizaje.

En resumen, el algoritmo de propagación hacia atrás para el perceptrón multicapa es:

1. Ajustar α. Inicializar los pesos.
2. Para cada tiempo $t = 1, 2, \ldots$, repetir los pasos a-e hasta que converja.
   a) Ajustar $\mathbf{a}^{(0)} = \mathbf{x}(t)$ tomado aleatoriamente del conjunto de entrenamiento.
   b) Para $\ell = 1, 2, \ldots, L$, calcular
   $$\mathbf{n}^{(\ell)} = \mathbf{a}^{(\ell-1)}\mathbf{W}^{(\ell)} + \mathbf{b}^{(\ell)} \qquad \mathbf{a}^{(\ell)} = f^{(\ell)}(\mathbf{n}^{(\ell)}).$$
   c) Calcular para $n = 1, 2, \ldots, N_L$
   $$s_n^{(L)} = 2\left(\mathbf{a}_n^{(L)} - \mathbf{t}_n(t)\right) \dot{f}^{(L)}(\mathbf{n}_n^{(L)}).$$
   d) Para $\ell = L-1, \ldots, 2, 1$ y $j = 1, 2, \ldots, N_\ell$, calcular
   $$s_j^{(\ell)} = \dot{f}^{(\ell)}(n_j^{(\ell)}) \sum_{i=1}^{N_{\ell+1}} w_{ji}^{(\ell+1)} s_i^{(\ell+1)}.$$

e) Para $\ell = 1, 2, \ldots, L$, actualizar los pesos

$$w_{ij}^{(\ell)}(t+1) = w_{ij}^{(\ell)}(t) - \alpha\, a_i^{(\ell-1)}(t)\, s_j^{(\ell)}(t),$$

$$b_j^{(\ell)}(t+1) = b_j^{(\ell)}(t) - \alpha\, s_j^{(\ell)}(t).$$

Para el cálculo de la actualización de los pesos se necesita encontrar la activación para todas las capas que depende de la función de activación en las neuronas. En el algoritmo de propagación hacia atrás la función de activación comúnmente utilizada es la función sigmoide:

$$f_{\text{logsig}}(x) = \frac{1}{1+e^{-x}}$$

la cual es diferenciable y tiene valores suaves que van entre 0 y 1 para x alrededor 0.

3.7.4. Red neuronal morfológica dendrítica (Dendrite Morphological Neural Networks DMNN)

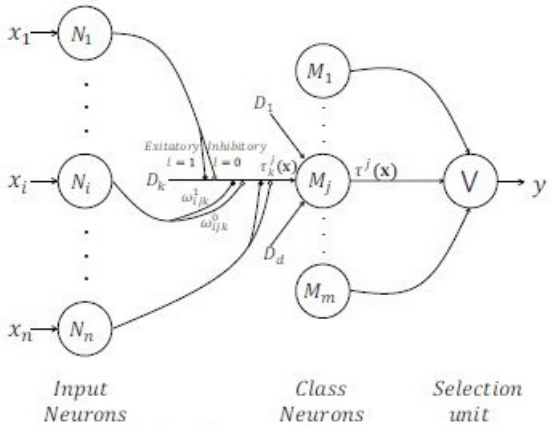
Con el propósito de producir planos de separación cerrados para discriminar los datos en diferentes clases, la Red neuronal morfológica dendrítica, RNMD en adelante, cumple con ese objetivo [32]. La novedad en este modelo es el incorporar estructuras computacionales en las dendritas de las neuronas. El uso de dendritas proporciona ventajas:

- Incrementa el poder computacional de los modelos neuronales.
- Permite la discriminación multiclase.
- Produce planos de separación cerrados entre clases.

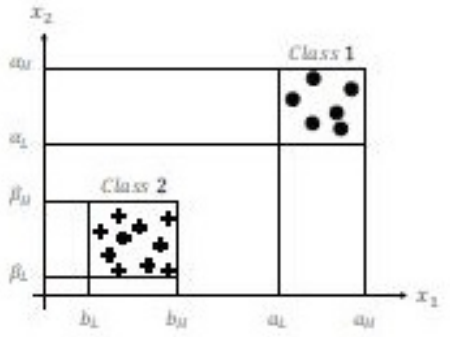
La arquitectura de las RNMD esta ilustrada en la figura 3.12. El modelo consiste de n neuronas de entrada (número de rasgos característicos), m neuronas de clase (número de clases) y una unidad de selección (Salida Final). Todas las neuronas de entrada están conectadas a cada una de las neuronas de clase.

Cada neurona de entrada tiene al menos dos pesos de conexión en cada dendrita dada, uno excitatorio y otro inhibitorio. El valor de estos pesos es desconocido y tiene que ser aprendido por medio de un conjunto de datos de entrenamiento.

La salida de la dendrita D_k en la neurona de clase M_j es $\tau_k^j(x)$ la cual depende del vector de atributos y de pesos. Cada neurona de clase provee un valor de salida $\tau^j(x)$ el cual es calculado mediante el valor de todas las dendritas. Finalmente, la unidad de selección determina la etiqueta de clase y tomando en cuenta todos los valores provistos por las neuronas de clase.



(a) Arquitectura



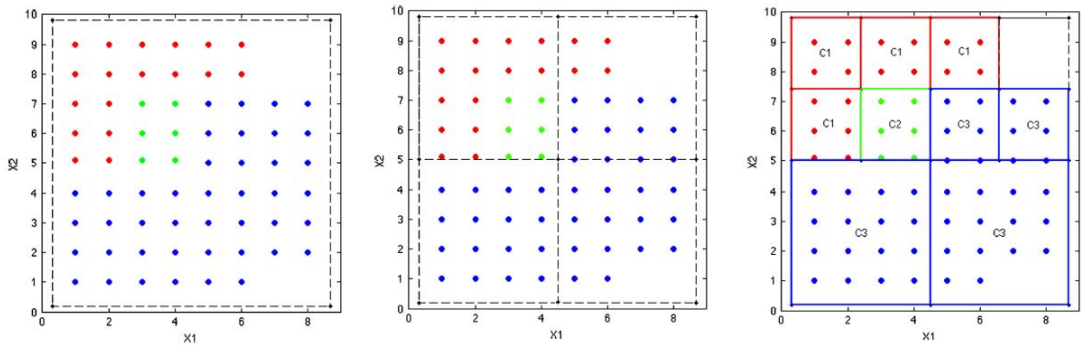
(b) Limites de decisión

Figura 3.12: Funcionamiento de una Red Neuronal Morfológica Dendrítica.

Algoritmo de entrenamiento "divide y venceras"para generación de hipercubos.

El método de entrenamiento usado en este proyecto fue el reportado en [32], el cual está basado en el paradigma de diseño de algoritmos conocido como "divide y venceras".

Algoritmo 3: Método de Entrenamiento para generación de hipercubos (figura 3.13).
Seleccionar los patrones de todas las clases y generar un hipercubo HC^n del tamaño que cubra todos los patrones. Dividir el hipercubo en 2^n hipercubos más pequeños. Verificar si cada hipercubo contiene solo patrones de una sola clase Mientras exista un hipercubo con más de una clase:. Seleccionar un hipercubo H el cual contenga más de una clase. Dividir el hipercubo H en 2^n hipercubos más pequeños. Verificar, si dos hipercubos tienen al menos un lado en común, agruparlos. Basado en las coordenadas de los hipercubos generar los pesos para la RNMD.



(a) Paso 1 (b) Paso 2 (c) Paso 3

Figura 3.13: Método de entrenamiento [32].

3.7.5. Accord.NET

Es un conjunto de librerías disponibles para el lenguaje C# en el que se encuentran algoritmos para clasificación de *Machine Learning*, así como algoritmos para implementar visión por computadora, procesamiento de señales y aplicaciones estadísticas.

De este framework se utilizaron las librerías de *Machine Learning* para implementar el algoritmo de KNN, de la Máquina de Soporte Vectorial y del Perceptrón Multicapa.

3.8. Control de la silla de ruedas

3.8.1. Entorno virtual

En la actualidad, la realidad virtual puede ser un recurso importante en el área de la investigación, ya que permite implementar ambientes para realizar entrenamiento y evaluación de diversos escenarios que pudieran ser peligrosos en la realidad o demasiado costosos.

Este tipo de tecnología es atractiva para una gran variedad de campos de investigación como lo es la robótica, el modelado de procesos, la medicina, incluso en la simulación de aplicaciones militares donde la experimentación puede ser costosa o incluso imposible de realizar [21].

En el presente proyecto se utiliza este tipo de tecnología para el diseño y la implementación de un ambiente virtual en el cual se emula el movimiento de una silla de ruedas mediante el uso de una interfaz cerebro - computadora, esto con la finalidad de presentar al usuario un entorno amigable y llevar a cabo una retroalimentación de las tareas mentales que ejecutará.

3.8.2. Motor de videojuegos Unity

Unity es un motor de videojuegos multiplataforma enfocado al desarrollo de entornos virtuales conformados por todo tipo de objetos 2D/3D y utilizando programas de comportamiento en el lenguaje C#.

Figura 3.14: Unity.

Además de eso tiene implementado el motor de física Box2D y PhysX de NVIDIA con lo cual permite simular modelos físicos. En el ambiente virtual que se implementará en este proyecto no profundizará en aspectos físicos como la fricción, peso, fuerza, velocidad, etc, el único objetivo es el de mostrar una representación gráfica de las tareas que se van ejecutando.

Capítulo 4

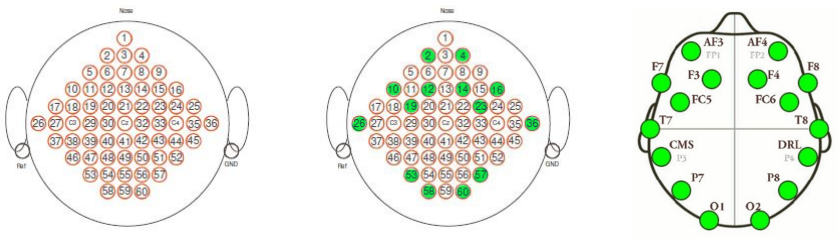
Metodología

4.1. Base de datos

Para el desarrollo de este proyecto solo se utilizaron los datos de dos sujetos de prueba nombrados k3b y k6b. Los datos del sujeto k3b estaban conformadas por 90 pruebas de cada clase, mientras que los del sujeto constaban de 60 pruebas de cada clase.

Para realizar el entrenamiento y el análisis del desempeño de los clasificadores, cada conjunto de datos de cada sujeto de prueba se dividieron a la mitad, 45 pruebas de cada clase conformaron los datos de entrenamiento y 45 pruebas de cada clase conformaron los datos de evaluación, en el caso del sujeto k3b. De igual forma se realizó esta división en el conjunto de datos del sujeto k6b, siendo 30 pruebas de cada clase para conformar el conjunto de entrenamiento y las 30 restantes de cada clase para el conjunto de evaluación.

Como fue explicado en la metodología, la base de datos cuenta con la grabación de 60 canales, con el objetivo de aproximarse a las señales adquiridas por el dispositivo Emotiv Epoc+ se realizó una selección solo de 14 de los 60 canales, eligiendo los más cercanos a los proporcionados por el Emotiv Epoc+ (figura 4.1).



(a) Base de Datos (b) Selección (c) Emotiv

Figura 4.1: Selección de canales en las señales de la BD.

4.1.1. Dos clases

En la primera etapa se realizó una clasificación binaria ya que el método de extracción CSP tiene como limitante el trabajar solo con dos clases. Al tener cuatro clases de tareas motoras se tuvieron seis posibilidades de emparejamiento:

1. Mano Izquierda vs Mano Derecha
2. Mano Izquierda vs Lengua
3. Mano Izquierda vs Pie
4. Mano Derecha vs Lengua
5. Mano Derecha vs Pie
6. Lengua vs Pie

Por lo tanto, se tuvieron 90 pruebas en el conjunto de entrenamiento y 90 pruebas en el conjunto de evaluación para implementar el proceso de clasificación.

La primer parte del procesamiento consiste en el filtrado de las señales, para realizar esto se utilizó el ritmo Mu, que como se mencionó en la metodología proporciona información motora, por lo cual se aplicó un filtro digital de tipo IIR pasa banda con frecuencia de 7-25 hz (figura 4.2).

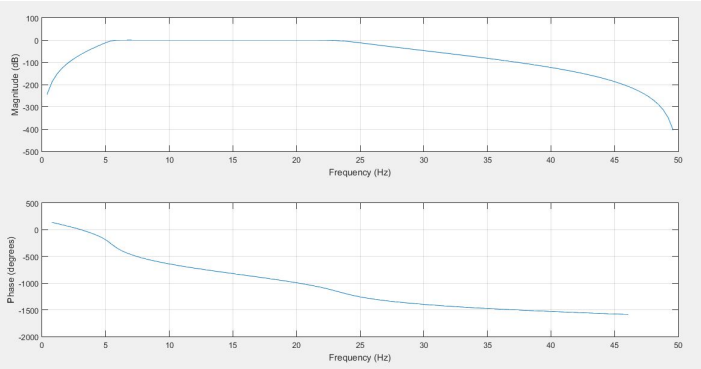
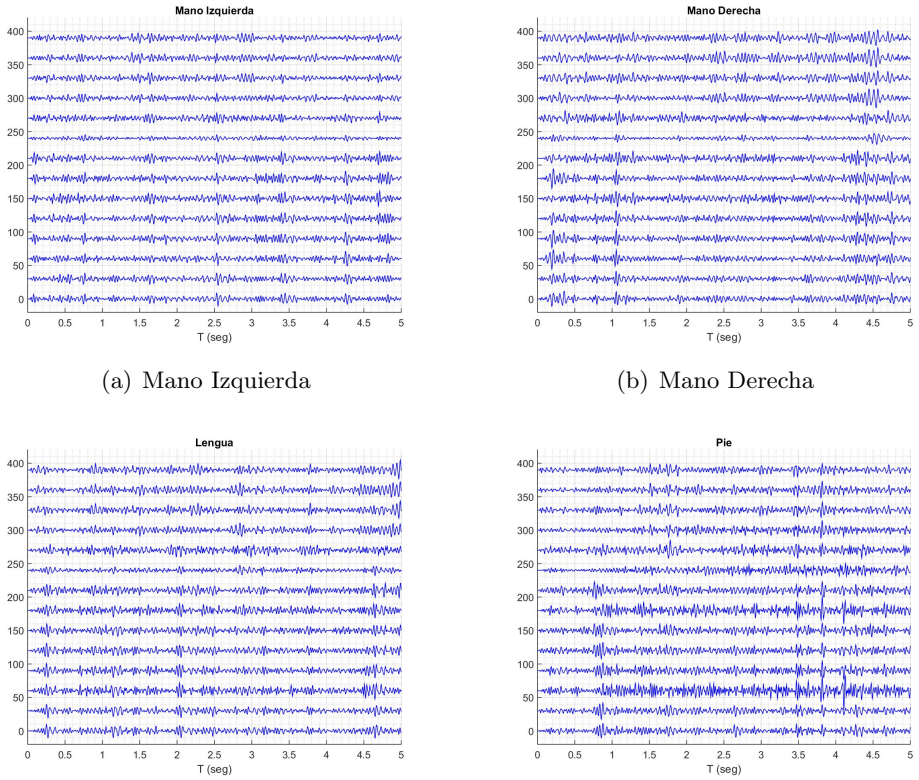


Figura 4.2: Respuesta del Filtro Pasa Banda.

Con esto se limita el espectro de frecuencias de los patrones a analizar, obteniendo señales como las mostradas en la figura 4.3.



(a) Mano Izquierda

(b) Mano Derecha

(c) Lengua

(d) Pie

Figura 4.3: Patrones Cerebrales Filtrados.

El filtro espacial se calculó utilizando las pruebas destinadas al entrenamiento mediante el siguiente algoritmo:

Algoritmo 4: Cálculo de los filtros espaciales en el conjunto de datos.

Para todos los patrones de una misma clase:
    Calcular la matriz de covarianza espacial de cada patrón.
    Calcular el promedio de las matrices de covarianza de los patrones de una misma clase.
Realizar el cálculo del filtro CSP para cada combinación de par de clases.

Para generar el conjunto de datos con el que se entrenaron los clasificadores se les aplicó el filtro al conjunto de entrenamiento y se realizó el cálculo de la media cuadrática y de la varianza, como indica la Figura 4.4.

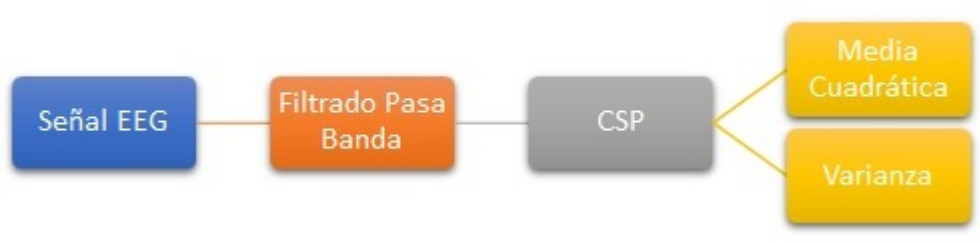
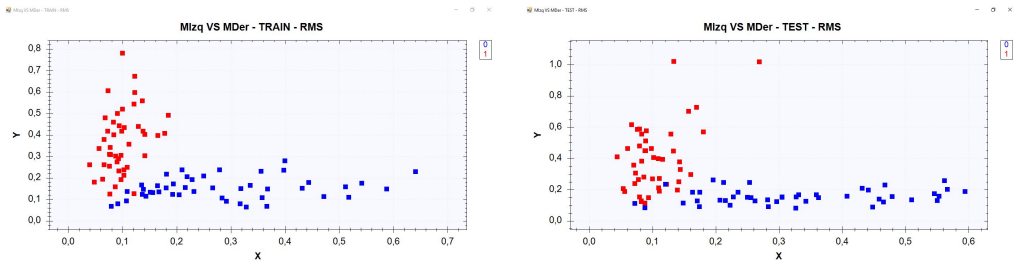


Figura 4.4: Metodología a seguir con cada patrón.

Al realizar el proceso de extracción de rasgos característicos se obtuvo un conjunto de patrones correspondiente al número de canales con los que contaban los patrones a analizar, en este caso fueron 14 rasgos característicos.

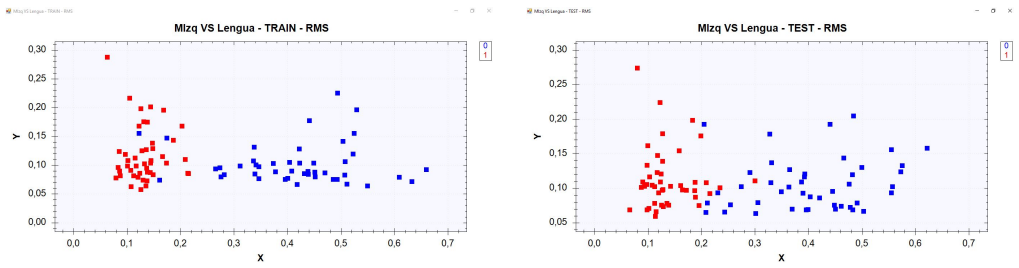
Media cuadrática (RMS)

Este proceso se aplicó a ambos sujetos de prueba obteniendo conjunto de patrones como el que se muestra en las siguientes figuras siendo graficado solo dos rasgos característicos.



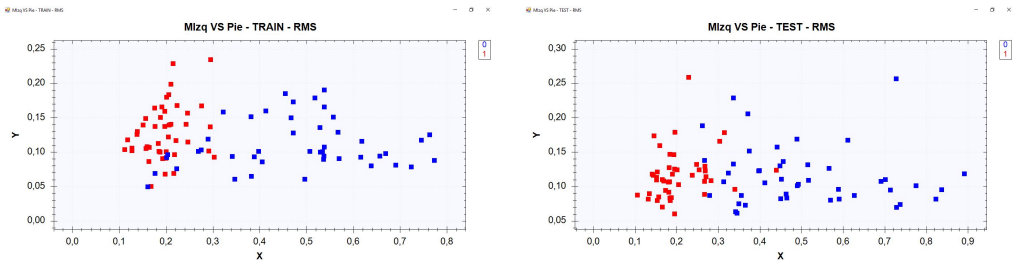
(a) MIzq.(Azul) vs MDer.(Rojo) (TRAIN) (b) MIzq.(Azul) vs MDer.(Rojo) (TEST)

Figura 4.5: Conjunto de datos M.Izq vs M.Der de k3b (RMS).



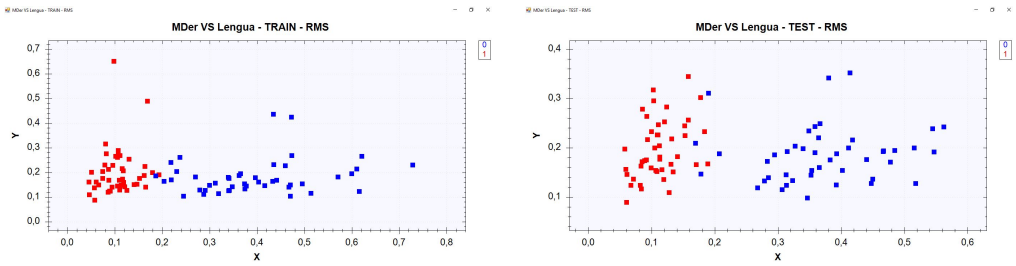
(a) MIzq.(Azul) vs Lengua(Rojo) (TRAIN) (b) MIzq.(Azul) vs Lengua(Rojo) (TEST)

Figura 4.6: Conjunto de datos M.Izq vs Lengua de k3b (RMS).



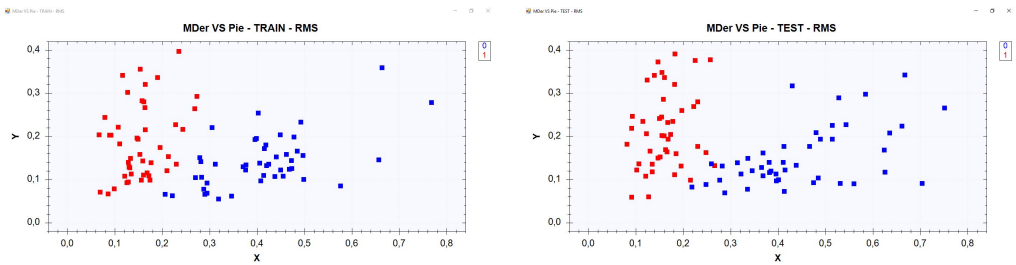
(a) MIzq.(Azul) vs Pie(Rojo) (TRAIN) (b) MIzq.(Azul) vs Pie(Rojo) (TEST)

Figura 4.7: Conjunto de datos M.Izq vs Pie de k3b (RMS).



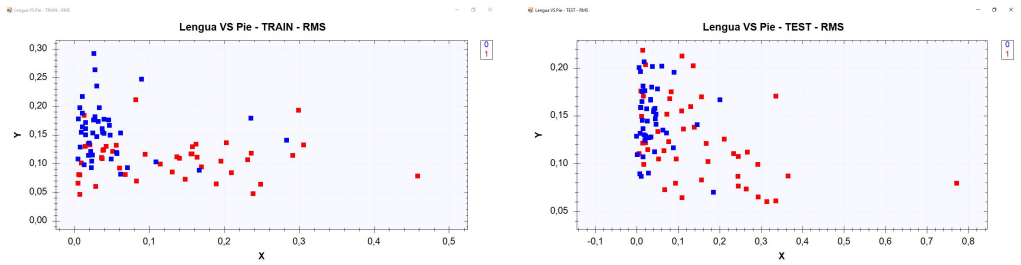
(a) MDer.(Azul) vs Lengua(Rojo) (TRAIN) (b) MDer.(Azul) vs Lengua(Rojo) (TEST)

Figura 4.8: Conjunto de datos M.Der vs Lengua de k3b (RMS).



(a) MDer.(Azul) vs Pie(Rojo) (TRAIN) (b) MDer.(Azul) vs Pie(Rojo) (TEST)

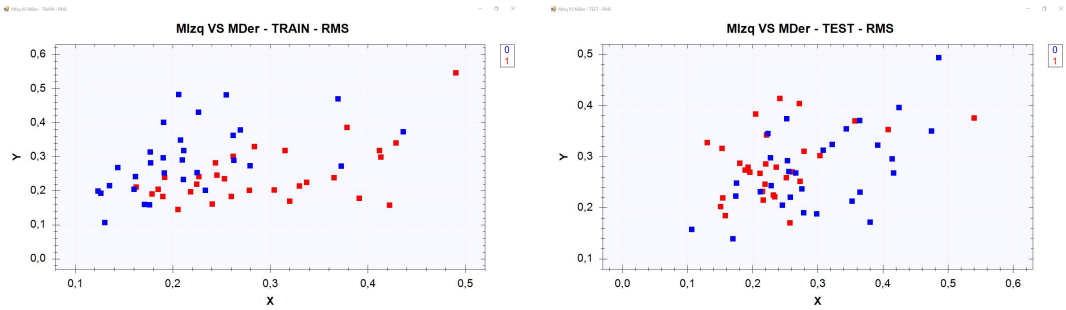
Figura 4.9: Conjunto de datos M.Der vs Pie de k3b (RMS).



(a) Lengua(Azul) vs Pie(Rojo) (TRAIN) (b) Lengua(Azul) vs Pie(Rojo) (TEST)

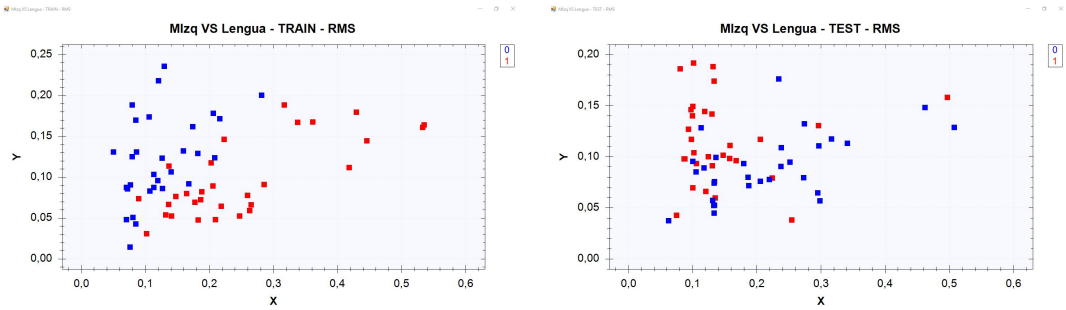
Figura 4.10: Conjunto de datos Lengua vs Pie de k3b (RMS).

Los datos obtenidos del sujeto k6b se muestran a continuación:



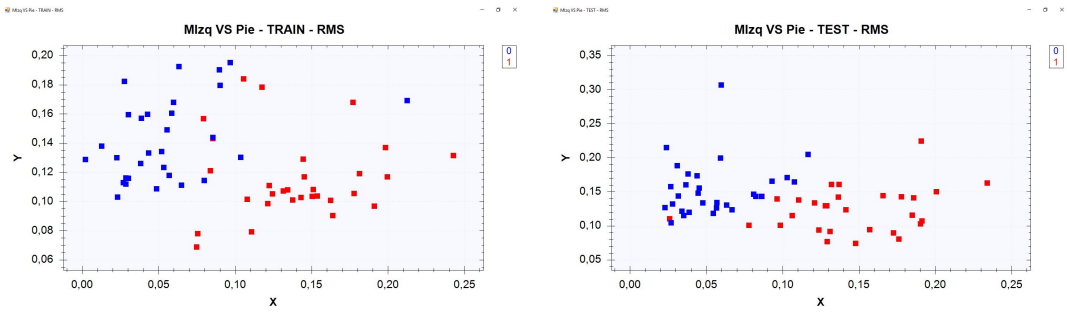
(a) MIzq.(Azul) vs MDer.(Rojo) (TRAIN) (b) MIzq.(Azul) vs MDer.(Rojo) (TEST)

Figura 4.11: Conjunto de datos M.Izq vs M.Der de k6b (RMS).



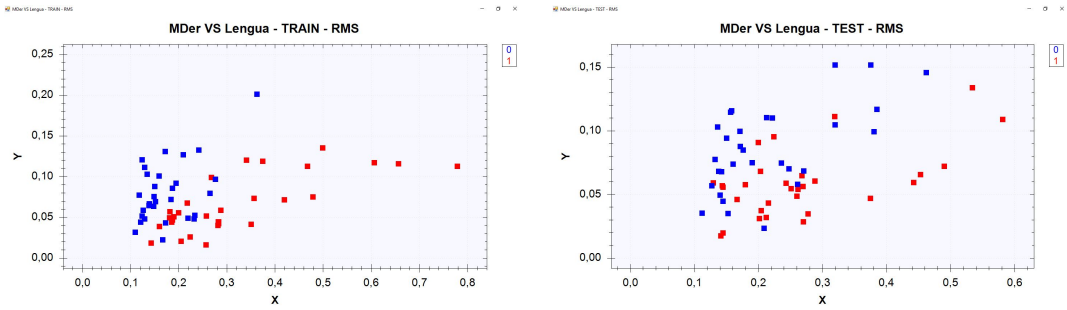
(a) MIzq.(Azul) vs Lengua(Rojo) (TRAIN) (b) MIzq.(Azul) vs Lengua(Rojo) (TEST)

Figura 4.12: Conjunto de datos M.Izq vs Lengua de k6b (RMS).



