

UNIVERSIDAD NACIONAL DEL LITORAL



**ALGORITMOS BIOINSPIRADOS  
PARA LA IMPLEMENTACIÓN DE  
INTERFACES CEREBRO COMPUTADORAS**

Iván Emilio Gareis

**FICH**

FACULTAD DE INGENIERIA  
Y CIENCIAS HIDRICAS

**INTEC**

INSTITUTO DE DESARROLLO TECNOLÓGICO  
PARA LA INDUSTRIA QUIMICA

Tesis de Doctorado **2017**





UNIVERSIDAD NACIONAL DEL LITORAL  
Facultad de Ingeniería y Ciencias Hídricas  
Instituto de Desarrollo Tecnológico para la Industria Química

**ALGORITMOS BIOINSPIRADOS  
PARA LA IMPLEMENTACIÓN DE  
INTERFACES CEREBRO COMPUTADORAS**

**Iván Emilio Gareis**

Tesis remitida al Comité Académico del Doctorado  
como parte de los requisitos para la obtención  
del grado de  
DOCTOR EN INGENIERIA  
Mención Señales Sistemas e Inteligencia Computacional  
de la  
UNIVERSIDAD NACIONAL DEL LITORAL

**2017**

Comisión de Posgrado, Facultad de Ingeniería y Ciencias Hídricas, Ciudad Universitaria, Paraje "El Pozo",  
S3000, Santa Fe, Argentina.





UNIVERSIDAD NACIONAL DEL LITORAL  
Facultad de Ingeniería y Ciencias Hídricas  
Instituto de Desarrollo Tecnológico para la Industria Química

**ALGORITMOS BIOINSPIRADOS  
PARA LA IMPLEMENTACIÓN DE  
INTERFACES CEREBRO COMPUTADORAS**

**Iván Emilio Gareis**

**Lugar de trabajo:**

*sinc(i)*

Instituto de Investigación en  
Señales, Sistemas e Inteligencia Computacional

Facultad de Ingeniería y Ciencias Hídricas

Universidad Nacional del Litoral

**Director:**

Dr. Hugo Leonardo Rufiner      UNL      CONICET

**Jurado Evaluador:**

Dr. Pablo Diez                      UNSJ      CONICET

Dr. Eric Laciari Leber              UNSJ      CONICET

Dr. Enrique Spinelli                UNLP      CONICET

Dr. Diego Tomassi                  UNL      CONICET

**2017**





### ACTA DE EVALUACIÓN DE TESIS DE DOCTORADO

En la sede de la Facultad de Ingeniería y Ciencias Hídricas de la Universidad Nacional del Litoral, a los treinta días del mes de marzo del año dos mil diecisiete, se reúnen los miembros del Jurado designado para la evaluación de la Tesis de Doctorado en Ingeniería titulada *“Algoritmos bioinspirados para la implementación de interfaces cerebro computadoras”*, desarrollada por el Bioing. Iván Emilio GAREIS, DNI N° 31.914.582. Ellos son: Dr. Eric Laciari Leber, Dr. Pablo Diez, Dr. Diego Tomassi y Dr. Enrique Spinelli.

Luego de escuchar la Defensa Pública y de evaluar la Tesis, el Jurado resuelve:

Otorgar la máxima calificación de sobresaliente (10) por la propuesta, implementación y validación de algoritmos originales; por la complejidad de las técnicas abordadas; la presentación fue clara y respondió con solvencia las preguntas del jurado. Su trabajo está respaldado por artículos en revistas y presentaciones en congresos.

Sin más, se da por finalizado el Acto Académico con la firma de los miembros del Jurado al pie de la presente.

-----  
Dr. Eric Laciari Leber

-----  
Dr. Diego Tomassi

-----  
Dr. Pablo Diez

-----  
Dr. JOSÉ LUIS MACOR  
SECRETARIO DE POSGRADO  
Facultad de Ingeniería y Cs. Hídricas  
Dr. Enrique Spinelli (\*)

(\*) El Dr. Spinelli participó por videoconferencia

Universidad Nacional del Litoral  
Facultad de Ingeniería y  
Ciencias Hídricas  
  
Secretaría de Posgrado

Ciudad Universitaria  
C.C. 217  
Ruta Nacional N° 168 - Km. 472,4  
(3000) Santa Fe  
Tel: (54) (0342) 4575 229  
Fax: (54) (0342) 4575 224  
E-mail: posgrado@fich.unl.edu.ar

Dr. JOSÉ LUIS MACOR  
SECRETARIO DE POSGRADO  
Facultad de Ingeniería y Cs. Hídricas








**UNIVERSIDAD NACIONAL DEL LITORAL**  
**Facultad de Ingeniería y Ciencias Hídricas**

Santa Fe, 30 de marzo de 2017.

Como miembros del Jurado Evaluador de la Tesis de Doctorado en Ingeniería titulada **“Algoritmos bioinspirados para la implementación de interfaces cerebro computadoras”**, desarrollada por el Bioing. Iván Emilio GAREIS, en el marco de la Mención “Inteligencia Computacional, Señales y Sistemas”, certificamos que hemos evaluado la Tesis y recomendamos que sea aceptada como parte de los requisitos para la obtención del título de Doctor en Ingeniería.

La aprobación final de esta disertación estará condicionada a la presentación de dos copias encuadernadas de la versión final de la Tesis ante el Comité Académico del Doctorado en Ingeniería.

  
-----  
Dr. Eric Laciari Leber

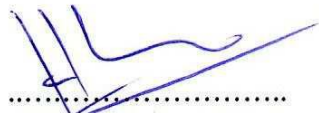
  
-----  
Dr. Diego Tomassi

  
-----  
Dr. Pablo Diez

  
-----  
Dr. JOSÉ LUIS MACOR  
SECRETARIO DE POSGRADO  
Facultad de Ingeniería y Cs. Hídricas  
Dr. Enrique Spinelli (x)

Santa Fe, 30 de marzo de 2017.

Certifico haber leído la Tesis, preparada bajo mi dirección en el marco de la Mención “Inteligencia Computacional, Señales y Sistemas” y recomiendo que sea aceptada como parte de los requisitos para la obtención del título de Doctor en Ingeniería.

  
-----  
Dr. Leonardo Rufiner  
Director de Tesis

*\* El Dr. Spinelli participo por videoconferencia*

Universidad Nacional del Litoral  
Facultad de Ingeniería y  
Ciencias Hídricas

Secretaría de Posgrado

  
-----  
Dr. JOSÉ LUIS MACOR  
SECRETARIO DE POSGRADO  
Facultad de Ingeniería y Cs. Hídricas  
Ciudad Universitaria  
C.C. 217  
Ruta Nacional N° 168 - Km. 472,4  
(3000) Santa Fe  
Tel: (54) (0342) 4575 229  
Fax: (54) (0342) 4575 224  
E-mail: posgrado@fich.unl.edu.ar



---

## **Declaración del autor**

Esta Tesis ha sido remitida como parte de los requisitos para la obtención del grado académico de Doctor en Ingeniería ante la Universidad Nacional del Litoral y ha sido depositada en la Biblioteca de la Facultad de Ingeniería y Ciencias Hídricas para que esté a disposición de sus lectores bajo las condiciones estipuladas por el reglamento de la mencionada Biblioteca.

Citaciones breves de esta Tesis son permitidas sin la necesidad de un permiso especial, en la suposición de que la fuente sea correctamente citada. Solicitudes de permiso para la citación extendida o para la reproducción parcial o total de ese manuscrito serán concebidos por el portador legal del derecho de propiedad intelectual de la obra.



---

*Para Mica.*



---

# Agradecimientos

Son muchas las personas que han colaborado, de una forma u otra, para poder hacer realidad esta tesis y a quienes deseo agradecer.

A mis colegas de trabajo del Instituto de Investigación en Señales Sistemas en Inteligencia Computacional, por el apoyo brindado durante la realización de mi doctorado.

A mis director, quien además de guiarme en términos académicos me brindó su confianza y apoyo personal en el desarrollo de este trabajo.

A Mica y Valentina por la paciencia y acompañamiento en el hogar.

Gracias a todos.





---

# Índice general

<b>Resumen</b>	IX
<b>Abstract</b>	XI
<b>1. Introducción</b>	1
<b>2. Interfaces Cerebro Computadora</b>	5
2.1. Bloques funcionales de una BCI	5
2.1.1. Preprocesamiento	6
2.1.2. Extracción de características	7
2.1.3. Selección de características	7
2.1.4. Clasificación	8
2.2. Registro de la Actividad Cerebral	8
2.2.1. Electroencefalografía	9
2.3. Potenciales relacionados con eventos	12
2.4. Promediación coherente	14
2.5. Aplicaciones de las BCI	15
2.6. Señales electrofisiológicas utilizadas en la implementación de BCI	17
2.6.1. BCI basadas en ritmos sensorimotoraes	17
2.6.2. BCI basadas en potenciales evocados visuales de estado estacionario	18
2.6.3. BCI basadas en potenciales corticales lentos	19
2.6.4. BCI basadas en potenciales P300	19
2.7. Software para BCI	21
<b>3. Métodos Computacionales</b>	23
3.1. Optimización	23
3.2. Regularización	25

3.3. Métodos de aprendizaje maquina	27
3.3.1. K-vecinos más cercanos	27
3.3.2. Máquina de vectores de soporte	28
<b>4. Redes Neuronales Artificiales</b>	<b>31</b>
4.1. Neuronas artificiales	32
4.2. Redes neuronales artificiales prealimentadas	34
4.3. Unidades softmax	36
4.4. Autocodificadores	37
4.5. Autocodificador ralo	38
4.6. Autocodificadores de limpieza de ruido	39
4.7. Otras variantes de autocodificador	40
4.8. Entrenamiento de redes neuronales artificiales	40
4.8.1. Retropropagación	41
4.9. Aprendizaje profundo	43
4.9.1. Pre-entrenamiento de redes multicapa	43
4.9.2. Redes neuronales convolucionales	45
<b>5. Autocodificador de Estimación de Promedios Coherentes</b>	<b>47</b>
5.1. ¿Qué es una buena representación?	48
5.2. Término de reconstrucción	50
5.3. Término de clasificación	51
5.4. Términos de rareza y regularización	52
5.5. AEP completo	52
5.6. Entrenamiento del AEP	53
5.7. Configuración de la red y ajuste de hiperparámetros	55
<b>6. Experimentos</b>	<b>59</b>
6.1. Evaluación de rendimiento	59
6.1.1. Medidas de rendimiento	60
6.2. Bases de datos	60
6.2.1. Bases de datos de competencias	61
6.2.2. Base de datos sintética	61
6.3. Preprocesamiento	63
6.4. Ajuste de hiperparámetros	65
6.5. Clasificación de potenciales relacionados con eventos	67

---

6.6. Estimación de potenciales relacionados con eventos . . . . .	72
<b>7. Conclusiones</b>	<b>75</b>
7.1. Discusión . . . . .	75
7.2. Aportes . . . . .	76
7.3. Trabajo futuro . . . . .	77
7.3.1. Autocodificadores de estimación de promedios coherentes apilados . . . . .	77
7.3.2. Habla imaginada . . . . .	78
<b>Acrónimos</b>	<b>81</b>
<b>Bibliografía</b>	<b>85</b>



---

# Índice de figuras

1.1. Tasa de transferencia de información en relación al número de épocas.	2
2.1. Diagrama de bloques de una BCI genérica.	6
2.2. (a) Sistema internacional 10-20 visto desde la izquierda y desde arriba de la cabeza. A = Lóbulo de la oreja, C = central, Pg = naso-faríngeo, P = parietal, F = frontal, Fp = frontal polar, O = occipital.(b) Localización y nomenclatura de los electrodos intermedios en las posiciones intermedias al 10 %. Adaptados de [58].	12
2.3. Señales pos-estimulo promediadas coherentemente.	15
2.4. Matriz de estimulación con el diseño propuesto originalmente por Farwell y Donchin.	20
3.1. Ilustración idealizada de curva-L. Se marcan el parámetro óptimo y su dirección de crecimiento.	27
4.1. Esquema de una neurona artificial básica.	32
4.2. Algunas funciones de activación.	33
4.3. Esquema desagregado de una red neuronal artificial básica.	36
4.4. Esquema funcional de una red neuronal artificial.	36
4.5. Esquema de entrenamiento de autocodificador apilado.	44
5.1. (a) AEP durante el entrenamiento. Las épocas objetivo están representadas con línea llena en rojo y las muestras no objetivo con línea azul punteada. (b) AEP durante la operación usado como extractor de características para clasificación. (c) AEP durante la operación usado para estimación.	53

---

6.1. Estimación de densidad espectral de frecuencia para las señales de entrenamiento originales y respuesta en frecuencia de los filtros lineales obtenidos por medio de LPC. . . . .	62
6.2. Muestras de ERP sintéticos utilizados para crear el conjunto de entrenamiento de la base de datos sintética (canal Pz). . . . .	63
6.3. Curvas-L utilizadas para obtener los hiperparámetros de regularización para el AEP entrenado con la base de datos sintéticos. . .	66
6.4. Representación espacio-temporal de los pesos de nueve unidades ocultas de un AEP entrenado. . . . .	71
6.5. Salidas con sus correspondientes entradas y salidas deseadas para el autocodificadores de estimación de promedios coherentes (AEP) (solo canal Pz). . . . .	72
6.6. Estimación de la PSD de las salidas del AEP para cada canal cuando se lo alimenta con ruido blanco. . . . .	73
7.1. Esquema del entrenamiento de AEP apilado. . . . .	78

---

# Índice de tablas

2.1. Resumen de métodos de registro de actividad cerebral. . . . .	9
6.1. Definiciones de la estructura y valores de los hiperparámetros obtenidos. . . . .	67
6.2. Resultados de clasificación de P300/no-P300 para el AEP utilizando los datos sintéticos. 1T, 2T indica época única y promedios de dos épocas respectivamente. . . . .	69
6.3. Resultados para el problema de clasificación binario P300/no P300 utilizando registros reales. . . . .	70





---

# Resumen

Las interfaces cerebro-computadora son sistemas que proporcionan comunicación a un dispositivo externo utilizando en forma directa mediciones de la actividad cerebral. Estos sistemas utilizan métodos de procesamiento de señales y aprendizaje maquina para traducir en comandos útiles la actividad cerebral registrada. Entre las señales empleadas para implementar interfaces cerebro-computadora podemos encontrar los potenciales relacionados con eventos, los cuales están comúnmente enmascarados por ruido no correlacionado de gran amplitud. Este es uno de los principales problemas de procesamiento de señales que se deben resolver en las interfaces cerebro-computadora basadas en potenciales P300. La clasificación y estimación de pequeñas señales repetitivas usualmente requiere del registro y procesamiento de varias realizaciones de la señal de interés, siendo cada repetición un proceso que consume tiempo y reduce la tasa de transferencia de información. Para afrontar este problema, hemos propuesto una nueva variante de autocodificador con una función de coste multiobjetivo, llamada autocodificador de estimación de promedios coherentes. En este trabajo se ilustra su uso y analiza su desempeño aplicándolo al problema del procesamiento de potenciales relacionados con eventos. También se presentan resultados experimentales que muestran las ventajas del enfoque propuesto.



---

# Abstract

Brain-computer interfaces are systems that provide communication to an external device by means of directly measuring the brain activity. These systems rely on signal processing and machine learning stages to successfully translate the registered brain activity into useful commands. Amongst the signals employed in brain-computer interfaces we can find different event related potentials, which are usually masked by large noise. This is one of the main signal processing problems to solve when estimating or classifying P300 event related potentials in brain-computer interfaces. Classification and estimation of small repetitive signals usually require recording and processing several different realizations of the signal of interest. Each repetition is a time consuming process that reduces the information transfer rate. To cope with this issue we propose a novel autoencoder variation, called coherent averaging estimation autoencoder with a new multi-objective cost function. We illustrate its use and analyze its performance in the problem of event related potentials processing. Experimental results showing the advantages of the proposed approach are presented as well.



---

# Capítulo 1

## Introducción

Una interfaz cerebro-computadora (BCI, del inglés Brain Computer Interface) es un sistema que provee vías de comunicación directas entre el cerebro y un dispositivo externo. Estas interfaces son una alternativa novedosa de comunicación particularmente interesantes para personas con discapacidades motoras severas, ya sea para ayudar a su integración en la sociedad o para proveer un medio de control sin asistencia permanente del entorno en el que se desenvuelven.

Las BCI detectan la presencia de patrones específicos de la actividad cerebral en curso de una persona, para luego traducirlos en comandos de control utilizables. Para identificar estos patrones, se emplean varios algoritmos de procesamiento de señales y de inteligencia artificial. Estos sistemas requieren de algún método de registro de la actividad cerebral y de una subsecuente traducción de dichos registros en señales de control aplicables a los dispositivos externos.

Puesto que la actividad cerebral es intrínsecamente no lineal y no estacionaria, el procesamiento y análisis de las señales que ésta genera presenta grandes desafíos. Estas señales se caracterizan por una gran variabilidad entre distintos sujetos e incluso entre distintas realizaciones. Además, cuando la captura de las señales se realiza por medio de electroencefalografía (EEG), los registros incluyen actividad proveniente de diversas fuentes, de las cuales sólo algunas aportan información de interés. Por otro lado, para los problemas de interés en la presente tesis, la relación señal a ruido (SNR, del inglés Signal to Noise Ratio) es muy desfavorable y las señales son de alta dimensión, con relativamente pocos patrones disponibles para el ajuste de los modelos a los datos. En el ámbito de las BCI, a las dificultades mencionadas se suma el hecho de que los métodos de procesamiento y clasificación deben ser aplicables en tiempo real. Debido a las

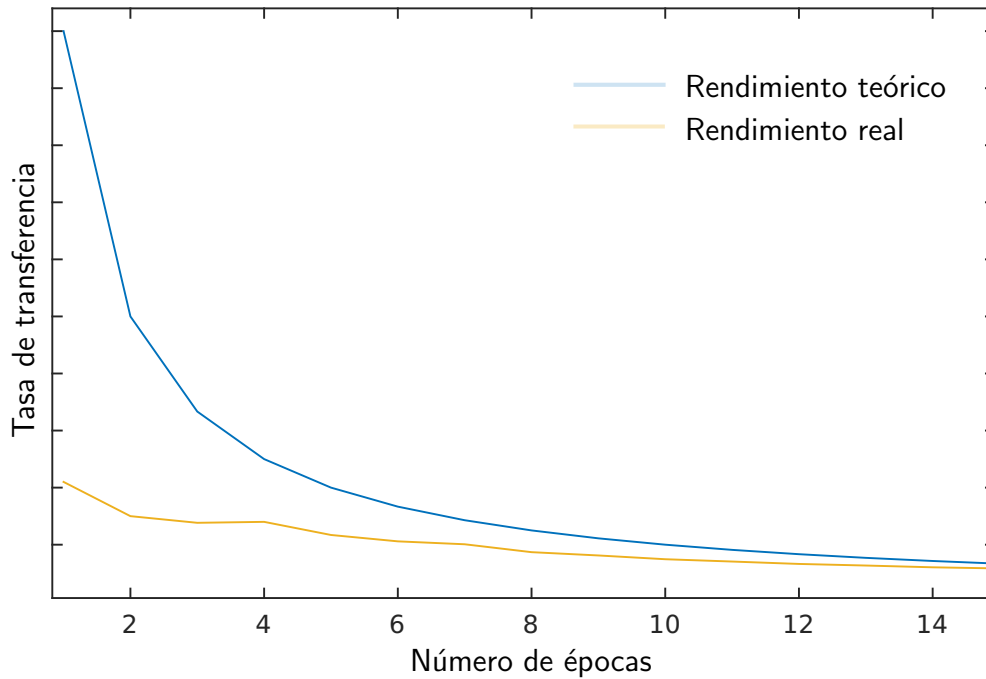


Figura 1.1: Tasa de transferencia de información en relación al número de épocas.

razones previamente mencionadas, los métodos de procesamiento de señales y reconocimiento de patrones constituyen un bloque crítico en el desarrollo de las BCI.

En el caso particular de las BCI basadas en potenciales P300, es común repetir múltiples veces el proceso de registro para aumentar las tasas de reconocimiento de las distintas señales. En la Figura 1.1 se presentan curvas de la tasa de transferencia de información (ITR, del inglés Information Transfer Rate) contra la cantidad de veces que se realizó el bloque de registro o épocas en una BCI basadas en potenciales P300. La curva azul representa la ITR que se obtendría si la tasa de reconocimiento de señales fuese perfecta, mientras que la curva amarilla muestra la ITR que se obtendría con las tasas de reconocimiento provistas utilizando una técnica del estado del arte [15]. Esta gráfica sugiere que hay lugar para incrementar el rendimiento de una BCI basada en potenciales P300 aumentando las tasas de clasificación de patrones para épocas únicas. Cabe resaltar que la ITR es solo una de las posibles medidas del rendimiento de una BCI, y que si bien tiene múltiples problemas al ser utilizada en esta aplicación [8], es sencilla y sirve para realizar una comparación rápida.

Por otro lado, a partir de la observación y entendimiento de ciertos mecanismos inteligentes empleados por los organismos vivos, es posible desarrollar nuevos

algoritmos que combinen técnicas inteligentes, capaces de aprender a partir de los datos con que se cuenta y logren representaciones suficientemente robustas que pongan en evidencia información discriminativa relevante de las señales analizadas. Una manera muy difundida de atacar estos problemas es a través del uso de una red neuronal artificial (ANN, del inglés Artificial Neural Network). Los modelos basados en ANN han sido desarrollados para una variedad de propósitos, inspirados generalmente en el funcionamiento de las neuronas biológicas. En los últimos años estos métodos han ganado popularidad, siendo aplicados con éxito a diversas problemáticas, motivándonos a desarrollar variantes de estos que puedan ser aplicables a las BCI.