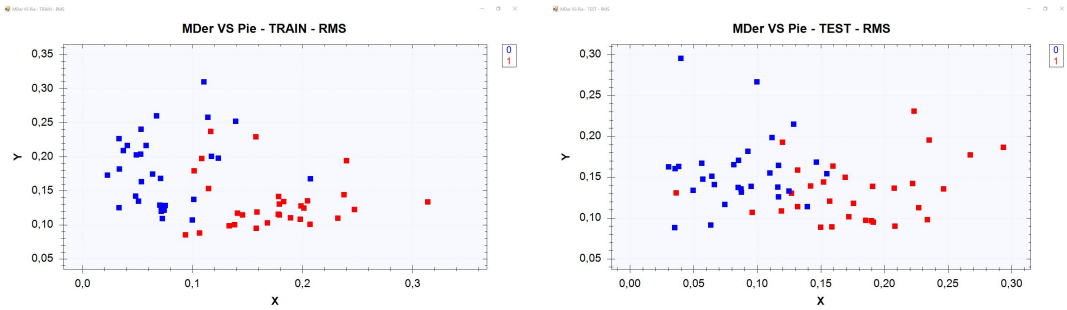
(a) MIzq.(Azul) vs Pie(Rojo) (TRAIN) (b) MIzq.(Azul) vs Pie(Rojo) (TEST)

Figura 4.13: Conjunto de datos M.Izq vs Pie de k6b (RMS).



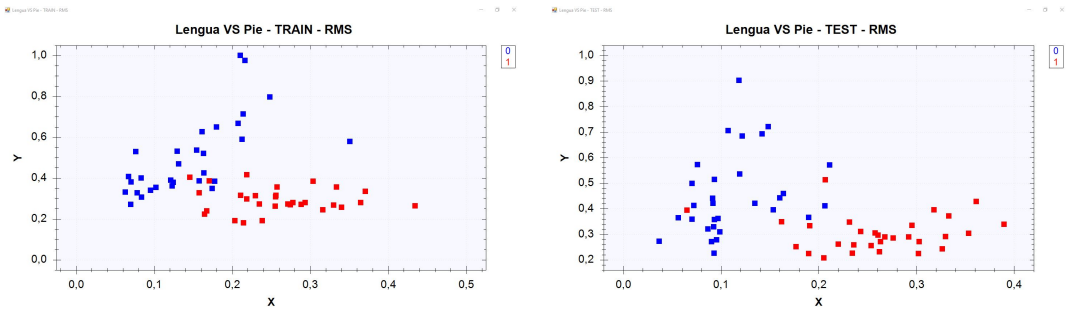
(a) MDer.(Azul) vs Lengua(Rojo) (TRAIN) (b) MDer.(Azul) vs Lengua(Rojo) (TEST)

Figura 4.14: Conjunto de datos M.Der vs Lengua de k6b (RMS).



(a) MDer.(Azul) vs Pie(Rojo) (TRAIN) (b) MDer.(Azul) vs Pie(Rojo) (TEST)

Figura 4.15: Conjunto de datos M.Der vs Pie de k6b (RMS).



(a) Lengua(Azul) vs Pie(Rojo) (TRAIN) (b) Lengua(Azul) vs Pie(Rojo) (TEST)

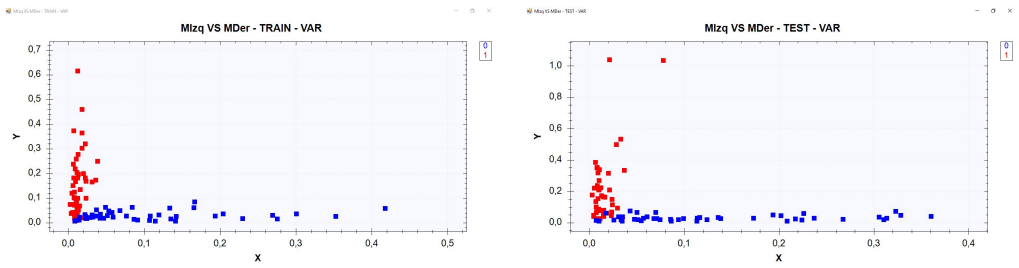
Figura 4.16: Conjunto de datos Lengua vs Pie de k6b (RMS).

Para llevar a cabo la clasificación en cada uno de los conjuntos de datos, se decidió establecer los hiperpárametros referentes al diseño según el tipo de clasificador:

- K Vecinos Cercanos: $k = 5$ y el tipo de distancia utilizada fue euclidiana.
- Perceptrón Multicapa: diseño con una sola capa oculta, la cual consiste de diez neuronas y la condiciones de paro para el algoritmo de propagación hacia atrás fueron el error cuadrático $< 0{,}001$ o el número de épocas máximas establecido en 100,000.
- Máquina de Soporte Vectorial: para establecer el kernel se utilizó un método implementado en la librería Accord, el cual elegía el kernel según el tipo de datos del conjunto de entrenamiento.
- Red Neuronal Morfológica Dendrítica: aquí solo se estableció el número máximo de épocas para evitar sobre entrenamiento, siendo éstas 1000.

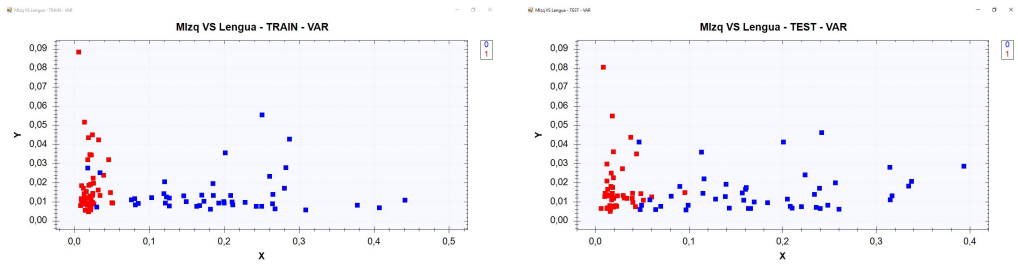
Varianza (VAR)

Como se mencionó en la metodología se utilizó una segunda medida estadística, la varianza. Se repitió el proceso solo que ahora utilizando esta medida estadística obteniendo los siguientes conjuntos de patrones:



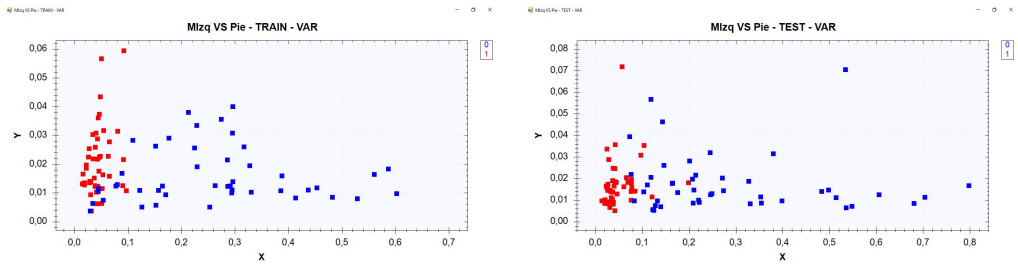
(a) MIzq.(Azul) vs MDer.(Rojo) (TRAIN) (b) MIzq.(Azul) vs MDer.(Rojo) (TEST)

Figura 4.17: Conjunto de datos M.Izq vs M.Der de k3b (VAR).



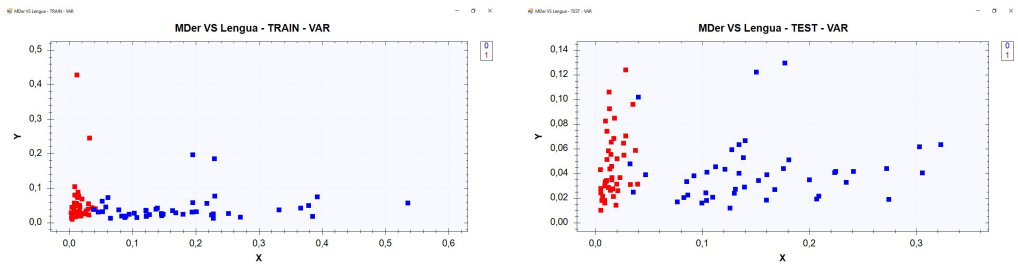
(a) MIzq.(Azul) vs Lengua(Rojo) (TRAIN) (b) MIzq.(Azul) vs Lengua(Rojo) (TEST)

Figura 4.18: Conjunto de datos M.Izq vs Lengua de k3b (VAR).



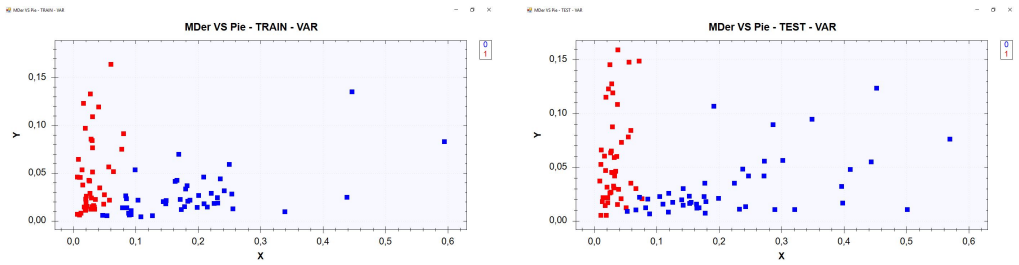
(a) MIzq.(Azul) vs Pie(Rojo) (TRAIN) (b) MIzq.(Azul) vs Pie(Rojo) (TEST)

Figura 4.19: Conjunto de datos M.Izq vs Pie de k3b (VAR).



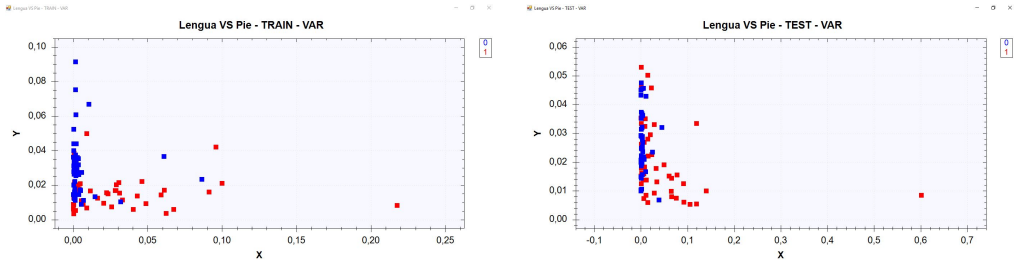
(a) MDer.(Azul) vs Lengua(Rojo) (TRAIN) (b) MDer.(Azul) vs Lengua(Rojo) (TEST)

Figura 4.20: Conjunto de datos M.Der vs Lengua de k3b (VAR).



(a) MDer.(Azul) vs Pie(Rojo) (TRAIN) (b) MDer.(Azul) vs Pie(Rojo) (TEST)

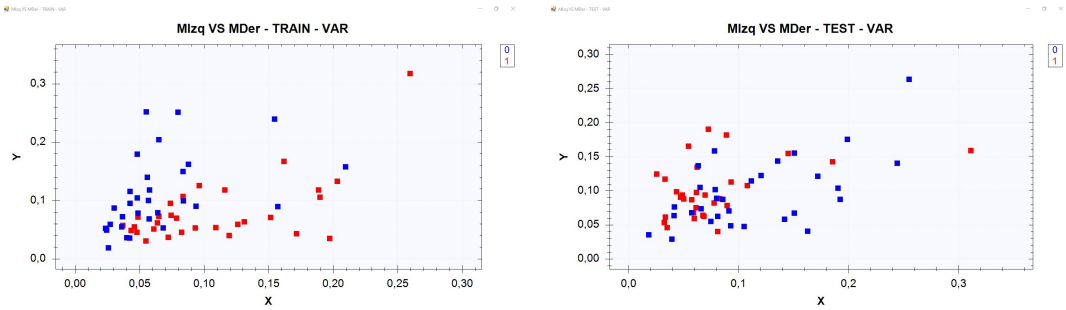
Figura 4.21: Conjunto de datos M.Der vs Pie de k3b (VAR).



(a) Lengua(Azul) vs Pie(Rojo) (TRAIN) (b) Lengua(Azul) vs Pie(Rojo) (TEST)

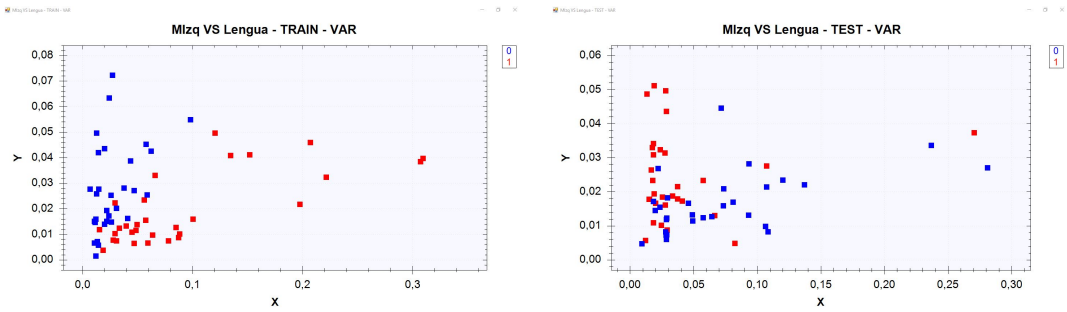
Figura 4.22: Conjunto de datos Lengua vs Pie de k3b (VAR).

Los datos obtenidos del sujeto k6b se muestran a continuación:



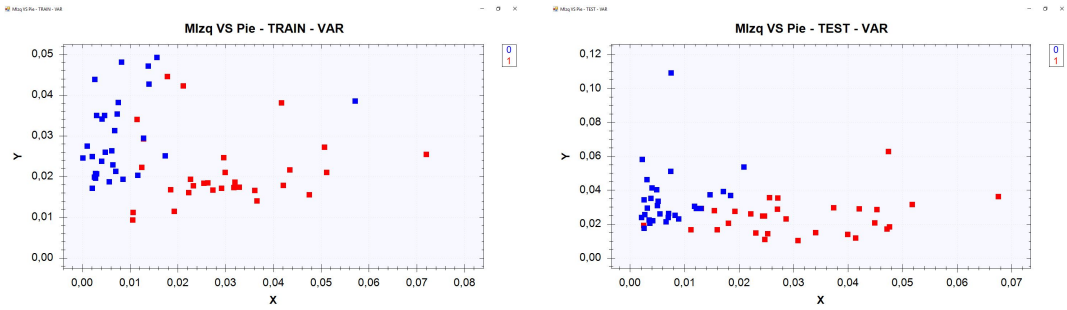
(a) MIzq.(Azul) vs MDer.(Rojo) (TRAIN) (b) MIzq.(Azul) vs MDer.(Rojo) (TEST)

Figura 4.23: Conjunto de datos M.Izq vs M.Der de k6b (VAR).



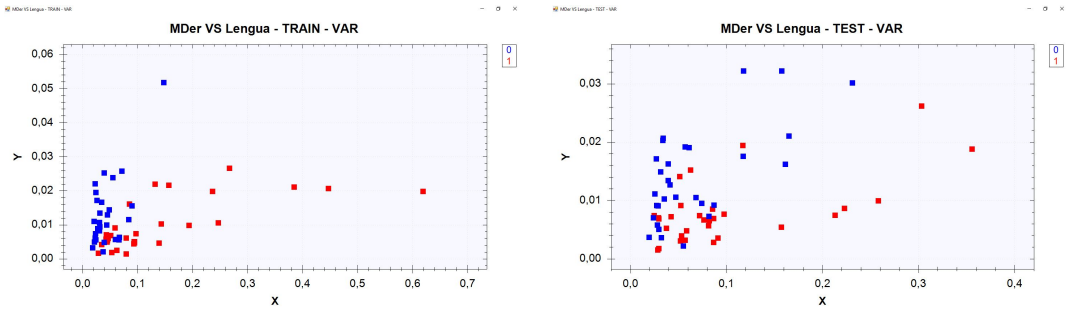
(a) MIzq.(Azul) vs Lengua(Rojo) (TRAIN) (b) MIzq.(Azul) vs Lengua(Rojo) (TEST)

Figura 4.24: Conjunto de datos M.Izq vs Lengua de k6b (VAR).



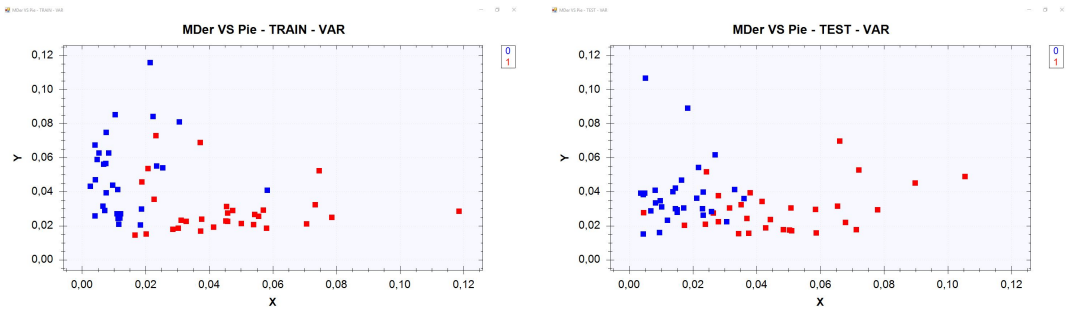
(a) MIzq.(Azul) vs Pie(Rojo) (TRAIN) (b) MIzq.(Azul) vs Pie(Rojo) (TEST)

Figura 4.25: Conjunto de datos M.Izq vs Pie de k6b (VAR).



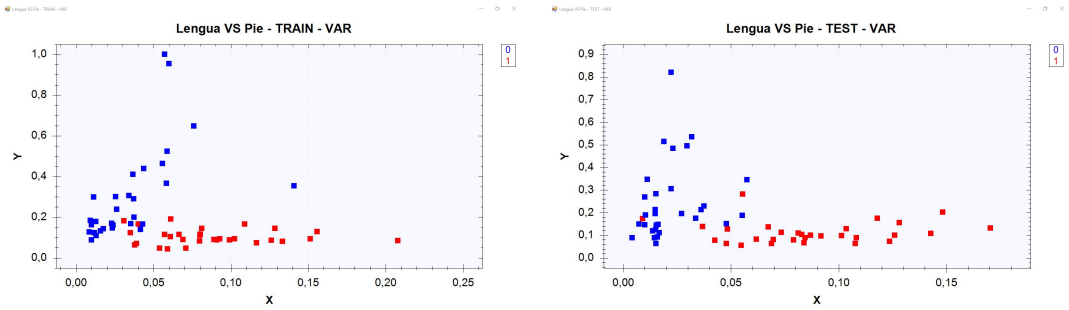
(a) MDer.(Azul) vs Lengua(Rojo) (TRAIN) (b) MDer.(Azul) vs Lengua(Rojo) (TEST)

Figura 4.26: Conjunto de datos M.Der vs Lengua de k6b (VAR).



(a) MDer.(Azul) vs Pie(Rojo) (TRAIN) (b) MDer.(Azul) vs Pie(Rojo) (TEST)

Figura 4.27: Conjunto de datos M.Der vs Pie de k6b (VAR).



(a) Lengua(Azul) vs Pie(Rojo) (TRAIN) (b) Lengua(Azul) vs Pie(Rojo) (TEST)

Figura 4.28: Conjunto de datos Lengua vs Pie de k6b (VAR).

Los clasificadores fueron los mismos antes mencionados variando el número de rasgos característicos.

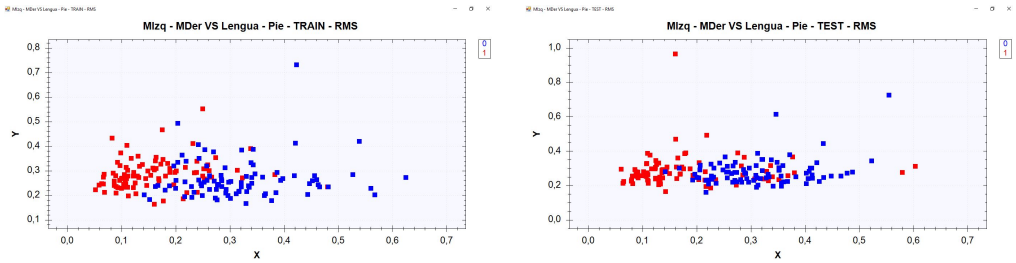
4.1.2. Cuatro clases (Mano Izquierda, Mano Derecha, Lengua, Pie)

Para cumplir el objetivo de clasificar 4 clases de acciones motoras se necesitó abordar el problema de diferente forma, el método de extracción por si solo tiene como limitante el trabajar solo con dos clases, por lo tanto se empleó la técnica *Pair-Wise (PW)* para establecer primero dos grandes clases, las cuales estaban constituidas, cada una de ellas, por dos clases, generando tres posibles combinaciones:

- Mano Izquierda - Mano Derecha VS Lengua - Pie
- Mano Izquierda - Lengua VS Mano Derecha - Pie
- Mano Izquierda - Pie VS Mano Derecha - Lengua

Media cuadrática (RMS)

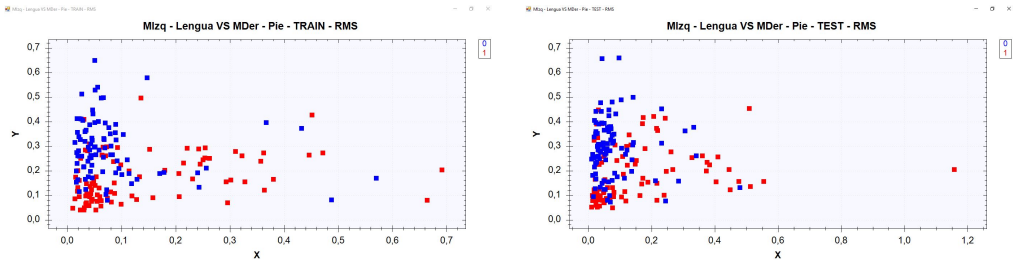
Los conjuntos de datos de entrenamiento con los que se trabajo se muestra en las figuras a continuación:



(a) MIzq.-MDer.(Azul) vs Lengua-Pie(Rojo) (TRAIN)

(b) MIzq.-MDer.(Azul) vs Lengua-Pie(Rojo) (TEST)

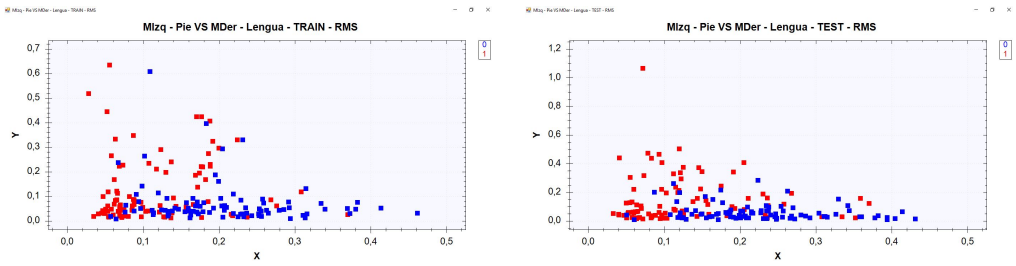
Figura 4.29: Conjunto de datos MIzq.-MDer vs Lengua-Pie (2vs2) de k3b (RMS).



(a) MIzq.-Lengua(Azul) vs MDer.-Pie(Rojo) (TRAIN)

(b) MIzq.-Lengua(Azul) vs MDer.-Pie(Rojo) (TEST)

Figura 4.30: Conjunto de datos MIzq.-Lengua vs MDer.-Pie (2vs2) de k3b (RMS).



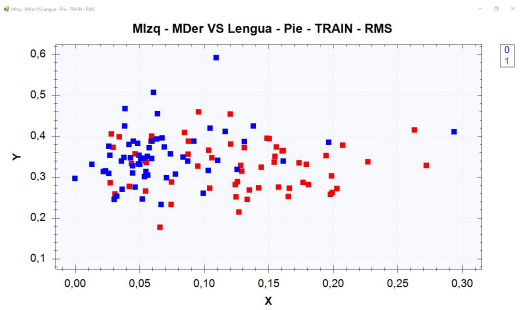
(a) MIzq.-Pie(Azul) vs MDer.-Lengua(Rojo) (TRAIN)

(b) MIzq.-Pie(Azul) vs MDer.-Lengua(Rojo) (TEST)

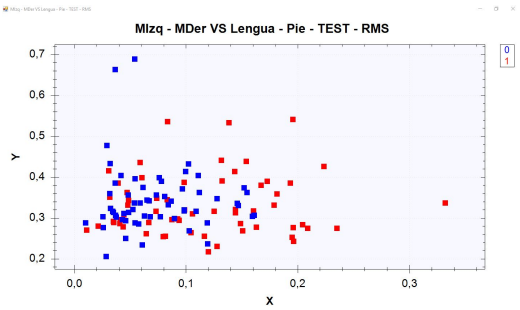
Figura 4.31: Conjunto de datos MIzq.-Pie vs MDer.-Lengua (2vs2) de k3b (RMS).

De igual forma se realizó el proceso con el sujeto de prueba k6b, a continuación se muestran

los conjuntos de datos de entrenamiento y prueba:

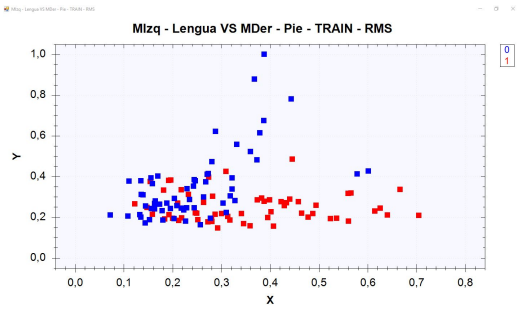


(a) MIzq.-MDer.(Azul) vs Lengua-Pie(Rojo) (TRAIN)

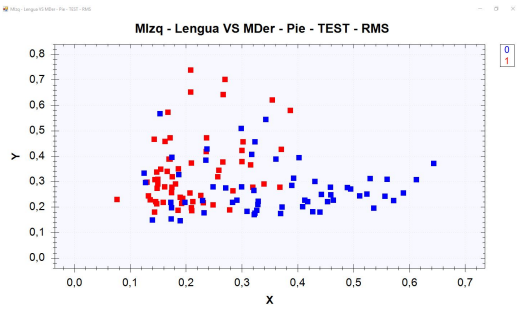


(b) MIzq.-MDer.(Azul) vs Lengua-Pie(Rojo) (TEST)

Figura 4.32: Conjunto de datos MIzq.-MDer vs Lengua-Pie (2vs2) de k6b (RMS).

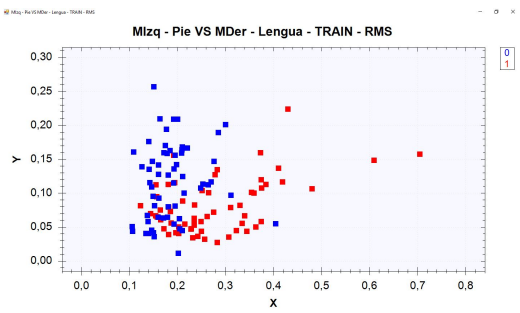


(a) MIzq.-Lengua(Azul) vs MDer.-Pie(Rojo) (TRAIN)

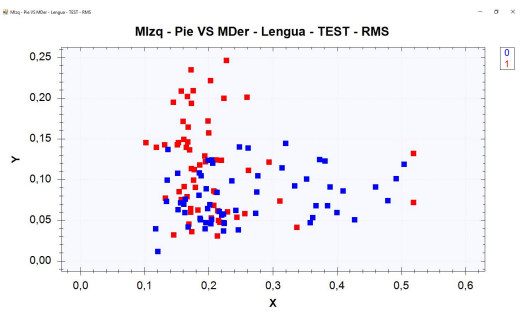


(b) MIzq.-Lengua(Azul) vs MDer.-Pie(Rojo) (TEST)

Figura 4.33: Conjunto de datos MIzq.-Lengua vs MDer.-Pie (2vs2) de k6b (RMS).



(a) MIzq.-Pie(Azul) vs MDer.-Lengua(Rojo) (TRAIN)

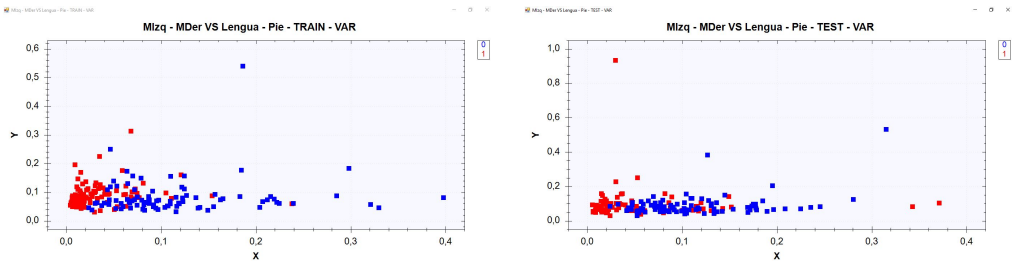


(b) MIzq.-Pie(Azul) vs MDer.-Lengua(Rojo) (TEST)

Figura 4.34: Conjunto de datos MIzq.-Pie vs MDer.-Lengua (2vs2) de k6b (RMS).

Varianza (VAR)

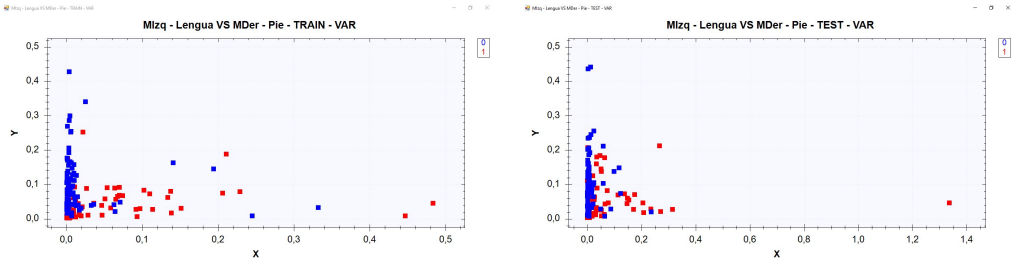
Al igual que en la sección anterior se utilizó una segunda medida estadística, la varianza. Se repitió el proceso obteniendo los siguientes conjuntos de patrones:



(a) MIzq.-MDer.(Azul) vs Lengua-Pie(Rojo) (TRAIN)

(b) MIzq.-MDer.(Azul) vs Lengua-Pie(Rojo) (TEST)

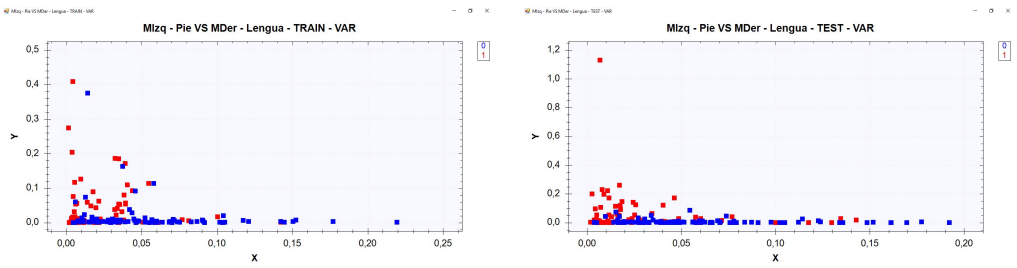
Figura 4.35: Conjunto de datos MIzq.-MDer vs Lengua-Pie (2vs2) de k3b (VAR).



(a) MIzq.-Lengua(Azul) vs MDer.-Pie(Rojo) (TRAIN)

(b) MIzq.-Lengua(Azul) vs MDer.-Pie(Rojo) (TEST)

Figura 4.36: Conjunto de datos MIzq.-Lengua vs MDer.-Pie (2vs2) de k3b (VAR).



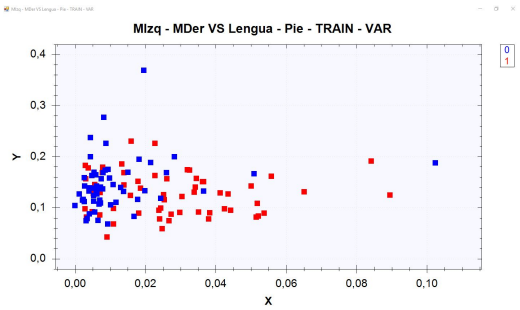
(a) MIzq.-Pie(Azul) vs MDer.-Lengua(Rojo) (TRAIN)

(b) MIzq.-Pie(Azul) vs MDer.-Lengua(Rojo) (TEST)

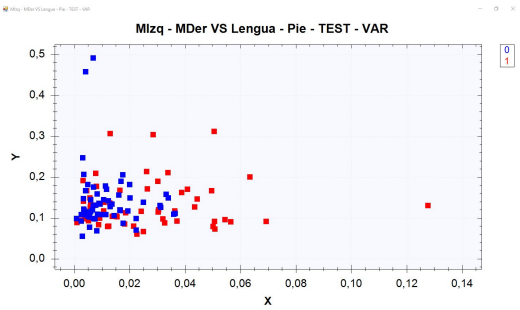
Figura 4.37: Conjunto de datos MIzq.-Pie vs MDer.-Lengua (2vs2) de k3b (VAR).

De igual forma se realizó el proceso con el sujeto de prueba k6b, a continuación se muestran

los conjuntos de datos de entrenamiento y prueba:

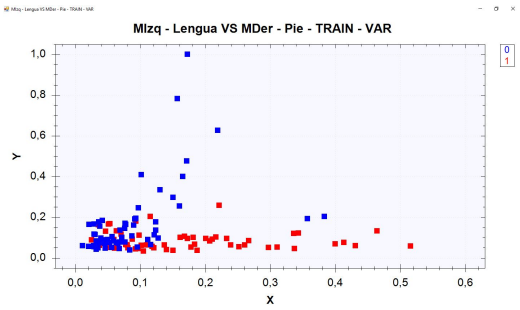


(a) MIzq.-MDer.(Azul) vs Lengua-Pie(Rojo) (TRAIN)

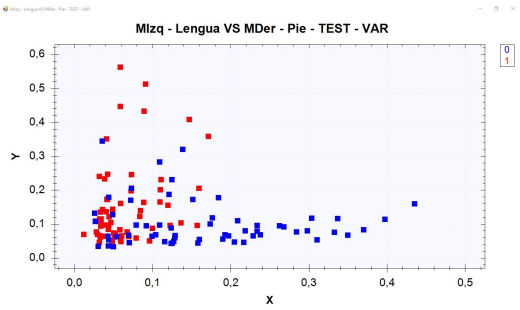


(b) MIzq.-MDer.(Azul) vs Lengua-Pie(Rojo) (TEST)

Figura 4.38: Conjunto de datos MIzq.-MDer vs Lengua-Pie (2vs2) de k6b (VAR).

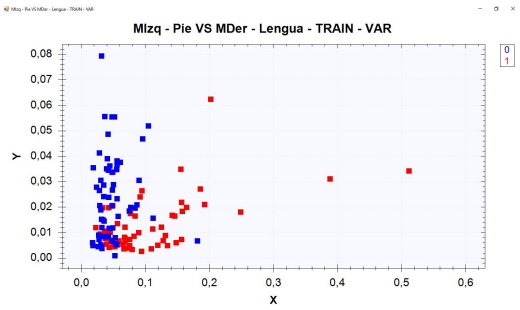


(a) MIzq.-Lengua(Azul) vs MDer.-Pie(Rojo) (TRAIN)

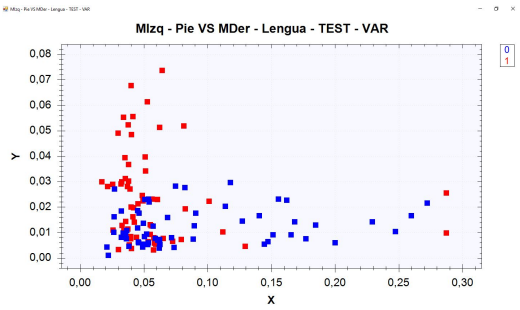


(b) MIzq.-Lengua(Azul) vs MDer.-Pie(Rojo) (TEST)

Figura 4.39: Conjunto de datos MIzq.-Lengua vs MDer.-Pie (2vs2) de k6b (VAR).



(a) MIzq.-Pie(Azul) vs MDer.-Lengua(Rojo) (TRAIN)



(b) MIzq.-Pie(Azul) vs MDer.-Lengua(Rojo) (TEST)

Figura 4.40: Conjunto de datos MIzq.-Pie vs MDer.-Lengua (2vs2) de k6b (VAR).

4.1.3. Red de clasificadores

Como se mencionó en la sección anterior, la clasificación de cuatro diferentes patrones se realizó de manera binaria diseñando así una red de clasificadores. Esta red de clasificadores primero clasifica entre dos grandes clases,a su vez cada una de ellas incluye dos subclases, dependiendo de la predicción en esta etapa se llevara a cabo una segunda clasificación para determinar la clase específica a la que pertenece el patrón (figura 4.41).

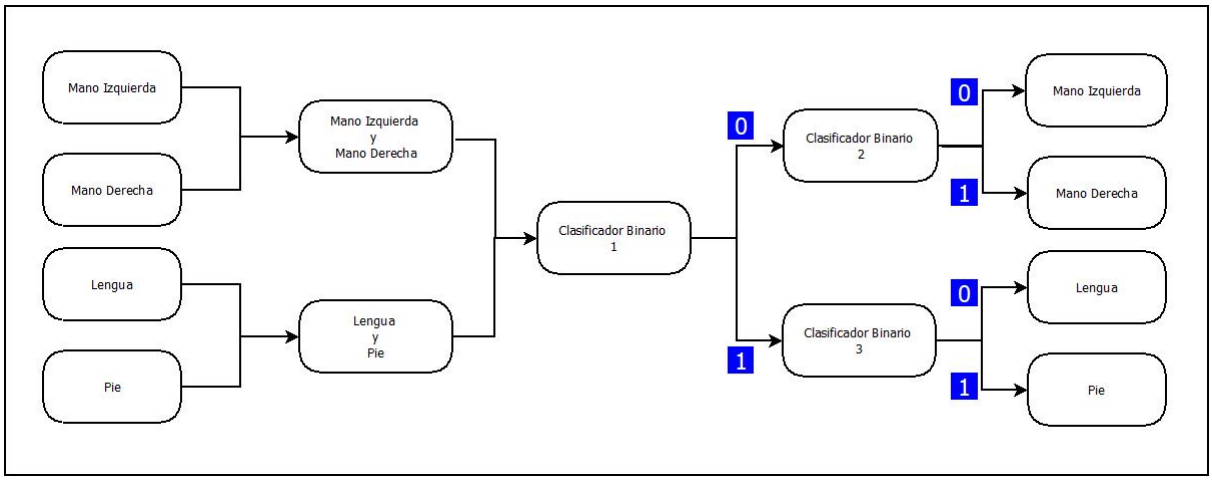


Figura 4.41: Funcionamiento Red de Clasificadores

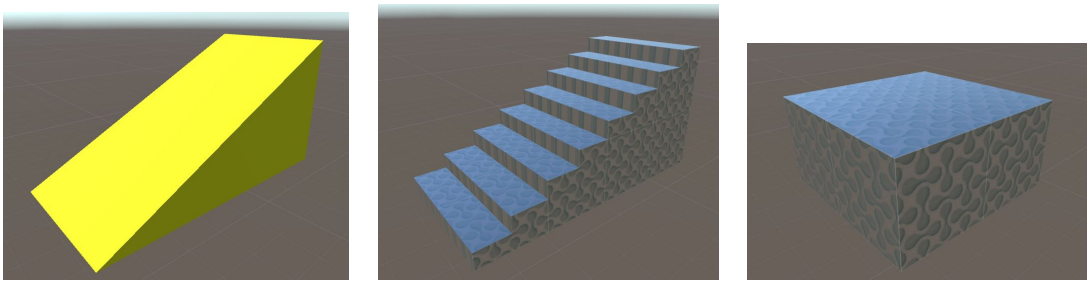
Para el diseño de esta red de clasificadores se tomaron los mejores clasificadores, la selección se realizó en cada una de las combinaciones posibles, los criterios para seleccionar fueron:

- Mismo método estadístico
- Menor error
- Menor número de rasgos característicos.

4.1.4. Entorno virtual

Después de tener establecidos los clasificadores, estos se implementaron en el entorno virtual para controlar el manejo de la silla de ruedas.

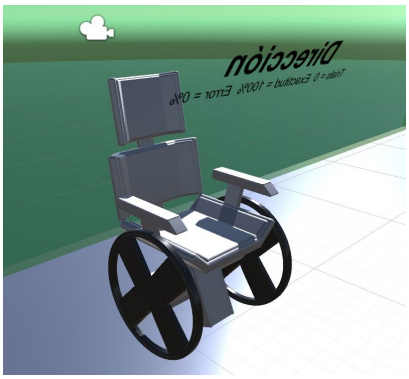
Primero se creó el entorno virtual en Unity utilizando componentes estándar que proporciona el software:



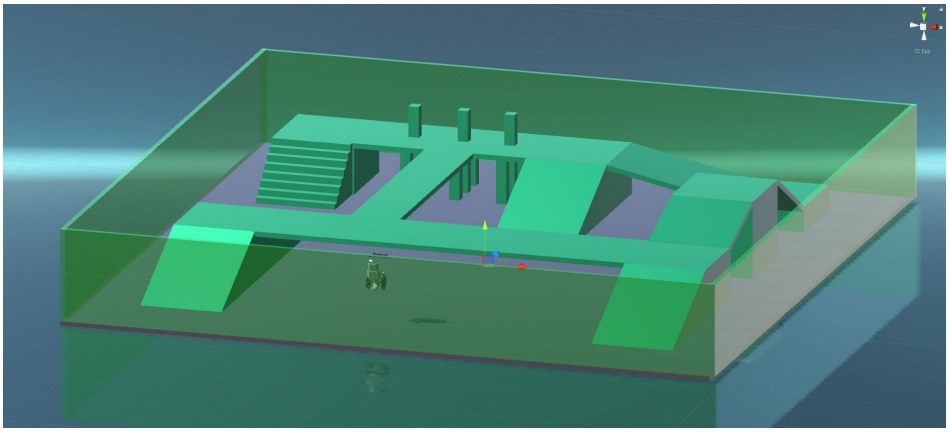
(a) Rampa (b) Escalera (c) Plataforma

Figura 4.42: Componentes estándar de Unity.

Estos componentes se pueden escalar permitiendo crear estructuras más complejas, éstas fueron agregadas al entorno virtual para hacerlo más amigable al usuario y permitir la interacción más fluida. De igual forma se utilizó un modelo de silla de ruedas que actuará como personaje principal en este escenario virtual (4.43).



(a) Modelo 3D de Silla de Ruedas



(b) Escenario

Figura 4.43: Entorno Virtual.

Este personaje tiene implementado un controlador que actúa de la siguiente forma:

Algoritmo 5: Controlador Silla de Ruedas

Datos: Datos de Prueba 4 Clases (180)
Para cada fotograma:
Si se presiona alguna de las teclas W-A-S-D.
Según la tecla que se presiono se selecciona un patrón según corresponda.
W - Lengua (random[1-45]).
A - Mano Izquierda (random[1-45]).
S - Pie (random[1-45]).
D - Mano Derecha (random[1-45]).
El patrón elegido ingresa al clasificador.
Dependiendo de la predicción la silla de ruedas actuará:
M. Izquierda: Giro a la Izquierda en 45 grados.
M. Derecha: Giro a la Derecha en 45 grados.
Lengua: Mover hacia adelante 50 cm.
Pie: Mover hacia atrás 50 cm.

La interfaz que se muestra al usuario es la mostrada en la figura 4.44.

Figura 4.44: Interfaz de Usuario.

Como se observa en la pantalla se mostrará el movimiento que el usuario quiso y la predicción del clasificador, así como ir contando los movimientos que realizan y el cálculo de exactitud y error.

Este entorno virtual tiene como objetivo el ofrecer una interfaz amigable al sujeto de prueba y una retroalimentación en las tareas mentales, ya que al observar que un objeto se mueve cuando él realiza la tarea mental existe en el sujeto un proceso de neuroplasticidad que mejorará el desempeño del proceso mental.

De igual forma este entorno tiene sus limitaciones, una de ellas es que no se realizó un control mecánico propio en la silla de ruedas virtual (se entiende esto como manejo de motores, instrumentación de la silla, seguridad, etc).

4.2. Emotiv EPOC+

En este proyecto se planteó el uso del dispositivo Emotiv EPOC+ para adquirir señales cerebrales, para esto se realizó un programa que llevara a cabo el procedimiento descrito en el capitulo anterior.

El programa fue desarrollado en el lenguaje de programación C#. El dispositivo permite la grabación de señales cerebrales por 30 min según la licencia que se adquirió, por lo mismo el programa realiza la grabación de 180 pruebas de tareas mentales consecutivas. Éste cuenta con un cronometro el cual va controlando los estímulos visuales que se presentan al sujeto de prueba (fig 4.45).

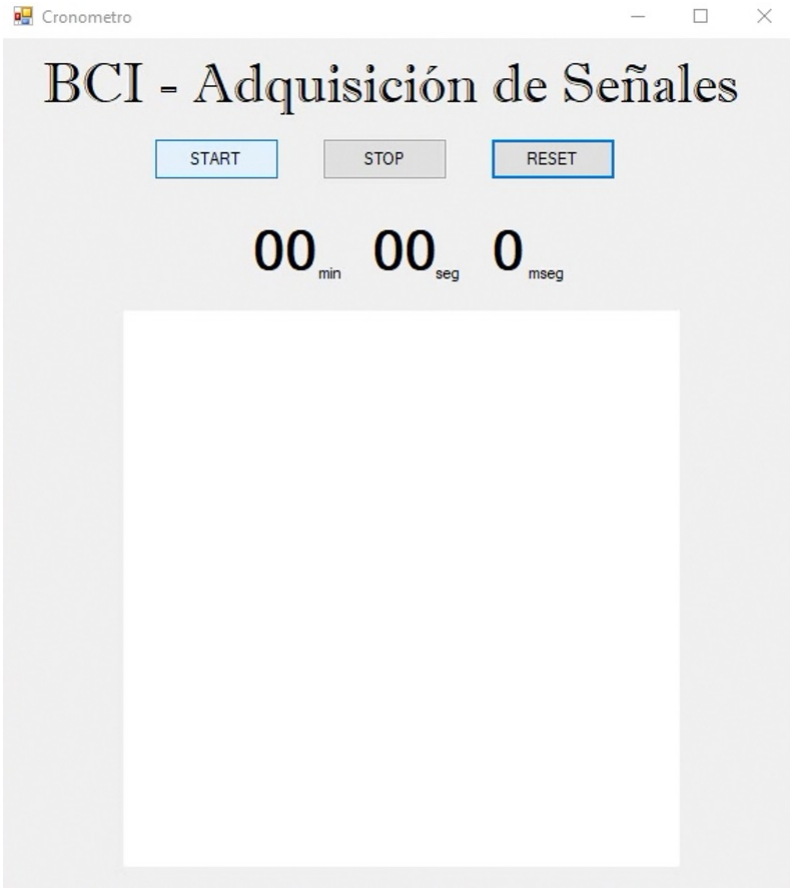
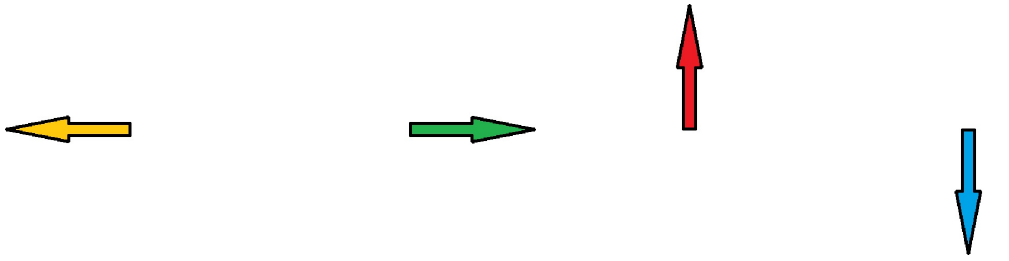


Figura 4.45: Software para adquisición de señales del Emotiv EPOC+.

Los patrones mostrados corresponden a cada una de las tareas mentales:



(a) Mano Izquierda (b) Mano Derecha (c) Lengua (d) Pie

Figura 4.46: Estímulos visuales presentados al sujeto de prueba.

Se tuvo el programa con todos los requerimientos para adquirir las señales cerebrales pero se presentó un problema en la infraestructura del dispositivo, ya que éste no realizaba las adquisiciones de manera adecuada. El error que se presentó fue la falta de información, ya que como se explicó este proceso debe respetar los tiempos de grabación ya que utiliza un protocolo de tipo sincróno. Con falta de información se refiere a que en el intervalo de grabación que consta de 10 segundos, solo se grababan seis o siete, el mayor problema es que los segundos ausentes no siempre fueron los mismos.

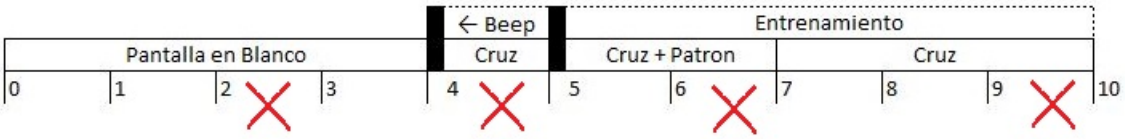


Figura 4.47: Error en adquisición de señales cerebrales.

Capítulo 5

Presentacion y discusión de resultados

Como se mencionó en el capítulo anterior, se realizó la clasificación de todos los conjuntos de datos en cada uno de los cuatro clasificadores implementados. De igual forma se realizó variando el número de rasgos característicos, desde dos hasta tomar los 14 rasgos.

5.1. Clasificación binaria: dos clases

Primero se realizó la clasificación con todos los pares de clases, el primer sujeto de análisis fue k3b obteniendo los resultados plasmados a continuación:

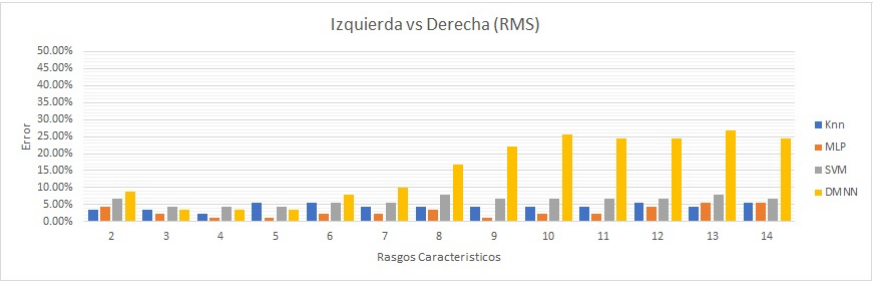


Figura 5.1: Desempeño de los clasificadores para MIzq. vs MDer. del sujeto k3b (RMS).

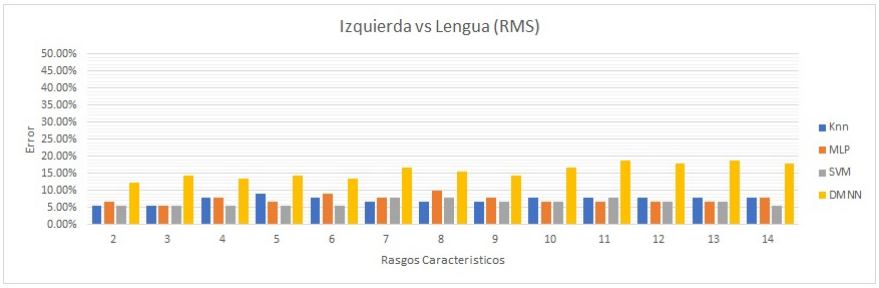


Figura 5.2: Desempeño de los clasificadores para MIzq. vs Lengua del sujeto k3b (RMS).

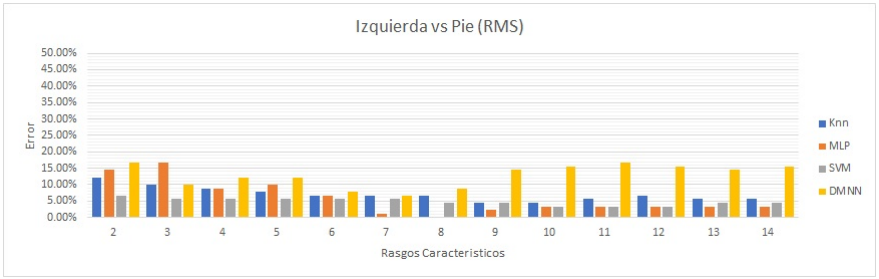


Figura 5.3: Desempeño de los clasificadores para MIzq. vs Pie del sujeto k3b (RMS).

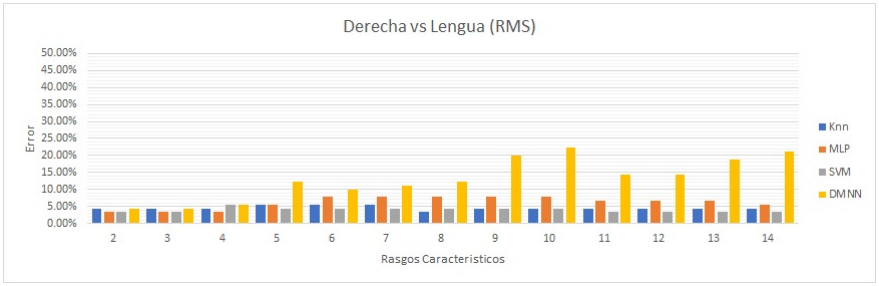


Figura 5.4: Desempeño de los clasificadores para MDer. vs Lengua del sujeto k3b (RMS).

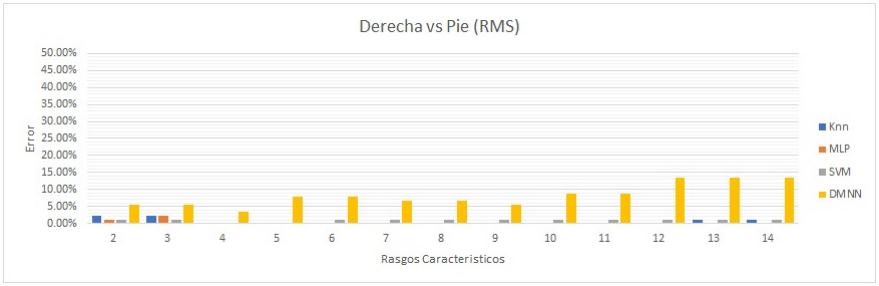


Figura 5.5: Desempeño de los clasificadores para MDer. vs Pie del sujeto k3b (RMS).

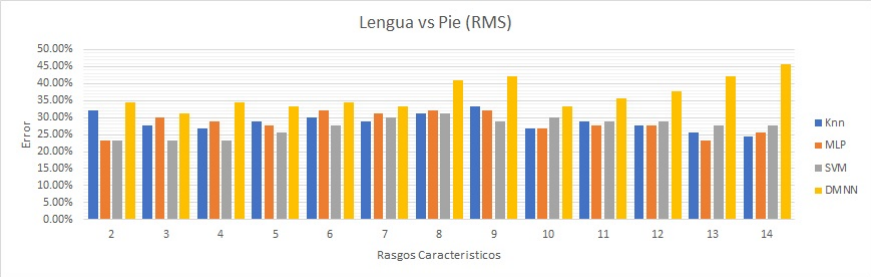


Figura 5.6: Desempeño de los clasificadores para Lengua vs Pie del sujeto k3b (RMS).

Como se observa, con este sujeto se obtuvieron resultados favorables para la mayoría de pares de clases ya que al menos se encuentra un clasificador con un error menor o igual al 5 %, incluso en el conjunto de datos de Mano Izquierda vs Pie y Mano Derecha vs Pie se encuentran clasificadores que tienen un 0 % de error. El mayor problema ocurre en el conjunto de datos de Lengua vs Pie, ya que aquí se presentan los errores más grandes siendo el menor de ellos con el clasificador de tipo máquina de soporte vectorial y un conjunto de datos con 4 rasgos, el cual presentó un error del 23,34 %.

Se observa que la neurona morfológica de procesamiento dendrítico tuvo problemas para clasificar las tareas mentales cuando se aumentaba el número de rasgos característicos.