El objetivo general de esta propuesta consiste en *diseñar y desarrollar nuevos algoritmos para análisis y procesamiento de señales de EEG con aplicación en el contexto de las BCI, y que brinden mejoras en el desempeño de estos sistemas.*

Los objetivos particulares incluyen:

- Explorar el estado del arte de las técnicas del procesamiento de señales aplicado a BCI.
- Proponer y desarrollar nuevos algoritmos de aprendizaje automático basados en ANN que contribuyan al análisis y procesamiento de señales de EEG utilizadas en BCI.
- Implementar las técnicas de análisis y procesamiento desarrolladas.
- Validar los métodos desarrollados mediante experimentos con datos artificiales y reales en el contexto de las BCI.
- Interpretar los resultados desde una perspectiva de análisis, modelado e identificación de señales y sistemas.

El resto de este documento se organizará como se describe a continuación. En el Capítulo [2](#) se presenta la problemática a abordar, dando una breve reseña de las técnicas de procesamiento más utilizadas en el área. En el Capítulo [3](#) se presentan brevemente algunos conceptos y métodos computacionales básicos utilizados en el desarrollo de los capítulos posteriores. En el Capítulo [4](#) se presentan los algoritmos que se utilizarán como base para el desarrollo de la estrategia propuesta y se discuten algunas consideraciones prácticas de estos métodos. También se presenta en dicho capítulo la mayor parte de la notación matemática a utilizar. En el

Capítulo [5](#) desarrollamos los algoritmos propuestos en esta tesis. En el Capítulo [6](#) se describen los experimentos realizados para analizar las capacidades de los métodos propuestos y se presentan los resultados de dichos experimentos. Por último, en el Capítulo [7](#) se discuten los resultados y conclusiones generales, y se presentan las líneas de trabajo futuro a desarrollar.



---

## Capítulo 2

# Interfaces Cerebro Computadora

En términos simples, una BCI se puede definir como un sistema que traduce señales cerebrales en nuevos tipos de salidas. Una definición más formal es la siguiente: una BCI es un sistema que adquiere señales representativas de la actividad cerebral y la traduce en una salida que puede ser usada para modificar las interacciones entre el cerebro y su ambiente interno o externo [18].

En este capítulo se discutirán algunos aspectos relevantes relacionados a las BCI.

### 2.1. Bloques funcionales de una BCI

En general, y de acuerdo a una de las primeras revisiones en el área [102], las partes de una BCI genérica se plantean como muestra el diagrama de la Figura 2.1. Se pueden distinguir claramente los siguientes bloques:

a) Instrumentación: es la etapa de adquisición y acondicionamiento de la señal de entrada. Capta la señal cerebral y devuelve una señal digitalizada en un rango dinámico adecuado para ser utilizada por los siguientes bloques.

b) Extracción de características: tiene como objetivo generar una representación adecuada de la señal de entrada que permita mejorar el desempeño de la etapa de clasificación.

c) Clasificación y traducción: en esta etapa se toman los patrones de la etapa de extracción de características, que se clasifican y traducen en las señales de salida de la BCI. Dichas señales de salida deben ser comandos adecuados para los dispositivos a controlar.

El usuario debe poder monitorear el estado del dispositivo a controlar. Esta

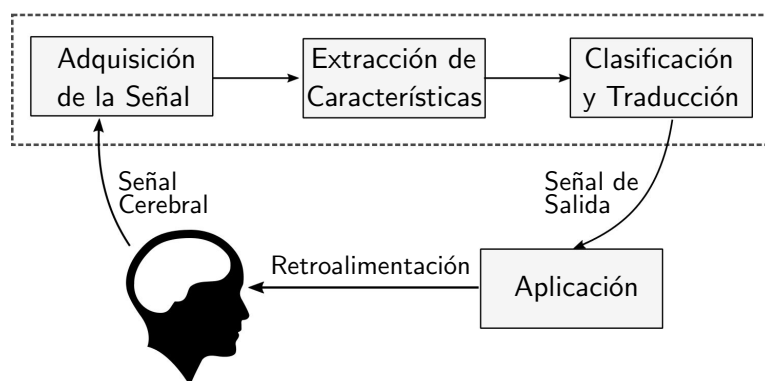


Figura 2.1: Diagrama de bloques de una BCI genérica.

realimentación permite al usuario determinar el resultado de sus esfuerzos por controlarlo, pudiendo así modular adecuadamente su actividad cerebral.

En la presente tesis los bloques sobre los que nos centraremos son los de extracción de características y de clasificación. Estos son los componentes de una BCI que aplican algoritmos de procesamiento de señales digitales y de reconocimiento de patrones. Estos dos bloques a su vez pueden ser subdivididos en preprocesamiento, extracción de características, selección de características, clasificación y traducción. Esta división en bloques debe ser vista como una simplificación para el estudio ya que muchas veces estas etapas pueden estar combinadas o no estar presentes.

### 2.1.1. Preprocesamiento

El objetivo principal del preprocesamiento es mejorar la relación señal a ruido presente en los registros. Las etapas de preprocesamiento que más comúnmente se realizan son el filtrado temporal, el filtrado espacial y la detección y remoción de artefactos [3].

Para aplicar filtrado temporal suelen utilizarse filtros sencillos. En el caso de las señales de EEG, y dependiendo del fenómeno particular a estudiar, la información útil se ubica por debajo de una frecuencia dada. En la etapa de preprocesamiento de las BCI suele utilizarse filtros pasa-bajos con frecuencias de corte que permitan mejorar la SNR del fenómeno bajo estudio. También es común la utilización de filtros pasa-banda, que además del ruido de alta frecuencia, rechazan también las tendencias de baja frecuencia, asociadas en general a la impedancia de la interfaz electrodo-piel.

Pueden ser utilizados distintos tipos de filtros espaciales, los más habituales son los de referencia promedio común (CAR, del inglés Common Average Reference) y los laplacianos. Otro método que suele emplearse es el los de patrones espaciales comunes (CSP, del inglés Common Spatial Patterns), particularmente con señales de imaginaria motora<sup>1</sup>. Dependiendo de como sean utilizadas las técnicas de análisis de componentes independientes (ICA, del inglés Independent Component Analysis) y análisis de componentes principales (PCA, del inglés Principal Components Analysis) también pueden ser consideradas filtros espaciales.

De las diversas fuentes de artefactos que pueden contaminar las señales de EEG, muchas contienen energía en bandas de frecuencia que se solapan con la información de interés (ver Sección 2.2.1). En estos casos los métodos de filtrado clásicos no son efectivos y es necesario utilizar otras técnicas como filtros adaptativos, filtros de Wiener o filtros de Bayes, técnicas de corrección utilizando electrooculografía (EOG), métodos de separación ciega de fuentes u otras estrategias como la descomposición modal empírica y la descomposición modal no-lineal. Una revisión detallada de estas y otras técnicas aplicadas a señales de EEG puede verse en [96].

### 2.1.2. Extracción de características

Las técnicas de extracción de características buscan encontrar parámetros de la señal reactivos al fenómeno que se desea detectar. Una gran variedad de estrategias se han utilizado para la extracción de características en el ámbito de las BCI. Muchas de estas estrategias están basadas en el diseño ad-hoc de características pero los métodos más comúnmente utilizados son estrategias generales de análisis de señales como las representaciones tiempo frecuencia y el modelado paramétrico. Más detalles se pueden encontrar en [3].

### 2.1.3. Selección de características

Posteriormente al proceso de extracción de características se puede seleccionar cuales son las que mas aportan a la resolución del problema. Estos son especialmente útiles para los diseños de BCI con dimensionalidad alta para los datos de entrada, o con estrategias de extracción de características que generen patrones

---

<sup>1</sup>del inglés “motor imagery”.

de dimensiones elevadas. Las técnicas más populares en este apartado para su uso en BCI han sido los algoritmo genético (GA, del inglés Genetic Algorithm), el análisis discriminante lineal (LDA, del inglés Linear Discriminant Analysis), el PCA, la selección secuencial de características y la búsquedas en grilla [3].

#### 2.1.4. Clasificación

La etapa de clasificación constituye uno de los bloques clave en las BCI. Una gran variedad de clasificadores se ha utilizado en este ámbito, y en general se observa que los clasificadores lineales son más robustos que los no lineales. Esto se debe a que este tipo de clasificadores tiene menos parámetros libres, y por tanto son menos propensos al sobre-ajuste [51]. Las técnicas más utilizadas son las máquina de soporte vectorial (SVM, del inglés Support Vector Machine) tanto lineales como no lineales, los clasificadores basados en LDA, las ANN (principalmente como perceptrón multi-capas (MLP, del inglés Multi Layer Perceptron) pero en algunos casos también perceptrones simples) y las selvas aleatorias<sup>2</sup> [3].

## 2.2. Registro de la Actividad Cerebral

Existen diversas tecnologías de registro que permiten captar la actividad cerebral, sin embargo no todas son aptas para el desarrollo de BCI. Entre las invasivas se encuentran los registros con microelectrodos implantables intracorticales [103]. Los primeros pueden registrar actividad de distinto tipo dependiendo de su ubicación (en relación a las neuronas corticales) y del tipo de filtrado temporal que se aplique. Los registros de cada uno de estos electrodos pueden corresponder a la actividad de una unidad única, de múltiples unidades o a el potencial de campo local (LFP, del inglés Local Field Potential). Otra técnica de registro de la actividad eléctrica cerebral invasiva es la electrocorticografía (ECoG) que consiste en la aplicación de electrodos directamente sobre la corteza cerebral. Con esta técnica se obtienen mejores resoluciones espacial y temporal en comparación con la EEG y además de menor sensibilidad a artefactos, sin embargo no deja de ser una técnica invasiva que requiere de una craneotomía y tiene riesgos asociados (a pesar de que no se penetra la masa encefálica) [97].

Entre las técnicas no invasivas existe gran variedad de métodos de registro con

---

<sup>2</sup>del inglés “random forests”.

Tabla 2.1: Resumen de métodos de registro de actividad cerebral.

Método	Actividad	Precio	Portabilidad	Riesgo	Resolución Temporal	Resolución Espacial
EEG	Eléctrica	++	Portable	+	$\sim 0,05s$	$\sim 10mm$
MEG	Magnética	$\sim$	No-Portable	+	$\sim 0,05s$	$\sim 5mm$
fNIRS	Metabólica	+	Portable	+	$\sim 1s$	$\sim 5mm$
fMRI	Metabólica	-	No-Portable	$\sim$	$\sim 1s$	$\sim 1mm$
ECoG	Eléctrica	+	Portable	-	$\sim 0,003s$	$\sim 1mm$
Registros Intracorticales	Eléctrica	$\sim$	Portable	-	$\sim 0,003s$	$\sim 0,5mm$ a $\sim 0,05mm$

diferentes grados de resolución espacial o temporal, amplio rango de precios y distintas posibilidades de portabilidad. Aunque la técnica no invasiva más popular es la EEG, es posible también utilizar la magnetoencefalografía (MEG) para medir los campos electromagnéticos del cerebro. Además hay diversas técnicas que permiten medir la actividad cerebral en base al flujo sanguíneo a través de sus tejidos, entre estas últimas tienen potencial aplicación en BCI la espectroscopía del infrarrojo cercano funcional (fNIRS, del inglés Functional Near-Infrared Spectroscopy) y la resonancia magnética nuclear funcional (fMRI, del inglés Functional Magnetic Resonance Imaging) [103]. En la Tabla 2.1 se presenta un resumen de las características principales de los métodos de registro mencionados [66]. Cabe resaltar que en la presente tesis se trabajará con señales de EEG.

### 2.2.1. Electroencefalografía

El potencial de acción fue observado por primera vez en el año 1848 por el fisiólogo Emil du Bois Rémond. Luego, en 1875, Richard Caton observó potenciales eléctricos a nivel cerebral efectuando registros invasivos en animales. Finalmente Hans Berger acuñó el término EEG y fue el primero en publicar registros de actividad eléctrica cerebral en seres humanos. Desde esa época se han descrito las ondas cerebrales, los potenciales evocados y alteraciones de los patrones que se observan por ejemplo durante los ataques de epilepsia. Actualmente la EEG

es una técnica muy usada en neurología clínica. Es comúnmente empleada para determinar el tipo y la localización de la actividad epiléptica o para el análisis de los trastornos del sueño [83]. También tiene aplicación en el diagnóstico y/o tratamiento de otras disfunciones neurológicas como encefalopatías, infecciones, demencia o desórdenes psiquiátricos [88].

La EEG mide diferencias de potencial eléctrico entre distintos puntos en el cuero cabelludo. Este potencial en el cuero cabelludo se genera como resultado de la actividad neuronal, cuya acción puede ser modelada como un conjunto de múltiples fuentes de corriente distribuidas en un cuerpo conductor (la cabeza). Hay dos tipos principales de actividad eléctrica asociada con las neuronas: los potenciales de acción y los potenciales postsinápticos (PSP, del inglés Postsynaptic Potentials). Los potenciales de acción son impulsos eléctricos consistentes en una serie autopropagante de polarizaciones y despolarizaciones, que viajan desde el comienzo del axón en el cuerpo de la célula a los terminales del axón, donde se liberan los neurotransmisores. Los PSP son las diferencias de voltaje que surgen cuando los neurotransmisores se unen a los receptores en la membrana de la célula postsináptica. La duración de un potencial de acción es de una milésima de segundo, los PSP pueden durar entre decenas a cientos de milisegundos. Además, los PSP están limitados esencialmente a las dendritas y al cuerpo celular en lugar de viajar por el axón a velocidad relativamente constante. Estas características son las que permiten que los dipolos generados por los PSP se sumen en lugar de cancelarse. Dicha agregación de dipolos generada por la suma de corrientes eléctricas que fluyen de múltiples neuronas cercanas en un pequeño volumen de tejido se conoce como LFP. La actividad sincronizada de un gran número de neuronas corticales que compartan una orientación similar, puede constituir una fuente de corriente agregada considerable. Cada configuración diferente de los conjuntos de fuentes de corriente en el cerebro puede ser asociada a un estado cerebral distinto, y se manifiesta como potenciales eléctricos en el cuero cabelludo que pueden ser registrados utilizando la EEG. Dichas fuentes de corriente son mezcladas y filtradas espacialmente, lo que vuelve la reconstrucción de su posición y geometría a partir de los registros de EEG un problema inverso mal definido. Sin embargo, la localización temporal de la actividad neuronal no se ve afectada, y por tanto las señales de EEG contienen información temporal precisa respecto a los patrones de activación neuronal.

En su forma más básica los electroencefalógrafos son sistemas relativamente

sencillos. Están compuestos fundamentalmente por los sensores y el amplificador diferencial de potenciales. Los sensores se denominan electrodos y convierten el flujo de corrientes iónicas en la cabeza en corrientes electrónicas en los conductores. La amplitud de los potenciales debidos a la actividad neuronal que se pueden registrar en el cuero cabelludo oscila entre los 10 y 200  $\mu$ Voltios, por lo que se debe amplificar las diferencias de potencial registradas entre los electrodos.

Hay dos métodos diferentes de montaje para realizar registros de EEG: monopolar y bipolar. El montaje monopolar recoge la señal en el sitio activo y la compara con un electrodo de referencia común, el cual se ubica de manera que no se vea afectado por la actividad cerebral. Comúnmente se utilizan los mastoides o los lóbulos de las orejas pero también se puede utilizar la CAR. Por otro lado en el montaje bipolar el potencial registrado se toma entre dos electrodos activos. Hay tres razones principales por las cuales los registros monopolares se recomiendan en EEG. La primera es que debido a que el amplificador diferencial rechaza todo lo que es común a sus dos entradas, rechazará también cualquier actividad común a dos electrodos, y dado que en los registros monopolares uno de los sensores se encuentra en un sitio pasivo hay menores probabilidades de registrar actividad eléctrica común entre este y un sitio activo. En cambio en los registros bipolares es mayor la probabilidad de que el amplificador rechace información de interés. La segunda razón es que un registro bipolar puede derivarse de uno monopolar utilizando aritmética simple, mientras lo contrario es imposible (un registro bipolar no puede ser transformado en un uno monopolar). La tercera razón es que con la colocación monopolar el electrodo con la amplitud más grande es probablemente el más cercano al centro generador, lo cual no es necesariamente cierto en los registros bipolares [80].

Con el objetivo de estandarizar la ubicación de los electrodos para los registros de EEG se utiliza comúnmente el sistema internacional 10-20. Como puede verse en la Figura 2.2a este sistema, que consiste en ubicar los electrodos a distancias del 10% o 20% de los perímetros mediales o transversales de la cabeza. Puede verse en la Figura 2.2b como el sistema 10-20 puede ser ampliado al llamado sistema 10-10 ubicando electrodos intermedios [58], o incluso ubicando electrodos separados hasta por un 5% de los perímetros en el sistema 10-5 [44, 69].

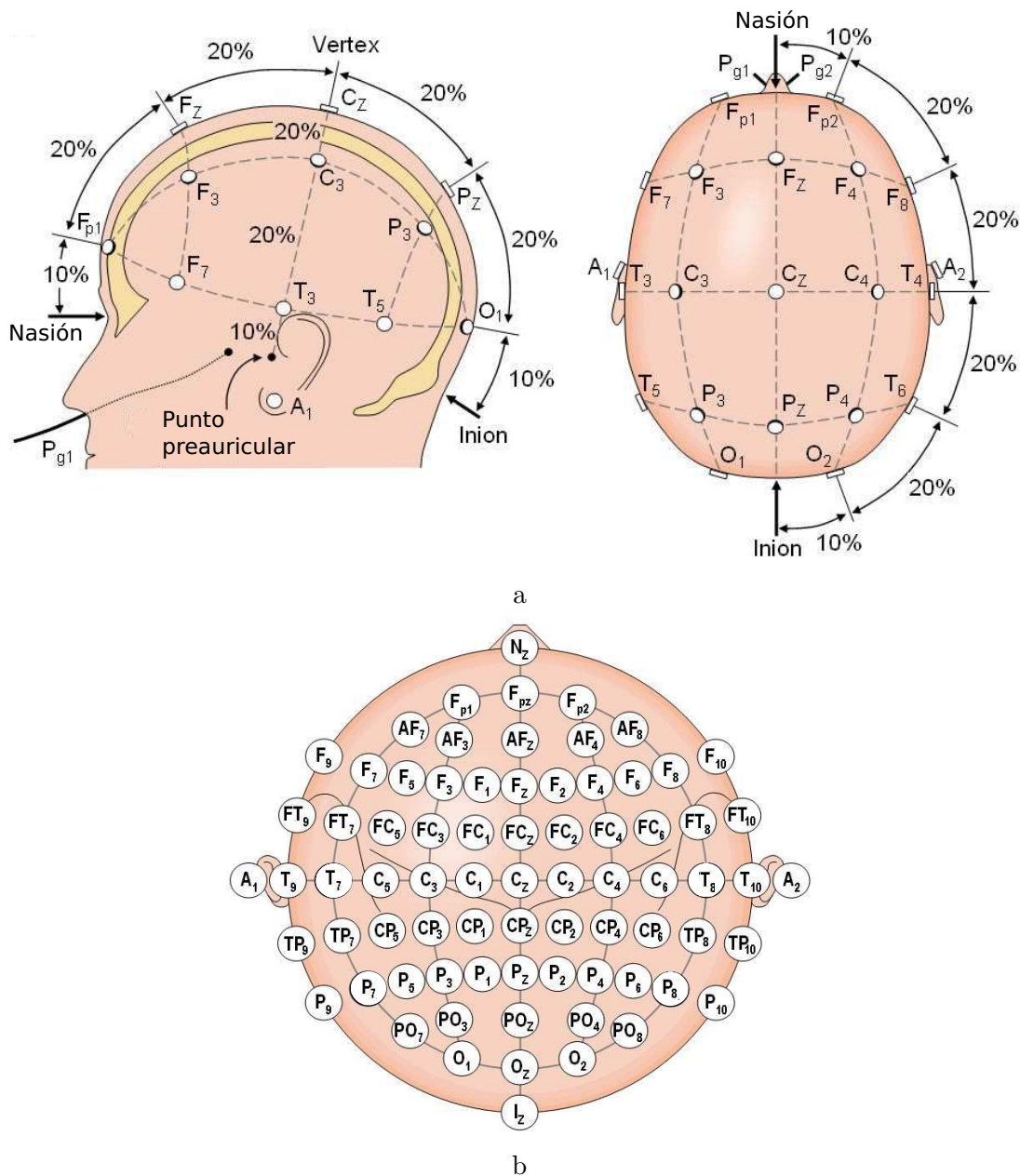


Figura 2.2: (a) Sistema internacional 10-20 visto desde la izquierda y desde arriba de la cabeza. A = Lóbulo de la oreja, C = central, Pg = naso-faríngeo, P = parietal, F = frontal, Fp = frontal polar, O = occipital.(b) Localización y nomenclatura de los electrodos intermedios en las posiciones intermedias al 10%. Adaptados de [58].

### 2.3. Potenciales relacionados con eventos

Los potenciales relacionados con eventos (ERP, del inglés Event Related Potentials) son variaciones que se observan en las señales de EEG y que se encuen-



tran en respuesta a algún proceso sensorial, motor o cognitivo. Estas respuestas pueden también ser observadas utilizando MEG, pero en estos casos no se habla ya de potenciales sino de campos relacionados con eventos.

Las formas de onda de los ERP consisten en una serie de deflexiones de potencial positivas y negativas, las cuales suelen relacionarse con diferentes componentes. La mayoría de estos componentes suele asociarse a fenómenos neurológicos subyacentes. Aunque algunos componentes de ERP se mencionan con siglas como por ejemplo la negatividad relacionada a errores (ERN, del inglés Error Related Negativity), la mayoría de los componentes son referidos por una letra (N/P) indicando su polaridad (negativo/positivo), seguida de un número que indica la latencia en milisegundos o la posición ordinaria del componente en la forma de onda. Por ejemplo, un pico negativo que es el primer pico sustancial en la forma de onda y que a menudo se produce alrededor de 100 milisegundos después de que se presenta un estímulo es comúnmente llamado N100 (indicando que es negativo y su latencia 100 ms después del estímulo) o N1 (indicando que es negativo y es el primer pico) [56].

Los ERP pueden dividirse en dos categorías, las ondas o componentes tempranos y los tardíos. Los primeros alcanzan sus picos aproximadamente dentro de los primeros 100 milisegundos después del estímulo y suele denominárselos como componentes sensoriales o exógenos, ya que dependen principalmente de las características del estímulo. Por otro lado, los ERP generados en partes posteriores reflejan la manera en que el sujeto evalúa el estímulo y se denominan cognitivos o endógenos.