De igual forma resta comparar con los resultados obtenidos de los conjuntos de datos con la medida estadística de la varianza, estos se verán más adelante, primero se reportan los resultados obtenidos del sujeto k6b en los gráficos que a continuación se muestran:

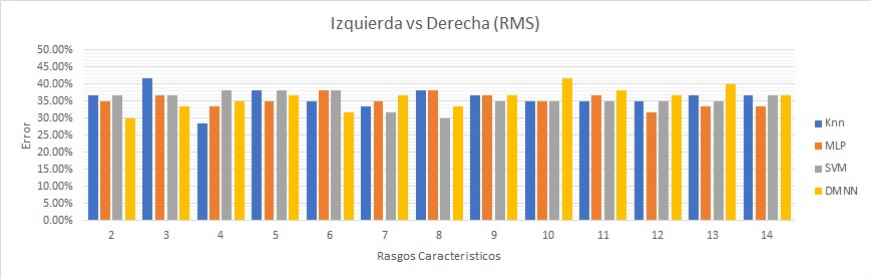


Figura 5.7: Desempeño de los clasificadores para MIzq. vs MDer. del sujeto k6b (RMS).

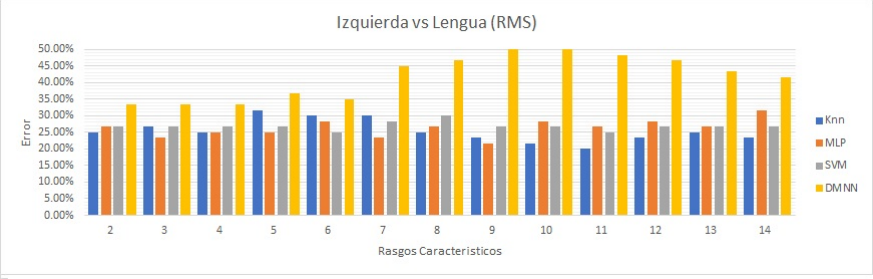


Figura 5.8: Desempeño de los clasificadores para MIzq. vs Lengua del sujeto k6b (RMS).

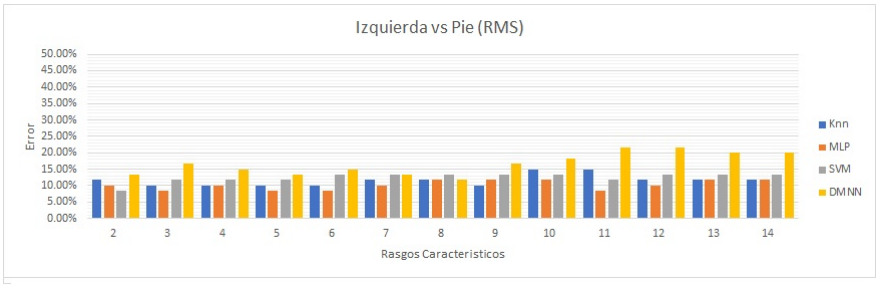


Figura 5.9: Desempeño de los clasificadores para MIzq. vs Pie del sujeto k6b (RMS).

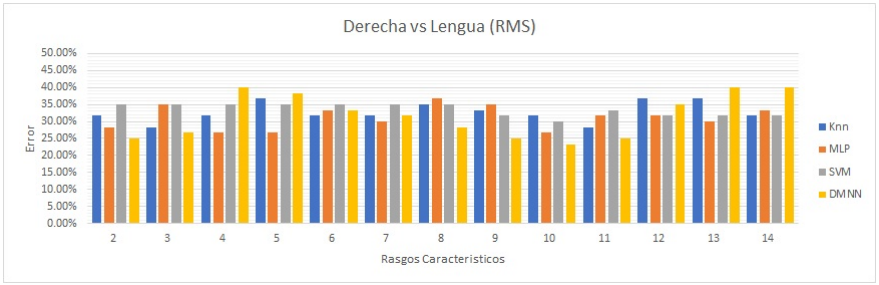


Figura 5.10: Desempeño de los clasificadores para MDer. vs Lengua del sujeto k6b (RMS).

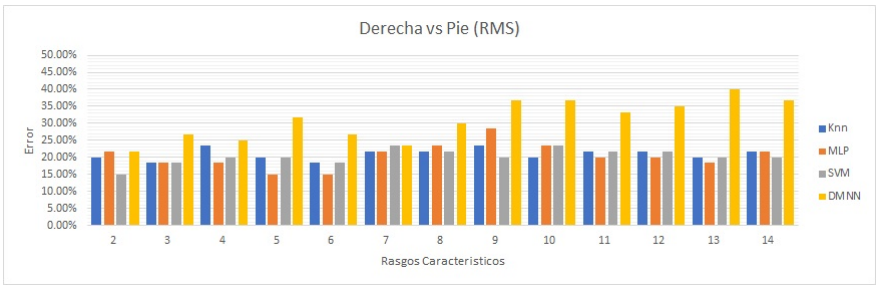


Figura 5.11: Desempeño de los clasificadores para MDer. vs Pie del sujeto k6b (RMS).

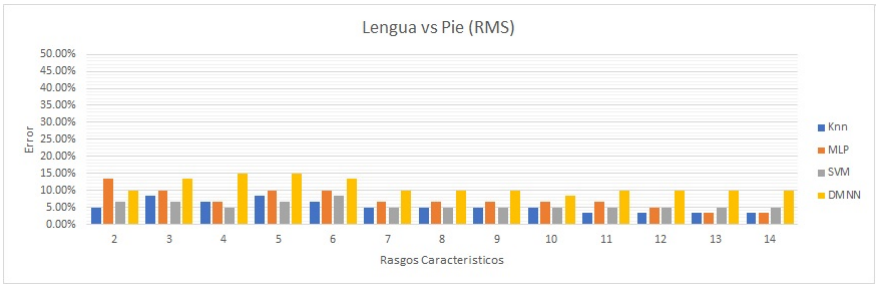


Figura 5.12: Desempeño de los clasificadores para Lengua vs Pie del sujeto k6b (RMS).

Al comparar los resultados obtenidos de los clasificadores con el sujeto k3b y el sujeto k6b se puede observar que el desempeño es menor en los del segundo sujeto, cabe aclarar que

el número de patrones es menor en el sujeto k6b, siendo 120 para entrenamiento y 120 para evaluación.

De igual forma se ve afectado el rendimiento de los clasificadores mostrando un error superior al 20 % en la mayoría de los conjuntos de datos, no obstante se mostró un buen desempeño en los conjuntos de datos de Mano Izquierda vs Pie y de Lengua vs Pie, en estos se presentaron errores menores al 10 %.

Como se mencionó anteriormente se realizó el proceso con conjunto de datos que utilizaba la varianza como medida, de igual forma primero se muestran los resultados obtenidos del sujeto k3b:

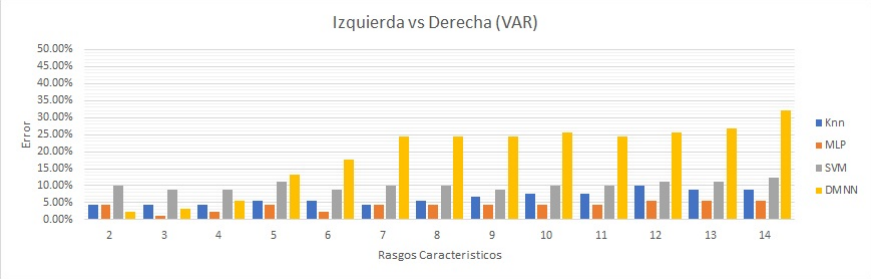


Figura 5.13: Desempeño de los clasificadores para MIzq. vs MDer. del sujeto k3b (VAR).

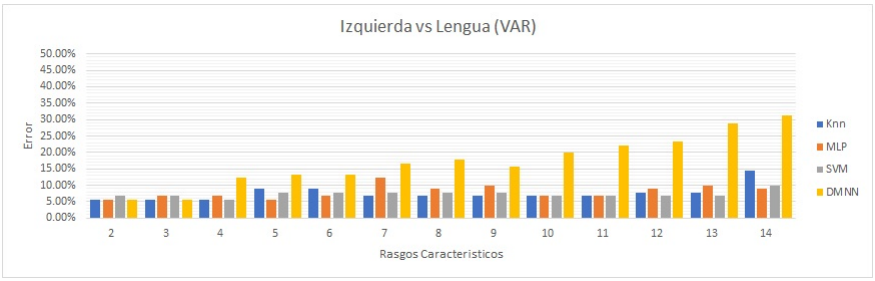


Figura 5.14: Desempeño de los clasificadores para MIzq. vs Lengua del sujeto k3b (VAR).

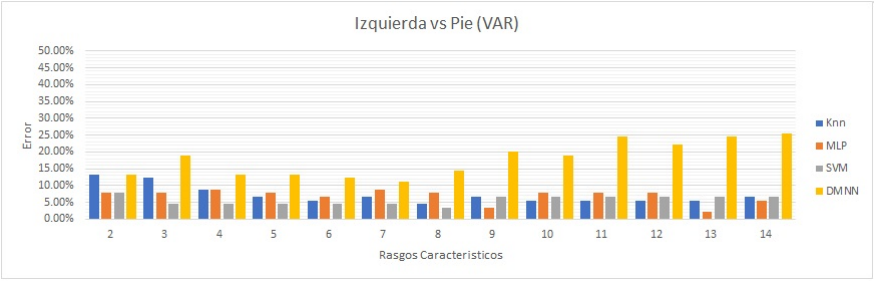


Figura 5.15: Desempeño de los clasificadores para MIzq. vs Pie del sujeto k3b (VAR).

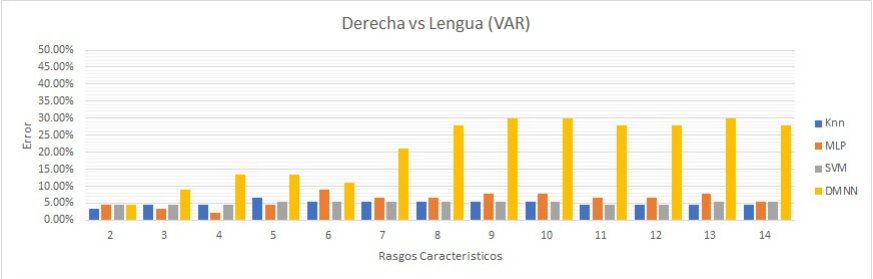


Figura 5.16: Desempeño de los clasificadores para MDer. vs Lengua del sujeto k3b (VAR).

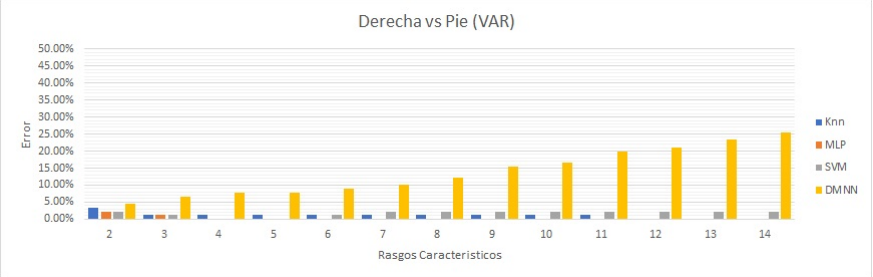


Figura 5.17: Desempeño de los clasificadores para MDer. vs Pie del sujeto k3b (VAR).

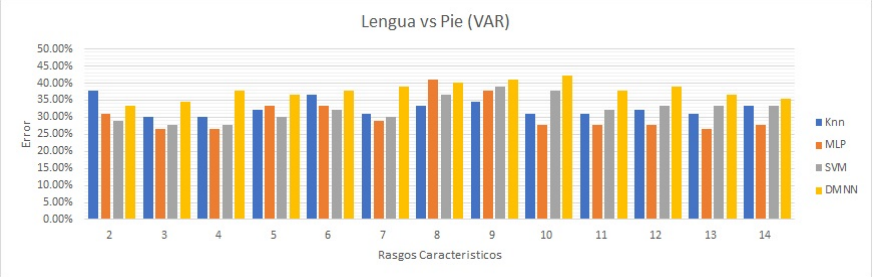


Figura 5.18: Desempeño de los clasificadores para Lengua vs Pie del sujeto k3b (VAR).

Como se puede observar los resultados son similares a los obtenidos con la media cuadrática, incluso se mantiene el desempeño de los clasificadores en el conjunto de datos de Lengua vs Pie con errores iguales o mayores al 25 %, de igual forma se repitió el proceso para el sujeto k6b obteniendo resultados similares:

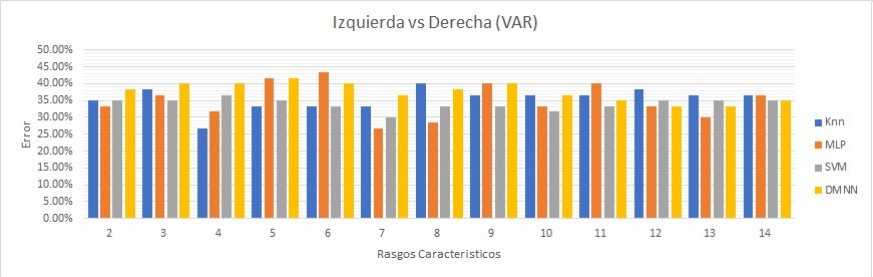


Figura 5.19: Desempeño de los clasificadores para MIzq. vs MDer. del sujeto k6b (VAR).

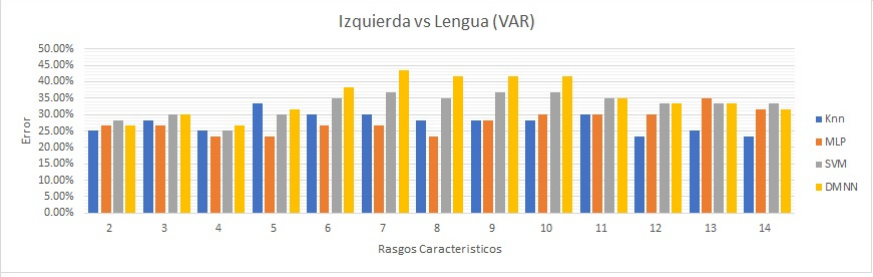


Figura 5.20: Desempeño de los clasificadores para MIzq. vs Lengua del sujeto k6b (VAR).

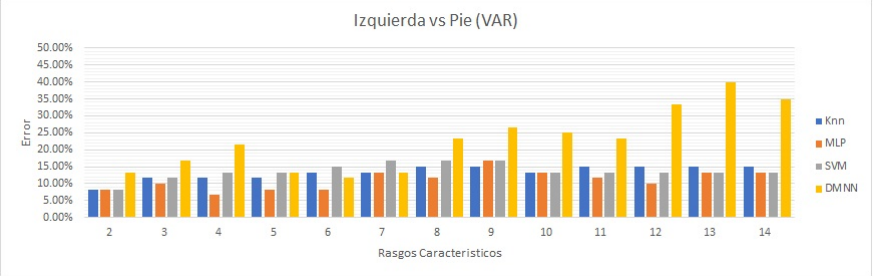


Figura 5.21: Desempeño de los clasificadores para MIzq. vs Pie del sujeto k6b (VAR).

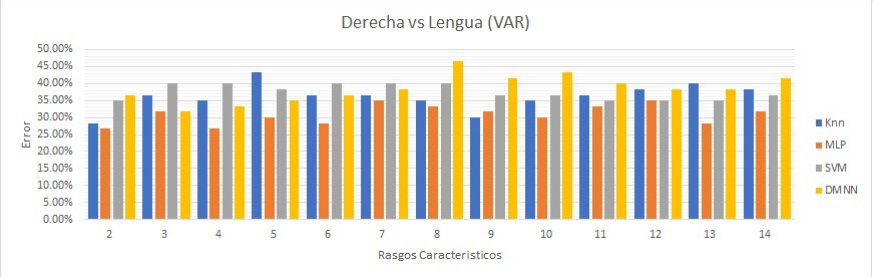


Figura 5.22: Desempeño de los clasificadores para MDer. vs Lengua del sujeto k6b (VAR).

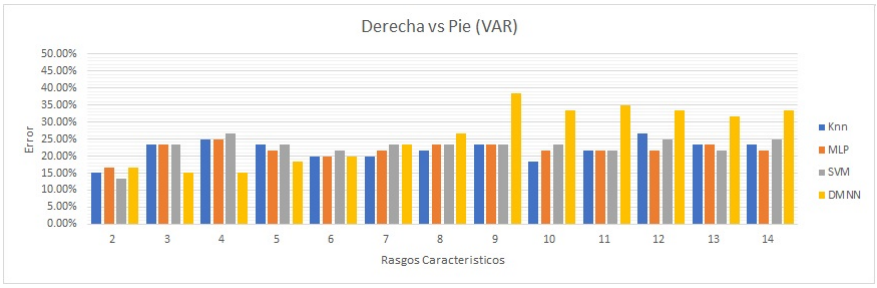


Figura 5.23: Desempeño de los clasificadores para MDer. vs Pie del sujeto k6b (VAR).

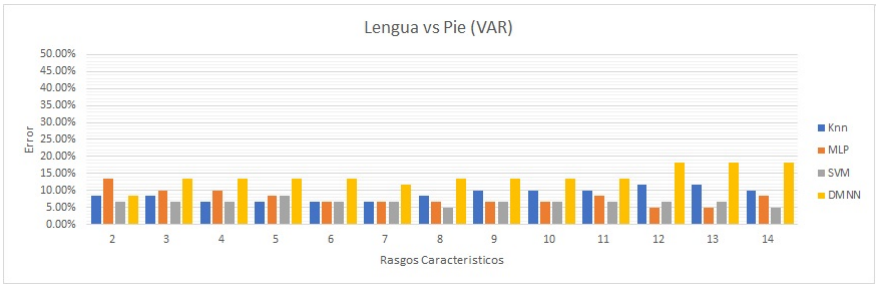


Figura 5.24: Desempeño de los clasificadores para Lengua vs Pie del sujeto k6b (VAR).

5.2. Clasificación binaria: cuatro clases

Para la clasificación de cuatro tipos de clases motoras, se realizó primero la clasificación binaria de dos grandes clases, como en los casos anteriores esto se realizó para ambos sujetos de prueba.

Comenzando con el sujeto k3b se aplicaron los clasificadores variando el número de rasgos característicos, obteniendo los siguientes resultados:

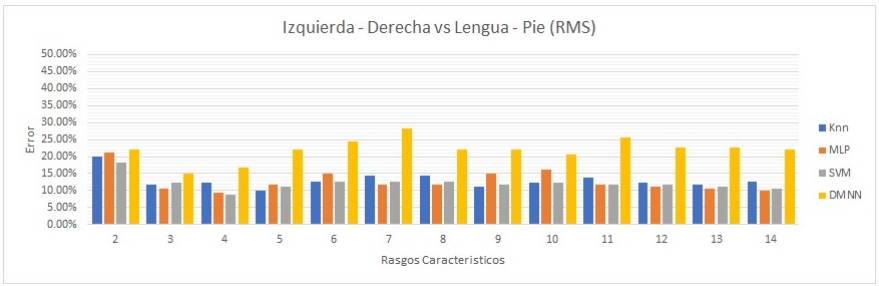


Figura 5.25: Desempeño de los clasificadores para MIzq.-MDer. vs Lengua-Pie del sujeto k3b (RMS).

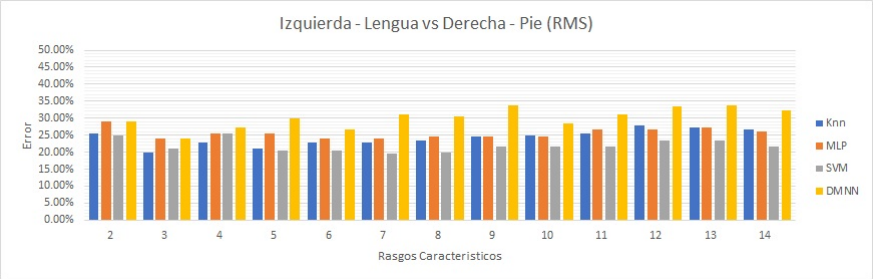


Figura 5.26: Desempeño de los clasificadores para MIzq.-Lengua vs MDer.-Pie del sujeto k3b (RMS).

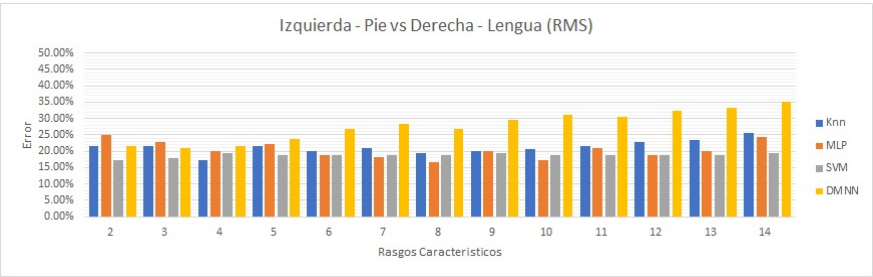


Figura 5.27: Desempeño de los clasificadores para MIzq.-Pie vs MDer.-Pie del sujeto k3b (RMS).

En este caso los clasificadores contaron con 180 patrones de entrenamiento y 180 para evaluación de los mismos, al analizar se observa que el conjunto Mano Izquierda - Mano Derecha vs Lengua - Pie es el que presenta mejores resultados, presentando un clasificador de tipo máquina de soporte vectorial con un error del 8,89 % mientras que en los otros dos conjuntos los errores rondan el 20 %. De igual forma en la red de clasificadores de la siguiente sección se trabajó con los tres conjuntos.

Posteriormente se realizó la clasificación de los datos obtenidos del sujeto k6b:

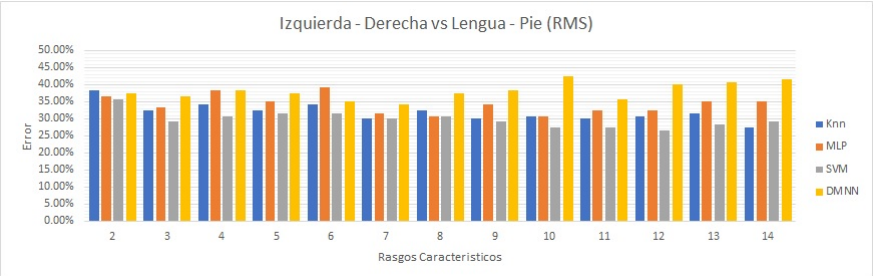


Figura 5.28: Desempeño de los clasificadores para MIzq.-MDer. vs Lengua-Pie del sujeto k6b (RMS).

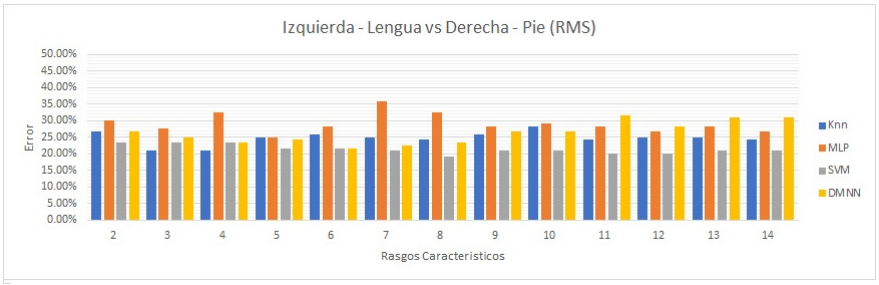


Figura 5.29: Desempeño de los clasificadores para MIzq.-Lengua vs MDer.-Pie del sujeto k6b (RMS).

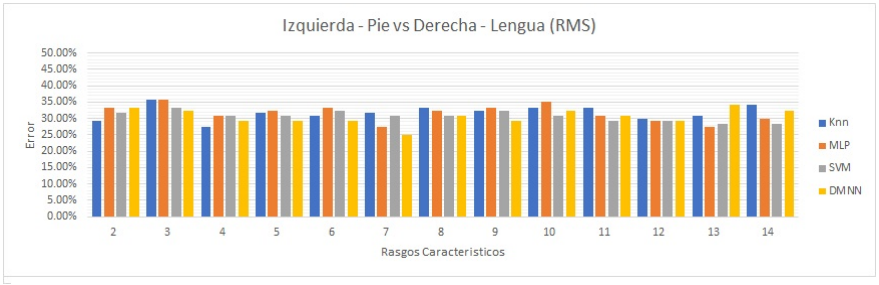


Figura 5.30: Desempeño de los clasificadores para MIzq.-Pie vs MDer.-Pie del sujeto k6b (RMS).

A diferencia del sujeto k3b, con el sujeto k6b se tuvieron 120 datos para entrenamiento y 120 datos para evaluación mostrando errores cercanos al 25 % en la mayoría de los clasificadores implementados, el mejor resultado se encuentra en el conjunto de datos de Mano Izquierda - Lengua vs Mano Derecha - Pie, en el cual se presenta un clasificador con un error del 19,17 %.

Todo este proceso se repitió tomando en cuenta el método estadístico de la varianza para comparar con los datos obtenidos anteriormente, primero se muestran los datos obtenidos del sujeto k3b:

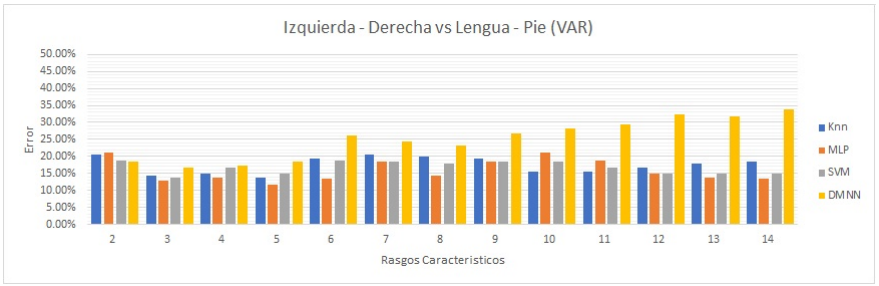


Figura 5.31: Desempeño de los clasificadores para MIzq.-MDer. vs Lengua-Pie del sujeto k3b (VAR).

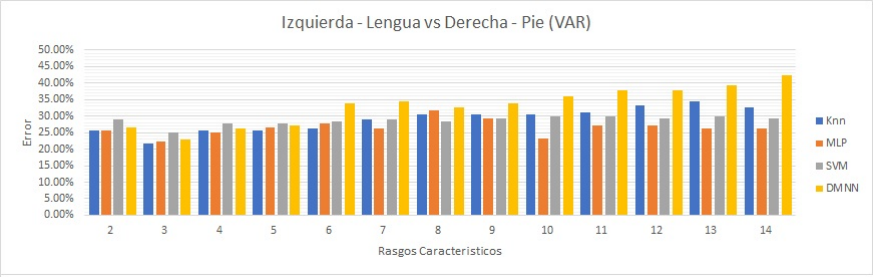


Figura 5.32: Desempeño de los clasificadores para MIzq.-Lengua vs MDer.-Pie del sujeto k3b (VAR).

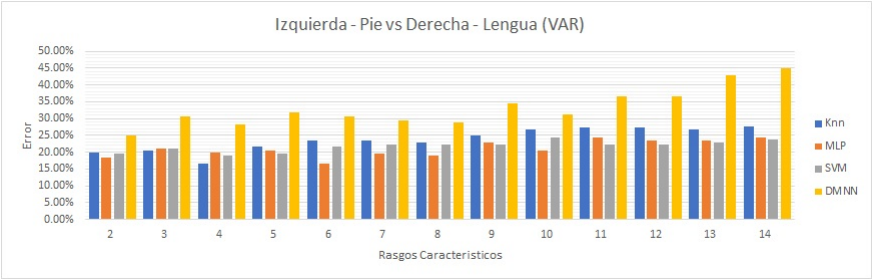


Figura 5.33: Desempeño de los clasificadores para MIzq.-Pie vs MDer.-Pie del sujeto k3b (VAR).

Como se puede observar los resultados son similares a los obtenidos con la media cuadrática, incluso se mantiene el desempeño de los clasificadores colocando al conjunto de datos de Mano Izquierda - Mano Derecha vs Lengua - Pie como la mejor opción para un clasificador final.

Para el sujeto k6b se obtuvieron resultados similares, mostrando resultados que rondan el 25 % de error.

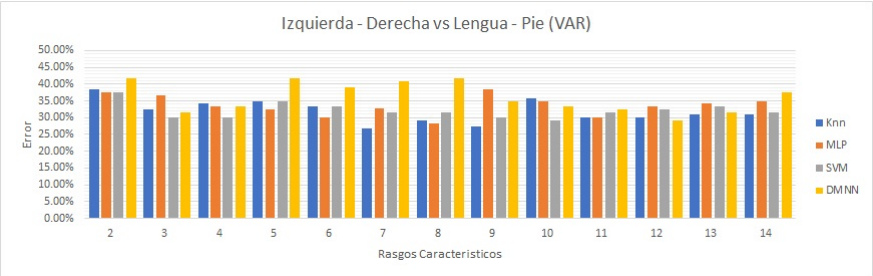


Figura 5.34: Desempeño de los clasificadores para MIzq.-MDer. vs Lengua-Pie del sujeto k6b (VAR).

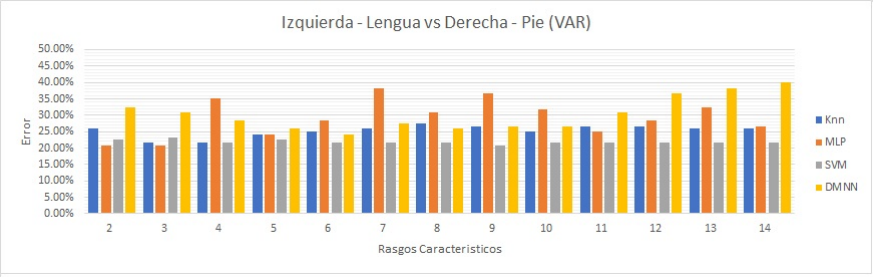


Figura 5.35: Desempeño de los clasificadores para MIzq.-Lengua vs MDer.-Pie del sujeto k6b (VAR).

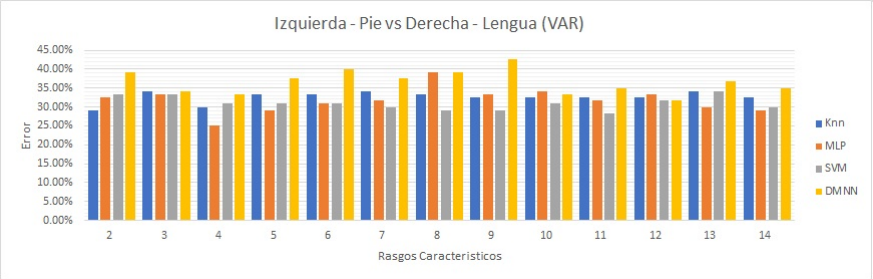


Figura 5.36: Desempeño de los clasificadores para MIzq.-Pie vs MDer.-Pie del sujeto k6b (VAR).

5.3. Red de clasificadores

Los resultados de los experimentos anteriores mostraron un buen desempeño en la clasificación binaria por lo cual se diseñó una red de clasificadores. La selección de los mejores clasificadores se realizó mediante los criterios mostrados en el capítulo anterior.

5.3.1. Sujeto de prueba: k3b

MIzq.-MDer. vs Lengua-Pie

Tabla 5.1: Red de Clasificadores

	Clasificador	Error
MIzq.-MDer. vs Lengua-Pie	SVM (4 Rasgos - RMS)	8,89 %
MIzq. vs MDer.	MLP (4 Rasgos - RMS)	1,12 %
Lengua vs Pie	SVM (4 Rasgos - RMS)	23,34 %

Tabla 5.2: Resultados de Clasificación (4 Clases)

	Red de Clasificadores	
	Error	Matriz de Confusión
MIzq.-MDer. vs Lengua-Pie	20,56 %	$\begin{bmatrix} 41 & 0 & 4 & 0 \\ 1 & 42 & 2 & 0 \\ 2 & 3 & 37 & 3 \\ 4 & 1 & 17 & 23 \end{bmatrix}$

MIzq.-Lengua vs MDer.-Pie

Tabla 5.3: Red de Clasificadores

	Clasificador	Error
MIzq.-Lengua vs MDer.-Pie	MLP (8 Rasgos - RMS)	19,45 %
MIzq. vs Lengua	MLP (8 Rasgos - RMS)	5,56 %
MDer. vs Pie	SVM (2 Rasgos - RMS)	0,00 %

Tabla 5.4: Resultados de Clasificación (4 Clases)

	Red de Clasificadores	
	Error	Matriz de Confusión
MIzq.-Lengua vs MDer.-Pie	21,67 %	$\begin{bmatrix} 33 & 5 & 1 & 6 \\ 2 & 43 & 0 & 0 \\ 3 & 2 & 32 & 8 \\ 2 & 0 & 10 & 33 \end{bmatrix}$

MIzq.-Pie vs MDer.-Lengua

Tabla 5.5: Red de Clasificadores

	Clasificador	Error
MIzq.-Pie vs MDer.-Lengua	MLP (8 Rasgos - RMS)	16,67 %
MIzq. vs Pie	MLP (8 Rasgos - RMS)	0,00 %
MDer. vs Lengua	SVM (2 Rasgos - RMS)	3,34 %

Tabla 5.6: Resultados de Clasificación (4 Clases)

	Red de Clasificadores	
	Error	Matriz de Confusión
MIzq.-Pie vs MDer.-Lengua	17,33 %	$\begin{bmatrix} 40 & 5 & 0 & 0 \\ 1 & 43 & 0 & 1 \\ 0 & 2 & 24 & 19 \\ 0 & 0 & 3 & 42 \end{bmatrix}$

5.3.2. Sujeto de prueba: k6b

MIzq.-MDer. vs Lengua-Pie

Tabla 5.7: Red de Clasificadores

	Clasificador	Error
MIzq.-MDer. vs Lengua-Pie	SVM (12 Rasgos - RMS)	26,67 %
MIzq. vs MDer.	KNN (4 Rasgos - RMS)	28,34 %
Lengua vs Pie	KNN (11 Rasgos - RMS)	3,34 %

Tabla 5.8: Resultados de Clasificación (4 Clases)

	Red de Clasificadores	
	Error	Matriz de Confusión
MIzq.-MDer. vs Lengua-Pie	38,34 %	$\begin{bmatrix} 18 & 7 & 5 & 0 \\ 6 & 14 & 5 & 5 \\ 4 & 8 & 17 & 1 \\ 2 & 3 & 0 & 25 \end{bmatrix}$

MIzq.-Lengua vs MDer.-Pie

Tabla 5.9: Red de Clasificadores

	Clasificador	Error
MIzq.-Lengua vs MDer.-Pie	SVM (8 Rasgos - RMS)	19,17 %
MIzq. vs Lengua	KNN (11 Rasgos - RMS)	20,00 %
MDer. vs Pie	SVM (2 Rasgos - RMS)	15,00 %

Tabla 5.10: Resultados de Clasificación (4 Clases)

	Red de Clasificadores	
	Error	Matriz de Confusión
MIzq.-Lengua vs MDer.-Pie	32,50 %	$\begin{bmatrix} 19 & 4 & 6 & 1 \\ 7 & 12 & 7 & 4 \\ 5 & 0 & 24 & 1 \\ 1 & 1 & 2 & 26 \end{bmatrix}$

MIzq.-Pie vs MDer.-Lengua

Tabla 5.11: Red de Clasificadores

	Clasificador	Error
MIzq.-Pie vs MDer.-Lengua	DMNN (7 Rasgos - RMS)	25,00 %
MIzq. vs Pie	SVM (2 Rasgos - RMS)	8,34 %
MDer. vs Lengua	DMNN (10 Rasgos - RMS)	23,34 %

Tabla 5.12: Resultados de Clasificación (4 Clases)

	Red de Clasificadores	
	Error	Matriz de Confusión
MIzq.-Pie vs MDer.-Lengua	35,84 %	$\begin{bmatrix} 17 & 8 & 5 & 0 \\ 6 & 15 & 5 & 4 \\ 2 & 7 & 21 & 0 \\ 1 & 2 & 3 & 24 \end{bmatrix}$

5.4. Clasificación en entorno virtual

Como se mencionó anteriormente, el objetivo del proyecto es el controlar una silla de ruedas, ésta se generó en un entorno virtual y se implementaron los mejores clasificadores.

El programa muestra en pantalla el movimiento que el usuario quiso realizar y la predicción del clasificador, así como un contador de los movimientos que se han realizado para el cálculo de la exactitud y error, esto permitió observar el comportamiento ya funcional de una silla de ruedas.

5.4.1. Sujeto de prueba: k3b

Tabla 5.13: Clasificador Entorno Virtual

Movimientos 1000	Error
MIzq.-MDer. vs Lengua-Pie	13,60 %
MIzq.-Lengua vs MDer.-Pie	29,62 %
MIzq.-Pie vs MDer.-Lengua	33,50 %

5.4.2. Sujeto de prueba: k6b

Tabla 5.14: Clasificador Entorno Virtual

Movimientos 1000	Error
MIzq.-MDer. vs Lengua-Pie	47,20 %
MIzq.-Lengua vs MDer.-Pie	35,50 %
MIzq.-Pie vs MDer.-Lengua	41,60 %

En el área práctica del manejo de una silla de ruedas generalmente el movimiento más común será el de ir hacia adelante, por lo cual se entiende que es preferible que el clasificador utilizado tenga el mejor desempeño en esta acción.

Analizando los resultados obtenidos en la aplicación práctica en el ambiente virtual del sujeto k3b se puede identificar que el conjunto de datos de Mano Izquierda - Mano Derecha vs Lengua - Pie es el que muestra el mejor resultado, esto se debe a que permite la movilidad de manera adecuada ya que el movimiento de avanzar hacia adelante es clasificado con poco error.

Hubo una notable diferencia con los resultados obtenidos con el sujeto k6b, ya que aquí la acción del movimiento hacia adelante ocurría erróneamente en la mayoría de ocasiones por lo cual, aunque los giros fueron bien realizados, no se permitía una movilidad fluida, una propuesta de solución para mejorar el desempeño práctico sería el cambiar los patrones de visualización motora con ese sujeto.

5.5. Contribuciones del proyecto

Este proyecto tuvo como objetivo la clasificación de señales cerebrales de dos y cuatro clases de tareas mentales para el control virtual de una silla de ruedas virtual. En el estado del arte se observó en la mayoría de casos solo la clasificación de dos tareas mentales (Izquierda vs Derecha), por lo mismo a continuación se comparan los resultados mostrados:

Tabla 5.15: Comparación con el Estado del Arte (2 Clases)

Estado del Arte (Mano Derecha vs Izquierda)			
	Método de Extracción	**Clases**	**Exactitud**
[1]	WDT + Promedio, min, max	KNN	84.28 %
		Clasificador Bayesiano	68.75 %
		MLP	74 %
		LDA	87.86 %
		SVM	88.57 %
[16]	Wavelet + CSP (10 canales)	FLDA	93 %
		SVM	90.9 %
		KNN	92.9 %

[3]	LDB + CSP	FLDA	75 %
		DBI	63 %
	LDB + LCT	FLDA	64 %
		DBI	71 %
[8]	FLDA	CSP (2 rasgos)	89.4 %
		CSP (10 rasgos)	89.4 %
[24]	RCSP	Arboles de Decision	79.8 %
		KNN	92.5 %
		LDA	95.4 %
		PSO-SVM	97 %
Proyecto	CSP (3 rasgos)	KNN	96.66 %
		MLP [10,1]	97.77 %
		SVM	95.55 %
		DMNN	96.66 %

- LDA: Linear Discriminant Analysis.
- LCT: Local Cosine Transform.
- LDB: Local Discriminant Bases.
- FLDA: Fisher Linear Discriminant Analysis.
- DBI: Davies-Bauldin Index.
- RCSP: Regularized Common Spatial Pattern.
- PSO-SVM: Particle Swarm Optimization SVM.

En la mayoría de casos los clasificadores utilizados en el proyecto presentan mejores resultados, sin embargo el sujeto de prueba no fue el mismo en todos los casos y como se observo en el desarrollo de este proyecto los resultados pueden variar dependiendo del sujeto de prueba.

Otro aspecto a resaltar es que solo se reporta la clasificación de dos tareas mentales, específicamente Mano Derecha vs Mano Izquierda. Por lo que la clasificación binaria de cuatro tareas mentales se considera un aporte de este proyecto. Así mismo se encontró un trabajo en el que se realizo la clasificación de cuatro clases:

Tabla 5.16: Comparación con el Estado del Arte (4 Clases)

Estado del Arte (Mano Derecha vs Izquierda)				
	Método de Extracción	**Clases**	**Clasificador (# rasgos)**	**Exactitud**
[11]	FBCSP - OVR	MIzq, MDer, Lengua, Pie	NBPW	77 %
Proyecto	CSP - Pair Wise	(MIzq vs MDer) vs (Lengua vs Pie)	SVM(4) + MLP(4) + SVM(4)	79.44 %
		(MIzq vs Lengua) vs (Derecha vs Pie)	MLP(8) + MLP(8) + SVM(2)	78.33 %
		(MIzq vs Pie) vs (MDer vs Lengua)	MLP(8) + MLP(8) + SVM(2)	82.67 %

- FBCSP: Filter Bank Common Spatial Pattern.
- NBPW: Naive Bayesian Parzen Window.

Asi mismo la aplicacion de la red neuronal de procesamiento dendrítico para la clasificacion de señales cerebrales solo se ha presentado en un articulo del estado del arte [2], aquí se

realizó la clasificación de dos tareas mentales: relajación y ejecución del movimiento de la mano. En este proyecto se realizó la clasificación de tareas mentales mas especificas del tipo motor, por lo anterior se considera como una contribución importante. De igual forma se hace resaltar que la red neuronal de procesamiento dendrítico mostró un desempeño que igualo en la mayoría de los casos los resultados mostrados por los clasificadores clásicos (KNN, SVM , MLP).

Tabla 5.17: Clasificación de señales cerebrales con DMNN

DMNN			
	Método de Extracción	**Clases**	**Exactitud**
[2]	PSD, Análisis r^2	Relajación vs Intención	68 %
		Relajación vs Ejecución	70 %
		Ejecución vs Intención	77 %
Proyecto	CSP (3 Rasgos)	Mano Izquierda vs Mano Derecha	96.66 %
		Mano Izquierda vs Lengua	85.55 %
		Mano Izquierda vs Pie	90 %
		Mano Derecha vs Lengua	95.55 %
		Mano Derecha vs Pie	94.44 %
		Lengua vs Pie	68.88 %

Capítulo 6

Conclusiones

La investigación en el campo de interfaces cerebro-computadora está aún en desarrollo ya que no se conoce exactamente como funciona el cerebro, lo que se hace actualmente es relacionar patrones encontrados en la señal cerebral con imágenes mediante la técnica *Motor Imagery* (Visualización Motora). Con esto se ha podido avanzar en este campo permitiendo hallar un enlace entre el cerebro y un dispositivo. La implementación de una interfaz cerebro-computadora completamente funcional impulsaría diferentes campos de investigación en los cuales se obtendría un controlador tan poderoso como lo es el cerebro humano en la operación de equipos o robots que lo requieran.

En la realización de este proyecto se pudo observar una aplicación en el campo de la asistencia en el ámbito médico, obteniendo resultados favorables, lo que permitió discriminar entre cuatro acciones motoras realizadas por dos diferentes sujetos de prueba.

Se implementó un sistema de procesamiento para la lectura y pre procesamiento de las señales obtenidas de la base de datos, en el cual se organizan los canales seleccionados para cada patrón y se implementa un filtrado en la banda perteneciente al ritmo Mu.

Se implementó el algoritmo de *Common Spatial Pattern* para la extracción de rasgos característicos de las señales cerebrales, dando como resultado conjuntos de datos cuasi linealmente separables en la mayoría de los casos. Lo anterior se atribuye a su modo de funcionamiento que fuerza la diferencia entre dos diferentes patrones, por lo mismo el enfoque del proyecto se llevo a cabo mediante clasificación binaria. Esta etapa puede considerarse como la mas critica en la discriminación de tareas mentales, ya que de aquí depende el correcto funcionamiento de los clasificadores.

En cuanto al diseño de clasificadores fueron utilizados algoritmos simples en su implementación ya que al observar los conjuntos de datos con los que se trabajó se encontraba una notoria separabilidad lineal en la mayoría, sin embargo es un área que se puede mejorar ya que en algunos casos no se percibe esa distinción entre clases. Y como se mencionó fue tomado un enfoque de clasificación binaria para trabajar con cuatro clases.

Se diseñó e implementó un entorno virtual en la plataforma Unity con el objetivo de ofrecer una interfaz gráfica amigable para el usuario y otorgar retroalimentación de las acciones que se fueran realizando.

Se implementó favorablemente el sistema de control al modelo de silla de ruedas virtual, en el cual cada una de las tareas de visualización motora establecidas fue enlazada a una acción del modelo virtual, así mismo se concluye que es de suma importancia un control a nivel local en la silla de ruedas, el cual delimite la magnitud de los movimientos y priorice la seguridad del paciente, en este proyecto no se realizó la implementación de este control.

Así mismo se hace notoria una diferencia entre los datos obtenidos de cada uno de los sujetos de prueba, ya que en el sujeto k3b se observó mejor desempeño en la clasificación, esto puede atribuirse a un gran número de factores, como por ejemplo, la forma de adquisición de las señales, el estado mental del sujeto a la hora de la prueba, la concentración en la tarea a realizar, inclusive la forma de pensar de cada uno de ellos y por supuesto la fisiología cerebral de los mismos.

Actualmente no se conoce completamente la forma de funcionamiento del cerebro humano, este tipo de investigación se basa en aproximaciones y por lo mismo sigue en desarrollo.

Se cumplieron los objetivos planteados en este proyecto, por lo cual se concluye que las señales cerebrales generadas por estímulos de visualización motora pueden ser clasificados y así mismo utilizados para el control de dispositivos.

Capítulo 7

Recomendaciones y trabajo futuro

En la realización de este proyecto se observaron diversas oportunidades de mejora en esta línea de investigación.

- Prueba con acciones de visualización motora diferentes: se puede realizar la adquisición de señales cerebrales utilizando el movimiento imaginario de otra parte del cuerpo.
- Prueba con señales de estímulos diferentes como figuras o colores: se puede ocupar variantes de los estímulos presentados al sujeto, no necesariamente acciones motoras, incluso utilizarse conceptos con mayor nivel de abstracción.
- Desarrollo de un método de extracción de rasgos característicos: este tema es de gran importancia, ya que de esta parte influye en el desempeño de los clasificadores. Actualmente se utilizan métodos basados en el espectro de frecuencia o como en este caso en el filtrado espacial, sin embargo tienen limites.
- Utilizar modelos de Aprendizaje Profundo o autoencoder para la extracción de rasgos característicos y clasificación de los patrones mentales.
- Implementar el procesamiento y adquisición en línea: como se mencionó la mayoría de las interfaces cerebro-computadora utilizan un protocolo de tipo sincróno, por lo cual el paciente no tiene la libertad de controlar cuando comienza una tarea mental. Esto dificulta la neuroplasticidad ya que el paciente no realiza las tareas mentales con naturalidad.
- Implementar un tipo clasificador diferente: en este proyecto se utilizaron algunos de los clasificadores mas sencillos, puede implementarse métodos con mayor poder computacional como lo son las redes pulso acopladas.
- Implementar un sistema para el control de la silla de ruedas utilizando la clasificación de señales cerebrales en conjunto con algoritmos de navegación autónoma y una adecuada instrumentación de la misma.

Bibliografía

[1] A. Ahangi, M. Karamnejad, N. Mohammadi, R. Ebrahimpour, and N. Bagheri. Multiple classifier system for EEG signal classification with application to brain-computer interfaces. *Neural Comput. Appl.*, 23(5):1319–1327, 2013.

[2] J. M. Antelis, B. Gudiño-mendoza, L. E. Falcón, and H. Sossa. Evaluating dendrite morphological neural networks in the recognition of voluntary movements from electroencephalographic signals. pages 1–19.

[3] J. Asensio Cubero, J. Q. Gan, and R. Palaniappan. Extracting optimal tempo-spatial features using local discriminant bases and common spatial patterns for brain computer interfacing. *Biomed. Signal Process. Control*, 8(6):772–778, 2013.

[4] Associated Press (Producer). Scientists develop wheelchairs controlled by the mind [Streaming video]. *Retrieved from Assoc. Press Video Collect. database*, 2009.

[5] Associated Press (Producer). Scientists develop 'sniff and go' device to help quadriplegics control their wheelchairs [Streaming video]. *Retrieved from Assoc. Press Video Collect. database*, 2010.

[6] R. Barea. Electroencefalografía. *Instrumentación Biomédica*, pages 1–26, 2005.

[7] J. D. Bayliss. Use of the evoked potential P3 component for control in a virtual apartment. *IEEE Trans. Neural Syst. Rehabil. Eng.*, 11(2):113–116, jun 2003.

[8] S. A. Belhadj, N. Benmoussat, and M. D. Krachai. CSP features extraction and FLDA classification of EEG-based motor imagery for Brain-Computer Interaction. *2015 4th Int. Conf. Electr. Eng. ICEE 2015*, pages 3–8, 2016.

[9] A. Campbell, T. Choudhury, S. Hu, H. Lu, M. K. Mukerjee, M. Rabbi, and R. D. Raizada. NeuroPhone: Brain-Mobile Phone Interface using a Wireless EEG Headset. *Proc. Second ACM SIGCOMM Work. Networking, Syst. Appl. Mob. handhelds - MobiHeld '10*, page 3, 2010.

[10] V. Cherkassky and V. Cherkassky. *The nature of statistical learning theory.*, volume 8. 1997.

[11] Z. Y. Chin, K. K. Ang, C. Wang, C. Guan, and H. Zhang. Multi-class Filter Bank common spatial pattern for four-class motor imagery BCI. *Proc. 31st Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. Eng. Futur. Biomed. EMBC 2009*, 138632:571–574, 2009.

[12] E. Donchin, K. M. Spencer, and R. Wijesinghe. The mental prosthesis: Assessing the speed of a P300-based brain- computer interface. *IEEE Trans. Rehabil. Eng.*, 8(2):174–179, jun 2000.

[13] EMOTIV. Emotiv Epoc and Testbench Specifications. *Brain Comput. Interface Sci. Context. EEG*, pages 1–7, 2014.

[14] H. Ghaheri and A. R. Ahmadyfard. Extracting common spatial patterns from EEG time segments for classifying motor imagery classes in a Brain Computer Interface (BCI). *Sci. Iran.*, 20(6):2061–2072, 2013.

[15] K. A. Gosmanova, C. S. Carmack, D. Goldberg, K. Fitzpatrick, B. Zoltan, D. M. Zeitlin, J. R. Wolpaw, O. A. Maehle, A. Borge, and M. Theresa. Eeg-Based Brain-Computer Interface Access To Tobii Dynavox Communicator 5. pages 5–8, 2017.

[16] R. X. Han and Q. G. Wei. Feature Extraction by Combining Wavelet Packet Transform and Common Spatial Pattern in Brain-Computer Interfaces. *Appl. Mech. Mater.*, 239:974–979, 2013.

[17] H. Higashi and T. Tanaka. Simultaneous design of FIR filter banks and spatial patterns for EEG signal classification. *IEEE Trans. Biomed. Eng.*, 60(4):1100–1110, 2013.

[18] S. M. Hosni, M. E. Gadallah, S. F. Bahgat, and M. S. AbdelWahab. Classification of EEG signals using different feature extraction techniques for mental-task BCI. *2007 Int. Conf. Comput. Eng. Syst.*, pages 220–226, 2007.

[19] Instituto Nacional de Estadística y Geografía. Censo de población y vivienda. 2010.

[20] J. Katona and A. Kovari. EEG-based computer control interface for brain-machine interaction. *Int. J. Online Eng.*, 11(6):43–48, 2015.

[21] R. Leeb, D. Friedman, G. R. Müller-Putz, R. Scherer, M. Slater, and G. Pfurtscheller. Self-paced (asynchronous) BCI control of a wheelchair in virtual environments: A case study with a tetraplegic. *Comput. Intell. Neurosci.*, 2007, 2007.

[22] M. Li, W. Li, J. Zhao, Q. Meng, M. Zeng, and G. Chen. *A p300 model for cerebot - A mind-controlled humanoid robot*, volume 274, pages 495–502. Springer International Publishing, Cham, 2014.

[23] Y. Li, J. Long, T. Yu, Z. Yu, C. Wang, H. Zhang, and C. Guan. An EEG-based BCI system for 2-D cursor control by combining Mu/Beta rhythm and P300 potential. *IEEE Trans. Biomed. Eng.*, 57(10 PART 1):2495–2505, oct 2010.

[24] Y. Ma, X. Ding, Q. She, Z. Luo, T. Potter, and Y. Zhang. Classification of Motor Imagery EEG Signals with Support Vector Machines and Particle Swarm Optimization. *Comput. Math. Methods Med.*, 2016(5):667–677, 2016.

[25] C. Mayor, J. G. Ochoa, and J. Burneo. *Manual de electroencefalografía. Handbook of Electroencephalography.* 2013.

[26] S. Member, J. M. Antelis, K. Andrea, and J. Minguez. A Noninvasive Brain-Actuated Wheelchair Based on a P300 Neurophysiological Protocol and Automated Navigation. *Robot. IEEE Trans.*, 25(3):614–627, 2009.

[27] T. Mulder. Motor imagery and action observation: Cognitive tools for rehabilitation. *J. Neural Transm.*, 114(10):1265–1278, 2007.

[28] Organizacion Muldial de la Salud. Informe mundial sobre la discapacidad. pages 1–27, 2011.

[29] J. A. Pineda. The functional significance of mu rhythms: Translating <seeing> and <hearing> into <doing>. *Brain Res. Rev.*, 50(1):57–68, 2005.

[30] J. C. Platt. Using Analytic QP and Sparseness to Speed Training of Support Vector Machines. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Adv. Neural Inf. Process. Syst. (NIPS' 99)*, volume 11, pages 557–563. MIT Press, 1999.

[31] H. Ramoser, J. Müller-Gerking, and G. Pfurtscheller. Optimal spatial filtering of single trial EEG during imagined hand movement. *IEEE Trans. Rehabil. Eng.*, 8(4):441–446, 2000.

[32] H. Sossa and E. Guevara. Efficient training for dendrite morphological neural networks. *Neurocomputing*, 131:132–142, 2014.

[33] M. L. Spring. *Machine Learning in Action.* 2015.

[34] S. Tong and N. V. Thankor. *Quantitative EEG Analysis Methods and Clinical Applications.* Number January. 2009.

[35] J. R. Wolpaw, D. J. McFarland, G. W. Neat, and C. A. Forneris. An EEG-based brain-computer interface for cursor control. *Electroencephalogr. Clin. Neurophysiol.*, 78(3):252–259, mar 1991.

[36] A. Yuksel and T. Olmez. A neural network-based optimal spatial filter design method for motor imagery classification. *PLoS One*, 10(5):1–22, 2015.

[37] K. T. Zeidan, F., Martucci. Brain Mechanisms Supporting Modulation of Pain by Mindfulness Meditation. *J. Neurosci.*, 31(14):5540–5548, 2011.

[38] Y. Zhang, G. Zhou, J. Jin, X. Wang, and A. Cichocki. Optimizing spatial patterns with sparse filter bands for motor-imagery based brain-computer interface. *J. Neurosci. Methods*, 255:85–91, 2015.