Históricamente, el primer componente cognitivo reportado fue la variación contingente negativa (CNV, del inglés Contingent Negative Variation) [56]. Este componente aparece (como su nombre lo indica) como un cambio negativo en el potencial sobre el cuero cabelludo durante la preparación para la realización de una tarea. El siguiente gran avance fue el descubrimiento del componente P300 en el año 1965 [56]. Dicho hallazgo desató mucho interés en la temática, motivando el descubrimiento de otros componentes. Todos estos componentes tienen latencias y características morfológicas particulares, se producen en respuesta a diferentes fenómenos y tienen diversas aplicaciones dentro de las ciencias médicas [91].

De particular interés es el componente P300, que fue observado por primera vez como una respuesta a estímulos no predecibles, y si bien en las primeras investigaciones al respecto aparecía en los 300 ms puede presentar el pico positivo entre

los 250 ms y 700 ms [56]. Se considera que el P300 esta asociado al procesamiento y clasificación de estímulos y se lo clasifica como un potencial endógeno ya que su aparición depende de la reacción del sujeto y no de los atributos del estímulo. Desde su descubrimiento se ha separado el componente P300 en dos componentes el P3a y el P3b. El P3a tiene su máxima amplitud en la región frontal y se genera en tareas con estímulos que requieren de atención mientras que el P3b tiene su máxima amplitud en la región parietal y esta se genera durante el procesamiento de eventos improbables [74].

Cuando se adquieren señales de ERP se registra simultáneamente actividad eléctrica cerebral de fondo. La amplitud de los ERP suele ser muy pequeña en relación con dichas señales de fondo, con una SNR del orden de -30dB. Sumadas a las señales cerebrales de fondo también es común la contaminación debida a la actividad ocular, respiratoria, cardíaca o muscular. Estas son características intrínsecas de las señales que dificultan la clasificación y visualización de los ERP. Para afrontar estos inconvenientes se utilizan diversas técnicas desde filtrado espacial y/o temporal hasta estrategias mas complejas como las de separación ciega de fuentes (BSS, del inglés Blind Source Separation), pasando por el uso filtros de Wiener, filtros de Kalman y distintos tipos de filtros no lineales [62, 96]. Una de las estrategias mas utilizadas para mejorar la SNR de los ERP es la promediación coherente (CA, del inglés Coherent Averaging) [2].

## 2.4. Promediación coherente

La CA es una técnica ampliamente utilizada para obtener una señal repetitiva pequeña enmascarada por ruido aditivo no correlacionado. Consiste simplemente en promediar un gran número de registros sincronizados, para lo cual debe conocerse con precisión el instante en que se produce el fenómeno que genera la señal de interés. Esta técnica fue propuesta originalmente en la década del 50 [20] y se ha convertido en una de las técnicas básicas para el procesamiento de señales cerebrales. En el caso de la aplicación de CA a ERP se usan los estímulos para alinear las épocas a promediar.

Una de las principales suposiciones de la CA es que la señal registrada  $x : [0, T] \rightarrow \mathbb{R}^m$  (donde  $m$  es la cantidad de sensores) puede ser modelada como  $x(t) = r(t) + n(t)$ , donde  $r(t)$  es la respuesta a los estímulos y  $n(t)$  es el ruido. Además  $r(t)$  debe ser invariante con el tiempo y  $n(t)$  debe estar descorrelacionada

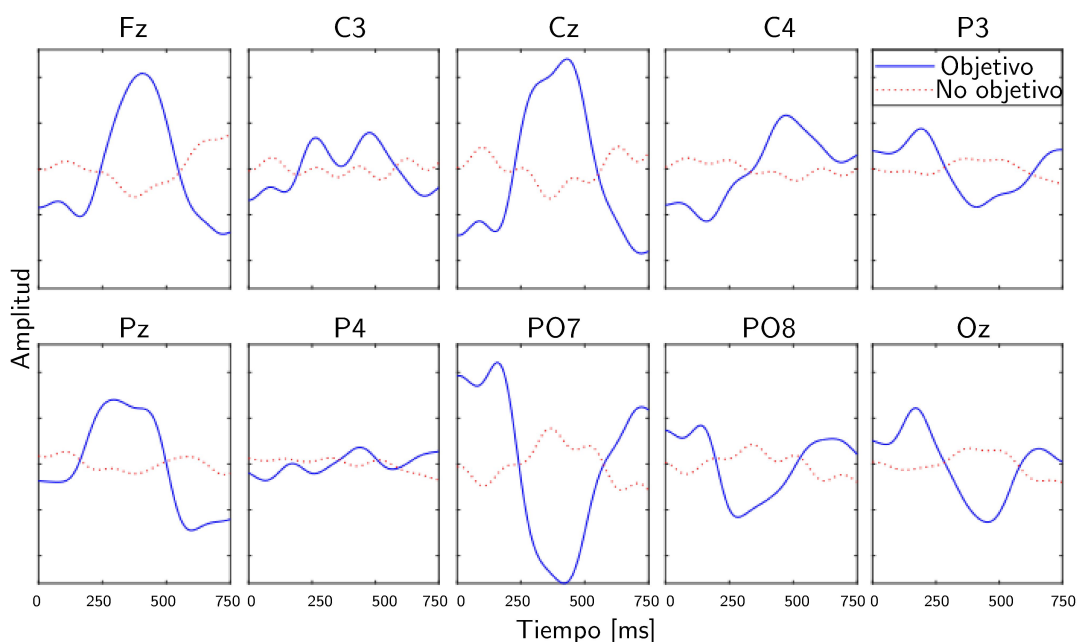


Figura 2.3: Señales pos-estimulo promediadas coherentemente.

con la respuesta, ser estacionaria y con media cero.

Si se cumplen dichas condiciones la CA mejora la SNR de la estimación de  $r(t)$  disminuyendo su varianza por un factor de  $\sqrt{N}$ , donde  $N$  es el número de muestras utilizado para la promediación [78]. Claramente aumentar  $N$  mejora la SNR de la estimación, pero agregar realizaciones es costoso en términos temporales, ya que el proceso bajo estudio debe repetirse para registrarlas. En general es deseable utilizar la menor cantidad posible de realizaciones necesarias para obtener las estimaciones de  $r(t)$ , y particularmente en el caso de las BCI utilizar valores de  $N$  muy grandes se vuelve prohibitivo.

Desde un principio la CA ha sido aplicada para estimar la morfología de potenciales evocados [21]. La técnica se ajusta bien a dicha tarea ya que las señales de potenciales evocados son repetitivas, se puede conocer en forma precisa cuando se producen y están enmascarados por ruido no correlacionado con los estímulos.

## 2.5. Aplicaciones de las BCI

Dadas las características de las BCI, su aplicación mas natural es la asistencia a personas con discapacidades motrices severas, tanto para su comunicación como para el control de dispositivos. Son múltiples las razones que pueden generar dis-

capacidades graves, tanto enfermedades neurodegenerativas y cerebro-vasculares como los traumatismos (particularmente en el tronco encefálico), pueden dañar o interrumpir de manera permanente o temporaria la comunicación del cerebro con gran parte del organismo. En ciertos casos las personas afectadas por este tipo de patologías conservan intacta su capacidad cognitiva, pero sufren afecciones a lo largo de las vías eferentes que imposibilitan efectuar acciones motoras. Se conoce como síndrome de enclaustramiento<sup>3</sup> el estado extremo en el que pueden derivar estas enfermedades. Quienes sufren de este síndrome conservan sus habilidades cognitivas pero son incapaces de realizar las acciones motoras deseadas. El síndrome de enclaustramiento clásico, según la definición de Bauer, Gerstenbrand y Ruml se caracteriza por inmovilidad total a excepción de los movimientos oculares verticales y el parpadeo [4]. A la fecha se han reportado varias patologías que en sus estadios avanzados conllevan al estado de enclaustramiento. Ejemplos de estas patologías son la esclerosis lateral amiotrófica (ELA), las lesiones de la médula espinal cervical (LMEC) y los infartos del tronco encefálico (ITE).

Diversos ejemplos de aplicaciones demuestran que las BCI pueden proporcionar beneficios reales en las vidas de sus usuarios. El primer ejemplo de uso de una BCI fue para permitir deletrear palabras sin la necesidad de realizar movimientos [27]. Se han utilizado BCI para controlar un brazo robótico que permita manipular el ambiente circundante [61]. Se las ha utilizado también para permitir comandar una silla de ruedas robótica [93]. En [60] se implementó una BCI que permite el movimiento de un cursor en tres dimensiones. Por otro lado, además de sus usos como sistema de comunicación y control, las BCI han demostrado que pueden ser utilizadas en la rehabilitación de pacientes tetraplégicos para aumentar la plasticidad neuronal [22].

Claramente los pacientes con discapacidades motrices son quienes pueden beneficiarse en mayor grado con el uso de BCI, sin embargo también se han propuesto otras aplicaciones que pueden asimismo resultar útiles a usuarios sanos. Las llamadas BCI pasivas utilizan las señales cerebrales como una herramienta de medición fisiológica que provee información sobre el estado emocional y/o cognitivo del individuo, brindando un medio de comunicación alternativo que permita mejorar la experiencia de uso de distintos tipos de interfaz hombre-máquina (HMI, del inglés Human Machine Interface) en general y de BCI activas en particular [13, 29]. En [68] se discute su uso en video-juegos, aplicación con una gran

---

<sup>3</sup>del inglés “locked-in syndrome”.

cantidad de potenciales usuarios. En el caso de utilizar BCI para el comando en video-juegos existe el desafío de que la respuesta de las interfaces tiene que ser precisa y rápida, ya que compite con otros medios de control que en el caso de la asistencia a personas con discapacidades motrices no siempre son una opción.

Las estrategias desarrolladas en esta tesis están orientadas a mejorar los métodos de asistencia a personas con discapacidades motrices, sin por eso estar necesariamente acotadas a dichas aplicaciones.

## 2.6. Señales electrofisiológicas utilizadas en la implementación de BCI

Las BCI pueden ser clasificadas en base a la señal electrofisiológica utilizada para implementarla. Asociada al tipo de señal es necesario plantear una estrategia que pueda explotar el fenómeno. Dichas combinaciones de fenómeno electrofisiológico y estrategia de funcionamiento suelen llamarse *paradigmas*. En esta sección se describen brevemente algunas de las señales más utilizadas para la implementación de BCI no invasivas y con un poco más de detalle las estrategias basadas en potenciales P300.

### 2.6.1. BCI basadas en ritmos sensorimotrices

Los ritmos sensorimotrices (SMR, del inglés Sensorimotor Rhythms) son oscilaciones en los campos electromagnéticos del cerebro registrados sobre las regiones sensorimotoras de la corteza cerebral. Estas oscilaciones pertenecen típicamente a las bandas de los ritmos mu, beta o gamma. Hace décadas que existe evidencia de que tanto la realización como la imaginación de movimientos de los miembros induce cambios en la actividad rítmica que se puede registrar en la corteza sensorimotora [65]. Estos cambios en los SMR pueden ser detectados y utilizados para implementar BCI.

Durante el movimiento voluntario se produce una disminución de los SMR denominada desincronización relacionada con eventos (ERD, del inglés Event Related Desynchronization) [72]. Los SMR también pueden aumentar debido a eventos sensorimotrices, como la finalización de una actividad motriz. Dichos incrementos se denominan sincronización relacionada con eventos (ERS, del inglés Event Related Synchronization) [71]. Los patrones espaciales y temporales con

los que se producen las ERD y ERS son variables y dependen de diversos factores, entre ellos de cual es el miembro que se mueve (o imagina mover). En [103] pueden encontrarse mas detalles al respecto de dichos patrones de sincronización y desincronización.

A través de diversas técnicas de procesamiento de señales basadas principalmente en análisis frecuencial y espacial es posible utilizar los SMR para implementar aplicaciones de comunicación y/o control [103]. Algunos estudios sugieren que incluso personas con discapacidades severas podrían utilizar este tipo de BCI [60].

### 2.6.2. BCI basadas en potenciales evocados visuales de estado estacionario

Los potenciales evocados visuales son deflecciones positivas o negativas de voltaje asociadas temporalmente a un estímulo visual específico. Dichos estímulos pueden consistir por ejemplo en destellos luminosos, cambios de color abruptos o la aparición de una imagen. Si la estimulación se produce en forma de eventos rápidos repetitivos, si bien cada estímulo particular evocará una respuesta, la sucesión de respuestas se ve como una oscilación de estado estacionario. Estas oscilaciones de voltaje se conocen como potenciales evocados visuales de estado estacionario (SSVEP, del inglés Steady State Visual Evoked Potentials). Si los SSVEP son procesados utilizando análisis frecuencial suele observarse como resultado espectros con picos en la frecuencia de estimulación correspondiente.

La forma más común en la que una BCI basada en SSVEP es implementada es presentado varias fuentes de destellos luminosos repetitivos, cada una con distinta frecuencia y ubicada en un lugar distinto del campo visual del usuario. Cada fuente representa una opción distinta (e.g. arriba/abajo/derecha/izquierda) que el usuario puede seleccionar fijando la mirada en la que desee. A través del análisis frecuencial de las señales de EEG (principalmente de la región occipital) es sistema puede identificar en cual de las fuentes el usuario está fijando la mirada. Algunas variantes de este sistema básico están descritas en [9]. Cabe mencionar que al utilizar estas estrategias es necesario que el usuario fije la mirada, para lo que se requiere control de los músculos oculares. Esto podría dificultar el uso de BCI basadas en SSVEP a sujetos con determinadas patologías [102].

### 2.6.3. BCI basadas en potenciales corticales lentos

Los potenciales corticales lentos (SCP, del inglés Slow Cortical Potentials) son cambios en el voltaje registrado con EEG que están asociados a eventos. Típicamente los SCP constituyen cambios negativos en el potencial, relativamente lentos, y que preceden movimientos (reales o imaginados) o la realización de tareas cognitivas. La modulación de los SCP se basa en el entrenamiento de los usuarios para modular su actividad mental realizando tareas que produzcan dichos potenciales [31].

Típicamente las BCI basadas en SCP presentan al usuario una sucesión de ensayos<sup>4</sup>, donde cada ensayo permite al usuario realizar una selección. Cada ensayo esta dividido en dos partes, un intervalo donde se registra la línea de base y un intervalo de control activo. Durante la etapa de control el usuario puede generar SCP si desea realizar la selección, o continuar en reposo, para no diferenciar su actividad de la actividad de base [48].

### 2.6.4. BCI basadas en potenciales P300

Como se explicó en la Sección 2.3, cuando estímulos infrecuentes se mezclan con estímulos frecuentes en tareas que requieren de la atención del sujeto, este evoca potenciales P300 en respuesta a los estímulos infrecuentes. Este paradigma de estimulación se conoce como paradigma del “bicho raro”<sup>5</sup> o del evento extraño.

El paradigma del bicho raro puede utilizarse para implementar BCI basadas en potenciales P300. Hay distintos métodos que permiten el uso de los potenciales P300 como método de comunicación, siendo el primero y mas conocido la matriz propuesta por Farwell y Donchin [27] que se utiliza para evocar ERP utilizando estímulos visuales. En este caso se presenta al usuario de la BCI una matriz de  $6 \times 6$  como la de la Figura 2.4 y se le solicita prestar atención al caracter que desea seleccionar. Durante su funcionamiento se intensifican repetidamente y al azar las filas y columnas de a una por vez. En cada bloque se intensifican cada una de las filas o columnas una vez, resultando en 12 intensificaciones por bloque. Al intensificarse las fila y columna que contienen el caracter que el usuario desea seleccionar, gracias a la atención prestada por el sujeto a ese estímulo en particular y a la menor frecuencia de estímulos objetivo contra los estímulos no

---

<sup>4</sup>del inglés “trial”.

<sup>5</sup>del inglés “oddball”.

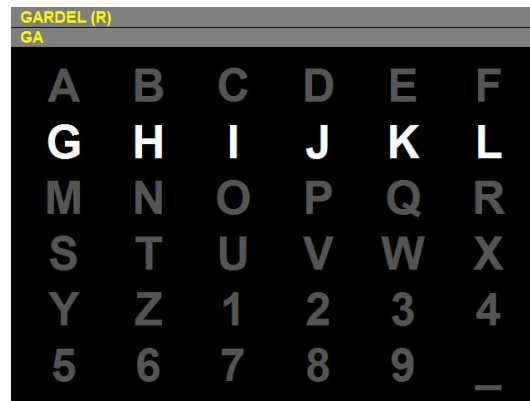


Figura 2.4: Matriz de estimulación con el diseño propuesto originalmente por Farwell y Donchin.

objetivo, se evocan potenciales P300 en la corteza cerebral del usuario. Suponiendo que las intensificaciones de las fila y columna correspondientes al caracter objetivo evocaron potenciales P300 y que las intensificaciones del resto de las filas y columnas no evocaron potenciales P300, se obtendrán dos épocas conteniendo respuestas evocadas P300 y diez épocas sin respuesta en cada bloque. Identificando que épocas tienen respuesta P300 y que épocas no presentan la respuesta se pueden identificar las fila y columna correspondientes al caracter elegido. Para determinar que intensificaciones de la matriz evocan ERP, el sistema debe ser capaz de resolver el problema de clasificación binaria entre épocas objetivo (con ERP) y épocas no objetivo (sin ERP). Este problema de clasificación binario es difícil de resolver debido a las características intrínsecas de las señales, en particular la desfavorable SNR. Debido a esto es en general necesario realizar varias repeticiones (entre 12 y 15 típicamente) del bloque de estimulación para así poder promediar entre si las respuestas correspondientes a cada fila/columna o mejorar la confianza de la clasificación de épocas únicas repitiéndola para varios ejemplos.

Diversas variables pueden modificarse en la estrategia del deletreador previamente descrito. Entre ellas se incluyen modificar los colores, la fuente y el tamaño de los caracteres [82] o el tamaño de la matriz de caracteres y la duración del intervalo inter estímulo (ISI, del inglés Inter Stimulus Interval) [87]. Además se han propuesto estrategias en las cuáles en lugar de realizar las intensificaciones en filas y columnas se intensifican caracteres individuales [33], o conjuntos de caracteres no adyacentes [95]. También se ha propuesto agrupar los caracteres en regiones y realizar la selección por etapas, primero seleccionando una región y posteriormente seleccionando alguno de los caracteres pertenecientes a la región



[28]. Se han implementado además BCI basadas en P300 donde los estímulos no son visuales sino táctiles [12] o auditivos [67]. Sin embargo, independientemente de la estrategia de estimulación utilizada o el ajuste de las variables de cada método todas estas BCI se basan en la clasificación entre porciones de señales de EEG con y sin potenciales P300.

## 2.7. Software para BCI

Para realizar experimentos o desarrollar BCI es necesaria la utilización de software que permita la implementación y coordinación de los distintos bloques del sistema. En particular el programa debe ser capaz de manipular y almacenar las señales registradas, realizar el análisis, procesamiento y clasificación de dichas señales y proveer salidas adecuadas tanto para el comando de un dispositivo externo como para la retro-alimentación al usuario. Además es deseable que las variables de cada bloque así como el protocolo de funcionamiento sean configurables.

Dependiendo de las necesidades específicas de cada experimento, las aplicaciones a desarrollar pueden demandar de desarrollos de software considerablemente grandes, especialmente en caso de ser necesarios sistemas que funcionen en tiempo real donde se deba coordinar en forma precisa el registro de las señales con los estímulos. Por otro lado el desarrollo individual de aplicaciones por cada laboratorio dificulta la colaboración entre distintos grupos. Debido a estos problemas surge naturalmente la necesidad de desarrollar plataformas de software generales para aplicaciones de BCI, que permitan a los investigadores realizar modificaciones en sus diseños de BCI minimizando la necesidad de programación y facilitando la cooperación entre grupos y la publicación de resultados.

Para permitir que estas plataformas generales sean de utilidad a diversos grupos de investigación es necesario que las mismas permitan la implementación de diversos diseños de BCI en forma fácil y modular. Además es deseable que el sistema sea compatible con distintos componentes de hardware (particularmente equipos de registro de señales) y distintos sistemas operativos. Hay varias plataformas desarrolladas que cumplen en distinta medida con estos requisitos, entre estas las más conocidas son el OpenVIBE [75] y el BCI2000 [84]. Sin embargo también existen otras alternativas como el BCI++, xBCI, BF++ y Pyff. Puede encontrarse una descripción mas detallada al respecto en [14].



---

## Capítulo 3

# Métodos Computacionales

En este capítulo se describirán algunos métodos computacionales utilizados en la presente tesis para entrenar, analizar o comparar los modelos propuestos.

### 3.1. Optimización

Los problemas de optimización matemática consisten básicamente en encontrar la mejor opción entre un conjunto de alternativas. En su forma más común un problema de optimización busca encontrar el valor óptimo de la variable  $x \in V$ , tal que se minimice (o maximice) una función matemática  $f : U \rightarrow \mathbb{R}$  (llamada función de costo) sujeta a la restricción de que la solución  $x \in V$ , donde  $V \subseteq U$ . Comúnmente esto se expresa con la siguiente notación:

$$\operatorname{argmin}_{x \in V} f(x). \tag{3.1}$$

Este tipo de problemas surge naturalmente en muchas áreas de las ciencias e ingeniería. En particular en el contexto del aprendizaje maquina y el procesamiento de señales, estos problemas tienen un rol preponderante.

Se han propuesto múltiples métodos para solucionar problemas de búsqueda de extremos, y comúnmente se los denomina algoritmos de optimización. Dichos algoritmos son en muchas ocasiones diseñados buscando maximizar el rendimiento computacional y garantizar la convergencia. Para conseguirlo se hace uso de las características de la función de costo  $f$  y del espacio de soluciones factibles  $V$ . Aquí describiremos únicamente métodos que pueden ser aplicables a problemas no lineales generales, excepto por la restricción de que  $f$  debe ser derivable.