Apéndice A

Tablas de resultados de clasificación

A.1. Sujeto de prueba: k3b

A.1.1. Dos Clases - media cuadrática

Tabla A.1: M.Izq VS M.Der (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	3,34 %	3,34 %	2,23 %	5,56 %	5,56 %	4,45 %	4,45 %	4,45 %	4,45 %	4,45 %	5,56 %	4,45 %	5,56 %
MLP	4,45 %	2,23 %	1,12 %	1,12 %	2,23 %	2,23 %	3,34 %	1,12 %	2,23 %	2,23 %	4,45 %	5,56 %	5,56 %
SVM	6,67 %	4,45 %	4,45 %	4,45 %	5,56 %	5,56 %	7,78 %	6,67 %	6,67 %	6,67 %	6,67 %	7,78 %	6,67 %
DMNN	8,89 %	3,34 %	3,34 %	3,34 %	7,78 %	10,00 %	16,67 %	22,23 %	25,56 %	24,45 %	24,45 %	26,67 %	24,45 %

Tabla A.2: M.Izq VS Lengua (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	5,56 %	5,56 %	7,78 %	8,89 %	7,78 %	6,67 %	6,67 %	6,67 %	7,78 %	7,78 %	7,78 %	7,78 %	7,78 %
MLP	6,67 %	5,56 %	7,78 %	6,67 %	8,89 %	7,78 %	10,00 %	7,78 %	6,67 %	6,67 %	6,67 %	6,67 %	7,78 %
SVM	5,56 %	5,56 %	5,56 %	5,56 %	5,56 %	7,78 %	7,78 %	6,67 %	6,67 %	7,78 %	6,67 %	6,67 %	5,56 %
DMNN	12,23 %	14,45 %	13,34 %	14,45 %	13,34 %	16,67 %	15,56 %	14,45 %	16,67 %	18,89 %	17,78 %	18,89 %	17,78 %

Tabla A.3: M.Izq VS Pie (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	12,23 %	10,00 %	8,89 %	7,78 %	6,67 %	6,67 %	6,67 %	4,45 %	4,45 %	5,56 %	6,67 %	5,56 %	5,56 %
MLP	14,45 %	16,67 %	8,89 %	10,00 %	6,67 %	1,12 %	0,00 %	2,23 %	3,34 %	3,34 %	3,34 %	3,34 %	3,34 %
SVM	6,67 %	5,56 %	5,56 %	5,56 %	5,56 %	5,56 %	4,45 %	4,45 %	3,34 %	3,34 %	3,34 %	4,45 %	4,45 %
DMNN	16,67 %	10,00 %	12,23 %	12,23 %	7,78 %	6,67 %	8,89 %	14,45 %	15,56 %	16,67 %	15,56 %	14,45 %	15,56 %

Tabla A.4: M.Der VS Lengua (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	4,45 %	4,45 %	4,45 %	5,56 %	5,56 %	5,56 %	3,34 %	4,45 %	4,45 %	4,45 %	4,45 %	4,45 %	4,45 %
MLP	3,34 %	3,34 %	3,34 %	5,56 %	7,78 %	7,78 %	7,78 %	7,78 %	7,78 %	6,67 %	6,67 %	6,67 %	5,56 %
SVM	3,34 %	3,34 %	5,56 %	4,45 %	4,45 %	4,45 %	4,45 %	4,45 %	4,45 %	3,34 %	3,34 %	3,34 %	3,34 %
DMNN	4,45 %	4,45 %	5,56 %	12,23 %	10,00 %	11,12 %	12,23 %	20,00 %	22,23 %	14,45 %	14,45 %	18,89 %	21,12 %

Tabla A.5: M.Der VS Pie (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	2,23 %	2,23 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	1,12 %	1,12 %
MLP	1,12 %	2,23 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %
SVM	1,12 %	1,12 %	0,00 %	0,00 %	1,12 %	1,12 %	1,12 %	1,12 %	1,12 %	1,12 %	1,12 %	1,12 %	1,12 %
DMNN	5,56 %	5,56 %	3,34 %	7,78 %	7,78 %	6,67 %	6,67 %	5,56 %	8,89 %	8,89 %	13,34 %	13,34 %	13,34 %

Tabla A.6: Lengua VS Pie (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	32,23 %	27,78 %	26,67 %	28,89 %	30,00 %	28,89 %	31,12 %	33,34 %	26,67 %	28,89 %	27,78 %	25,56 %	24,46 %
MLP	23,34 %	30,00 %	28,89 %	27,78 %	32,23 %	31,12 %	32,23 %	32,23 %	26,67 %	27,78 %	27,78 %	23,34 %	25,56 %
SVM	23,34 %	23,34 %	23,34 %	25,56 %	27,78 %	30,00 %	31,12 %	28,89 %	30,00 %	28,89 %	28,89 %	27,78 %	27,78 %
DMNN	34,45 %	31,12 %	34,45 %	33,34 %	34,45 %	33,34 %	41,12 %	42,23 %	33,34 %	35,56 %	37,78 %	42,23 %	45,56 %

A.1.2. Dos clases - varianza

Tabla A.7: M.Izq VS M.Der (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	4,45 %	4,45 %	4,45 %	5,56 %	5,56 %	4,45 %	5,56 %	6,67 %	7,78 %	7,78 %	10,00 %	8,89 %	8,89 %
MLP	4,45 %	1,12 %	2,23 %	4,45 %	2,23 %	4,45 %	4,45 %	4,45 %	4,45 %	4,45 %	5,56 %	5,56 %	5,56 %
SVM	10,00 %	8,89 %	8,89 %	11,12 %	8,89 %	10,00 %	10,00 %	8,89 %	10,00 %	10,00 %	11,12 %	11,12 %	12,23 %
DMNN	2,23 %	3,34 %	5,56 %	13,34 %	17,78 %	24,45 %	24,45 %	24,45 %	25,56 %	24,45 %	25,56 %	26,67 %	32,23 %

Tabla A.8: M.Izq VS Lengua (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	5,56 %	5,56 %	5,56 %	8,89 %	8,89 %	6,67 %	6,67 %	6,67 %	6,67 %	6,67 %	7,78 %	7,78 %	14,45 %
MLP	5,56 %	6,67 %	6,67 %	5,56 %	6,67 %	12,23 %	8,89 %	10,00 %	6,67 %	6,67 %	8,89 %	10,00 %	8,89 %
SVM	6,67 %	6,67 %	5,56 %	7,78 %	7,78 %	7,78 %	7,78 %	7,78 %	6,67 %	6,67 %	6,67 %	6,67 %	10,00 %
DMNN	5,56 %	5,56 %	12,23 %	13,34 %	13,34 %	16,67 %	17,78 %	15,56 %	20,00 %	22,23 %	23,34 %	28,89 %	31,12 %

Tabla A.9: M.Izq VS Pie (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	13,34 %	12,23 %	8,89 %	6,67 %	5,56 %	6,67 %	4,45 %	6,67 %	5,56 %	5,56 %	5,56 %	5,56 %	6,67 %
MLP	7,78 %	7,78 %	8,89 %	7,78 %	6,67 %	8,89 %	7,78 %	3,33 %	7,78 %	7,78 %	7,78 %	2,23 %	5,56 %
SVM	7,78 %	4,45 %	4,45 %	4,45 %	4,45 %	4,45 %	3,33 %	6,67 %	6,67 %	6,67 %	6,67 %	6,67 %	6,67 %
DMNN	13,34 %	18,89 %	13,34 %	13,34 %	12,23 %	11,11 %	14,45 %	20,00 %	18,89 %	24,45 %	22,23 %	24,45 %	25,56 %

Tabla A.10: M.Der VS Lengua (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	3,34 %	4,45 %	4,45 %	6,67 %	5,56 %	5,56 %	5,56 %	5,56 %	5,56 %	4,45 %	4,45 %	4,45 %	4,45 %
MLP	4,45 %	3,34 %	2,23 %	4,45 %	8,89 %	6,67 %	6,67 %	7,78 %	7,78 %	6,67 %	6,67 %	7,78 %	5,56 %
SVM	4,45 %	4,45 %	4,45 %	5,56 %	5,56 %	5,56 %	5,56 %	5,56 %	5,56 %	4,45 %	4,45 %	5,56 %	5,56 %
DMNN	4,45 %	8,89 %	13,34 %	13,34 %	11,12 %	21,11 %	27,78 %	30,00 %	30,00 %	27,78 %	27,78 %	30,00 %	27,78 %

Tabla A.11: M.Der VS Pie (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	3,34 %	1,12 %	1,12 %	1,12 %	1,12 %	1,12 %	1,12 %	1,12 %	1,12 %	1,12 %	0,00 %	0,00 %	0,00 %
MLP	2,23 %	1,12 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %
SVM	2,23 %	1,12 %	0,00 %	0,00 %	1,12 %	2,23 %	2,23 %	2,23 %	2,23 %	2,23 %	2,23 %	2,23 %	2,23 %
DMNN	4,45 %	6,67 %	7,78 %	7,78 %	8,89 %	10,00 %	12,23 %	15,56 %	16,67 %	20,00 %	21,12 %	23,34 %	25,56 %

Tabla A.12: Lengua VS Pie (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	37,78 %	30,00 %	30,00 %	32,23 %	36,67 %	31,12 %	33,34 %	34,45 %	31,12 %	31,12 %	32,23 %	31,12 %	33,34 %
MLP	31,12 %	26,67 %	26,67 %	33,34 %	33,34 %	28,89 %	41,12 %	37,78 %	27,78 %	27,78 %	27,78 %	26,67 %	27,78 %
SVM	28,89 %	27,78 %	27,78 %	30,00 %	32,23 %	30,00 %	36,67 %	38,89 %	37,78 %	32,23 %	33,34 %	33,34 %	33,34 %
DMNN	33,34 %	34,45 %	37,78 %	36,67 %	37,78 %	38,89 %	40,00 %	41,12 %	42,23 %	37,78 %	38,89 %	36,67 %	35,56 %

A.1.3. Cuatro clases (Pair-Wise) - media cuadrática

Tabla A.13: M.Izq - M.Der VS Lengua - Pie (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	20,00 %	11,67 %	12,23 %	10,00 %	12,78 %	14,45 %	14,45 %	11,12 %	12,23 %	13,89 %	12,23 %	11,67 %	12,78 %
MLP	21,12 %	10,56 %	9,45 %	11,67 %	15,00 %	11,67 %	11,67 %	15,00 %	16,12 %	11,67 %	11,12 %	10,56 %	10,00 %
SVM	18,34 %	12,23 %	8,89 %	11,12 %	12,78 %	12,78 %	12,78 %	11,67 %	12,23 %	11,67 %	11,67 %	11,12 %	10,56 %
DMNN	22,23 %	15,00 %	16,67 %	22,23 %	24,45 %	28,34 %	22,23 %	22,23 %	20,56 %	25,56 %	22,78 %	22,78 %	22,23 %

Tabla A.14: M.Izq - Lengua VS M.Der - Pie (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	25,56 %	20,00 %	22,78 %	21,12 %	22,78 %	22,78 %	23,34 %	24,45 %	25,00 %	25,56 %	27,78 %	27,23 %	26,67 %
MLP	28,89 %	23,89 %	25,56 %	25,56 %	23,89 %	23,89 %	24,45 %	24,45 %	24,45 %	26,67 %	26,67 %	27,23 %	26,12 %
SVM	25,00 %	21,12 %	25,56 %	20,56 %	20,56 %	19,45 %	20,00 %	21,67 %	21,67 %	21,67 %	23,34 %	23,34 %	21,67 %
DMNN	28,89 %	23,89 %	27,23 %	30,00 %	26,67 %	31,12 %	30,56 %	33,89 %	28,34 %	31,12 %	33,34 %	33,89 %	32,23 %

Tabla A.15: M.Izq - Pie VS M.Der - Lengua (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	21,67 %	21,67 %	17,23 %	21,67 %	20,00 %	21,12 %	19,45 %	20,00 %	20,56 %	21,67 %	22,78 %	23,34 %	25,56 %
MLP	25,00 %	22,78 %	20,00 %	22,23 %	18,89 %	18,34 %	16,67 %	20,00 %	17,23 %	21,12 %	18,89 %	20,00 %	24,45 %
SVM	17,23 %	17,78 %	19,45 %	18,89 %	18,89 %	18,89 %	18,89 %	19,45 %	18,89 %	18,89 %	18,89 %	18,89 %	19,45 %
DMNN	21,67 %	21,12 %	21,67 %	23,89 %	26,67 %	28,34 %	26,67 %	29,45 %	31,12 %	30,56 %	32,23 %	33,34 %	35,00 %

A.1.4. Cuatro clases (Pair-Wise) - varianza

Tabla A.16: M.Izq - M.Der VS Lengua - Pie (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	20,56 %	14,45 %	15,00 %	13,89 %	19,45 %	20,56 %	20,00 %	19,45 %	15,56 %	15,56 %	16,67 %	17,78 %	18,34 %
MLP	21,12 %	12,78 %	13,89 %	11,67 %	13,34 %	18,34 %	14,45 %	18,34 %	21,12 %	18,89 %	15,00 %	13,89 %	13,34 %
SVM	18,89 %	13,89 %	16,67 %	15,00 %	18,89 %	18,34 %	17,78 %	18,34 %	18,34 %	16,67 %	15,00 %	15,00 %	15,00 %
DMNN	18,34 %	16,67 %	17,23 %	18,34 %	26,12 %	24,45 %	23,34 %	26,67 %	28,34 %	29,45 %	32,23 %	31,67 %	33,89 %

Tabla A.17: M.Izq - Lengua VS M.Der - Pie (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	25,56 %	21,67 %	25,56 %	25,56 %	26,12 %	28,89 %	30,56 %	30,56 %	30,56 %	31,12 %	33,34 %	34,45 %	32,78 %
MLP	25,56 %	22,23 %	25,00 %	26,67 %	27,78 %	26,12 %	31,67 %	29,45 %	23,34 %	27,23 %	27,23 %	26,12 %	26,12 %
SVM	28,89 %	25,00 %	27,78 %	27,78 %	28,34 %	28,89 %	28,34 %	29,45 %	30,00 %	30,00 %	29,45 %	30,00 %	29,45 %
DMNN	26,67 %	22,78 %	26,12 %	27,23 %	33,89 %	34,45 %	32,78 %	33,89 %	36,12 %	37,78 %	37,78 %	39,45 %	42,23 %

Tabla A.18: M.Izq - Pie VS M.Der - Lengua (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	20,00 %	20,56 %	16,67 %	21,67 %	23,34 %	23,34 %	22,78 %	25,00 %	26,67 %	27,23 %	27,23 %	26,67 %	27,78 %
MLP	18,34 %	21,12 %	20,00 %	20,56 %	16,67 %	19,45 %	18,89 %	22,78 %	20,56 %	24,45 %	23,34 %	23,34 %	24,45 %
SVM	19,45 %	21,12 %	18,89 %	19,45 %	21,67 %	22,23 %	22,22 %	22,22 %	24,45 %	22,23 %	22,23 %	22,78 %	23,89 %
DMNN	25,00 %	30,56 %	28,34 %	31,67 %	30,56 %	29,45 %	28,89 %	34,45 %	31,12 %	36,67 %	36,67 %	42,78 %	45,00 %

A.2. Sujeto de prueba: k6b

A.2.1. Dos clases - media cuadrática

Tabla A.19: M.Izq VS M.Der (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	36,67 %	41,67 %	28,34 %	38,34 %	35,00 %	33,34 %	38,34 %	36,67 %	35,00 %	35,00 %	35,00 %	36,67 %	36,67 %
MLP	35,00 %	36,67 %	33,34 %	35,00 %	38,34 %	35,00 %	38,34 %	36,67 %	35,00 %	36,67 %	31,67 %	33,34 %	33,34 %
SVM	36,67 %	36,67 %	38,34 %	38,34 %	38,34 %	31,67 %	30,00 %	35,00 %	35,00 %	35,00 %	35,00 %	35,00 %	36,67 %
DMNN	30,00 %	33,34 %	35,00 %	36,67 %	31,67 %	36,67 %	33,34 %	36,67 %	41,67 %	38,34 %	36,67 %	40,00 %	36,67 %

Tabla A.20: M.Izq VS Lengua (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	25,00 %	26,67 %	25,00 %	31,67 %	30,00 %	30,00 %	25,00 %	23,34 %	21,67 %	20,00 %	23,34 %	25,00 %	23,34 %
MLP	26,67 %	23,34 %	25,00 %	25,00 %	28,34 %	23,34 %	26,67 %	21,67 %	28,34 %	26,67 %	28,34 %	26,67 %	31,67 %
SVM	26,67 %	26,67 %	26,67 %	26,67 %	25,00 %	28,34 %	30,00 %	26,67 %	26,67 %	25,00 %	26,67 %	26,67 %	26,67 %
DMNN	33,34 %	33,34 %	33,34 %	36,67 %	35,00 %	45,00 %	46,67 %	50,00 %	50,00 %	48,34 %	46,67 %	43,34 %	41,67 %

Tabla A.21: M.Izq VS Pie (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	11,67%	10,00%	10,00%	10,00%	10,00%	11,67%	11,67%	10,00%	15,00%	15,00%	11,67%	11,67%	11,67%
MLP	10,00%	8,34%	10,00%	8,34%	8,34%	10,00%	11,67%	11,67%	11,67%	8,34%	10,00%	11,67%	11,67%
SVM	8,34%	11,67%	11,67%	11,67%	13,34%	13,34%	13,34%	13,34%	13,34%	11,67%	13,34%	13,34%	13,34%
DMNN	13,34%	16,67%	15,00%	13,34%	15,00%	13,34%	11,67%	16,67%	18,34%	21,67%	21,67%	20,00%	20,00%

Tabla A.22: M.Der VS Lengua (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	31,67%	28,34%	31,67%	36,67%	31,67%	31,67%	35,00%	33,34%	31,67%	28,34%	36,67%	36,67%	31,67%
MLP	28,34%	35,00%	26,67%	26,67%	33,34%	30,00%	36,67%	35,00%	26,67%	31,67%	31,67%	30,00%	33,34%
SVM	35,00%	35,00%	35,00%	35,00%	35,00%	35,00%	35,00%	31,67%	30,00%	33,34%	31,67%	31,67%	31,67%
DMNN	25,00%	26,67%	40,00%	38,34%	33,34%	31,67%	28,34%	25,00%	23,34%	25,00%	35,00%	40,00%	40,00%

Tabla A.23: M.Der VS Pie (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	20,00%	18,34%	23,34%	20,00%	18,34%	21,67%	21,67%	23,34%	20,00%	21,67%	21,67%	20,00%	21,67%
MLP	21,67%	18,34%	18,34%	15,00%	15,00%	21,67%	23,34%	28,34%	23,34%	20,00%	20,00%	18,34%	21,67%
SVM	15,00%	18,34%	20,00%	20,00%	18,34%	23,34%	21,67%	20,00%	23,34%	21,67%	21,67%	20,00%	20,00%
DMNN	21,67%	26,67%	25,00%	31,67%	26,67%	23,34%	30,00%	36,67%	36,67%	33,34%	35,00%	40,00%	36,67%

Tabla A.24: Lengua VS Pie (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	5,00%	8,34%	6,67%	8,34%	6,67%	5,00%	5,00%	5,00%	5,00%	3,34%	3,34%	3,34%	3,34%
MLP	13,34%	10,00%	6,67%	10,00%	10,00%	6,67%	6,67%	6,67%	6,67%	6,67%	5,00%	3,34%	3,34%
SVM	6,67%	6,67%	5,00%	6,67%	8,34%	5,00%	5,00%	5,00%	5,00%	5,00%	5,00%	5,00%	5,00%
DMNN	10,00%	13,34%	15,00%	15,00%	13,34%	10,00%	10,00%	10,00%	8,34%	10,00%	10,00%	10,00%	10,00%

A.2.2. Dos clases - varianza

Tabla A.25: M.Izq VS M.Der (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	35,00%	38,34%	26,67%	33,34%	33,34%	33,34%	40,00%	36,67%	36,67%	36,67%	38,34%	36,67%	36,67%
MLP	33,34%	36,67%	31,67%	41,67%	43,34%	26,67%	28,34%	40,00%	33,34%	40,00%	33,34%	30,00%	36,67%
SVM	35,00%	35,00%	36,67%	35,00%	33,34%	30,00%	33,34%	33,34%	31,67%	33,34%	35,00%	35,00%	35,00%
DMNN	38,34%	40,00%	40,00%	41,67%	40,00%	36,67%	38,34%	40,00%	36,67%	35,00%	33,34%	33,34%	35,00%

Tabla A.26: M.Izq VS Lengua (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	25,00%	28,34%	25,00%	33,34%	30,00%	30,00%	28,34%	28,34%	28,34%	30,00%	23,34%	25,00%	23,34%
MLP	26,67%	26,67%	23,34%	23,34%	26,67%	26,67%	23,34%	28,34%	30,00%	30,00%	30,00%	35,00%	31,67%
SVM	28,34%	30,00%	25,00%	30,00%	35,00%	36,67%	35,00%	36,67%	36,67%	35,00%	33,34%	33,34%	33,34%
DMNN	26,67%	30,00%	26,67%	31,67%	38,34%	43,34%	41,67%	41,67%	41,67%	35,00%	33,34%	33,34%	31,67%

Tabla A.27: M.Izq VS Pie (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	8,34%	11,67%	11,67%	11,67%	13,34%	13,34%	15,00%	15,00%	13,34%	15,00%	15,00%	15,00%	15,00%
MLP	8,34%	10,00%	6,67%	8,34%	8,34%	13,34%	11,67%	16,67%	13,34%	11,67%	10,00%	13,34%	13,34%
SVM	8,34%	11,67%	13,34%	13,34%	15,00%	16,67%	16,67%	16,67%	13,34%	13,34%	13,34%	13,34%	13,34%
DMNN	13,34%	16,67%	21,67%	13,34%	11,67%	13,34%	23,34%	26,67%	25,00%	23,34%	33,34%	40,00%	35,00%

Tabla A.28: M.Der VS Lengua (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	28,34%	36,67%	35,00%	43,34%	36,67%	36,67%	35,00%	30,00%	35,00%	36,67%	38,34%	40,00%	38,34%
MLP	26,67%	31,67%	26,67%	30,00%	28,34%	35,00%	33,34%	31,67%	30,00%	33,34%	35,00%	28,34%	31,67%
SVM	35,00%	40,00%	40,00%	38,34%	40,00%	40,00%	40,00%	36,67%	36,67%	35,00%	35,00%	35,00%	36,67%
DMNN	36,67%	31,67%	33,34%	35,00%	36,67%	38,34%	46,67%	41,67%	43,34%	40,00%	38,34%	38,34%	41,67%

Tabla A.29: M.Der VS Pie (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	15,00%	23,34%	25,00%	23,34%	20,00%	20,00%	21,67%	23,34%	18,34%	21,67%	26,67%	23,34%	23,34%
MLP	16,67%	23,34%	25,00%	21,67%	20,00%	21,67%	23,34%	23,34%	21,67%	21,67%	21,67%	23,34%	21,67%
SVM	13,34%	23,34%	26,67%	23,34%	21,67%	23,34%	23,34%	23,34%	23,34%	21,67%	25,00%	21,67%	25,00%
DMNN	16,67%	15,00%	15,00%	18,34%	20,00%	23,34%	26,67%	38,34%	33,34%	35,00%	33,34%	31,67%	33,34%

Tabla A.30: Lengua VS Pie (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	8,34%	8,34%	6,67%	6,67%	6,67%	6,67%	8,34%	10,00%	10,00%	10,00%	11,67%	11,67%	10,00%
MLP	13,34%	10,00%	10,00%	8,34%	6,67%	6,67%	6,67%	6,67%	6,67%	8,34%	5,00%	5,00%	8,34%
SVM	6,67%	6,67%	6,67%	8,34%	6,67%	6,67%	5,00%	6,67%	6,67%	6,67%	6,67%	6,67%	5,00%
DMNN	8,34%	13,34%	13,34%	13,34%	13,34%	11,67%	13,34%	13,34%	13,34%	13,34%	18,34%	18,34%	18,34%

A.2.3. Cuatro clases (Pair-Wise) - media cuadrática

Tabla A.31: M.Izq - M.Der VS Lengua - Pie (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	38,34 %	32,50 %	34,17 %	32,50 %	34,17 %	30,00 %	32,50 %	30,00 %	30,84 %	30,00 %	30,84 %	31,67 %	27,50 %
MLP	36,67 %	33,34 %	38,34 %	35,00 %	39,17 %	31,67 %	30,84 %	34,17 %	30,84 %	32,50 %	32,50 %	35,00 %	35,00 %
SVM	35,84 %	29,17 %	30,84 %	31,67 %	31,67 %	30,00 %	30,84 %	29,17 %	27,50 %	27,50 %	26,67 %	28,34 %	29,17 %
DMNN	37,50 %	36,67 %	38,34 %	37,50 %	35,00 %	34,17 %	37,50 %	38,34 %	42,50 %	35,84 %	40,00 %	40,84 %	41,67 %

Tabla A.32: M.Izq - Lengua VS M.Der - Pie (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	38,34 %	32,50 %	34,17 %	32,50 %	34,17 %	30,00 %	32,50 %	30,00 %	30,84 %	30,00 %	30,84 %	31,67 %	27,50 %
MLP	36,67 %	33,34 %	38,34 %	35,00 %	39,17 %	31,67 %	30,84 %	34,17 %	30,84 %	32,50 %	32,50 %	35,00 %	35,00 %
SVM	35,84 %	29,17 %	30,84 %	31,67 %	31,67 %	30,00 %	30,84 %	29,17 %	27,50 %	27,50 %	26,67 %	28,34 %	29,17 %
DMNN	37,50 %	36,67 %	38,34 %	37,50 %	35,00 %	34,17 %	37,50 %	38,34 %	42,50 %	35,84 %	40,00 %	40,84 %	41,67 %

Tabla A.33: M.Izq - Pie VS M.Der - Lengua (RMS)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	29,17 %	35,84 %	27,50 %	31,67 %	30,84 %	31,67 %	33,34 %	32,50 %	33,34 %	33,34 %	30,00 %	30,84 %	34,17 %
MLP	33,34 %	35,84 %	30,84 %	32,50 %	33,34 %	27,50 %	32,50 %	33,34 %	35,00 %	30,84 %	29,17 %	27,50 %	30,00 %
SVM	31,67 %	33,34 %	30,84 %	30,84 %	32,50 %	30,84 %	30,84 %	32,50 %	30,84 %	29,17 %	29,17 %	28,34 %	28,34 %
DMNN	33,34 %	32,50 %	29,17 %	29,17 %	29,17 %	25,00 %	30,84 %	29,17 %	32,50 %	30,84 %	29,17 %	34,17 %	32,50 %

A.2.4. Cuatro clases (Pair-Wise) - varianza

Tabla A.34: M.Izq - M.Der VS Lengua - Pie (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	38,34 %	32,50 %	34,17 %	35,00 %	33,34 %	26,67 %	29,17 %	27,50 %	35,84 %	30,00 %	30,00 %	30,84 %	30,84 %
MLP	37,50 %	36,67 %	33,34 %	32,50 %	30,00 %	32,84 %	28,34 %	38,34 %	35,00 %	30,00 %	33,34 %	34,17 %	35,00 %
SVM	37,50 %	30,00 %	30,00 %	35,00 %	33,34 %	31,67 %	31,67 %	30,00 %	29,17 %	31,67 %	32,50 %	33,34 %	31,67 %
DMNN	41,67 %	31,67 %	33,34 %	41,67 %	39,17 %	40,84 %	41,67 %	35,00 %	33,34 %	32,50 %	29,17 %	31,67 %	37,50 %

Tabla A.35: M.Izq - Lengua VS M.Der - Pie (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	25,84 %	21,67 %	21,67 %	24,17 %	25,00 %	25,84 %	27,50 %	26,67 %	25,00 %	26,67 %	26,67 %	25,84 %	25,84 %
MLP	20,84 %	20,84 %	35,00 %	24,17 %	28,34 %	38,34 %	30,84 %	36,67 %	31,67 %	25,00 %	28,34 %	32,50 %	26,67 %
SVM	22,50 %	23,34 %	21,67 %	22,50 %	21,67 %	21,67 %	21,67 %	20,84 %	21,67 %	21,67 %	21,67 %	21,67 %	21,67 %
DMNN	32,50 %	30,84 %	28,34 %	25,84 %	24,17 %	27,50 %	25,84 %	26,67 %	26,67 %	30,84 %	36,67 %	38,34 %	40,00 %

Tabla A.36: M.Izq - Pie VS M.Der - Lengua (VAR)

	Error												
Rasgos	2	3	4	5	6	7	8	9	10	11	12	13	14
Knn	29,17 %	34,17 %	30,00 %	33,34 %	33,34 %	34,17 %	33,34 %	32,50 %	32,50 %	32,50 %	32,50 %	34,17 %	32,50 %
MLP	32,50 %	33,34 %	25,00 %	29,17 %	30,84 %	31,67 %	39,17 %	33,34 %	34,17 %	31,67 %	33,34 %	30,00 %	29,17 %
SVM	33,34 %	33,34 %	30,84 %	30,84 %	30,84 %	30,00 %	29,17 %	29,17 %	30,84 %	28,34 %	31,67 %	34,17 %	30,00 %
DMNN	39,17 %	34,17 %	33,34 %	37,50 %	40,00 %	37,50 %	39,17 %	42,50 %	33,34 %	35,00 %	31,67 %	36,67 %	35,00 %

Apéndice B

Codigo de programas implementados

B.1. Calculo del filtro espacial CSP

```
1       clf;clc;clear;close all;
2       load('A01.mat')
3       fm=250;
4       nt=length(class_train);
5       nin=1;
6
7       mi=class_train;
8       PB=load('Butter7_30.mat');
9       ncl=0;
10      ncr=0;
11      ncf=0;
12      nct=0;
13      lsum=0;
14      rsum=0;
15      fsum=0;
16      tsum=0;
17
18      ncl1=0;
19      ncr1=0;
20      ncl2=0;
21      ncr2=0;
22      ncl3=0;
23      ncr3=0;
24      lsum1=0;
25      rsum1=0;
26      lsum2=0;
27      rsum2=0;
28      lsum3=0;
29      rsum3=0;
30
31      for ntrial=nin:nin+nt-1
32          trial=ntrial;
33          ttr=trial-nin+1;
```

```
34          for nch=1:14
35              ch=nch;
36              if mi(trial)==1
37                  tmov='Izq';
38                  color='blue';
39                  marc='+';
40              end
41              if mi(trial)==2
42                  tmov='Der';
43                  color='red';
44                  marc='*';
45              end
46              if mi(trial)==4
47                  tmov='food';
48                  color='black';
49                  marc='+';
50              end
51              if mi(trial)==3
52                  tmov='tongue';
53                  color='green';
54                  marc='*';
55              end
56              eeg=squeeze(seeg_train(trial,:,ch)');
57              mimagery=eeg;
58              mimagery=filter(PB.Num,PB.Den,mimagery);
59              mcov(nch,:)= mimagery;
60              mch(ttr,nch,:)= mimagery;
61          end
62          disp(tmov)
63          if mi(trial)==1
64              ncl=ncl+1;
65              nco=(mcov*mcov')/trace(mcov*mcov');
66              lsum=lsum+nco;
67          end
68          if mi(trial)==2
69              ncr=ncr+1;
70              nco=(mcov*mcov')/trace(mcov*mcov');
71              rsum=rsum+nco;
72          end
73          if mi(trial)==3
74              ncf=ncf+1;
75              nco=(mcov*mcov')/trace(mcov*mcov');
76              fsum=fsum+nco;
77          end
78          if mi(trial)==4
79              nct=nct+1;
80              nco=(mcov*mcov')/trace(mcov*mcov');
81              tsum=tsum+nco;
82          end
83          if mi(trial)==1 || mi(trial)==2
84              ncl1=ncl1+1;
85              nco=(mcov*mcov')/trace(mcov*mcov');
86              lsum1=lsum1+nco;
87          end
88          if mi(trial)==3 || mi(trial)==4
89              ncr1=ncr1+1;
```

```
90              nco=(mcov*mcov')/trace(mcov*mcov');
91              rsum1=rsum1+nco;
92          end
93          if mi(trial)==1 || mi(trial)==4
94              ncl2=ncl2+1;
95              nco=(mcov*mcov')/trace(mcov*mcov');
96              lsum2=lsum2+nco;
97          end
98          if mi(trial)==2 || mi(trial)==3
99              ncr2=ncr2+1;
100             nco=(mcov*mcov')/trace(mcov*mcov');
101             rsum2=rsum2+nco;
102         end
103         if mi(trial)==1 || mi(trial)==3
104             ncl3=ncl3+1;
105             nco=(mcov*mcov')/trace(mcov*mcov');
106             lsum3=lsum3+nco;
107         end
108         if mi(trial)==2 || mi(trial)==4
109             ncr3=ncr3+1;
110             nco=(mcov*mcov')/trace(mcov*mcov');
111             rsum3=rsum3+nco;
112         end
113     end
114
115     covc1=lsum/ncl;
116     covc2=rsum/ncr;
117     covc3=fsum/ncf;
118     covc4=tsum/nct;
119
120     covc11=lsum1/ncl1;
121     covc21=rsum1/ncr1;
122     covc12=lsum2/ncl2;
123     covc22=rsum2/ncr2;
124     covc13=lsum3/ncl3;
125     covc23=rsum3/ncr3;
126
127     [SFLVSR] = CSP(covc1,covc2);
128     [SFLVSF] = CSP(covc1,covc3);
129     [SFLVST] = CSP(covc1,covc4);
130     [SFRVSF] = CSP(covc2,covc3);
131     [SFRVST] = CSP(covc2,covc4);
132     [SFFVST] = CSP(covc3,covc4);
133
134     [SFLRVSFT] = CSP(covc11,covc21);
135     [SFLTVSRF] = CSP(covc12,covc22);
136     [SFLFVSRT] = CSP(covc13,covc23);
137
138     [PWA_SFLRVSFT] = CSP((covc1+covc2)/2,(covc3+covc4)/2);
139     [PWA_SFLFVSRT] = CSP((covc1+covc3)/2,(covc2+covc4)/2);
140     [PWA_SFLTVSRF] = CSP((covc1+covc4)/2,(covc2+covc3)/2);
141
142     [OVR_SFLVSALL] = CSP(covc1,(covc2+covc3+covc4)/3);
143     [OVR_SFRVSALL] = CSP(covc2,(covc1+covc3+covc4)/3);
144     [OVR_SFFVSALL] = CSP(covc3,(covc2+covc1+covc4)/3);
145     [OVR_SFTVSALL] = CSP(covc4,(covc2+covc3+covc1)/3);
```

```
146
147     save('SFG14_A01.mat','SFLVSR','SFLVSF','SFLVST','SFRVSF','SFRVST','
            SFFVST','SFLRVSFT','SFLFVSRT','SFLTVSRF','PWA_SFLRVSFT','
            PWA_SFLFVSRT','PWA_SFLTVSRF','OVR_SFLVSALL','OVR_SFRVSALL','
            OVR_SFFVSALL','OVR_SFTVSALL');
```

B.2. Método CSP

```
1   function [result] = CSP(C1,C2)
2   Rsum=0;
3       R ={C1 C2};
4       Rsum=C1+C2;
5       [EVecsum,EValsum] = eig(Rsum);
6       [EValsum,ind] = sort(diag(EValsum),'descend');
7       EVecsum = EVecsum(:,ind);
8       W = sqrt(inv(diag(EValsum))) * EVecsum';
9       for k = 1:nargin
10          S{k} = W * R{k} * W';
11      end
12      [B,D] = eig(S{1},S{2});
13      [D,ind]=sort(diag(D));
14      B=B(:,ind);
15      result = B'*W;
16  end
```

B.3. Extracción rasgos característicos

```
1   clf;clc;clear;close all;
2   load('A01.mat')
3   fm=250;
4   nt=length(class_train);
5   nin=1;
6   nf=14;
7   PB=load('Butter7_30.mat');
8   load SFG14_A01.mat
9   mcov = zeros(14,1000);
10  mcov_TEST = zeros(14,1000);
11  dataSetLRVSFT=zeros(nt,nf);
12  dataSetLVSR=zeros(nt,nf);
13  dataSetFVST=zeros(nt,nf);
14  dataSetLFVSRT=zeros(nt,nf);
15  dataSetLVSF=zeros(nt,nf);
16  dataSetRVST=zeros(nt,nf);
17  dataSetLTVSRF=zeros(nt,nf);
18  dataSetLVST=zeros(nt,nf);
19  dataSetRVSF=zeros(nt,nf);
20  dataSetLRVSFT_VAR=zeros(nt,nf);
21  dataSetLVSR_VAR=zeros(nt,nf);
22  dataSetFVST_VAR=zeros(nt,nf);
23  dataSetLFVSRT_VAR=zeros(nt,nf);
```

```
24        dataSetLVSF_VAR=zeros(nt,nf);
25        dataSetRVST_VAR=zeros(nt,nf);
26        dataSetLTVSRF_VAR=zeros(nt,nf);
27        dataSetLVST_VAR=zeros(nt,nf);
28        dataSetRVSF_VAR=zeros(nt,nf);
29        dataSetLRVSFT_AVG=zeros(nt,nf);
30        dataSetLVSR_AVG=zeros(nt,nf);
31        dataSetFVST_AVG=zeros(nt,nf);
32        dataSetLFVSRT_AVG=zeros(nt,nf);
33        dataSetLVSF_AVG=zeros(nt,nf);
34        dataSetRVST_AVG=zeros(nt,nf);
35        dataSetLTVSRF_AVG=zeros(nt,nf);
36        dataSetLVST_AVG=zeros(nt,nf);
37        dataSetRVSF_AVG=zeros(nt,nf);
38
39        dataSetLRVSFT_TEST=zeros(nt,nf);
40        dataSetLVSR_TEST=zeros(nt,nf);
41        dataSetFVST_TEST=zeros(nt,nf);
42        dataSetLFVSRT_TEST=zeros(nt,nf);
43        dataSetLVSF_TEST=zeros(nt,nf);
44        dataSetRVST_TEST=zeros(nt,nf);
45        dataSetLTVSRF_TEST=zeros(nt,nf);
46        dataSetLVST_TEST=zeros(nt,nf);
47        dataSetRVSF_TEST=zeros(nt,nf);
48        dataSetLRVSFT_TEST_VAR=zeros(nt,nf);
49        dataSetLVSR_TEST_VAR=zeros(nt,nf);
50        dataSetFVST_TEST_VAR=zeros(nt,nf);
51        dataSetLFVSRT_TEST_VAR=zeros(nt,nf);
52        dataSetLVSF_TEST_VAR=zeros(nt,nf);
53        dataSetRVST_TEST_VAR=zeros(nt,nf);
54        dataSetLTVSRF_TEST_VAR=zeros(nt,nf);
55        dataSetLVST_TEST_VAR=zeros(nt,nf);
56        dataSetRVSF_TEST_VAR=zeros(nt,nf);
57        dataSetLRVSFT_TEST_AVG=zeros(nt,nf);
58        dataSetLVSR_TEST_AVG=zeros(nt,nf);
59        dataSetFVST_TEST_AVG=zeros(nt,nf);
60        dataSetLFVSRT_TEST_AVG=zeros(nt,nf);
61        dataSetLVSF_TEST_AVG=zeros(nt,nf);
62        dataSetRVST_TEST_AVG=zeros(nt,nf);
63        dataSetLTVSRF_TEST_AVG=zeros(nt,nf);
64        dataSetLVST_TEST_AVG=zeros(nt,nf);
65        dataSetRVSF_TEST_AVG=zeros(nt,nf);
66        for trial=nin:nt
67        disp(trial)
68            ttr=trial-nin+1;
69            for nch=1:14
70                eeg=squeeze(seeg_train(trial,:,nch)');
71                mimagery=eeg;
72                mimagery=filter(PB.Num,PB.Den,mimagery);
73                mcov(nch,:)= mimagery;
74                eeg=squeeze(seeg_test(trial,:,nch)');
75                mimagery=eeg;
76                mimagery=filter(PB.Num,PB.Den,mimagery);
77                mcov_TEST(nch,:)= mimagery;
78            end
79            %
```

```
80          %%
81          xxx=spatFilt(mcov,SFLRVSFT,nf);
82          xxx_TEST=spatFilt(mcov_TEST,SFLRVSFT,nf);
83          for ll=1:nf
84              dataSetLRVSFT(trial,ll) = rms(squeeze(xxx(ll,:)))^2;
85              dataSetLRVSFT_TEST(trial,ll) = rms(squeeze(xxx_TEST(ll,:)))
                    ^2;
86              dataSetLRVSFT_VAR(trial,ll) = var(squeeze(xxx(ll,:)))^2;
87              dataSetLRVSFT_TEST_VAR(trial,ll) = var(squeeze(xxx_TEST(ll
                    ,:)))^2;
88              dataSetLRVSFT_AVG(trial,ll) = mean(squeeze(xxx(ll,:)))^2;
89              dataSetLRVSFT_TEST_AVG(trial,ll) = mean(squeeze(xxx_TEST(ll
                    ,:)))^2;
90          end
91          xxx=spatFilt(mcov,SFLVSR,nf);
92          xxx_TEST=spatFilt(mcov_TEST,SFLVSR,nf);
93          for ll=1:nf
94              dataSetLVSR(trial,ll) = rms(squeeze(xxx(ll,:)))^2;
95              dataSetLVSR_TEST(trial,ll) = rms(squeeze(xxx_TEST(ll,:)))^2;
96              dataSetLVSR_VAR(trial,ll) = var(squeeze(xxx(ll,:)))^2;
97              dataSetLVSR_TEST_VAR(trial,ll) = var(squeeze(xxx_TEST(ll,:))
                    )^2;
98              dataSetLVSR_AVG(trial,ll) = mean(squeeze(xxx(ll,:)))^2;
99              dataSetLVSR_TEST_AVG(trial,ll) = mean(squeeze(xxx_TEST(ll,:)
                    ))^2;
100         end
101         xxx=spatFilt(mcov,SFFVST,nf);
102         xxx_TEST=spatFilt(mcov_TEST,SFFVST,nf);
103         for ll=1:nf
104             dataSetFVST(trial,ll) = rms(squeeze(xxx(ll,:)))^2;
105             dataSetFVST_TEST(trial,ll) = rms(squeeze(xxx_TEST(ll,:)))^2;
106             dataSetFVST_VAR(trial,ll) = var(squeeze(xxx(ll,:)))^2;
107             dataSetFVST_TEST_VAR(trial,ll) = var(squeeze(xxx_TEST(ll,:))
                    )^2;
108             dataSetFVST_AVG(trial,ll) = mean(squeeze(xxx(ll,:)))^2;
109             dataSetFVST_TEST_AVG(trial,ll) = mean(squeeze(xxx_TEST(ll,:)
                    ))^2;
110         end
111         %
112         %%
113         xxx=spatFilt(mcov,SFLFVSRT,nf);
114         xxx_TEST=spatFilt(mcov_TEST,SFLFVSRT,nf);
115         for ll=1:nf
116             dataSetLFVSRT(trial,ll) = rms(squeeze(xxx(ll,:)))^2;
117             dataSetLFVSRT_TEST(trial,ll) = rms(squeeze(xxx_TEST(ll,:)))
                    ^2;
118             dataSetLFVSRT_VAR(trial,ll) = var(squeeze(xxx(ll,:)))^2;
119             dataSetLFVSRT_TEST_VAR(trial,ll) = var(squeeze(xxx_TEST(ll
                    ,:)))^2;
120             dataSetLFVSRT_AVG(trial,ll) = mean(squeeze(xxx(ll,:)))^2;
121             dataSetLFVSRT_TEST_AVG(trial,ll) = mean(squeeze(xxx_TEST(ll
                    ,:)))^2;
122         end
123         xxx=spatFilt(mcov,SFLVSF,nf);
124         xxx_TEST=spatFilt(mcov_TEST,SFLVSF,nf);
125         for ll=1:nf
```

```
126                 dataSetLVSF(trial,ll) = rms(squeeze(xxx(ll,:)))^2;
127                 dataSetLVSF_TEST(trial,ll) = rms(squeeze(xxx_TEST(ll,:)))^2;
128                 dataSetLVSF_VAR(trial,ll) = var(squeeze(xxx(ll,:)))^2;
129                 dataSetLVSF_TEST_VAR(trial,ll) = var(squeeze(xxx_TEST(ll,:))
                        )^2;
130                 dataSetLVSF_AVG(trial,ll) = mean(squeeze(xxx(ll,:)))^2;
131                 dataSetLVSF_TEST_AVG(trial,ll) = mean(squeeze(xxx_TEST(ll,:)
                        ))^2;
132             end
133             xxx=spatFilt(mcov,SFRVST,nf);
134             xxx_TEST=spatFilt(mcov_TEST,SFRVST,nf);
135             for ll=1:nf
136                 dataSetRVST(trial,ll) = rms(squeeze(xxx(ll,:)))^2;
137                 dataSetRVST_TEST(trial,ll) = rms(squeeze(xxx_TEST(ll,:)))^2;
138                 dataSetRVST_VAR(trial,ll) = var(squeeze(xxx(ll,:)))^2;
139                 dataSetRVST_TEST_VAR(trial,ll) = var(squeeze(xxx_TEST(ll,:))
                        )^2;
140                 dataSetRVST_AVG(trial,ll) = mean(squeeze(xxx(ll,:)))^2;
141                 dataSetRVST_TEST_AVG(trial,ll) = mean(squeeze(xxx_TEST(ll,:)
                        ))^2;
142             end
143             %
144             %%
145             xxx=spatFilt(mcov,SFLTVSRF,nf);
146             xxx_TEST=spatFilt(mcov_TEST,SFLTVSRF,nf);
147             for ll=1:nf
148                 dataSetLTVSRF(trial,ll) = rms(squeeze(xxx(ll,:)))^2;
149                 dataSetLTVSRF_TEST(trial,ll) = rms(squeeze(xxx_TEST(ll,:)))
                        ^2;
150                 dataSetLTVSRF_VAR(trial,ll) = var(squeeze(xxx(ll,:)))^2;
151                 dataSetLTVSRF_TEST_VAR(trial,ll) = var(squeeze(xxx_TEST(ll
                        ,:)))^2;
152                 dataSetLTVSRF_AVG(trial,ll) = mean(squeeze(xxx(ll,:)))^2;
153                 dataSetLTVSRF_TEST_AVG(trial,ll) = mean(squeeze(xxx_TEST(ll
                        ,:)))^2;
154             end
155             xxx=spatFilt(mcov,SFLVST,nf);
156             xxx_TEST=spatFilt(mcov_TEST,SFLVST,nf);
157             for ll=1:nf
158                 dataSetLVST(trial,ll) = rms(squeeze(xxx(ll,:)))^2;
159                 dataSetLVST_TEST(trial,ll) = rms(squeeze(xxx_TEST(ll,:)))^2;
160                 dataSetLVST_VAR(trial,ll) = var(squeeze(xxx(ll,:)))^2;
161                 dataSetLVST_TEST_VAR(trial,ll) = var(squeeze(xxx_TEST(ll,:))
                        )^2;
162             dataSetLVST_AVG(trial,ll) = mean(squeeze(xxx(ll,:)))^2;
163                 dataSetLVST_TEST_AVG(trial,ll) = mean(squeeze(xxx_TEST(ll,:)
                        ))^2;
164             end
165             xxx=spatFilt(mcov,SFRVSF,nf);
166             xxx_TEST=spatFilt(mcov_TEST,SFRVSF,nf);
167             for ll=1:nf
168                 dataSetRVSF(trial,ll) = rms(squeeze(xxx(ll,:)))^2;
169                 dataSetRVSF_TEST(trial,ll) = rms(squeeze(xxx_TEST(ll,:)))^2;
170                 dataSetRVSF_VAR(trial,ll) = var(squeeze(xxx(ll,:)))^2;
171                 dataSetRVSF_TEST_VAR(trial,ll) = var(squeeze(xxx_TEST(ll,:))
                        )^2;
```

```
172                 dataSetRVSF_AVG(trial,ll) = mean(squeeze(xxx(ll,:)))^2;
173                 dataSetRVSF_TEST_AVG(trial,ll) = mean(squeeze(xxx_TEST(ll,:)
                        ))^2;
174             end
175             %
176         end
177         %%
178         BCI=horzcat(dataSetLRVSFT,dataSetLVSR,dataSetFVST);
179         minimo=min(min(BCI));
180         maximo=max(max(BCI));
181         save('BCI_TRAIN_MAXMIN_LRVSFT.mat','minimo','maximo')
182         BCI=(BCI-minimo)/(maximo-minimo);
183         BCI_TRAIN=horzcat(class_train',BCI);
184         BCI_TRAIN_TABLA=table(BCI_TRAIN);
185         writetable(BCI_TRAIN_TABLA,'BCI_TRAIN_LRVSFT.csv','Delimiter',',');
186
187         BCI=horzcat(dataSetLRVSFT_TEST,dataSetLVSR_TEST,dataSetFVST_TEST);
188         BCI=(BCI-minimo)/(maximo-minimo);
189         BCI_TEST=horzcat(class_test(1:length(class_train))',BCI);
190         BCI_TEST_TABLA=table(BCI_TEST);
191         writetable(BCI_TEST_TABLA,'BCI_TEST_LRVSFT.csv','Delimiter',',');
192
193         BCI=horzcat(dataSetLRVSFT_VAR,dataSetLVSR_VAR,dataSetFVST_VAR);
194         minimo=min(min(BCI));
195         maximo=max(max(BCI));
196         save('BCI_TRAIN_MAXMIN_LRVSFT_VAR.mat','minimo','maximo')
197         BCI=(BCI-minimo)/(maximo-minimo);
198         BCI_TRAIN=horzcat(class_train',BCI);
199         BCI_TRAIN_TABLA=table(BCI_TRAIN);
200         writetable(BCI_TRAIN_TABLA,'BCI_TRAIN_LRVSFT_VAR.csv','Delimiter',',
                ');
201
202         BCI=horzcat(dataSetLRVSFT_TEST_VAR,dataSetLVSR_TEST_VAR,
                dataSetFVST_TEST_VAR);
203         BCI=(BCI-minimo)/(maximo-minimo);
204         BCI_TEST=horzcat(class_test(1:length(class_train))',BCI);
205         BCI_TEST_TABLA=table(BCI_TEST);
206         writetable(BCI_TEST_TABLA,'BCI_TEST_LRVSFT_VAR.csv','Delimiter',',')
                ;
207
208         BCI=horzcat(dataSetLRVSFT_AVG,dataSetLVSR_AVG,dataSetFVST_AVG);
209         minimo=min(min(BCI));
210         maximo=max(max(BCI));
211         save('BCI_TRAIN_MAXMIN_LRVSFT_AVG.mat','minimo','maximo')
212         BCI=(BCI-minimo)/(maximo-minimo);
213         BCI_TRAIN=horzcat(class_train',BCI);
214         BCI_TRAIN_TABLA=table(BCI_TRAIN);
215         writetable(BCI_TRAIN_TABLA,'BCI_TRAIN_LRVSFT_AVG.csv','Delimiter',',
                ');
216
217         BCI=horzcat(dataSetLRVSFT_TEST_AVG,dataSetLVSR_TEST_AVG,
                dataSetFVST_TEST_AVG);
218         BCI=(BCI-minimo)/(maximo-minimo);
219         BCI_TEST=horzcat(class_test(1:length(class_train))',BCI);
220         BCI_TEST_TABLA=table(BCI_TEST);
```

```
221         writetable(BCI_TEST_TABLA, 'BCI_TEST_LRVSFT_AVG.csv','Delimiter',',')
                ;
222         %
223         %%
224         BCI=horzcat(dataSetLFVSRT,dataSetLVSF,dataSetRVST);
225         minimo=min(min(BCI));
226         maximo=max(max(BCI));
227         save('BCI_TRAIN_MAXMIN_LFVSRT.mat','minimo','maximo')
228         BCI=(BCI-minimo)/(maximo-minimo);
229         BCI_TRAIN=horzcat(class_train',BCI);
230         BCI_TRAIN_TABLA=table(BCI_TRAIN);
231         writetable(BCI_TRAIN_TABLA, 'BCI_TRAIN_LFVSRT.csv','Delimiter',',');
232
233         BCI=horzcat(dataSetLFVSRT_TEST,dataSetLVSF_TEST,dataSetRVST_TEST);
234         BCI=(BCI-minimo)/(maximo-minimo);
235         BCI_TEST=horzcat(class_test(1:length(class_train))',BCI);
236         BCI_TEST_TABLA=table(BCI_TEST);
237         writetable(BCI_TEST_TABLA, 'BCI_TEST_LFVSRT.csv','Delimiter',',');
238
239         BCI=horzcat(dataSetLFVSRT_VAR,dataSetLVSF_VAR,dataSetRVST_VAR);
240         minimo=min(min(BCI));
241         maximo=max(max(BCI));
242         save('BCI_TRAIN_MAXMIN_LFVSRT_VAR.mat','minimo','maximo')
243         BCI=(BCI-minimo)/(maximo-minimo);
244         BCI_TRAIN=horzcat(class_train',BCI);
245         BCI_TRAIN_TABLA=table(BCI_TRAIN);
246         writetable(BCI_TRAIN_TABLA, 'BCI_TRAIN_LFVSRT_VAR.csv','Delimiter',',
                ');
247
248         BCI=horzcat(dataSetLFVSRT_TEST_VAR,dataSetLVSF_TEST_VAR,
                dataSetRVST_TEST_VAR);
249         BCI=(BCI-minimo)/(maximo-minimo);
250         BCI_TEST=horzcat(class_test(1:length(class_train))',BCI);
251         BCI_TEST_TABLA=table(BCI_TEST);
252         writetable(BCI_TEST_TABLA, 'BCI_TEST_LFVSRT_VAR.csv','Delimiter',',')
                ;
253
254         BCI=horzcat(dataSetLFVSRT_AVG,dataSetLVSF_AVG,dataSetRVST_AVG);
255         minimo=min(min(BCI));
256         maximo=max(max(BCI));
257         save('BCI_TRAIN_MAXMIN_LFVSRT_AVG.mat','minimo','maximo')
258         BCI=(BCI-minimo)/(maximo-minimo);
259         BCI_TRAIN=horzcat(class_train',BCI);
260         BCI_TRAIN_TABLA=table(BCI_TRAIN);
261         writetable(BCI_TRAIN_TABLA, 'BCI_TRAIN_LFVSRT_AVG.csv','Delimiter',',
                ');
262
263         BCI=horzcat(dataSetLFVSRT_TEST_AVG,dataSetLVSF_TEST_AVG,
                dataSetRVST_TEST_AVG);
264         BCI=(BCI-minimo)/(maximo-minimo);
265         BCI_TEST=horzcat(class_test(1:length(class_train))',BCI);
266         BCI_TEST_TABLA=table(BCI_TEST);
267         writetable(BCI_TEST_TABLA, 'BCI_TEST_LFVSRT_AVG.csv','Delimiter',',')
                ;
268         %
269         %%
```

```
270 BCI=horzcat(dataSetLTVSRF,dataSetLVST,dataSetRVSF);
271 minimo=min(min(BCI));
272 maximo=max(max(BCI));
273 save('BCI_TRAIN_MAXMIN_LTVSRF.mat','minimo','maximo')
274 BCI=(BCI-minimo)/(maximo-minimo);
275 BCI_TRAIN=horzcat(class_train',BCI);
276 BCI_TRAIN_TABLA=table(BCI_TRAIN);
277 writetable(BCI_TRAIN_TABLA,'BCI_TRAIN_LTVSRF.csv','Delimiter',',');
278
279 BCI=horzcat(dataSetLTVSRF_TEST,dataSetLVST_TEST,dataSetRVSF_TEST);
280 BCI=(BCI-minimo)/(maximo-minimo);
281 BCI_TEST=horzcat(class_test(1:length(class_train))',BCI);
282 BCI_TEST_TABLA=table(BCI_TEST);
283 writetable(BCI_TEST_TABLA,'BCI_TEST_LTVSRF.csv','Delimiter',',');
284
285 BCI=horzcat(dataSetLTVSRF_VAR,dataSetLVST_VAR,dataSetRVSF_VAR);
286 minimo=min(min(BCI));
287 maximo=max(max(BCI));
288 save('BCI_TRAIN_MAXMIN_LTVSRF_VAR.mat','minimo','maximo')
289 BCI=(BCI-minimo)/(maximo-minimo);
290 BCI_TRAIN=horzcat(class_train',BCI);
291 BCI_TRAIN_TABLA=table(BCI_TRAIN);
292 writetable(BCI_TRAIN_TABLA,'BCI_TRAIN_LTVSRF_VAR.csv','Delimiter',',
      ');
293
294 BCI=horzcat(dataSetLTVSRF_TEST_VAR,dataSetLVST_TEST_VAR,
      dataSetRVSF_TEST_VAR);
295 BCI=(BCI-minimo)/(maximo-minimo);
296 BCI_TEST=horzcat(class_test(1:length(class_train))',BCI);
297 BCI_TEST_TABLA=table(BCI_TEST);
298 writetable(BCI_TEST_TABLA,'BCI_TEST_LTVSRF_VAR.csv','Delimiter',',')
      ;
299
300 BCI=horzcat(dataSetLTVSRF_AVG,dataSetLVST_AVG,dataSetRVSF_AVG);
301 minimo=min(min(BCI));
302 maximo=max(max(BCI));
303 save('BCI_TRAIN_MAXMIN_LTVSRF_AVG.mat','minimo','maximo')
304 BCI=(BCI-minimo)/(maximo-minimo);
305 BCI_TRAIN=horzcat(class_train',BCI);
306 BCI_TRAIN_TABLA=table(BCI_TRAIN);
307 writetable(BCI_TRAIN_TABLA,'BCI_TRAIN_LTVSRF_AVG.csv','Delimiter',',
      ');
308
309 BCI=horzcat(dataSetLTVSRF_TEST_AVG,dataSetLVST_TEST_AVG,
      dataSetRVSF_TEST_AVG);
310 BCI=(BCI-minimo)/(maximo-minimo);
311 BCI_TEST=horzcat(class_test(1:length(class_train))',BCI);
312 BCI_TEST_TABLA=table(BCI_TEST);
313 writetable(BCI_TEST_TABLA,'BCI_TEST_LTVSRF_AVG.csv','Delimiter',',')
      ;
314 %
```

B.4. Método filtrado espacial

```
function Y = spatFilt(x, coef, dimm)
[m, n] = size(coef);
    if(m<dimm)
        disp('Cannot reduce to a higher dimensional space!');
        return
    end
    Ptild = zeros(dimm,n);
    i=0;
    for d = 1:dimm
     if(mod(d,2)==0)
        Ptild(d,:) = coef(m-i,:);
        i=i+1;
     else
        Ptild(d,:) = coef(1+i,:);
     end
    end
    T = length(x);
    Y=zeros(dimm,T);
    for d = 1:dimm
       for t = 1:T
       Y(d,t) = dot(Ptild(d,:),x(:,t));
       end
    end
    return
```

B.5. Lectura de archivos CSV

```
public List<Trial> LeerCSV(string rutacompleta)
 {
     //ALMACENAR LOS RENGLONES DEL DOCUMENTO EN UN ARRAY
     string[] renglones = File.ReadAllLines(rutacompleta);
     string datoIndividual;
     List<Trial> dataSet = new List<Trial>();
     try {
         //LEER LINEA POR LINEA EMPEZANDO DE LA 0 HASTA LA ANTEPENULTIMA
         for (int i = 1; i < renglones.Length; i++)
         //for (long i = 1; i < 2; i++){
             //SEPARAR SEGUN EL CARACTER "," Y ALMACENAR LOS DATOS EN UN
                 ARRAY
             datoIndividual = renglones[i];
             Char delimiter = ',';
             string[] datos = datoIndividual.Split(delimiter);
             int claseT = Convert.ToInt32(datos[0]);
             double[] featuresT = new double[datos.Length - 1];
             for (int j = 1; j < datos.Length; j++){
                 featuresT[j - 1] = double.Parse(datos[j], CultureInfo.
                     InvariantCulture);
             }
             Trial p1 = new Trial(claseT, featuresT);
             dataSet.Add(p1);
             //Console.WriteLine(datos[1]);
         }
     }
```

```
25      catch (Exception e) {
26          Console.WriteLine(e.Message);
27      }
28      return dataSet;
29  }
```