Cuando la función de costo es derivable, es conveniente utilizar su gradiente para determinar sus puntos estacionarios y/o encontrar las direcciones en las que disminuye. Los métodos que hacen uso solo de los gradientes de la función objetivo suelen llamarse algoritmos de primer orden, en contraste con los métodos que también hacen uso de la matriz Hessiana que son llamados de segundo orden [104].

Entre los métodos de primer orden se destaca el del gradiente descendente, que busca sucesivamente las direcciones de mayor gradiente de la función, e itera con pasos en dichas direcciones. Son múltiples las variantes de este método que han sido propuestas [104].

Entre los algoritmos de segundo orden el más representativo es el método de Newton, que se basa en encontrar las raíces del gradiente de la función objetivo utilizando la información provista por la matriz Hessiana. Para la mayoría de los problemas el método de Newton converge en menos iteraciones que los algoritmos de primer orden, sin embargo, obtener y guardar los valores de la Hessiana resulta computacionalmente costoso, especialmente para problemas con muchas variables. Debido a esto se han propuesto múltiples métodos que utilizan estimaciones de la Hessiana y son conocidos como cuasi-Newtonianos. Entre estos métodos se destacan los que utilizan la aproximación de rango simétrico 1 (SR1, del inglés Symmetric Rank 1) y la de Broyden-Fletcher-Goldfarb-Shanno (BFGS) [104].

Otra familia de algoritmos de optimización es la de los métodos de gradiente conjugado (CG, del inglés Conjugate Gradient). Estos fueron originalmente diseñados para resolver problemas lineales, pero se los extendió posteriormente para ser aplicables también a problemas no lineales. Se trata de un método iterativo, donde en cada iteración  $k$  primero se obtiene la dirección del paso  $d_k$ , luego se calcula la longitud de paso  $\alpha_k$  y finalmente se actualiza la posición de la iteración al nuevo punto  $x_{k+1}$ . Para obtener la dirección  $d_k$  se utiliza el gradiente en el punto actual  $\nabla f(x_k)$ , modificado por la dirección de la iteración anterior  $d_{k-1}$ . La fórmula utilizada para determinar el uso de la dirección previa para el cálculo de la dirección actual varía entre distintos métodos [35]. Para calcular la longitud del paso se plantea un problema de optimización en una dimensión, donde se evalúa la función en la dirección deseada. A pesar de requerir generalmente más iteraciones que los métodos cuasi-Newtonianos para converger los algoritmos de CG son muy utilizados debido al bajo costo computacional por iteración.

La gran mayoría de los métodos de optimización son incapaces de diferenciar entre mínimos globales y locales, por lo que la existencia de mínimos locales puede constituir un problema, particularmente si son numerosos y/o significativamente peores que el mínimo global. Este problema es común en el entrenamiento de ANN.

## 3.2. Regularización

Usualmente en las áreas de aprendizaje maquina y problemas inversos se utilizan métodos de regularización. Estos ayudan a prevenir el sobreajuste y/o a estabilizar las soluciones de problemas mal condicionados.

En particular, en el contexto del aprendizaje maquina, es difícil, o incluso imposible conocer la complejidad y estructura óptima para modelar los sistemas o procesos de interés. En general la solución óptima para este problema es utilizar modelos regularizados que en cierta medida puedan ajustar su complejidad en forma adecuada.

Consideremos un problema donde se busca encontrar los parámetros  $w$  que optimicen la función  $f$  según un criterio determinado por la función de costo  $J$  dado el conjunto de datos  $X$ . Si dicho problema es mal condicionado o particularmente propenso al sobreajuste, suele agregarse un término de regularización  $R$  a la función a optimizar obteniéndose,

$$\min_w (J(f(\cdot; w), X) + \lambda R(w)), \quad (3.2)$$

donde  $\lambda \in \mathbb{R}$  determina el peso relativo del término de regularización. Al resolver el problema presentado en [3.2](#) se minimizan simultáneamente  $J$  y  $R$ . En general  $R(w)$  penaliza la complejidad de  $f$ , por ejemplo favoreciendo su suavidad. Los beneficios de minimizar la complejidad pueden entenderse a través del principio de la navaja de Occam.

La forma mas sencilla de la función de penalización  $R$  consiste en una suma del cuadrado de los parámetros de la función  $f$ . En otras palabras, sea  $w = (w_1, \dots, w_n)^T$  un vector que contiene los parámetros de  $f$  entonces  $R(w) = \|w\|^2$ . Las técnicas que utilizan este tipo de términos cuadráticos han surgido en variadas disciplinas, en el campo de la estadística se las llama técnicas de encogimiento<sup>1</sup>

---

<sup>1</sup>del inglés “shrinkage”.

porque tienden a reducir el valor de los parámetros, mientras que en el ámbito de las redes neuronales se le llama decaimiento de pesos<sup>2</sup>.

Para definir completamente el problema dado por [3.2](#), además de  $R$  es necesario también determinar el valor de  $\lambda$ . Para determinar dicho parámetro no existe un método óptimo, pero se han propuesto en el contexto de los problemas inversos varios métodos clásicos como el principio de discrepancia de Morozov [\[64\]](#), la curva-L y la validación cruzada generalizada [\[30\]](#). A pesar de esto no se trata aún de un problema cerrado, ya que se siguen proponiendo métodos para atacarlo [\[36\]](#).

En particular, el método de la curva-L es aplicable a problemas mas generales que los otros métodos [\[24\]](#), [\[64\]](#). Este consiste en elegir varios candidatos para  $\lambda$ , dados por  $\lambda_i$  y que cumplan con  $0 < \lambda_1 < \lambda_2 < \dots < \lambda_n < \infty$ , y resolver  $n$  veces el problema dado por [3.2](#) tomando  $\lambda = \lambda_i$  con  $i = 1, \dots, n$ . Para cada uno de los  $\Theta_i$  óptimos obtenidos como solución a los problemas correspondientes (planteados con  $\lambda_i$ ) se grafican posteriormente los puntos dados por  $(\log(J(f(\cdot; \Theta_i), X)), \log(R(\Theta_i))) \in \mathbb{R}^2$ , formando una curva al unirlos. Es importante resaltar aquí que el codominio de las funciones que componen cada término de la función de costo regularizada ( $J$  y  $R$ ), debe estar incluido en  $\mathbb{R}_{>0}$ . Empíricamente se considera que el punto óptimo de  $\lambda$  se encuentra lo mas cerca posible del punto de inflexión convexo de la curva.

Se puede ver un ejemplo de curva-L idealizada en la Figura [3.1](#). En este caso se la presenta en forma de curva continua parametrizada por el valor de  $\lambda$ . Se puede observar también en dicha gráfica la posición que ocuparía el  $\lambda$  óptimo. Si bien en este ejemplo la curva-L presentada es continua, en la práctica las curvas se construyen utilizando solo un conjunto finito de puntos.

Comúnmente el método de la curva-L se utiliza para problemas lineales, sin embargo, su uso se puede extender para la aplicación a problemas no lineales. De hecho puede probarse que si la función de costo original  $J$  puede escribirse como la norma-2 de alguna función no lineal de los parámetros  $\Theta$  y el término de regularización se escribe como la norma-2 de dichos parámetros, entonces la curva-L correspondiente será monotónicamente decreciente y convexa [\[34\]](#).

En algunos casos el método de la curva-L no converge a los valores óptimos de los parámetros de regularización [\[100\]](#). A pesar de este inconveniente y debido a su flexibilidad, la curva-L es uno de los métodos mas utilizados en el contexto

---

<sup>2</sup>del inglés “weight decay”.

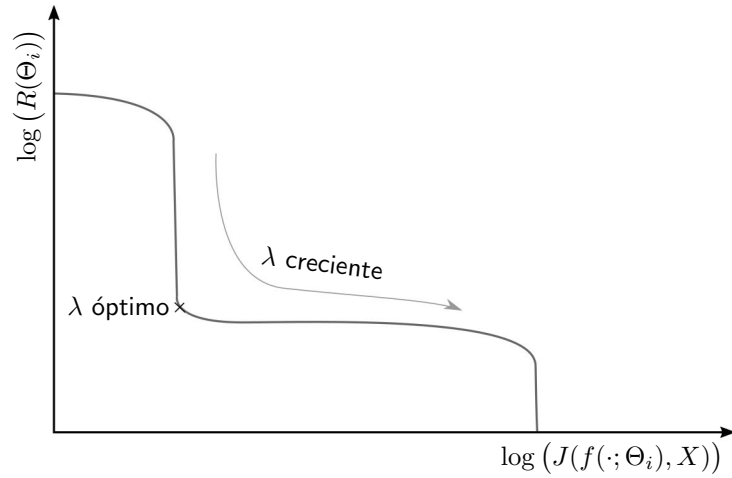


Figura 3.1: Ilustración idealizada de curva-L. Se marcan el parámetro óptimo y su dirección de crecimiento.

de la regularización de problemas inversos.

### 3.3. Métodos de aprendizaje maquina

En esta sección describiremos dos métodos de reconocimiento de patrones que fueron utilizados durante el desarrollo de la tesis. Estos algoritmos son el de  $k$ -vecinos más cercanos (KNN, del inglés  $k$ -Nearest Neighbors) y las SVM.

#### 3.3.1. K-vecinos más cercanos

El método de KNN es un algoritmo de reconocimiento de patrones no paramétrico. El mismo estima el valor de la función de densidad de probabilidad, o directamente la probabilidad a posteriori de que un elemento pertenezca a una clase determinada, a partir de la información proporcionada por el conjunto de patrones de entrenamiento. El KNN es un ejemplo de método de aprendizaje perezoso<sup>3</sup> ya que no realiza generalizaciones del espacio de búsqueda (mas allá de los datos pertenecientes al conjunto de entrenamiento) hasta que no se realiza una consulta con un nuevo patrón.

Sea  $V$  el espacio que contiene a los patrones del problema de interés y  $d : V \times V \rightarrow \mathbb{R}$  una función capaz de medir la distancia entre dos elementos pertenecientes a dicho espacio. Dado un patrón desconocido  $x \in V$ , se calcula su distancia

<sup>3</sup>del inglés “lazy learning”.

$d(x, x_j)$  con cada uno de los  $x_j$  patrones de entrenamiento, se identifican los  $k$  vecinos con las distancias mas pequeñas, y se determina la clase del patrón desconocido  $x$  como la clase mayoritaria entre el conjunto de dichos  $k$  vecinos.

Para problemas binarios se selecciona preferentemente valores de  $k$  impar. Como medida de distancia suelen utilizarse las distancias euclídea, Manhattan y de Hamming, entre otras. También es posible siempre definir distancias nuevas en forma arbitraria y se han propuesto métodos para definir de manera automática métricas o pseudo-métricas basadas en el conjunto de datos.

A pesar de la simplicidad del algoritmo detrás del KNN, este método puede definir hipersuperficies de separación no lineales extremadamente complejas.

Los clasificadores basados en KNN comúnmente tienen rendimientos inferiores a los de otros métodos, sin embargo, los destaca entre los algoritmos de clasificación no lineales, convirtiéndolo en una buena opción para el presente trabajo en el que se utilizan los clasificadores como una medida de la bondad de las técnicas de extracción de características utilizadas.

### 3.3.2. Máquina de vectores de soporte

Las SVM constituyen un conjunto de algoritmos de aprendizaje maquina sumamente exitoso y popular. Fueron propuestas inicialmente en la década del 60 y popularizadas a mediados de los años 90 [17] y desde entonces han sido ampliamente utilizadas.

Las SVM originalmente están constituidas por un clasificador lineal binario supervisado. La particularidad que diferencia a las SVM de otros métodos lineales es el hecho de que no solo se busca un hiperplano que separe las clases, sino que las separe de forma óptima. En el contexto de las SVM, óptimo significa que el hiperplano debe ubicarse de modo de maximizar la mínima distancia a los patrones de entrenamiento (el doble de dicha distancia mínima es llamado margen). En general el ajuste de una SVM se reduce a solucionar un problema de optimización que maximice el margen. Se han propuesto múltiples métodos para solucionar el problema de optimización que define a las SVM, alcanzándose un excelente rendimiento en términos computacionales. Los clasificadores basados en SVM son además robustos y presentan gran capacidad de generalización.

Si bien las SVM originalmente sirven para solucionar problemas de clasificación binarios linealmente separables, se han propuesto métodos que reducen en gran medida dichas restricciones. Actualmente se puede utilizar las SVM para



resolver problemas de regresión, problemas de clasificación multiclase, problemas de agrupamiento<sup>4</sup> no supervisado y especialmente problemas no lineales utilizando transformaciones a espacios de mayor dimensión por medio del uso del “truco del núcleo”<sup>5</sup>.

---

<sup>4</sup>del inglés “clustering”.

<sup>5</sup>del inglés “kernel trick”.



---

## Capítulo 4

# Redes Neuronales Artificiales

Las ANN son modelos de aprendizaje maquina que consisten en unidades simples que pueden ser conectadas entre si. Estas unidades sencillas son llamadas *neuronas*, planteado una analogía con las neuronas biológicas que son las unidades fundamentales de procesamiento del sistema nervioso animal. Existen diversos modelos de neuronas artificiales, que varían tanto en su aplicación como en la similitud que presentan con sus contrapartes biológicas.

Los primeros modelos de ANN con mas de una capa de neuronas fueron propuestos en los años 60 y se popularizaron recién a finales de los años 80. Sin embargo, y a pesar de que las ANN con al menos una capa oculta pueden aproximar cualquier función [41], otras técnicas obtuvieron durante mucho tiempo mejores resultados tanto en problemas de clasificación como en problemas de regresión.

Los métodos de aprendizaje profundo (DL, del inglés Deep Learning), avances en el hardware y la disponibilidad de grandes cantidades de datos han favorecido el resurgimiento de las ANN. De hecho en los últimos 10 años las técnicas de DL han adquirido un rol protagónico en la comunidad del aprendizaje maquina. Luego de mejorar los rendimientos de referencia en muchas tareas como la clasificación de dígitos escritos a mano [40], el reconocimiento de fonemas [32], clasificación de imágenes [47] y detección de objetos [16]. Puede encontrarse una revisión histórica sobre ANN, particularmente centrada en aprendizaje profundo en [85].

En esta tesis se usan modelos de neurona básicos y de aplicación práctica. Se priorizó el rendimiento de los modelos como métodos de aprendizaje maquina por sobre la similitud con los sistemas biológicos.

## 4.1. Neuronas artificiales

Entre los modelos de neuronas artificiales mas sencillos se destaca el que consiste en una combinación lineal de las entradas a la que se le aplica una función de activación  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ . Las neuronas artificiales con la capacidad de aprender a partir del ajuste de sus pesos fueron propuestas por primera vez por Rosenblatt, y posteriormente analizadas y refinadas en los 60s por Minsky y Papert [63]. El modelo básico propuesto en estos primeros años a seguido siendo utilizado, no solo como una ANN completa en si misma (e.g. perceptrón) sino también interconectándola con otras unidades para formar ANN mas complejas.

Dados el vector de pesos  $w = (w_1, \dots, w_n)^T \in \mathbb{R}^n$ , el valor del sesgo  $b \in \mathbb{R}$  y el vector de entrada  $x = (x_1, \dots, x_n)^T \in \mathbb{R}^n$  la salida de cada neurona artificial esta dada por

$$y = \sigma \left( \sum_{i=1}^n w_i x_i + b \right). \quad (4.1)$$

Es de utilidad además expresar la salida de una red neuronal como una función arbitraria  $f$  de los parámetros y de las entradas, por ejemplo para el caso de la red definida en la Ecuación 4.1 podemos reescribirla como

$$y = f(x; w, b). \quad (4.2)$$

La Figura 4.1 se puede ver un esquema del modelo básico de neurona artificial que comprende una combinación lineal de las entradas seguida de una función de activación  $\sigma$ .

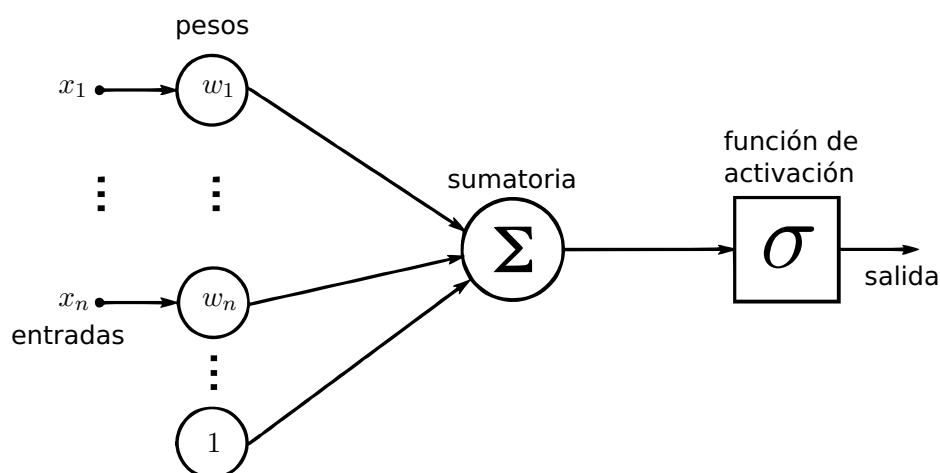


Figura 4.1: Esquema de una neurona artificial básica.

Se han propuesto varias funciones de activación con diferentes propiedades. Entre ellas se destacan la función identidad, la escalón, la sigmoidea y la unidad lineal rectificada (ReLU, del inglés Rectified Linear Unit) cuyas expresiones matemáticas son respectivamente:

$$\sigma_{\text{identidad}}(u) = u, \quad \sigma_{\text{escalón}}(u) = \begin{cases} 0, & \text{si } u < 0 \\ 1, & \text{si } u \geq 0 \end{cases}, \quad (4.3)$$

$$\sigma_{\text{sigmoidea}}(u) = \frac{1}{1 + e^u}, \quad \sigma_{\text{ReLU}}(u) = \begin{cases} 0, & \text{si } u < 0 \\ u, & \text{si } u \geq 0 \end{cases}. \quad (4.4)$$

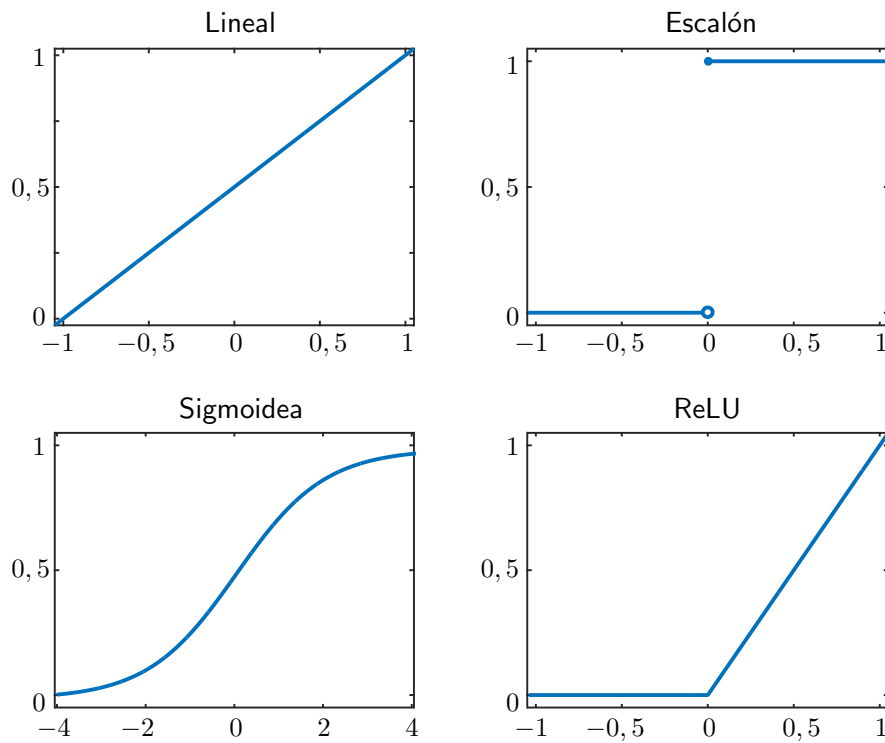


Figura 4.2: Algunas funciones de activación.

Para entrenar los modelos basados en redes neuronales es necesario plantear sus correspondientes funciones de costo a optimizar. Estas dependerán de la tarea que se desea desempeñar con la red ya que, dependiendo de como se defina la función de costo, una misma estructura puede cumplir funciones muy distintas. Una forma básica de función de costo es a través del promedio de una función de pérdida<sup>1</sup>  $L : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$ , la tiene la ventaja de que permite utilizar una cantidad

<sup>1</sup>del inglés “loss function”.

$N$  de patrones arbitraria en cada iteración del entrenamiento y es fácilmente diferenciable (dada la derivada de  $L$ ). Esta forma general de función de costo  $J$  puede definirse como

$$J \doteq \frac{1}{N} \sum_{i=1}^N L(x^i, y_i), \quad (4.5)$$

donde  $x^i = (x_1^i, \dots, x_n^i)^T$  y  $y_i$  son las  $i$ -ésimas entrada y salida. Hay múltiples definiciones de función de pérdida para redes neuronales, entre las cuales mencionaremos dos de las más básicas que nos servirán particularmente para desarrollar el resto de las funciones de costo utilizadas en esta tesis. La primera se define usando el error cuadrático medio como:

$$L_{ECM}(f(x^i; w, b); y_i) \doteq \|f(x^i; w, b) - y_i\|^2. \quad (4.6)$$

Es importante notar aquí que no es necesario aplicar la norma en el caso de la red neurona básica presentada en esta sección cuya salida  $y \in \mathbb{R}$ , sin embargo para modelos mas complejos la salida de la red puede ser vectorial. La segunda función de pérdida a definir esta basada en la entropía cruzada<sup>2</sup> y puede expresarse como

$$L_{CE}(f(x^i; w, b); y_i) \doteq -(y_i) \log(f(x^i; w, b)) + (1 - y_i) \log(1 - f(x^i; w, b)). \quad (4.7)$$

En este caso por simplicidad se presenta la función de pérdida de entropía cruzada para una salida escalar, sin embargo para extenderla a múltiples salidas solo es necesario sumar la pérdida para cada salida. En el caso del modelo presentado en esta sección es posible utilizar cualquiera de las funciones de pérdida presentadas para construir una función de costo a partir de la Ecuación 4.5

## 4.2. Redes neuronales artificiales prealimentadas

Para construir una ANN, suele ser necesario conectar varias neuronas artificiales. Respecto a la forma en que se realizan dichas conexiones, una de las principales taxonomías entre las ANN está dada por la existencia o no de conexiones recurrentes, denominándose las ANN recurrentes y ANN prealimentadas<sup>3</sup>.

<sup>2</sup>del inglés “cross entropy” (también conocida como ).

<sup>3</sup>del inglés “feedforward”.

respectivamente. Las ANN prealimentadas son significativamente más sencillas de entrenar y analizar, evitando la necesidad de agregar retardos y de sincronizar las unidades. En esta tesis trabajaremos exclusivamente con ANN prealimentadas.

En la arquitectura más comúnmente utilizada de ANN prealimentada, se conectan las salidas de un conjunto de neuronas denominado capa, a las entradas de neuronas pertenecientes a una segunda capa, la cual puede a su vez ser conectada con otra capa subsiguiente.

Para incluir varias unidades y capas es necesario reescribir la notación utilizada en la Sección 4.1 e introducir elementos nuevos. Denominaremos así a cada parámetro de peso o sesgo como  $w_{j,i}^l \in \mathbb{R}$  y  $b_j^l \in \mathbb{R}$  respectivamente, con  $i, j, l \in \mathbb{N}$  donde  $l$  indica la capa,  $j$  distingue las unidades de una misma capa e  $i$  indica de cual de los pesos de la unidad se trata. La salida de cada unidad, también llamada activación de la unidad se escribe como  $a_j^l \in \mathbb{R}$  donde  $l \in \mathbb{N}$  y  $j \in \mathbb{N}$  indican la capa y unidad respectivamente. Por último  $n_l \in \mathbb{N}$  es el número de unidades de la capa  $l$ . Utilizando esta notación la salida de cada unidad puede escribirse como:

$$a_j^l = \sigma \left( \sum_{i=1}^{n_l} w_{j,i}^l a_i^{l-1} + b_j^l \right). \quad (4.8)$$

Puede verse como cada unidad recibe como entradas las salidas de la capa sucesiva. La salida  $y$  está constituida por las activaciones de las neuronas de la última capa. Cabe resaltar que si bien por simplicidad se utilizó  $\sigma$  para denotar la función de activación, esta no es necesariamente la misma para todas las unidades.

En la Figura 4.3 se puede observar una ANN con la estructura de capas descrita. En dicha Figura  $W^l = (w_{j,i}^l) \in \mathbb{R}^{n_l \times n_{l+1}}$  y  $b^l = (b_1^l, \dots, b_{n_l}^l)^T$  son respectivamente la matriz de pesos y vector de sesgos de la capa  $l$ .

Otra forma de representar la ANN es expresar la transformación aplicada por la capa  $l$  como una función de la salida de la capa  $l - 1$  y sus correspondientes parámetros ajustables  $\Theta^l$ . De este modo la salida de la red podría escribirse como  $y = h(g(f(x; \Theta^1); \Theta^2); \Theta^3)$ , donde  $f$ ,  $g$  y  $h$  son las transformaciones que aplican las primera, segunda y tercera capas, respectivamente. En la Figura 4.4 se puede ver una red equivalente a la de la Figura 4.3 expresada con esta otra notación.

Las funciones de costo que se pueden utilizar para entrenar estos modelos son las mismas que fueron presentadas en la Sección 4.1.

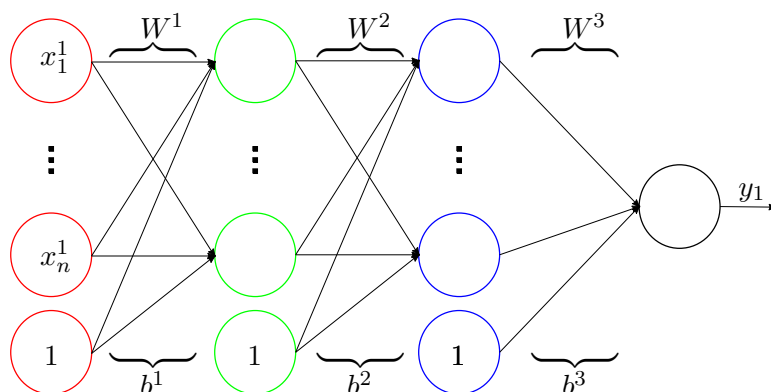


Figura 4.3: Esquema desagregado de una red neuronal artificial básica.

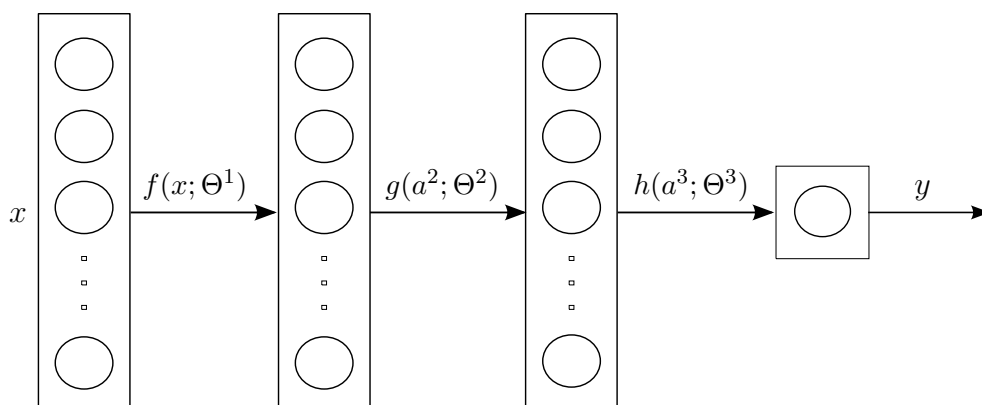


Figura 4.4: Esquema funcional de una red neuronal artificial.

### 4.3. Unidades softmax

La función softmax (SF, del inglés Softmax Function) es una generalización de la función logística de una a varias dimensiones [11]. Como en el caso de la función logística, la SF puede implementarse a través de una ANN llamada red neuronal artificial softmax (SNN, del inglés Softmax Artificial Neural Network) donde cada unidad de salida está típicamente asociada a cada clase. Cada salida puede ser interpretada como la probabilidad de que el patrón de entrada  $x \in \mathbb{R}^n$  corresponde a una clase dada. La salida de la  $\ell$ -ésima unidad en una SNN con  $M$  unidades de salida está dada por la función  $h_\ell : \mathbb{R}^n \times \mathbb{R}^{M \times n} \rightarrow \mathbb{R}$  cuya expresión es

$$h_\ell(x; \Gamma) \doteq \frac{e^{\Gamma_{\ell,:} x}}{\sum_{j=1}^M e^{\Gamma_{j,:} x}}, \quad (4.9)$$

donde  $\Gamma \in M \times n$  es una matriz cuyos componentes son los pesos de la red y cuya  $j$ -ésima fila es  $\Gamma_{j,:}$ .



Sea  $M \in \mathbb{R}$  el número de unidades de salida de la SNN. Sean además  $X = (x^1, \dots, x^N)$  y  $y = (y_1, \dots, y_N)^T$  donde  $x^i = (x_1^i, \dots, x_n^i)^T$  y  $y_i$  son las  $i$ -ésimas muestra y su correspondiente etiqueta. El costo asociado con la SNN es la función  $J_S : \mathbb{R}^{n \times N} \times \mathbb{R}^N \times \mathbb{R}^{M \times n} \rightarrow \mathbb{R}$  definida como

$$J_S(X, y; \Gamma) \doteq -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M \mathbf{I}_j(y_i) \log(h_j(x^i; \Gamma)), \quad (4.10)$$

donde  $\mathbf{I}_j : \mathbb{R} \rightarrow \{0, 1\}$  se define como

$$\mathbf{I}_j(y_i) = \begin{cases} 1, & \text{si } y_i = j \\ 0, & \text{si } y_i \neq j \end{cases}. \quad (4.11)$$

## 4.4. Autocodificadores

Un tipo particular de ANN a considerar esta constituido es el autocodificador (AE, del inglés Autoencoder). Estos modelos se entrenan para replicar las entradas en las salidas de la red, con al menos una capa oculta que restrinja la representación y evite que el modelo aprenda la función de identidad. En su forma más simple un AE consiste en una ANN prealimentada con una capa oculta con menos unidades que la entrada (esta es la restricción más simple).

Sean  $\Theta \in \mathbb{R}^{(n+1) \times m}$  y  $\Phi \in \mathbb{R}^{(m+1) \times n}$  definidas como  $\Theta = [W_\Theta^T, b_\Theta]^T$  y  $\Phi = [W_\Phi^T, b_\Phi]^T$  donde  $W_\Theta \in \mathbb{R}^{n \times m}$ ,  $b_\Theta \in \mathbb{R}^m$ ,  $W_\Phi \in \mathbb{R}^{m \times n}$  y  $b_\Phi \in \mathbb{R}^n$  son las matrices de pesos y los vectores de sesgo para el codificador y el decodificador respectivamente. Sean además  $f(\cdot; \Theta) : \mathbb{R}^n \rightarrow \mathbb{R}^m$  la función de codificación que transforma las entradas al espacio del código interno,  $g(\cdot; \Phi) : \mathbb{R}^m \rightarrow \mathbb{R}^n$  la función de decodificación que transforma el código de vuelta al espacio de las entradas y  $L : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$  es la función de pérdida. En base a esta notación, la función de costo  $J_A : \mathbb{R}^{n \times N} \times \mathbb{R}^{(n+1) \times m} \times \mathbb{R}^{(m+1) \times n} \rightarrow \mathbb{R}$  puede definirse como:

$$J_A(X; \Theta, \Phi) \doteq \frac{1}{N} \sum_{i=1}^N L(g(f(x^i; \Theta); \Phi), x^i) \quad (4.12)$$

donde  $X$  y  $N$  se definen como previamente.

Con el objetivo de prevenir el sobre ajuste de los parámetros suele aplicarse penalizaciones para suavizar las soluciones. Agregando términos de regularización

cuadráticos sobre los pesos se obtiene la función de costo

$$J_{AR}(X; \Theta, \Phi) \doteq \frac{1}{N} \sum_{i=1}^N L\left(g(f(x^i; \Theta); \Phi), x^i\right) + \lambda(\|\Theta\|^2 + \|\Phi\|^2), \quad (4.13)$$

donde  $\lambda$  es el hiperparámetro que pesa la importancia relativa del término de penalización.

Utilizando el error cuadrático como función de pérdida (i.e.  $L(x, y) = \|x - y\|^2$ ) obtenemos la siguiente función de costo regularizada:

$$J_{AR}(X; \Theta, \Phi) \doteq \frac{1}{N} \sum_{i=1}^N \|g(f(x^i; \Theta); \Phi) - x^i\|^2 + \lambda(\|\Theta\|^2 + \|\Phi\|^2). \quad (4.14)$$

## 4.5. Autocodificador ralo

Para restringir la capacidad de representación de la capa oculta se puede diseñar el autocodificador de modo que el número de características del código sea menor al número de características de la entrada (i.e.  $m < n$ ). Otra restricción posible (pero no excluyente) consiste en imponer al código un requerimiento de rareza<sup>4</sup>. Hay varias maneras de estimular la rareza del código, pero en el caso del autocodificador ralo (SAE, del inglés Sparse Autoencoder) consiste en agregar un término de penalización  $P_s : \mathbb{R}^{m \times N} \rightarrow \mathbb{R}$ , convirtiendo la función de costo en una función multiobjetivo de la forma

$$J_{SAE}(X; \Theta, \Phi) \doteq \frac{1}{N} \sum_{i=1}^N L\left(g(f(x^i; \Theta); \Phi), x^i\right) + \beta P_s(f(X; \Theta); \rho) \quad (4.15)$$

donde  $\beta$  determina la importancia relativa del término de rareza y  $\rho$  es un parámetro de la función de penalización.

Sea

$$\hat{\rho}_j = \hat{\rho}_j(X; \Theta) = \frac{1}{N} \sum_{i=1}^N (x^i; \Theta) \quad (4.16)$$

la activación media de la  $j$ -ésima unidad oculta sobre el conjunto de datos  $X$ , el término  $P_s$  puede ser definido utilizando la divergencia de Kullback-Leibler

---

<sup>4</sup>del inglés “sparsness”.

(KLD, del inglés Kullback-Leibler Divergence) obteniéndose:

$$P_s(X, \Theta; \rho) \doteq \sum_{j=1}^m KL(\rho \parallel \hat{\rho}_j) = \sum_{j=1}^m \left( \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \right), \quad (4.17)$$

donde  $\rho$  es la activación media deseada. Es importante notar que para asegurar que la ecuación 4.17 está bien definida, es conveniente excluir el 0 y el 1 como posibles valores de  $f_j$ ,  $\forall j = 1, \dots, m$ .

Otra posible definición de la penalización de rareza consiste en usar la norma  $\ell_1$  del vector cuyos componentes son la activación media de las unidades ocultas,  $\hat{\rho} = (\hat{\rho}_1, \dots, \hat{\rho}_n)^T \in \mathbb{R}^n$  [42]. Utilizando esta idea se puede definir el término de penalización de la siguiente forma:

$$P_{\ell_1}(X, \Theta) \doteq \sum_{j=1}^m |\hat{\rho}_j| = \sum_{j=1}^m \left| \frac{1}{N} \sum_{i=1}^N f_j(x^i; \Theta) \right|. \quad (4.18)$$

## 4.6. Autocodificadores de limpieza de ruido

El autocodificador de limpieza de ruido (DAE, del inglés Denoising Autoencoder) es una variante de AE optimizado para reconstruir señales limpias  $x^i \in \mathbb{R}^n$  partiendo de sus versiones artificialmente corrompidas  $\tilde{x}^i \in \mathbb{R}^n$  [98]. La función de costo de los DAE se puede escribir, en forma análoga a la ecuación 4.12, como

$$J_{DAE}(X, \tilde{X}; \Theta, \Phi) \doteq \frac{1}{N} \sum_{i=1}^N L(g(f(\tilde{x}^i; \Theta); \Phi), x^i), \quad (4.19)$$

donde  $X$ ,  $\Theta$  y  $\Phi$  son como previamente definidos y  $\tilde{X} = (\tilde{x}^1, \dots, \tilde{x}^N)$ .

Utilizando una función de pérdida cuadrática, la función de costo se puede escribir como

$$J_{DAE}(X, \tilde{X}; \Theta, \Phi) \doteq \frac{1}{N} \sum_{i=1}^N \|g(f(\tilde{x}^i; \Theta); \Phi) - x^i\|^2. \quad (4.20)$$

Distintas transformaciones han sido propuestas para el proceso de corrupción de los datos. Entre estas se incluyen (1) la adición de ruido Gaussiano isotrópico, (2) el ruido de enmascaramiento, donde se selecciona al azar una determinada cantidad de las características a las que se modifica su valor llevando su valor a cero y (3) el ruido de “sal y pimienta” donde una fracción de las características

(seleccionadas al azar) se fija al máximo o mínimo valor (seleccionado al azar con 50 % de chance para cada opción) [99].

Los DAE son apropiados para encontrar características descriptoras robustas para patrones limpios. Sin embargo en el caso de que los datos originales estén ya significativamente contaminados con ruido la transformación utilizando DAE no necesariamente mejora la representación.

## 4.7. Otras variantes de autocodificador

Existen muchas otras variaciones de AE que agregan restricciones o modifican de alguna manera la función de costo original, siempre buscando mejorar las representaciones ocultas que el modelo puede encontrar. Entre estas variantes hay que destacar al autocodificador de suavizado (SMAE, del inglés Smoothing Autoencoder), donde la codificación de cada muestra se utiliza para reconstruir ejemplos de la que se encuentren en una determinada vecindad definida en base a dicha muestra [54]. Por otro lado en [76] proponen el autocodificador contractivo (CAE, del inglés Contractive Autoencoder), que incluye un término de regularización que mide la sensibilidad de las representaciones que obtiene con respecto a la entrada. Otro modelo muy exitoso, especialmente en aplicación para la generación de imágenes, es el autocodificador variacional (VAE, del inglés Variational Autoencoder), propuesto en [46], donde se desarrollan las bases de este modelo utilizando métodos variacionales.

Todos estos modelos buscan evitar que la ANN aprenda la función identidad con distintas estrategias y favorecer su capacidad de capturar información importante aprendiendo así representaciones útiles.

## 4.8. Entrenamiento de redes neuronales artificiales

El proceso de entrenamiento de una ANN es una de las etapas fundamentales del diseño de estos sistemas, y que por medio de las distintas variantes que se pueden aplicar, les otorga una gran flexibilidad. Los métodos de entrenamiento pueden dividirse en entrenamiento supervisado y entrenamiento no supervisado [77]. La diferencia entre estos métodos es básicamente que en un caso la salida

que la red debería producir es conocida, y se utiliza para supervisar como tiende a comportarse dicha red, mientras en el otro caso no se conoce la salida de la red. Esta diferenciación es importante porque condiciona directamente la estructura de la función de costo  $J$ . Además, la disponibilidad de datos no etiquetados a dado lugar en parte a la evolución de varios métodos de entrenamiento no supervisado o parcialmente supervisado que pueden encuadrarse dentro de las técnicas de DL.

Básicamente entrenar una ANN consiste en ajustar sus parámetros con el fin de obtener un determinado comportamiento. En general el problema del entrenamiento de la red puede escribirse como

$$\Theta_O = \underset{\Theta}{\operatorname{argmin}} J(X; \Theta), \quad (4.21)$$

donde  $J$  es la función de costo,  $\Theta$  contiene los parámetros ajustables del modelo,  $\Theta_O$  contiene los parámetros óptimos y  $X$  contiene los datos disponibles para el entrenamiento.

### 4.8.1. Retropropagación

Entre los métodos fundamentales utilizados en el entrenamiento de las ANN se destaca el algoritmo de retropropagación (BP, del inglés Backpropagation). Este método fue introducido originalmente en los años 70 y popularizado posteriormente luego de la publicación del trabajo de Rumelhart et. al. [79]. Dicho artículo describe varias redes neuronales donde la BP resulta mas rápida que el resto de los algoritmos de aprendizaje que se utilizaban hasta la fecha.

Esencialmente la BP permite calcular eficientemente la derivada parcial de la función de costo  $J(X; \Theta)$ , respecto a cada uno de los parámetros en  $\Theta$ . Las derivadas parciales nos permitan saber que tanto cambia el costo al variar cada uno de los pesos resultando en una herramienta muy útil para resolver el problema dado por la ecuación 4.21.

Para poder escribir explícitamente las expresiones de las derivadas parciales dadas por el método de BP desagregamos  $\Theta$  en sus componentes escalares a los que, siguiendo la notación convencionalmente utilizada, denominaremos  $w_{j,i}^l$  y  $b_j^l$  con  $i, j, l \in \mathbb{N}$  donde  $w$  indica que se trata de uno de los pesos de la sumatoria de entradas a la unidad,  $b$  que se trata de un valor de sesgo,  $l$  indica la capa,  $j$  distingue las unidades de una misma capa e  $i$  indica de cual de los pesos de la unidad se trata. Reescribimos también la ecuación 4.1 separándola en dos por-

ciones y agregando los subíndices correspondientes para que cada unidad básica pueda ser ubicada dentro de una ANN con múltiples capas:

$$z_j^l = \sum_{i=1}^n w_{j,i}^l a_i^{l-1} + b_j^l, \quad (4.22a)$$

$$a_j^l = \sigma(z_j^l), \quad (4.22b)$$

donde  $z_j^l \in \mathbb{R}$  y  $a_j^l \in \mathbb{R}$  representan respectivamente las entrada y salida de la  $j$ -ésima unidad de la  $l$ -ésima capa. Para resolver el problema que esta notación presenta en la primer capa (i.e.  $l = 1$ ) definimos  $a_i^0 = x_i$ .

Utilizando esta notación podemos presentar las cuatro ecuaciones básicas del algoritmo de BP. La primera es una medida del error en la capa de salida:

$$\delta_j^L = \frac{\partial J}{\partial a_j^L} \sigma'(z_j^L), \quad (4.23)$$

donde  $L$  es el índice de la capa de salida ( $l = 1, \dots, L$ ) y  $\sigma' : \mathbb{R} \rightarrow \mathbb{R}$  es la derivada de la función de activación. El primer factor del lado derecho de la Ecuación [4.23](#) ( $\partial J / \partial a_j^L$ ) suele ser fácil de calcular teniendo en cuenta las funciones de costo comúnmente utilizadas, como las basadas error cuadrático medio o la entropía cruzada.

La segunda ecuación es una medida del error en la  $l$ -ésima capa en base al error de la  $(l+1)$ -ésima capa:

$$\delta_j^l = \sum_k w_{k,j}^{l+1} \delta_k^{l+1} \sigma'(z_j^l). \quad (4.24)$$

En este caso se hace evidente el porque del nombre del algoritmo, el error se “retropropaga” de las capas superiores hacia la entrada.

Las tercera y cuarta ecuaciones nos permiten escribir las expresiones para las derivadas parciales respecto a las conexiones  $w$  y a los sesgos  $b$  respectivamente:

$$\frac{\partial J}{\partial b_j^l} = \delta_j^l, \quad (4.25a)$$

$$\frac{\partial J}{\partial w_j^l} = a_k^{l-1} \delta_j^l. \quad (4.25b)$$

Se puede encontrar este algoritmo desarrollado en mas profundidad en [\[77\]](#).

## 4.9. Aprendizaje profundo

Como vimos, las ANN están inspiradas en el funcionamiento de las neuronas biológicas. Si bien las redes neuronales reales utilizan muchísimas capas de neuronas, se ha encontrado que cuando se utilizan ANN con muchas capas ocultas los métodos de entrenamiento y arquitecturas clásicos no funcionan bien. Como alternativa han surgido nuevas arquitecturas y métodos de entrenamiento de ANN que dieron lugar a una nueva rama del aprendizaje maquina denominada DL [85]. El DL engloba una serie de algoritmos que intentan modelar los datos a partir de abstracciones de alto nivel empleando múltiples capas de procesamiento y transformaciones no-lineales [52].