B.6. KNN: entrenamiento y evaluación

```
1 public KNearestNeighbors TrainKNN(List<Trial> dataTrainTemp, int nfeatures
      , int k, int tdistancia)
2   {
3       List<Trial> dataTrain = new List<Trial>();
4       foreach (Trial trial in dataTrainTemp)
5       {
6           dataTrain.Add(trial);
7       }
8       dataTrain = DesordenarLista(dataTrain);
9       double[][] inputs = new double[dataTrain.Count][];
10      int[] outputs = new int[dataTrain.Count];
11      int i = 0;
12      foreach (Trial currentTrial in dataTrain)
13      {
14          double[] temFeatures = new double[nfeatures];
15          for (int j = 0; j < nfeatures; j++)
16          {
17              temFeatures[j] = currentTrial.Features[j];
18          }
19          inputs[i] = temFeatures;
20          outputs[i] = currentTrial.Clase - 1;
21          i++;
22      }
23
24      var knn = new KNearestNeighbors(k);
25      if (tdistancia == 0)
26      {
27          knn = new KNearestNeighbors(k, new Euclidean());
28      }
29      else if (tdistancia == 1)
30      {
31          knn = new KNearestNeighbors(k, new Mahalanobis());
32      }
33      else if (tdistancia == 2)
34      {
35          knn = new KNearestNeighbors(k, new Manhattan());
36      }
37      knn.Learn(inputs, outputs);
38      return knn;
39
40  }
41
42  public int TestKNN(KNearestNeighbors knn, Trial inX, int nfeatures)
43  {
44      int prediccion = 0;
```

```
45          double[] input = new double[nfeatures];
46          for (int j = 0; j < nfeatures; j++)
47          {
48              input[j] = inX.Features[j];
49          }
50          prediccion = knn.Decide(input) + 1;
51
52          return prediccion;
53      }
```

B.7. MLP: entrenamiento y evaluación

```
1   public ActivationNetwork TrainMLP(List<Trial> dataTrainTemp, int
        nfeatures, int[] neuronas)
2   {
3       int epochs = 100000;
4       List<Trial> dataTrain = new List<Trial>();
5       foreach (Trial trial in dataTrainTemp)
6       {
7           dataTrain.Add(trial);
8       }
9       dataTrain = DesordenarLista(dataTrain);
10      double[][] input = new double[dataTrain.Count][];
11      double[][] output = new double[dataTrain.Count][];
12      int i = 0;
13      foreach (Trial currentTrial in dataTrain)
14      {
15          double[] temFeatures = new double[nfeatures];
16          output[i] = new double[] { currentTrial.Clase - 1 };
17          for (int j = 0; j < nfeatures; j++)
18          {
19              temFeatures[j] = currentTrial.Features[j];
20          }
21          input[i] = temFeatures;
22          i++;
23      }
24
25      ActivationNetwork network = new ActivationNetwork(new
            SigmoidFunction(2), nfeatures, neuronas);
26      BackPropagationLearning teacher = new BackPropagationLearning(
            network);
27      teacher.LearningRate = 0.15;
28      teacher.Momentum = 0.4;
29      bool needToStop = false;
30      int epochNumber = 0;
31      while (!needToStop)
32      {
33          epochNumber++;
34          double error = teacher.RunEpoch(input, output);
35          //Console.WriteLine(epochNumber + " --- " + error);
36          if (error <= 0.001 || epochNumber >= epochs)
37          {
38              needToStop = true;
```

```
39          }
40      }
41      return network;
42  }
43
44  public int TestMLP(Network modelMLP, Trial inX, int nfeatures)
45  {
46      int prediccion = 0;
47      double[] input = new double[nfeatures];
48      for (int j = 0; j < nfeatures; j++)
49      {
50          input[j] = inX.Features[j];
51      }
52      double[] label = modelMLP.Compute(input);
53      prediccion = (int)Math.Round(label[0]) + 1;
54      return prediccion;
55  }
```

B.8. SVM: entrenamiento y evaluación

```
1 public SupportVectorMachine<Gaussian> TrainSVM(List<Trial> dataTrainTemp,
    int nfeatures)
2  {
3      List<Trial> dataTrain = new List<Trial>();
4      foreach (Trial trial in dataTrainTemp)
5      {
6          dataTrain.Add(trial);
7      }
8      dataTrain = DesordenarLista(dataTrain);
9      double[][] inputs = new double[dataTrain.Count][];
10     int[] labels = new int[dataTrain.Count];
11     int i = 0;
12     foreach (Trial currentTrial in dataTrain)
13     {
14         double[] temFeatures = new double[nfeatures];
15         if (currentTrial.Clase == 1)
16         {
17             labels[i] = -1;
18         }
19         else if (currentTrial.Clase == 2)
20         {
21             labels[i] = 1;
22         }
23
24         for (int j = 0; j < nfeatures; j++)
25         {
26             temFeatures[j] = currentTrial.Features[j];
27         }
28         inputs[i] = temFeatures;
29         i++;
30     }
31
32     var learn = new SequentialMinimalOptimization<Gaussian>()
```

```
33      {
34          UseComplexityHeuristic = true,
35          UseKernelEstimation = true
36      };
37      SupportVectorMachine<Gaussian> svm = learn.Learn(inputs, labels);
38      return svm;
39
40  }
41
42  public int TestSVM(SupportVectorMachine<Gaussian> svm, Trial inX, int
        nfeatures)
43  {
44      int prediccion = 0;
45      double[] input = new double[nfeatures];
46      for (int j = 0; j < nfeatures; j++)
47      {
48          input[j] = inX.Features[j];
49      }
50      bool prediction = svm.Decide(input);
51      if (prediction)
52      {
53          prediccion = 2;
54      }
55      else
56      {
57          prediccion = 1;
58      }
59      return prediccion;
60  }
```

B.9. DMNN: entrenamiento y evaluación

```
1 public DendriteMorphologicalNeuralNetwork TrainDMNN(List<Trial>
      dataTrainTemp, int nfeatures, int nclases)
2   {
3       List<Trial> dataTrain = new List<Trial>();
4       foreach (Trial currentTrial in dataTrainTemp)
5       {
6           double[] temFeatures = new double[nfeatures];
7           for (int j = 0; j < nfeatures; j++)
8           {
9               temFeatures[j] = currentTrial.Features[j];
10          }
11          dataTrain.Add(new Trial(currentTrial.Clase, temFeatures));
12      }
13
14      DendriteMorphologicalNeuralNetwork DMNN = new
            DendriteMorphologicalNeuralNetwork(nfeatures, nclases);
15      DMNN.learn(dataTrain);
16      return DMNN;
17
18  }
19
```

```
20  public int TestDMNN(DendriteMorphologicalNeuralNetwork DMNN, Trial inX,
        int nfeatures)
21  {
22      int prediccion = 0;
23      double[] input = new double[nfeatures];
24      for (int j = 0; j < nfeatures; j++)
25      {
26          input[j] = inX.Features[j];
27      }
28      prediccion = DMNN.Decide(input);
29
30      return prediccion;
31  }
```

B.10. Clase Trial

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Trial
6 {
7   public int Clase { get; set; }
8   public double[] Features { get; set; }
9   public double Distancia { get; set; }
10
11  public Trial(int clase, double[] features)
12  {
13      Clase = clase;
14      Features = features;
15      Distancia = 0;
16  }
17 }
```

B.11. Clase DMNN

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4
5 public class DendriteMorphologicalNeuralNetwork
6 {
7   public int boxNumber { get; set; }
8   public int featureNumber { get; set; }
9   public int nClasses { get; set; }
10  private double[][] inputs { get; set; }
11  private int[] labels { get; set; }
12  private List<BoxDNN> finalBoxes = new List<BoxDNN>();
13  private List<BoxDNN> tempBoxesObjetive = new List<BoxDNN>();
14  private List<BoxDNN> tempBoxesOneClass = new List<BoxDNN>();
```

```
15  private List<BoxDNN> tempBoxesMoreClasses = new List<BoxDNN>();
16  private List<BoxDNN> tempBoxesMoreClasses2 = new List<BoxDNN>();
17  public int nEpochs = 100;
18
19  public DendriteMorphologicalNeuralNetwork(int nfeatures, int nclases)
20  {
21      featureNumber = nfeatures;
22      nClasses = nclases;
23  }
24
25  public void learn(List<Trial> dataTrain)
26  {
27      List<double[]> wBigBox = new List<double[]>();
28      for (int i = 0; i < featureNumber; i++)
29      {
30          double[] stemp = new double[dataTrain.Count];
31          int ss = 0;
32          foreach (Trial trial in dataTrain)
33          {
34              stemp[ss] = trial.Features[i];
35              ss++;
36          }
37          wBigBox.Add(new double[] { MinArray(stemp), MaxArray(stemp) });
38      }
39      BoxDNN bigBox = new BoxDNN(wBigBox);
40      bigBox.ClassCount(dataTrain, nClasses);
41      divBox(bigBox, dataTrain);
42      foreach (BoxDNN box in tempBoxesOneClass)
43      {
44          finalBoxes.Add(new BoxDNN(box));
45      }
46      foreach (BoxDNN box in tempBoxesMoreClasses)
47      {
48          tempBoxesObjetive.Add(new BoxDNN(box));
49      }
50      tempBoxesOneClass.Clear();
51      tempBoxesMoreClasses.Clear();
52      int epoch = 0;
53      while (tempBoxesObjetive.Count > 0 && epoch < nEpochs)
54      {
55          tempBoxesOneClass.Clear();
56          tempBoxesMoreClasses.Clear();
57          tempBoxesMoreClasses2.Clear();
58          foreach (BoxDNN box in tempBoxesObjetive)
59          {
60              divBox(box, dataTrain);
61              foreach (BoxDNN box2 in tempBoxesOneClass)
62              {
63                  finalBoxes.Add(new BoxDNN(box2));
64              }
65              foreach (BoxDNN box2 in tempBoxesMoreClasses)
66              {
67                  tempBoxesMoreClasses2.Add(new BoxDNN(box2));
68              }
69          }
70          tempBoxesObjetive.Clear();
```

```
71          foreach (BoxDNN box in tempBoxesMoreClasses2)
72          {
73              tempBoxesObjetive.Add(new BoxDNN(box));
74          }
75          epoch++;
76          if (epoch == nEpochs)
77          {
78              foreach (BoxDNN box in tempBoxesObjetive)
79              {
80                  finalBoxes.Add(new BoxDNN(box));
81              }
82          }
83      }
84      foreach (BoxDNN box in finalBoxes)
85      {
86          box.DefPrincipalClass();
87      }
88      boxNumber = finalBoxes.Count;
89  }
90
91  public int Decide(double[] input)
92  {
93      int prediction = 0;
94      List<double[]> d = new List<double[]>();
95      foreach (BoxDNN box in finalBoxes)
96      {
97          d.Add(new double[] { box.PrincipalClass, box.BoxDNNActTest(input
                ) });
98      }
99      d = d.OrderByDescending(o => o[1]).ToList();
100     prediction = (int)d[0][0];
101     return prediction;
102 }
103
104 private void divBox(BoxDNN box, List<Trial> dataTrain)
105 {
106     tempBoxesOneClass.Clear();
107     tempBoxesMoreClasses.Clear();
108     List<double[]> wtemp = new List<double[]>();
109     foreach (double[] wp in box.w)
110     {
111         wtemp.Add(new double[] { wp[0], wp[1] / 2, wp[1] });
112     }
113     int nd = box.w.Count;
114     int boxDim = (int)Math.Pow(2, nd);
115     List<double[]> w2 = new List<double[]>();
116     List<BoxDNN> tempBoxes = new List<BoxDNN>();
117     for (int contb = 0; contb < boxDim; contb++)
118     {
119         w2.Clear();
120         int[] s = Dec2Bin(contb, nd);
121         for (int i = 0; i < nd; i++)
122         {
123             w2.Add(new double[] { wtemp[i][s[i]], wtemp[i][s[i] + 1] });
124         }
125         tempBoxes.Add(new BoxDNN(w2));
```

```
126             tempBoxes[contb].ClassCount(dataTrain, nClasses);
127         }
128
129         foreach (BoxDNN tempbox in tempBoxes)
130         {
131             int presentClasses = 0;
132             for (int p = 1; p < tempbox.AllClasses.Count + 1; p++)
133             {
134                 if (tempbox.AllClasses[p] != 0)
135                 {
136                     presentClasses++;
137                 }
138             }
139             if (presentClasses == 1)
140             {
141                 tempBoxesOneClass.Add(new BoxDNN(tempbox));
142             }
143             else if (presentClasses > 1)
144             {
145                 tempBoxesMoreClasses.Add(new BoxDNN(tempbox));
146             }
147         }
148     }
149
150     public int[] Dec2Bin(int num, int res)
151     {
152         int[] ind = new int[res];
153         string cadena = "";
154         char[] mascara = new char[res];
155         for (int s = 0; s < res; s++)
156         {
157             mascara[s] = '0';
158         }
159         char[] bitnum = new char[mascara.Length];
160         cadena = System.Convert.ToString(num, 2);
161         int diftam = mascara.Length - cadena.Length;
162         int index = 0;
163         for (int j = 0; j < mascara.Length; j++)
164         {
165             bitnum[j] = mascara[j];
166             if (j >= diftam)
167             {
168                 bitnum[j] = (char)cadena[index];
169                 index++;
170             }
171         }
172         for (int s = 0; s < res; s++)
173         {
174             ind[s] = (int)Char.GetNumericValue(bitnum[s]);
175         }
176         return ind;
177     }
178
179     public double MaxArray(double[] arraynumber)
180     {
181         double max = 0;
```

```
182         max = arraynumber[0];
183         for (int i = 1; i < arraynumber.Length; i++)
184         {
185             max = Math.Max(max, arraynumber[i]);
186         }
187         return max;
188     }
189 
190     public double MinArray(double[] arraynumber)
191     {
192         double min = 0;
193         min = arraynumber[0];
194         for (int i = 1; i < arraynumber.Length; i++)
195         {
196             min = Math.Min(min, arraynumber[i]);
197         }
198         return min;
199     }
200 
201 }
```

B.12. Clase BoxDNN

```
1  using System.Collections.Generic;
2  using System;
3  
4  public class BoxDNN
5  {
6      public int featuresNumber { get; set; }
7      public int PrincipalClass { get; set; }
8      public List<double[]> w = new List<double[]>();
9      public Dictionary<int, int> AllClasses = new Dictionary<int, int>();
10 
11     public BoxDNN(List<double[]> wi)
12     {
13         foreach (double[] s in wi)
14         {
15             this.w.Add(new double[] { s[0], s[1] });
16         }
17         featuresNumber = w.Count;
18     }
19 
20     public BoxDNN(BoxDNN box)
21     {
22         foreach (double[] s in box.w)
23         {
24             w.Add(s);
25         }
26         featuresNumber = w.Count;
27         PrincipalClass = box.PrincipalClass;
28         for (int i = 1; i < box.AllClasses.Count + 1; i++)
29         {
30             AllClasses[i] = box.AllClasses[i];
```

```
31      }
32  }
33
34  public void ClassCount(List<Trial> dataTrain, int nclases)
35  {
36      for (int i = 1; i < nclases + 1; i++)
37      {
38          AllClasses[i] = 0;
39      }
40      foreach (Trial trial in dataTrain)
41      {
42          int rs = BoxDNNAct(trial.Features);
43          if (rs == 1)
44          {
45              AllClasses[trial.Clase] += 1;
46          }
47      }
48  }
49
50  public void DefPrincipalClass()
51  {
52      int index = 1;
53      int contmax = AllClasses[index];
54      for (int i = 2; i < AllClasses.Count + 1; i++)
55      {
56          if (AllClasses[i] > contmax)
57          {
58              contmax = AllClasses[i];
59              index = i;
60          }
61      }
62      PrincipalClass = index;
63  }
64
65  private int BoxDNNAct(double[] xi)
66  {
67      int r = 0;
68      double[] dmin = new double[xi.Length];
69      for (int j = 0; j < xi.Length; j++)
70      {
71          dmin[j] = Math.Min(xi[j] - w[j][0], w[j][1] - xi[j]);
72      }
73      double d = MinArray(dmin);
74      if (d > 0)
75      {
76          r = 1;
77      }
78      return r;
79  }
80
81  public double BoxDNNActTest(double[] xi)
82  {
83      int r = 0;
84      double[] dmin = new double[xi.Length];
85      for (int j = 0; j < xi.Length; j++)
86      {
```

```
87              dmin[j] = Math.Min(xi[j] - w[j][0], w[j][1] - xi[j]);
88          }
89          double d = MinArray(dmin);
90          return d;
91      }
92
93      public double MaxArray(double[] arraynumber)
94      {
95          double max = 0;
96          max = arraynumber[0];
97          for (int i = 1; i < arraynumber.Length; i++)
98          {
99              max = Math.Max(max, arraynumber[i]);
100         }
101         return max;
102     }
103
104     public double MinArray(double[] arraynumber)
105     {
106         double min = 0;
107         min = arraynumber[0];
108         for (int i = 1; i < arraynumber.Length; i++)
109         {
110             min = Math.Min(min, arraynumber[i]);
111         }
112         return min;
113     }
114 }
```

B.13. Controlador silla de ruedas en Unity

```
1 using UnityEngine;
2 using System.Collections.Generic;
3 using Accord.MachineLearning.VectorMachines;
4 using Accord.Statistics.Kernels;
5 using Accord.Math;
6
7 public class WheelchairController : MonoBehaviour {
8       public float moveSpeed = 0.43f;
9       private float turnSpeed = 2.5f;
10      private float slopeResistance = 25;
11      private float stickToGroundForce = 10;
12      private CharacterController cc;
13      private UnityEngine.Vector3 velocity;
14      private bool previouslyGrounded;
15  private string textScreen;
16  private string textScreen2;
17  private TextMesh pantalla;
18  private TextMesh pantallaStatus;
19  private float hor;
20  private float ver;
21  private float cp;
22  private float cn;
```

```
23  private float exactitud;
24  private float error;
25
26  private string ruta = "";
27
28  private string rutaTRAIN = "";
29  private string rutaTEST = "";
30  private int cl11 = 0;
31  private int cl12 = 0;
32  private int cl21 = 0;
33  private int cl22 = 0;
34
35  private Accord.Neuro.Network model1VS2;
36  private Accord.Neuro.Network model11VS12;
37  private SupportVectorMachine<Gaussian> model21VS22;
38
39  private List<Trial> BCI_TRAIN;
40  List<Trial> BCI_TRAIN_1VS2 = new List<Trial>();
41  List<Trial> BCI_TRAIN_11VS12 = new List<Trial>();
42  List<Trial> BCI_TRAIN_21VS22 = new List<Trial>();
43
44  private List<Trial> BCI_TEST;
45  private List<Trial> BCI_TEST_L = new List<Trial>();
46  private List<Trial> BCI_TEST_R = new List<Trial>();
47  private List<Trial> BCI_TEST_F = new List<Trial>();
48  private List<Trial> BCI_TEST_T = new List<Trial>();
49  private int featuresNumber = 14;
50  private System.Random rnd = new System.Random();
51  private Trial testPattern;
52
53  private Procesamiento procesamiento = new Procesamiento();
54
55  void Awake()
56  {
57      rutaTRAIN = ruta + "BCI_TRAIN_LFVSRT.csv";
58      rutaTEST = ruta + "BCI_TEST_LFVSRT.csv";
59      int nfeatures = 14;
60      cl11 = 1;
61      cl12 = 4;
62      cl21 = 2;
63      cl22 = 3;
64      BCI_TRAIN = procesamiento.LeerCSV(rutaTRAIN);
65      foreach (Trial trial in BCI_TRAIN)
66      {
67          double[] tempFeature = new double[nfeatures];
68          for (int i = 0; i < nfeatures; i++)
69          {
70              tempFeature[i] = trial.Features[i];
71          }
72          double[] tempFeature2 = new double[nfeatures];
73          for (int i = 14; i < nfeatures + 14; i++)
74          {
75              tempFeature2[i - 14] = trial.Features[i];
76          }
77          double[] tempFeature3 = new double[nfeatures];
78          for (int i = 28; i < nfeatures + 28; i++)
```

```
79          {
80              tempFeature3[i - 28] = trial.Features[i];
81          }
82
83          if (trial.Clase == cl11)
84          {
85              BCI_TRAIN_1VS2.Add(new Trial(1, tempFeature));
86              BCI_TRAIN_11VS12.Add(new Trial(1, tempFeature2));
87          }
88          else if (trial.Clase == cl12)
89          {
90              BCI_TRAIN_1VS2.Add(new Trial(1, tempFeature));
91              BCI_TRAIN_11VS12.Add(new Trial(2, tempFeature2));
92          }
93          else if (trial.Clase == cl21)
94          {
95              BCI_TRAIN_1VS2.Add(new Trial(2, tempFeature));
96              BCI_TRAIN_21VS22.Add(new Trial(1, tempFeature3));
97          }
98          else if (trial.Clase == cl22)
99          {
100             BCI_TRAIN_1VS2.Add(new Trial(2, tempFeature));
101             BCI_TRAIN_21VS22.Add(new Trial(2, tempFeature3));
102         }
103     }
104     model1VS2 = procesamiento.TrainMLP(BCI_TRAIN_1VS2, 8, new int[] {
            10, 1 });
105     model11VS12 = procesamiento.TrainMLP(BCI_TRAIN_11VS12, 8, new int[]
            { 10, 1 });
106     model21VS22 = procesamiento.TrainSVM(BCI_TRAIN_21VS22, 2);
107     BCI_TEST = procesamiento.LeerCSV(rutaTEST);
108     //featuresNumber = model1VS2.InputsCount;
109     cn = 0;
110     cp = 0;
111     foreach (Trial trial in BCI_TEST)
112     {
113         if (trial.Clase == 1)
114         {
115             BCI_TEST_L.Add(new Trial(trial.Clase, trial.Features));
116         }
117         else if (trial.Clase == 2)
118         {
119             BCI_TEST_R.Add(new Trial(trial.Clase, trial.Features));
120         }
121         else if (trial.Clase == 3)
122         {
123             BCI_TEST_F.Add(new Trial(trial.Clase, trial.Features));
124         }
125         else if (trial.Clase == 4)
126         {
127             BCI_TEST_T.Add(new Trial(trial.Clase, trial.Features));
128         }
129     }
130 }
131 // Use this for initialization
132 void Start () {
```

```
133             cc = GetComponent<CharacterController>();
134     }
135         // Update is called once per frame
136         void Update () {
137         bool mov = false;
138         if (cc.isGrounded) {
139                         velocity.y = -stickToGroundForce;
140                         previouslyGrounded = true;
141                 } else {
142                         if (previouslyGrounded) {
143                                 velocity.y = 0;
144                         }
145                         previouslyGrounded = false;
146                 }
147         //hor = Input.GetAxis("Horizontal");
148         //ver = Input.GetAxis("Vertical");
149         hor = 0;
150         ver = 0;
151         int predict = 0;
152         testPattern = null;
153         //Dependiendode la flecha oprimida se toma un patron de esa clase de
                manera aleatoria
154         if (Input.GetKeyUp(KeyCode.RightArrow))
155         {
156             textScreen = "Derecha";
157             int rndnumber = rnd.Next(0, 44);
158             testPattern = BCI_TEST_L[rndnumber];
159         }
160         if (Input.GetKeyUp(KeyCode.LeftArrow))
161         {
162             textScreen = "Izquierda";
163             int rndnumber = rnd.Next(0, 44);
164             testPattern = BCI_TEST_R[rndnumber];
165         }
166         if (Input.GetKeyUp(KeyCode.UpArrow))
167         {
168             textScreen = "Adelante";
169             int rndnumber = rnd.Next(0, 44);
170             testPattern = BCI_TEST_F[rndnumber];
171         }
172         if (Input.GetKeyUp(KeyCode.DownArrow))
173         {
174             textScreen = "Atras";
175             int rndnumber = rnd.Next(0, 44);
176             testPattern = BCI_TEST_L[rndnumber];
177         }
178
179         if (testPattern != null)
180         {
181             mov = true;
182             double[] tempFeature = new double[featuresNumber];
183             for (int j = 0; j < featuresNumber; j++)
184             {
185                 tempFeature[j] = testPattern.Features[j];
186             }
187             double[] tempFeature2 = new double[featuresNumber];
```

```
188         for (int j = 14; j < featuresNumber + 14; j++)
189         {
190             tempFeature2[j - 14] = testPattern.Features[j];
191         }
192         double[] tempFeature3 = new double[featuresNumber];
193         for (int j = 28; j < featuresNumber + 28; j++)
194         {
195             tempFeature3[j - 28] = testPattern.Features[j];
196         }
197 
198         int predict1VS2 = procesamiento.TestMLP(model1VS2, new Trial(
                testPattern.Clase, tempFeature), 8);
199         int predict11VS12 = 0;
200         int predict21VS22 = 0;
201         if (predict1VS2 == 1)
202         {
203             predict11VS12 = procesamiento.TestMLP(model11VS12, new Trial
                    (testPattern.Clase, tempFeature2), 8);
204             if (predict11VS12 == 1)
205             {
206                 predict = cl11;
207             }
208             else if (predict11VS12 == 2)
209             {
210                 predict = cl12;
211             }
212         }
213         else if (predict1VS2 == 2)
214         {
215             predict21VS22 = procesamiento.TestSVM(model21VS22, new Trial
                    (testPattern.Clase, tempFeature3), 2);
216             if (predict21VS22 == 1)
217             {
218                 predict = cl21;
219 
220             }
221             else if (predict21VS22 == 2)
222             {
223                 predict = cl22;
224             }
225         }
226         string labelCor = "Error";
227         if (predict == testPattern.Clase)
228         {
229             labelCor = "";
230         }
231         Debug.Log("---->" + predict1VS2 + "--" + predict11VS12 + "--"
                + predict21VS22 + "->>" + predict + "->>" + testPattern.
                Clase + "->>>" + labelCor + "<<<");
232     }
233     //Envio de Senal de Control
234     if (predict == 1)
235     {
236         hor = 1;
237         ver = 0;
238         textScreen2 = "Derecha";
```

```
239         } else if (predict == 2)
240         {
241             hor = -1;
242             ver = 0;
243             textScreen2 = "Izquierda";
244         } else if (predict == 3)
245         {
246             hor = 0;
247             ver = 1;
248             textScreen2 = "Adelante";
249         } else if (predict == 4)
250         {
251             hor = 0;
252             ver = -1;
253             textScreen2 = "Atras";
254         }
255         pantalla = (TextMesh)GameObject.Find("Text_Receiver").GetComponent<
                TextMesh>();
256         pantallaStatus = (TextMesh)GameObject.Find("Text_Receiver_Status").
                GetComponent<TextMesh>();
257         //pantalla.color = Color.black;
258         pantalla.text = "" + textScreen + "   -   " + textScreen2;
259         if (mov == true)
260         {
261             if (textScreen == textScreen2)
262             {
263                 //Debug.Log("Correcto");
264                 cp++;
265                 pantalla.color = Color.green;
266             }
267             else
268             {
269                 //Debug.Log("Error");
270                 cn++;
271                 pantalla.color = Color.red;
272             }
273             exactitud = cp * 100 / (cp + cn);
274             error = cn * 100 / (cp + cn);
275             pantallaStatus.text = "Trials = " + (cp + cn) + "  Exactitud = "
                    + exactitud.ToString("F2") +"%  Error = " + error.ToString()
                    + "%";
276         }
277         float magRotacion = hor * turnSpeed;
278         //Debug.Log(magRotacion);
279         transform.Rotate(0, magRotacion, 0);
280         UnityEngine.Vector3 desiredMove = transform.forward * ver *
                moveSpeed;
281                 RaycastHit hitInfo;
282                 Physics.SphereCast(transform.position, cc.radius,
                        UnityEngine.Vector3.down, out hitInfo, cc.height / 2f);
283                 desiredMove = UnityEngine.Vector3.ProjectOnPlane(desiredMove
                        , hitInfo.normal);
284                 // Slide down slopes
285                 float hitAngle = UnityEngine.Vector3.Angle(hitInfo.normal,
                        UnityEngine.Vector3.up);
```

```
286             float slopeEffect = Mathf.Clamp01((hitAngle -
                    slopeResistance) / cc.slopeLimit);
287         UnityEngine.Vector3 slideForce = new UnityEngine.Vector3(hitInfo.
                normal.x, -hitInfo.normal.y, hitInfo.normal.z) * slopeEffect;
288             velocity += Physics.gravity * Time.deltaTime;
289             cc.Move((velocity + desiredMove + slideForce) * Time.
                    deltaTime);
290         }
291     }
```