En esta sección presentaremos algunos aspectos y modelos de ANN profundas relevantes para el desarrollo o discusión de esta tesis. Sin embargo, si bien no se desarrollarán, no se puede dejar de mencionar las redes neuronales recursivas (RNN, del inglés Recursive Neural Networks) y en particular entre estas a las redes de gran memoria de corto plazo (LSTM, del inglés Long Short Term Memory), arquitectura que han tenido un amplio desarrollo en los últimos años [50].

### 4.9.1. Pre-entrenamiento de redes multicapa

Las redes prealimentadas descritas en la Sección 4.2 probablemente constituyan la arquitectura más importante de las ANN. Entre las primeras estrategias exitosas desarrolladas para poder obtener ANN prealimentadas con múltiples capas se destaca la de *pre-entrenar* la red de una capa a la vez utilizando métodos no supervisados y luego, utilizando BP, realizar un ajuste fino de la red pre-entrenada. El primer modelo propuesto que utiliza dicha estrategia se denominó red de creencia profunda (DBN, del inglés Deep Belief Network) [39].

Si bien en [39] se utiliza la red de Boltzman restringida (RBM, del inglés Restricted Boltzman Machine) como arquitectura de base para construir cada una de las capas pre-entrenadas, también se han utilizado los AE con mucho éxito para el pre-entrenamiento de cada capa [7]. La mayoría de las variantes de AE pueden ser apiladas para crear o inicializar ANN con múltiples capas, dando énfasis a distintas características de cada representación [6].

El proceso de entrenamiento de un AE apilado consiste en entrenar de forma voraz muchos AE. Cada capa de decodificación se descarta y las representaciones de características obtenidas con las capas de codificación se usan para alimentar

las entradas de otro AE. Dicho proceso se repite tantas veces como capas se desee apilar. En el ajuste final se suele utilizar BP sobre toda la red en base a la comparación con la salida deseada. En la Figura 4.5 se esquematiza el preproceso de entrenamiento de un AE apilado típico. Las razones por las que este enfoque voraz da buenos resultados han sido ampliamente estudiadas [6].

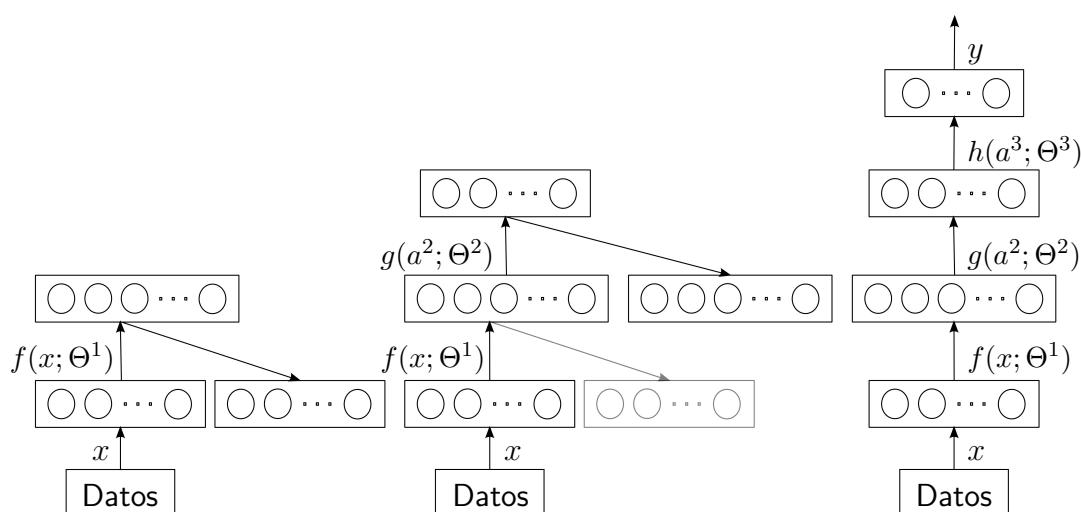


Figura 4.5: Esquema de entrenamiento de autoencoder apilado.

La potencialidad de este enfoque se hace evidente a partir del desempeño logrado mediante un pre-entrenamiento generativo seguido por un ajuste discriminativo en varias tareas de reconocimiento de patrones estándar [35]. Estas técnicas han sido aplicadas exitosamente a la señal de voz [38] y en particular al reconocimiento de emociones utilizando dicha señal [90]. El análisis y clasificación mediante redes profundas también se ha aplicado con éxito en el campo de las señales biomédicas [43].

En los últimos años se ha observado que en los casos en que se cuenta con gran cantidad de datos etiquetados el pre-entrenamiento capa por capa no es necesario, ya que es posible entrenar la ANN profunda completa de una sola vez con menores probabilidades de caer en mínimos locales malos [50]. Para las arquitecturas de ANN con muchas capas en las que no se utiliza pre-entrenamiento se han diseñado estrategias que previenen el sobreajuste y aceleran el entrenamiento, en particular cabe destacar la técnica de *dropout* [89].



### 4.9.2. Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN, del inglés Convolutional Neural Networks) están diseñadas para aprovechar las características de datos que estén naturalmente estructurados en la forma de arreglos (1D para señales y secuencias; 2D para imágenes o espectrogramas de audio; y 3D para vídeo o imágenes volumétricas). Esto se debe a cuatro características clave de estas redes: las conexiones locales, los pesos compartidos, la puesta en común<sup>5</sup> y el uso de muchas capas. La arquitectura típica de una CNN está estructurada como una serie de etapas con tres tipos fundamentales de capas: capas convolucionales, capas de transformación no lineal y capas de puesta en común [49].

Se han aplicado las CNN en forma exitosa en múltiples ocasiones que se remontan a principios de 1990, comenzando con las ANN con retardos temporales para reconocimiento de habla [101] hasta trabajos mucho más recientes en reconocimiento de rostros [92]. Se las ha usado para reconocimiento de emociones en señales de voz [59] y en video [45]. En el contexto de las BCI se han utilizado para aprender filtros espaciales óptimos en base a los datos [15].

---

<sup>5</sup>del inglés “pooling”.



---

## Capítulo 5

# Autocodificador de Estimación de Promedios Coherentes

Como se describió en el Capítulo 4 en los últimos años se han producido múltiples avances en el campo del aprendizaje maquina de la mano de las nuevas técnicas basadas en ANN. Algunas de estas técnicas han sido aplicadas en la comunidad de las BCI, adaptándolas en algunos casos para resolver problemas específicos del área. En [15] se utilizan CNN para aprovechar las correlaciones espaciales y temporales en las señales de EEG. En dicho trabajo se busca maximizar las tasas de reconocimiento de señales de ERP a través del uso de filtros espaciales y temporales obtenidos a partir de los datos. En [1] los autores proponen la utilización de una DBN como los clasificadores débiles de un conjunto<sup>1</sup> para mejorar la tasa de reconocimiento de señales de SMR registradas con EEG. En [53] se propone la utilización de DAE con el objetivo atacar el problema de los datos faltantes en las señales de EEG. Este es un problema común que puede ser el resultado de artefactos que obligan a descartar segmentos enteros de la señal, o situaciones como la desconexión de sensores. En dicho artículo se muestra, que la potencia espectral de señales simuladas puede ser estimada satisfactoriamente partiendo de datos incompletos y que los resultados de clasificación de señales de SMR obtenidos utilizando la técnica propuesta son comparables con los obtenidos utilizando otras técnicas del estado del arte.

Partiendo de algunos de los modelos propuestos en el contexto del DL, en esta sección se describe la propuesta un nuevo algoritmo para clasificación y estimación de señales repetitivas enmascaradas por ruido aditivo descorrelacionado. Ésta

---

<sup>1</sup>del inglés “ensemble”.

nueva ANN, a la que nos referiremos como AEP, incluye en su función de costo términos de reconstrucción, clasificación y rareza, convirtiendo su entrenamiento en un problema de optimización multiobjetivo complejo. El AEP está inspirado por las técnicas de CA y por los DAE.

## 5.1. ¿Qué es una buena representación?

Para obtener buenas representaciones de nuestras señales de interés es necesario definir antes que hace a una representación buena. En una primera instancia podemos decir que dicha representación debe ser útil para que un sistema alcance un rendimiento mayor del que podría alcanzar en caso de no utilizarla. En este sentido, dependiendo del objetivo final del sistema, una buena representación podría ser la que le permita obtener, por ejemplo, tasas de reconocimiento mas altas o mejorar en mayor medida la SNR de las entradas. Sin embargo, es importante considerar que no es necesariamente la mejor opción plantear como objetivo una medida de rendimiento que maximice directamente el una función de costo relacionada con el objetivo final del sistema. Esto se ha hecho evidente en el éxito de las estrategias que utilizan entrenamiento no supervisado para posteriormente obtener un mayor rendimiento en las tareas de clasificación. A continuación mencionaremos algunos de dichos criterios, potencialmente capaces de mejorar las representaciones de las señales de interés.

Claramente es necesario que la representación conserve información de la señal original. Esto puede medirse en forma directa a través del la información mutua [55]. Sin embargo también puede darse un interpretación mas intuitiva: si una representación nos permite reconstruir al menos parcialmente, las señales de entrada entonces estará conservando información de dichas señales. Este es el criterio que se utiliza al entrenar AE. En el caso en que la interpretación intuitiva no sea suficiente puede demostrarse que entrenar un AE para minimizar el error de reconstrucción es equivalente a maximizar una cota inferior de la información mutua entre la entrada y la representación latente [99].

Como mencionamos previamente, el solo hecho de conservar la información no provee necesariamente de representaciones útiles. Dimos como ejemplo de la función identidad que si bien conserva perfectamente la información de la entrada, no nos provee de ninguna nueva revelación respecto a la estructura subyacente de los datos. Para obtener representaciones de mayor utilidad es en general necesario

introducir restricciones en los modelos. Una posible restricción es a través de la compresión de la representación, ajustando la cantidad de unidades ocultas del modelo a un valor menor al de la entrada. Esta reducción de la dimensión es justificable si suponemos que la entrada tiene menos grados de libertad que dimensiones o que gran parte de la información importante en la entrada esta contenida en un subespacio del espacio original. En general es difícil probar dichas suposiciones, sin embargo es un enfoque muy utilizado en la práctica y que resulta de utilidad.

Otra forma de restringir la representación es a través de restricciones suaves<sup>2</sup>. Estas toman la forma de penalizantes y entre otras opciones pueden forzar a los modelos a aprender códigos raros utilizando por ejemplo la norma- $\ell_1$  o la divergencia de Kullback-Leibler. Los AE raros utilizan este enfoque. Los códigos raros presentan múltiples características interesantes. Una de ellas es que se ha mostrado que la utilización de códigos raros permite en múltiples casos la obtención de representaciones similares a las obtenidas utilizando criterios de independencia estadística computacionalmente mas costosos [81]. Si bien son múltiples la características útiles de los códigos raros, su utilización no garantiza que la representación será discriminativa.

En el caso de los problemas de clasificación siempre es importante que el sistema retenga información discriminativa. Comúnmente no es necesario aplicar condiciones explícitas para que esto se cumpla, y en muchos problemas con cumplir con alguno de los otros criterios se puede guardar información útil para la clasificación en forma implícita. Sin embargo en los problemas con muy mala SNR corremos el riesgo de modelar unicamente el ruido y perder la información asociada a la clase. Esto es particularmente cierto para el caso de la clasificación de ERP, por lo que consideramos que en este problema es necesario forzar explícitamente al modelo a conservar información discriminativa.

Es también deseable que una representación sea robusta en el sentido de presentar estabilidad ante versiones corruptas de la entrada. Esto puede considerarse como un requisito equivalente a que la representación oculta esté menos contaminada por el ruido que la señal original, o a que el proceso extraiga las características mas representativas de los datos en lugar de modelar el ruido. En esta idea se basan los DAE [98].

Resulta de interés también que los modelos permitan la concatenación de va-

---

<sup>2</sup>del inglés “soft constraints”.

rias etapas de codificación en forma consecutiva. Esto permitiría pre-entrenar una red profunda utilizando como base cada uno de los codificadores. Desde el punto de vista teórico, el pre-entrenamiento no supervisado guiaría en forma natural al modelo a aprender representaciones incrementalmente abstractas. Hay indicios de esto al analizar las salidas de cada capa de algunos sistemas de reconocimiento de imágenes, donde van pasando por ejemplo del reconocimiento de bordes, al de formas y al de objetos sucesivamente. Desde el punto de vista práctico, si bien las redes con muchas capas tienen la capacidad de construir modelos altamente complejos, es difícil conseguir que la red aprenda los mejores modelos entre la gran cantidad de posibles de mínimos locales. El pre-entrenamiento inicializa la red en regiones más cercanas a mejores mínimos locales [25].

## 5.2. Término de reconstrucción

Como se describió en la Sección 2.4 al utilizar la CA se supone que cada respuesta tiene la misma latencia respecto a su correspondiente estímulo, que el ruido es aditivo y que no tiene correlación con la respuesta. Cuando se cumplen estas suposiciones la CA atenúa la amplitud del ruido, mejorando así la SNR. Para el caso de las BCI hay un compromiso entre la velocidad de operación y la cantidad de muestras que es necesario obtener. Es deseable obtener tasas de reconocimiento similares a las obtenidas para las señales promediadas utilizando la menor cantidad de registros posible. Para ello se propone el AEP como método para estimar las señales promediadas coherentemente usando una menor cantidad de épocas.

El AEP es una ANN con una capa oculta a la que se le solicita obtener en la salida una versión limpia de la entrada obtenida a través de CA, a diferencia de los AE, donde se establece la entrada como la salida deseada. Repitiendo por comodidad parte de la notación introducida en el el Capítulo 4,  $X = (x^1, \dots, x^N)$  y  $y = (y_1, \dots, y_N)^T$  donde  $x^i = (x_1^i, \dots, x_n^i)^T$  y  $y_i$  son los  $i$ -ésimas patrón y su correspondiente etiqueta y  $N \in \mathbb{R}$  es el tamaño muestral. Definimos también las matrices de pesos y los vectores de sesgo para el codificador y el decodificador respectivamente como  $\Theta = [W_\Theta^T, b_\Theta]^T$  y  $\Phi = [W_\Phi^T, b_\Phi]^T$  donde  $W_\Theta \in \mathbb{R}^{n \times m}$ ,  $b_\Theta \in \mathbb{R}^m$ ,  $W_\Phi \in \mathbb{R}^{n \times m}$  y  $b_\Phi \in \mathbb{R}^n$ . Además  $f(\cdot; \Theta) : \mathbb{R}^n \rightarrow \mathbb{R}^m$  es la función de codificación,  $g(\cdot; \Phi) : \mathbb{R}^m \rightarrow \mathbb{R}^n$  es la función de decodificación y  $L : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$  es la función de pérdida. En base a esta notación, el término de costo del AEP

debido a la reconstrucción  $J_R : \mathbb{R}^{n \times N} \times \mathbb{R}^{(n+1) \times m} \times \mathbb{R}^{(m+1) \times n} \rightarrow \mathbb{R}$  puede definirse como:

$$J_R(f(X; \Theta); \Phi) \doteq \frac{1}{N} \sum_{i=1}^N \left\| g(f(x^i; \Theta); \Phi) - \frac{x^i + \sum_{j \in R_i^K} x^j}{K+1} \right\|^2, \quad (5.1)$$

donde  $R_i^K$  denota un subconjunto de  $K$  índices seleccionados al azar sin reemplazo, correspondientes a  $K$  patrones (columnas de  $X$ ) de la misma clase que  $x^i$ , excluyendo a  $x^i$ . Este término de costo es similar a la función de costo del DAE (ecuación 4.20), con la diferencia de que en lugar de corromper la entrada se estima su versión limpia y se la propone como salida deseada.

Si extendemos la definición de la función de activación  $\sigma$  presentada en la Sección 4.1 para que al ser aplicada a un vector se obtenga su aplicación componente a componente, podemos definir la función de codificación para el AEP como  $f(x; \Theta) = \sigma(W_\Theta^T x + b_\Theta)$ . La salida de la función de codificación a una entrada  $x$  puede entenderse como su representación en la capa oculta. En forma análoga, sea  $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$  otra función de activación aplicada componente a componente, la función de decodificación se define como  $g(f; \Phi) = \phi(W_\Phi^T f(x; \Theta) + b_\Phi)$ . En este caso hay que prestar especial atención al hecho de que el rango de la función de activación  $\phi$  debe contener todos los valores posibles de la entrada  $x$ , ya que de otro modo nunca podrá realizar correctamente las aproximaciones a la salida deseada.

### 5.3. Término de clasificación

Con el fin de asegurar que las representaciones ocultas del AEP conservan información discriminativa, se agregan a la red unidades softmax que toman como entrada la salida de la capa oculta. En otras palabras, la  $\ell$ -ésima unidad softmax (ver Ec. 4.9) se alimenta con las activaciones de la capa oculta del AEP:

$$h_\ell(f(x; \Theta); \Gamma) = \frac{e^{\Gamma_{\ell,:} f(x; \Theta)}}{\sum_{j=1}^M e^{\Gamma_{j,:} f(x; \Theta)}}, \quad (5.2)$$

donde  $\Gamma$  es la matriz de pesos softmax. El término de costo asociado al poder de discriminación se define como:

$$J_D(f(X; \Theta), y; \Gamma) \doteq -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M \mathbf{I}_j(y_i) \log(h_j(f(x^i; \Theta); \Gamma)), \quad (5.3)$$

donde  $M \in \mathbb{R}$  es el número de unidades de salida de la SNN y  $\Theta$ ,  $\Gamma$  y  $y$  se definen como previamente.

Durante el entrenamiento la señal de error dada por la ecuación 5.3 es retropropagada a través de sus conexiones a los pesos de la capa de codificación, direccionándolos a que extraigan (o no descarten) información discriminativa. La capa softmax puede ser descartada cuando se finaliza el entrenamiento del AEP.

## 5.4. Términos de rareza y regularización

Para inducir rareza en la representación oculta se aplica un término de penalización  $P_S$  en la capa correspondiente. Este término es equivalente al utilizado con los SAE (ver Sección 4.5).

Esto permite además definir redes que busquen representaciones sobrecompletas.

También se agregan términos cuadráticos de regularización a todos los parámetros ajustables de la red. Estos términos ayudan a prevenir el sobre-ajuste de la red a los datos de entrenamiento y a suavizar las soluciones, brindándoles robustez frente a la presencia de *outliers*.

## 5.5. AEP completo

Sumando los términos de reconstrucción, discriminación y las penalizaciones, podemos escribir la función de costo del AEP como:

$$J(X, y; \Theta, \Phi, \Gamma) \doteq J_R(f(X; \Theta); \Phi) + \gamma J_D(f(X; \Theta), y; \Gamma) + \beta P_S(f(X; \Theta); \rho) + \lambda_\Theta \|\Theta\|^2 + \lambda_\Phi \|\Phi\|^2 + \lambda_\Gamma \|\Gamma\|^2, \quad (5.4)$$

donde  $\gamma$ ,  $\beta$ ,  $\lambda_\Theta$ ,  $\lambda_\Phi$  y  $\lambda_\Gamma$  son todos parámetros positivos que permiten ajustar el peso relativo de cada término.



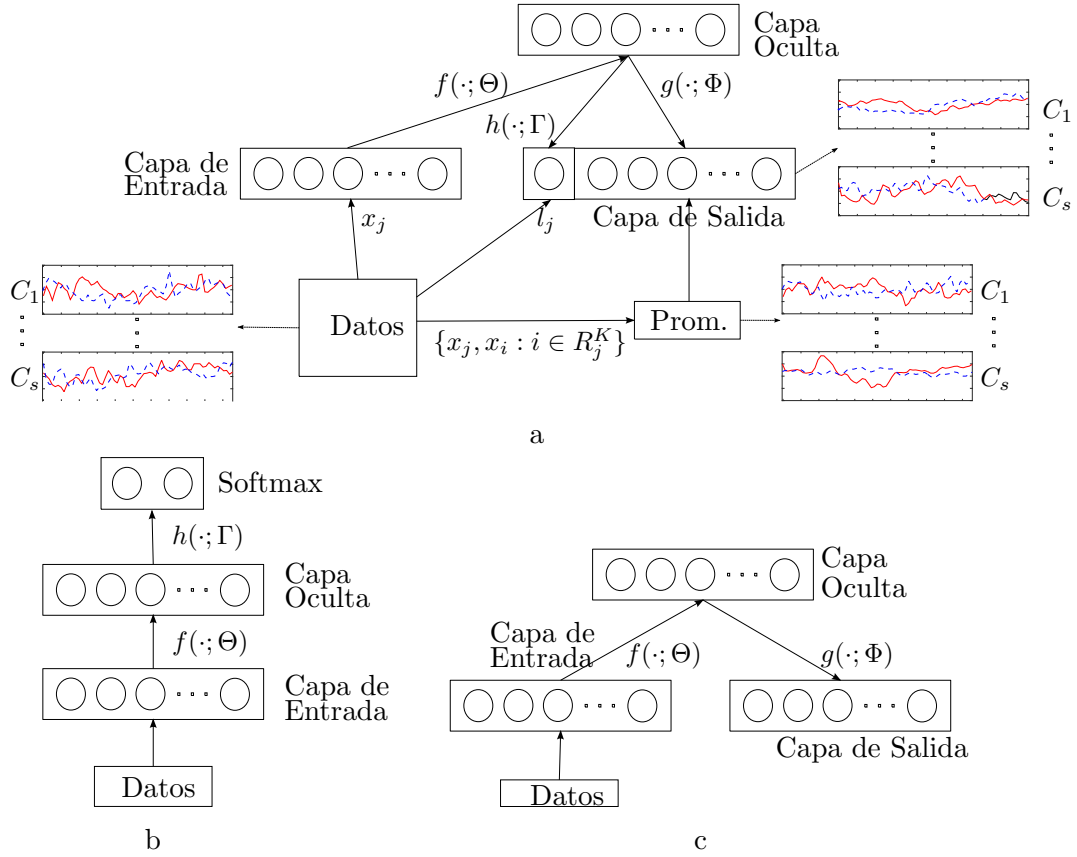


Figura 5.1: (a) AEP durante el entrenamiento. Las épocas objetivo están representadas con línea llena en rojo y las muestras no objetivo con línea azul punteada. (b) AEP durante la operación usado como extractor de características para clasificación. (c) AEP durante la operación usado para estimación.

En la Figura 5.1a se presenta la estructura básica de un AEP. Las entradas  $x_i$  se toman directamente de los patrones en la base de datos y se transforman a través de  $f(\cdot; \Theta)$  para obtener la representación oculta. Dicha representación se alimenta luego a la capa de decodificación, la cual aplica una transformación  $g(\cdot; \Phi)$  con el objetivo de reconstruir una versión limpia de la entrada presentada  $x_i$ . La salida deseada se estima a través de CA. Por otro lado, la representación oculta también se utiliza para alimentar la capa softmax  $h(\cdot; \Gamma)$ , que intenta predecir la etiqueta correspondiente.

## 5.6. Entrenamiento del AEP

El proceso de entrenamiento del AEP es muy similar al típico que se utiliza para el entrenamiento de AE y se presenta en detalle en el Algoritmo 1. Los valores

de los parámetros deben inicializarse al azar, pueden usarse distintas distribuciones estadísticas para obtenerlos. El cálculo del paso de optimización dependerá del método de optimización elegido, no todos los algoritmos de optimización devuelven necesariamente la misma solución en cada iteración, especialmente para problemas con múltiples mínimos locales. Para definir el criterio de parada se suele fijar como condición que el costo de validación no aumente durante épocas sucesivas del entrenamiento.

---

**Algorithm 1** Entrenamiento del AEP.
 

---

Inicializar  $\Theta_0, \Phi_0, \Gamma_0$ .

$i \leftarrow 0$

**while** el criterio de parada no sea satisfecho **do**

Tomar  $N$  muestras del conjunto de entrenamiento y construir el mini-batch  $X^i$ .

Tomar las correspondientes  $N$  etiquetas  $y^i$ .

$a^i \leftarrow f(X^i; \Theta_i)$ . ▷ obtener las activaciones ocultas

$o^i \leftarrow g(a^i; \Phi_i)$ . ▷ obtener las salidas

$s^i \leftarrow h(o^i; \Gamma_i)$ . ▷ obtener las salidas softmax

$J_i \leftarrow J(X_i, y; \Theta_i, \Phi_i, \Gamma_i)$ . ▷ obtener el costo

Calcular el gradiente del costo  $\nabla_{(\Theta_i, \Phi_i, \Gamma_i)} J_i$  utilizando retropropagación.

Calcular la dirección y magnitud del paso de optimización  $\Delta\Theta_i, \Delta\Phi_i$  y  $\Delta\Gamma_i$  en base a  $J$  y  $\nabla_{(\Theta, \Phi, \Gamma)} J$ .

$\Theta_{i+1} \leftarrow \Theta_i + \Delta\Theta_i$ . ▷ actualizamos  $\Theta$ .

$\Phi_{i+1} \leftarrow \Phi_i + \Delta\Phi_i$ . ▷ actualizamos  $\Phi$ .

$\Gamma_{i+1} \leftarrow \Gamma_i + \Delta\Gamma_i$ . ▷ actualizamos  $\Gamma$ .

Verificar si se cumple el criterio de parada.

$i \leftarrow i + 1$ .

---

Hay que tener en cuenta que, dependiendo del número de unidades ocultas y del tamaño de los patrones de entrada, el número de parámetros en la red podría llegar a ser bastante alto, de hecho, hay en principio  $m(M + 2n + 1) + n$  parámetros libres en el modelo. Estos parámetros son las componentes de las matrices y vectores de codificación, decodificación y softmax, a saber  $W_\Theta, b_\Theta, W_\Phi, b_\Phi$  y  $\Gamma$ .

En ciertos casos, particularmente cuando la relación entre el número de patrones de entrenamiento y la cantidad de parámetros se vuelve desfavorable, toda suposición razonable que conduzca a una reducción en el número de parámetros libres es altamente deseable. Reducir el número de parámetros contribuye a prevenir el sobre-ajuste y además disminuye el costo computacional del proceso de entrenamiento. Una manera de reducir el número de parámetros libres es

“atando” los pesos de las matrices de codificación y decodificación de modo que  $W_{\Theta} = W_{\Phi}^T$ . Esta suposición es razonable porque, puesto que tanto los patrones originales como sus promedios tienen formas de onda similares y pertenecen al mismo espacio ( $\mathbb{R}^n$ ), la acción del decodificador podría ser pensada como la inversa de la acción del codificador. Cuando se aplica la restricción de atar los pesos del AEP se modifica el Algoritmo 1 ya que primero, no es necesario actualizar  $\Phi$  ya que está asociada al valor de  $\Theta$  y segundo, al calcular el gradiente se debe tomar el promedio entre  $\nabla_{(\Theta_i)} J_i$  y  $\nabla_{(\Phi_i)} J_i$ .

## 5.7. Configuración de la red y ajuste de hiperparámetros

Se ha mostrado que el rendimiento de algunas arquitecturas de ANN puede variar significativamente con la elección de los hiperparámetros [73]. Como la mayoría de las ANN, el AEP tiene muchos hiperparámetros, por lo que encontrar sus valores óptimos puede constituir una tarea compleja. Aunque algunos autores abordan este problema dando recomendaciones prácticas y directrices de diseño, cada conjunto de datos y arquitectura tiene diferentes hiperparámetros óptimos y son necesarios procedimientos apropiados para su estimación [5, 37].

Para ajustar los parámetros del AEP proponemos un enfoque ad hoc de cinco etapas para el establecimiento de todos los hiperparámetros. A continuación describiremos brevemente las etapas propuestas. En primer lugar se realizan las elecciones de diseño de la arquitectura de la ANN, es decir, elegimos las funciones de activación y decidimos si los pesos del codificador y el decodificador están o no atados ( $W_{\Theta} = W_{\Phi}^T$ ). En segundo lugar, se utiliza un análisis objetivo, apoyado por experimentos preliminares, para determinar la cantidad  $K$  de épocas a promediar para calcular las señales de entrenamiento de salida. En tercer lugar, se proponen los pesos que ajustan la importancia relativa de los términos de discriminación y rareza de la función de coste (ecuación 5.4), es decir  $\gamma$  y  $\beta$ . En cuarto lugar se elige mediante el procedimiento de curva L el valor de los hiperparámetros de regularización de los términos de penalización en la función de coste, es decir,  $\lambda_{\Theta}$ ,  $\lambda_{\Gamma}$  y  $\lambda_{\Phi}$ , [24]. Por último, se realiza una búsqueda de grilla para encontrar los valores óptimos para los restantes hiperparámetros discretos, es decir, el número de unidades ocultas  $m$ , el tamaño de mini-batch  $N$  y el número máximo de iteraciones para cada mini-batch  $L$ . A continuación describiremos algunas consi-

deraciones de diseño necesarias y detalles necesarios para la implementación de cada una estas etapas.

Se han propuesto muchas funciones de activación para las unidades de las ANN prealimentadas [5]. Una de las funciones más conocidas y ampliamente utilizadas es la sigmoidea, que es una función suave con derivada acotada en  $\mathbb{R}$ . Esta función de activación proporciona estabilidad numérica para el proceso de entrenamiento ya que su derivada está siempre entre cero y uno. Para ANN con muchas capas, este tipo de función de activación puede producir gradientes desvanecientes, y por lo tanto, se usan generalmente otras alternativas tales como las ReLU. Como se menciona en la Sección 5.6, atar los pesos del codificador y el decodificador reduce significativamente los grados de libertad del AEP, y puede acelerar el proceso de entrenamiento, a expensas de reducir la capacidad de representación del modelo. La decisión de que estructura utilizar dependerá de la relación entre la cantidad de muestras disponibles para el entrenamiento del sistema, que tan ruidosas sean dichas muestras, la cantidad de parámetros a ajustar y la complejidad de la función a aproximar. Una medida útil para saber si es necesario reducir la cantidad de parámetros es analizando la capacidad de generalización de la red. Si se evidencia sobreajuste en general es necesario reducir la cantidad de parámetros libres.

Los hiperparámetros  $\gamma$  y  $\beta$  determinan la importancia relativa de los tres primeros términos del segundo miembro de la ecuación 5.4, sin embargo hay que tener en cuenta que no necesariamente los factores que estos multiplican (i.e.  $J_D(f(X; \Theta), y; \Gamma)$  y  $P_S(f(X; \Theta); \rho)$ ) comparten el mismo orden de magnitud, ni entre si, ni con el término de reconstrucción (i.e.  $J_R(f(X; \Theta); \Phi)$ ). De hecho, si bien estos elementos están normalizados respecto a la cantidad de unidades ( $m$ ) y al número de muestras de cada *mini-batch* ( $N$ ), dependiendo de la estructura de los datos a modelar pueden tomar distintos valores. Por esta razón es importante evaluar en una primera instancia cada término por separado con los datos de entrenamiento para verificar sus respectivos ordenes de magnitud, esto nos permite luego incluir estos factores en los respectivos hiperparámetros  $\gamma$  y  $\beta$ . Una vez que el orden de magnitud de los tres primeros términos es ajustado, resta tomar la decisión de diseño de definir que tanto debe pesar cada término en la función de costo. Para esto se debe incluir en los hiperparámetros  $\gamma$  y  $\beta$  el factor correspondiente.

El número  $K$  de épocas promediadas para obtener la salida del AEP puede

tener un gran efecto en las representaciones aprendidas. En primera instancia podríamos tender a fijar  $K$  a un valor grande para conseguir una salida tan limpia como sea posible. Sin embargo, debido a que se agregan épocas distintas a la que alimentamos a la entrada del AEP, a medida que  $K$  aumenta la similitud entre la entrada y la salida promediada disminuye. Por lo tanto hay una relación de compromiso entre obtener salidas más limpias y conservar una relación entre la entrada y la salida.

Para ajustar los parámetros de regularización de la función de costo (i.e.  $\Theta$ ,  $\Phi$  y  $\Gamma$ ). Se implementó un proceso de tres pasos, mediante el método de la curva-L. En el paso 1 se inicializa  $\Theta$  al azar como  $\Theta_0$ , y de acuerdo con la teoría de la curva L [24], se fija  $\lambda_\Gamma$  a un valor  $\lambda_\Gamma^0$ , tomado aproximadamente igual al punto correspondiente a la máxima curvatura de la curva  $\|\Gamma\|^2$  como una función de  $J_D(f(X; \Theta_0), y; \Gamma)$  parametrizada por  $\lambda_\Gamma$ . Luego se calcula:

$$\Gamma_0 \doteq \underset{\Gamma \in \mathbb{R}^{2 \times m}}{\operatorname{argmin}} (\gamma J_D(f(X; \Theta_0), y; \Gamma) + \lambda_\Gamma^0 \|\Gamma\|^2). \quad (5.5)$$

En el paso 2 se utiliza un procedimiento similar para aproximar el  $\lambda_\Phi$  óptimo, tomándolo como  $\lambda_\Phi^0$ , que es el valor asociado al punto de máxima curvatura de la curva  $J_R(f(X; \Theta_0); \Phi) - \|\Phi\|^2$  parametrizada por  $\lambda_\Phi$ . Luego calculamos:

$$\Phi_0 \doteq \underset{\Phi \in \mathbb{R}^{(n+1) \times m}}{\operatorname{argmin}} (J_R(f(X; \Theta_0); \Phi) + \lambda_\Phi^0 \|\Phi\|^2). \quad (5.6)$$

En el paso 3 el valor óptimo de  $\lambda_\Theta$  (i.e.  $\lambda_\Theta^0$ ) se estima por el método de la curva-L como el valor correspondiente al punto de máxima curvatura en la curva de  $\|\Theta\|^2$  en función de  $J_R(f(X; \Theta); \Phi_0) + \gamma J_D(f(X; \Theta), y; \Gamma_0) + \beta P_S(f(X; \Theta); \rho)$ , donde  $\Gamma_0$  y  $\Phi_0$  son las soluciones de los problemas presentados en las Ecuaciones 5.5 y 5.6, respectivamente.

Es importante notar que para cada uno de los puntos de cada una de las curvas es necesario ajustar un modelo. Cada uno de estos modelos tiene una estructura diferente ya que se fijan algunos de los pesos de la red original, ajustando solo parte de la red. En el paso 1 se fijan los pesos del codificador a  $\Theta_0$  y se ajusta únicamente los valores de  $\Gamma$ . En el paso 2, también con los parámetros de  $\Theta$  fijos se ajusta el valor de los pesos del decodificador  $\Phi$  para cada uno de los puntos de la curva. En el paso 3, con  $\Gamma$  y  $\Phi$  fijos se ajusta el valor de los parámetros  $\Theta$  de la función de codificación. Durante estos entrenamientos parciales de la red

realizados para obtener los distintos puntos de las curvas-L, en lugar de separar el conjunto de entrenamiento en *mini-batches* se utilizaron todos los datos de entrenamiento para obtener resultados mas estables y comparables entre los distintos puntos de la curva. Por otro lado, ninguno de los hiperparámetros  $N$ ,  $l$  y  $m$  fue establecido en las etapas anteriores. Aunque los valores de  $N$  y  $l$  no se usan en los procesos de entrenamiento modificados utilizados para obtener los puntos de las curvas-L, es inevitable establecer la cantidad de unidades ocultas  $m$ . Para analizar el efecto de el valor de  $m$  sobre el resultado obtenido tras el ajuste de los parámetros de regularización, hemos realizado el proceso de ajuste de  $\lambda_{\Theta}$ ,  $\lambda_{\Phi}$  y  $\lambda_{\Gamma}$  para varios valores de  $m$  (algunos más grandes, algunos iguales y algunos más pequeños que el tamaño de los patrones de entrada) y encontramos que los valores óptimos de los hiperparámetros de penalización son independientes de  $m$ .

La última etapa consiste en una búsqueda de grilla para establecer los tres hiperparámetros restantes (es decir,  $m$ ,  $N$  y  $l$ ). Para evaluar el desempeño de cada AEP entrenado con una distinta configuración de estos hiperparámetros, es necesario utilizar alguna medida sencilla de la bondad de cada AEP. Para esto se selecciona como óptima la configuración de la red que provea el mayor rendimiento en términos de la capacidad de clasificación sobre los datos de validación. Para reducir la incertidumbre es conveniente en estos casos utilizar la misma configuración y semillas para el generador de números aleatorios de cada red a ser evaluada.

---

# Capítulo 6

## Experimentos

En este capítulo se describen los experimentos realizados para analizar las capacidades de los algoritmos propuestos y se presentan y discuten los resultados de dichos experimentos. Para ello previamente se definen las medidas de rendimiento utilizadas y se introducen las bases de datos utilizadas.

### 6.1. Evaluación de rendimiento

Con el objetivo de medir la bondad de las características de las representaciones obtenidas mediante el AEP, se evaluó el desempeño de clasificación utilizando tres clasificadores diferentes: SNN, SVM y KNN.

La elección de la SNN es natural dado que el AEP incluye salidas softmax. Estas salidas direccionan el entrenamiento del modelo, llevándolo hacia la búsqueda de una representación oculta que favorezca la tasa de clasificación obtenida con la SNN. La estructura pre-entrenada de las unidades de clasificación del AEP pueden incluso utilizarse como punto de partida para entrenar la SNN que se utilizará como clasificador.

Además de la SNN, para analizar si existen diferencias en el ajuste de la representación a dicho clasificador en comparación con otros métodos, se utilizó otro clasificador lineal. Para esto elegimos una SVM lineal, principalmente en base a su popularidad y robustez.

La selección del KNN permite observar si existen diferencias entre el uso de clasificadores lineales y no lineales, tanto antes como después de la extracción de características con el AEP.

Las SNN fueron entrenadas utilizando la implementación del algoritmo Broyden-

Fletcher-Goldfarb-Shanno de memoria limitada (LBFGS, del inglés Limited Memory Broyden-Fletcher-Goldfarb-Shanno) incluido en el paquete de optimización minFunc [86]. Todos los parámetros del optimizador se dejaron en sus valores por defecto, y el número máximo de iteraciones se estableció en 400.

Se utilizó la implementación de SVM provista por el paquete LIBLINEAR [26]. Se utilizó como ya mencionamos un núcleo lineal. Todos los parámetros se dejaron por defecto y se añadió un término de bias.

La implementación de KNN utilizada es la que se proporciona con PRTools [23]. Para definir las vecindades se utilizó la distancia euclídea. Luego de evaluar algunos resultados preliminares obtenidos estableciendo el número de vecinos en tres, cinco, siete y nueve, se terminó fijando este parámetro en nueve vecinos ya que con este valor se obtuvieron los mejores resultados en forma consistente.

### 6.1.1. Medidas de rendimiento

Se midió el desempeño de los clasificadores binarios propuestos utilizando la Especificidad (Spe), Sensibilidad (Sen), Precisión (Prec) y Exactitud (Acc), definidas como:

$$\text{Spe} = \frac{TN}{FP + TN}, \quad \text{Sen} = \frac{TP}{FN + TP}, \quad \text{Prec} = \frac{TP}{FP + TP}, \quad \text{Acc} = \frac{TP + TN}{NT}. \quad (6.1)$$

donde  $TP$ ,  $TN$ ,  $FP$  y  $FN$  son el número de verdaderos positivos, negativos verdaderos, falsos positivos y falsos negativos respectivamente y  $NT = FP + TP + FN + TN$  es el número total de muestras. Dado que en el caso del deletreador de Donchin las clases están desbalanceadas, se utilizó también la exactitud balanceada (BalAcc). Esta medida está diseñada para ser insensible a los datos desbalanceados y puede facilitar el análisis de los resultados. Se define como:

$$\text{BalAcc} = \frac{\text{Spe} + \text{Sen}}{2}. \quad (6.2)$$

## 6.2. Bases de datos

Para ajustar los parámetros y evaluar el rendimiento de el AEP utilizamos los dos conjuntos de datos que se describen debajo.



### 6.2.1. Bases de datos de competencias

Con el objetivo de popularizar las BCI y contribuir al desarrollo de las técnicas de procesamiento y clasificación de señales, entre los años 2001 y 2008 el equipo del *Berlin Brain-Computer Interface project* organizó cuatro concursos centrados en la clasificación de señales de BCI basadas en EEG. Para la tercera edición, se publicó un conjunto de datos con señales de EEG registradas utilizando el deletreador basado en P300 incluido en el sistema BCI2000 [10].

Este conjunto de datos consiste en registros de EEG muestreados a 240 Hz, tomados de 64 canales. La base de datos cuenta con señales correspondientes a dos individuos (A y B). Cada individuo deletreó un total de 85 caracteres de entrenamiento y 100 caracteres de prueba, con bloques de 15 intensificaciones por carácter. Este procedimiento produce un total de 15300 señales pos-estímulo de entrenamiento y 18000 señales de pos-estímulo de prueba por individuo, con una relación objetivo/no-objetivo de 1:5. Más detalles sobre la configuración de adquisición de datos, así como el conjunto de datos en sí mismo, están disponibles en la página web de las competencias <sup>1</sup>.

### 6.2.2. Base de datos sintética

Para evaluar el comportamiento del AEP y ajustar sus hiperparámetros se utilizó, además de la base de datos real descrita en la Sección 6.2.1 una base de datos sintética. Esto se decidió para evitar en las primeras instancias complejidades tales como conjuntos de datos con clases desequilibradas, artefactos de registro, limitaciones en el número de épocas disponibles para el entrenamiento y la presencia de épocas extremadamente ruidosas y/o mal etiquetadas. Utilizando datos sintéticos pudimos reducir el espacio de búsqueda para la configuración de hiperparámetros de la red y ganar conocimiento respecto al comportamiento del sistema propuesto.

Para generar los datos sintéticos se alimentó el algoritmo de codificación predictiva lineal (LPC, del inglés Linear Predictive Coding) con los datos de las competencias (ver Sección 6.2.1) para estimar los parámetros de varios sistemas lineales autorregresivos (AR) que modelen las señales de EEG de fondo para cada canal [57]. Los órdenes óptimos de los sistemas se fijaron individualmente para cada modelo utilizando el criterio de información de Akaike. Los modelos AR se

<sup>1</sup>[www.bbc.de/competition](http://www.bbc.de/competition) accedido el 1 de junio de 2016

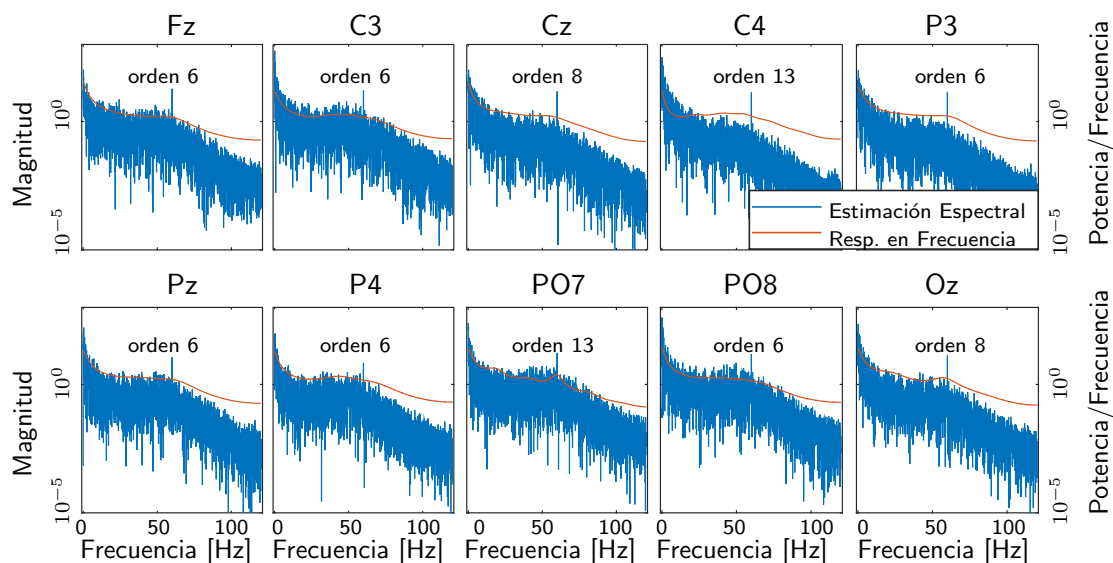


Figura 6.1: Estimación de densidad espectral de frecuencia para las señales de entrenamiento originales y respuesta en frecuencia de los filtros lineales obtenidos por medio de LPC.

alimentaron con ruido blanco y las salidas se utilizaron como señales de EEG de fondo simulado. Se entrenó un modelo AR para cada uno de los diez canales seleccionados (Fz, C3, Cz, C4, P3, Pz, P4, PO7, PO8 y Oz), y sólo se realizó el proceso para los registros del individuo A. Siguiendo la separación original del modelo en conjuntos de entrenamiento y prueba también se construyeron modelos distintos para simular las señales de entrenamiento y prueba de EEG de fondo.

Las señales de ruido blanco alimentadas al modelo AR correspondiente a un determinado canal son independientes del ruido asociado al resto de los canales. Debido a esto las señales correspondientes a cada canal no están espacialmente correlacionadas, en oposición a las señales de EEG en los registros reales donde dicha correlación está presente. En la Figura [6.1](#) se pueden ver las respuestas en frecuencia de los filtros obtenidos junto con las densidades espectrales de potencia estimadas a partir de las señales de entrenamiento originales (el orden mostrado corresponde al orden de cada filtro y el eje ordenado tiene escala logarítmica).

Además de las señales de EEG de fondo, también fue necesario construir ERP sintéticos. Para esto se realizó la promediación de distintos subconjuntos de 500 ERP objetivo elegidas al azar. Dichos ERP son seccionados de los registros originales, tomando la respuesta desde los 100 ms previos a cada estímulo hasta los 850 ms posteriores al estímulo. Con el fin de evitar la introducción de artefactos de borde cada muestra se multiplicó con una ventana gaussiana elegida apropiadamente. En cada segundo, se simuló un estímulo con una probabilidad de 50/50

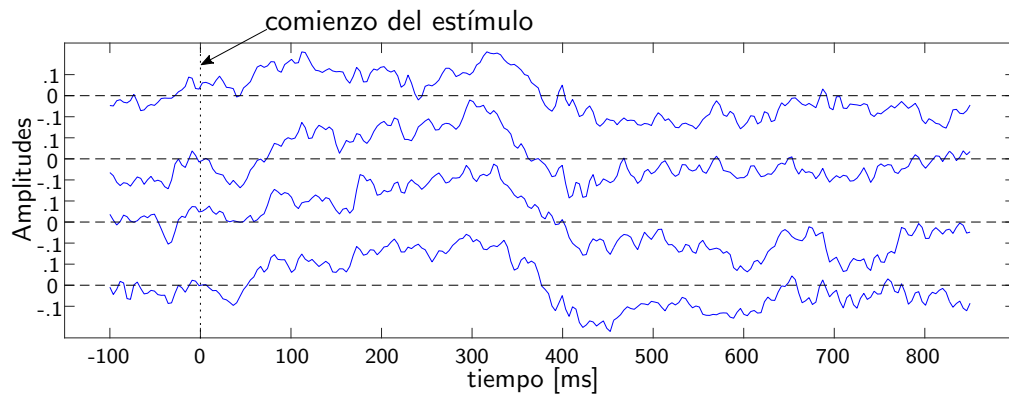


Figura 6.2: Muestras de ERP sintéticos utilizados para crear el conjunto de entrenamiento de la base de datos sintética (canal Pz).

de ser objetivo o no objetivo. Por último para cada una de las muestras donde el estímulo simulado fuese objetivo se añadió una muestra de ERP sintético. Se tomó el cuidado de mantener la SNR en la base de datos sintética aproximadamente igual a la de los datos reales. La SNR se estimó independientemente para cada canal, obteniéndose como SNR promedio para los diez canales  $-19$  dB. Los ERP sintéticos de entrenamiento se generaron promediando épocas del conjunto de datos reales de entrenamiento, mientras que los de prueba se generaron promediando épocas del conjunto de prueba. La Figura 6.2 muestra cuatro ejemplos de ERP sintéticos obtenidas del conjunto de entrenamiento para el canal Pz.

Cabe destacar que para los registros sintéticos generados las clases están balanceadas, en contraposición a los registros reales donde la relación de objetivo a no-objetivo es  $1/5$ . Dado que cada ERP sintético fue generado tomando al azar conjuntos de épocas reales diferentes, esta base de datos artificiales es más realista y difícil de clasificar que otras bases de datos con una SNR similar, pero obtenidas utilizando siempre la misma plantilla de ERP. Debido a que tanto los ERP como la señal de EEG de fondo fueron generados a partir de los correspondientes datos reales de entrenamiento y prueba, los datos simulados también están separados en estas categorías y no hay información cruzada entre los conjuntos.

### 6.3. Preprocesamiento

Como vimos en la Sección 2.1.1, el preprocesamiento es una etapa fundamental para acondicionar las señales crudas a ser usadas por la BCI. Para este trabajo se buscó realizar una mínima cantidad de preprocesamiento para acondicionar las

señales. Tanto los registros reales como los sintéticos fueron preprocesados de la misma manera.

La etapa de preprocesamiento se inició normalizando las señales llevando los registros de entrenamiento a media cero y desvío estándar unitario. Utilizando las medias y desvíos estándar calculados para los datos de entrenamiento se normalizaron también los datos de prueba. Dicho proceso de normalización se aplicó en forma independiente para cada electrodo.

Posteriormente se evaluó las frecuencias de corte óptimas de los filtros pasa-bajos. Para esto se realizaron pruebas de rendimiento en tareas de clasificación tomando diferentes frecuencias de corte. Sobre la base de datos reales se generaron 14 conjuntos de patrones distintos, cada uno con distintas frecuencias de corte. Se evaluaron las frecuencias de corte entre 4 y 30 Hz inclusive, saltando de a 2 Hz (4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28 y 30). Los patrones registrados fueron clasificados utilizando los tres clasificadores KNN, SVM y SNN. Los conjuntos de entrenamiento se balancearon quitando patrones sin respuesta P300 para obtener la misma cantidad de patrones objetivo y no-objetivo. La selección de patrones no-objetivo a ser removidos para construir los conjuntos de entrenamiento se realizó al azar. Tanto para el sujeto A como para el sujeto B utilizando la SVM y la SNN los mejores resultados se obtienen con frecuencias de corte de alrededor de los 10 Hz. Con los KNN no se ve una tendencia tan marcada, únicamente se puede ver diferencias para el sujeto B, siendo mejores los resultados utilizando dicho clasificador en las frecuencias bajas. En lugar de definir la frecuencia de corte en 10 Hz se fijó en 12 Hz por ser un divisor entero de la frecuencia de muestreo de las señales.

En base a dichos resultados se filtró los registros con un filtro respuesta finita al impulso (FIR, del inglés Finite Impulse Response) pasa-bandas de orden 20, con frecuencias de corte ubicadas en 0,5 y 12 Hz. Después de aplicar el filtrado temporal se segmentaron las épocas individuales (objetivo y no objetivo) de los registros. El segmento correspondiente a cada época comienza en el instante del estímulo y termina 750 ms después.

Dado que los datos fueron registrados con una frecuencia de muestreo de 240 Hz y las frecuencias más altas presentes en los ERP de interés son significativamente más bajas, las señales pueden ser submuestreadas. Para las búsquedas de hiperparámetros óptimos y para los experimentos de clasificación se redujo la frecuencia de muestreo a 24 Hz, conservando una muestra de cada diez y des-

cartando las nueve restantes. Para los experimentos donde se busca analizar la capacidad de limpieza de ruido del AEP no se reduce la frecuencia de muestreo. Cada época se tiene por lo tanto 180 o 1800 muestras, dependiendo de si fue o no sometida al sub-muestreo.

Los procesos descritos se realizaron para cada uno de los canales por separado. Por último, los segmentos de cada canal se concatenaron en un único vector de características para cada época.

## 6.4. Ajuste de hiperparámetros

Siguiendo la estrategia propuesta en la Sección 5.7 describimos a continuación las decisiones de diseño y los resultados obtenidos para cada etapa.

Como mencionamos previamente, las ANN con muchas capas pueden verse perjudicadas por los gradientes desvanecientes. Exploramos dicha posibilidad analizando los valores de las señales de error retropropagadas, concluyendo que el AEP no sufre particularmente del problema (al menos para las bases de datos utilizadas). Por otro lado, en experimentos preliminares con distintos tipos de unidades se obtuvieron resultados peores o iguales que los obtenidos con unidades sigmoides. En base a esto se decidió utilizar dicho tipo de unidades.

El valor del objetivo de rareza  $\rho$  se estableció en 0,1, siguiendo las recomendaciones proporcionadas en 37. Como se menciona en la Sección 5.6, atar los pesos del codificador y el decodificador reduce significativamente los grados de libertad del AEP a expensas de reducir la capacidad del modelo. Tanto como para el proceso de ajuste de hiperparámetros, como para los experimentos descritos en este capítulo se dejó libres los parámetros del codificador y decodificador (i.e.  $W_{\Theta} \neq W_{\Phi}^T$ ).

El hiperparámetro  $\gamma$ , asociado con el término de discriminación de la función de costo, se ajustó de manera que el primer y segundo términos en el miembro derecho de la ec. 5.4) tengan magnitudes similares. Para esto fue necesario fijar el valor de  $\gamma = 20$ . El valor del hiperparámetro  $\beta$  se estableció de modo que el término de rareza (tercer término en el miembro derecho de la ec. 5.4) sea aproximadamente un orden de magnitud menor que los términos de reconstrucción y de discriminación. Para esto se fijó  $\beta = 100$ . Esta elección se justifica por el hecho de que preferimos priorizar las capacidades de reconstrucción y la discriminación por sobre la rareza de la representación oculta. Para estimar la magnitud de cada

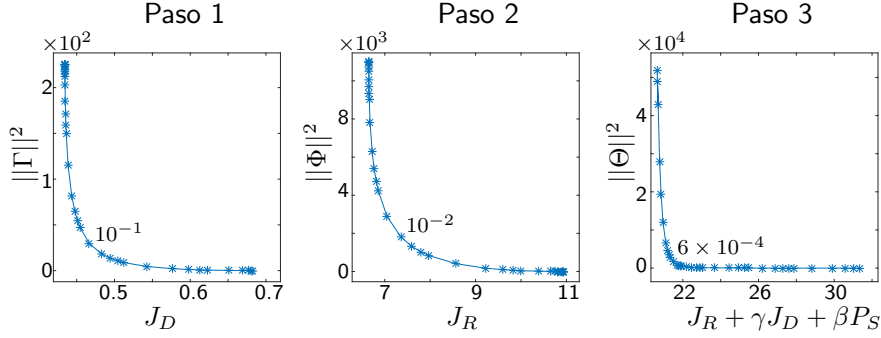


Figura 6.3: Curvas-L utilizadas para obtener los hiperparámetros de regularización para el AEP entrenado con la base de datos sintéticos.

término se evaluó la función de costo utilizando los datos de entrenamiento.

Como se mencionó en la Sección 5.7 existe una relación de compromiso entre reducir el nivel de ruido y mantener cierta similitud entre la entrada y la salida deseada. Los resultados experimentales obtenidos con las bases de datos utilizadas, han mostrado una notable disminución en el rendimiento para valores de  $K$  mayores que 3, y resultados ligeramente mejores tomando  $K = 2$ . Por estas razones ese fue el valor de  $K$  utilizado a lo largo de este trabajo.

Siguiendo el proceso propuesto en la Sección 5.7 como la cuarta etapa de ajuste, para determinar el óptimo de los hiperparámetros de optimización  $\lambda_\Theta$ ,  $\lambda_\Phi$  y  $\lambda_\Gamma$  se utilizó el criterio de la curva-L. En la Tabla 6.1 se presentan los resultados obtenidos para cada uno de los conjuntos de datos. En la Figura 6.3 se ejemplifica el tipo de curvas que se obtiene utilizando esta metodología.

Por último, se realizó la búsqueda de grilla para establecer  $m$ ,  $N$  y  $l$ . La búsqueda del valor óptimo de  $m$  se realizó entre 18 y 360 unidades ocultas (correspondientes a una décima y el doble de las características de entrada, respectivamente). La búsqueda de  $N$  se realizó para *mini-batches* entre 10 y 5000 muestras y también utilizando todos los datos de entrenamiento (i.e. sin utilizar *mini-batches*). La búsqueda de  $l$  se realizó entre 2 y 1000. En estos experimentos los AEP fueron entrenados utilizando el método del CG implementado en minFunc [86]. Los conjunto de datos de entrenamiento se dividió en subconjuntos de entrenamiento y validación, conteniendo el 90% y 10% de los datos, respectivamente. Estos subconjuntos se utilizaron posteriormente para todos los modelos. Con el fin de reducir la incertidumbre, todos los modelos con el mismo valor de  $m$  se entrenaron con los mismos pesos iniciales. Para evaluar el desempeño de los diferentes AEP, utilizamos las representaciones de características ocultas que cada una produjo para entrenar y evaluar una SNN. La bondad de cada AEP se midió

Tabla 6.1: Definiciones de la estructura y valores de los hiperparámetros obtenidos.

	Datos Sintéticos	Datos Reales	
		Suj. A	Suj. B
función de activación de las unidades ocultas	Sigmoidea	Sigmoidea	Sigmoidea
pesos atados?	Falso	Falso	Falso
$\rho$	0.1	0.1	0.1
$\gamma$	20	20	20
$\beta$	100	100	100
$K$ (# de promedios en la salida)	2	2	2
$\lambda_{\Theta}$	$6 \times 10^{-4}$	$10^{-2}$	$4 \times 10^{-5}$
$\lambda_{\Phi}$	$10^{-2}$	$2 \times 10^{-2}$	$10^{-1}$
$\lambda_{\Gamma}$	$10^{-1}$	$4 \times 10^{-2}$	$2 \times 10^{-1}$
$m$ (# unidades ocultas)	90	45	90
$N$ (Tamaño del mini-batch)	todos los datos	800	todos los datos
$l$ (límite de iteraciones del optimizador)	2	250	50

entonces en términos del rendimiento de clasificación de su SNN correspondiente.

Los parámetros y configuraciones obtenidos o definidos por los procesos descritos en esta sección se presentan en la Tabla [6.1](#).

Cabe señalar que las diferencias entre los parámetros obtenidos utilizando datos reales y simulados para todos los experimentos fueron pequeños. Esto sugiere que los hiperparámetros de AEP son robustos y estables para diferentes conjuntos de datos. También se puede observar que la utilización de datos simulados para realizar la búsqueda devuelve hiperparámetros similares a los obtenidos utilizando datos reales.

## 6.5. Clasificación de potenciales relacionados con eventos

El AEP está diseñado principalmente para ser utilizado como un bloque de extracción de características para clasificación, en la presente sección se analiza su rendimiento en este sentido. Para esto utilizamos la configuración dada en la segunda columna de la Tabla [6.1](#) en todos los experimentos desarrollados en esta

sección. Cabe mencionar que para la base de datos reales también se realizaron los experimentos usando los hiperparámetros óptimos ajustados específicamente (columnas 3 y 4 de la Tabla 6.1), sin embargo los resultados obtenidos fueron similares en ambos casos. Para los experimentos descritos en esta Sección, se utilizaron los datos con submuestreados, por lo que los patrones de entrada consisten de 180 características.

En primera instancia se entrenaron los AEP, y a continuación las representaciones ocultas generadas por estos fueron utilizadas para entrenar y probar clasificadores basados en SNN, SVM y KNN.

Para generar resultados a los fines comparativos se utilizaron dos configuraciones: una utilizando los datos crudos y otra utilizando los datos promediados dos veces de forma coherente (cada muestra se promedió con una muestra perteneciente a la misma clase). Estas señales también se utilizaron para entrenar y probar clasificadores basados en SNN, SVM y KNN. En términos de tiempo de registro y/o rendimiento de una BCI lo justo es comparar los resultados obtenidos usando los datos sin promediar con los obtenidos por el AEP. Los datos promediados, por otro lado, proporcionarían un “techo” para el desempeño ya que tiene una mejor SNR que las señales presentadas al AEP.

Ambas bases de datos están originalmente separadas en un conjunto de entrenamiento y uno de prueba. Todos los resultados presentados corresponden a los rendimientos sobre los conjuntos de prueba, que no fueron utilizados para ajustar los hiperparámetros ni para la selección de los modelos. Cada conjunto de datos de entrenamiento se dividió al azar en dos subconjuntos: uno (que consta de 90 % de los datos) que se utilizó para el entrenamiento propiamente dicho y el otro (que consiste en el 10 % restante de los datos de entrenamiento originales) y que fue utilizado para la validación. El proceso de dividir los datos en subconjuntos y el posterior proceso de entrenamiento se repitió 100 veces para cada configuración. A continuación, en base a los resultados obtenidos para los subconjuntos de validación, se seleccionaron los mejores modelos para cada configuración. Dichos modelos fueron posteriormente evaluados utilizando los conjuntos de prueba. Todas las muestras de entrenamiento y de prueba disponibles fueron utilizadas en los experimentos, por lo tanto el conjunto de datos sintéticos consistió en 10000 muestras de entrenamiento y 10000 muestras de prueba y el conjunto de datos real consistió en 15300 muestras de entrenamiento y 18000 muestras de prueba para cada sujeto (A y B). Es oportuno recordar aquí que mientras el conjunto de



Tabla 6.2: Resultados de clasificación de P300/no-P300 para el AEP utilizando los datos sintéticos. 1T, 2T indica época única y promedios de dos épocas respectivamente.

Método	Spe %	Sen %	Prec %	Acc %
SNN (2T)	72	97	77	84
AEP + SNN (1T)	<b>85</b>	81	<b>85</b>	<b>83</b>
SNN (1T)	65	<b>91</b>	72	78
SVM (2T)	91	89	91	90
AEP + SVM (1T)	<b>84</b>	<b>84</b>	<b>84</b>	<b>84</b>
SVM (1T)	82	81	82	82
KNN (2T)	79	77	78	78
AEP + KNN (1T)	<b>67</b>	<b>95</b>	<b>74</b>	<b>81</b>
KNN (1T)	<b>67</b>	65	66	66

datos simulado es equilibrado, el conjunto de datos reales no lo es.

Los resultados sobre el conjunto de pruebas para los mejores modelos obtenidos utilizando el conjunto de datos sintéticos se resumen en la Tabla [6.2](#). Como se puede ver, el rendimiento obtenido usando KNN es el peor para todas las configuraciones. Sin embargo, es importante señalar que cuando se utilizan las representaciones provistas por la capa oculta del AEP y el KNN para clasificar, el rendimiento mejora sustancialmente, incluso resultando mayor al de los KNN entrenados utilizando promedios de dos épocas. Por otra parte, las representaciones latentes obtenidas a través de los AEP nos permitió mejorar la exactitud, especificidad y precisión obtenidas utilizando épocas únicas con la SNN y con la SVM. Además, el AEP + SNN (usando épocas únicas) mejoró la especificidad y la precisión en comparación con los resultados obtenidos para la SNN alimentada con épocas promediadas. Estos resultados sugieren que el enfoque propuesto es capaz de extraer información relevante, obteniendo mejores resultados de clasificación.

La mayoría de los artículos que presentan métodos de clasificación de datos registrados utilizando el deletreador de Donchin informan la tasa de error en la clasificación de caracteres. Sin embargo creemos que medir directamente el desempeño del sistema de clasificación binario (objetivo/no objetivo) es más fácil de analizar en el contexto del modelo propuesto (dado que nuestro objetivo es procesar y clasificar los potenciales relacionados con eventos (ERP, del inglés Event Related Potentials)). En la Tabla [6.3](#) se resumen los resultados obtenidos

Tabla 6.3: Resultados para el problema de clasificación binario P300/no P300 utilizando registros reales.

Método	Spe %			Sen %			BalAcc %		
	A	B	avg.	A	B	avg.	A	B	avg.
LDB	65	65	65.0	63	75	69.0	64	70	67.0
CNN	69	77	73.0	61	64	62.5	65	70	67.5
AEP	68	73	70.5	62	70	66.0	65	71	68.0

con los mejores modelos entrenados para el conjunto de datos reales. En este caso se presentan los resultados obtenidos para los datos de prueba con cada sujeto junto con el resultado promedio. Se comparan las medidas de rendimiento de clasificación de las salidas softmax con dos métodos del estado del arte.

El primero de los métodos utilizados para la comparación se describe en detalle en [70]. En este trabajo los autores utilizan un método de extracción de características utilizando el algoritmo de bases discriminantes locales (LDB, del inglés Local Discriminant Bases) utilizando la descomposición paquete de ondas (WP, del inglés Wavelet Packet), para posteriormente clasificar las muestras usando un clasificador de análisis discriminante lineal. Esta técnica propuesta obtuvo los mejores resultados entre varios otros enfoques probados por los autores<sup>2</sup>. Aquí replicamos el método y lo probamos usando los mismos datos que usamos para el AEP.

La segunda estrategia utilizada para la comparación se presenta en [15]. En dicho artículo se propone el uso de una CNN para realizar la extracción y clasificación de características en un único sistema. Aunque los autores proponen varias redes diferentes, sólo una (llamada CNN-2a en el artículo original) hace uso de un número predefinido y limitado de los 64 sensores originales proporcionados en la base de datos. Puesto que este es el mismo enfoque que se tomó en esta tesis, elegimos precisamente esa red para la comparación. Los números de TP, TN, FP y FN se informan en [15] y usamos dichos resultados para calcular los otros estimadores de desempeño. Como se puede ver en la tabla 6.3, los tres métodos producen rendimientos similares en términos de exactitud balanceada, mientras que se pueden observar variaciones más altas en las especificidades y sensibilidades.

<sup>2</sup>comunicación privada de resultados aún no publicados.

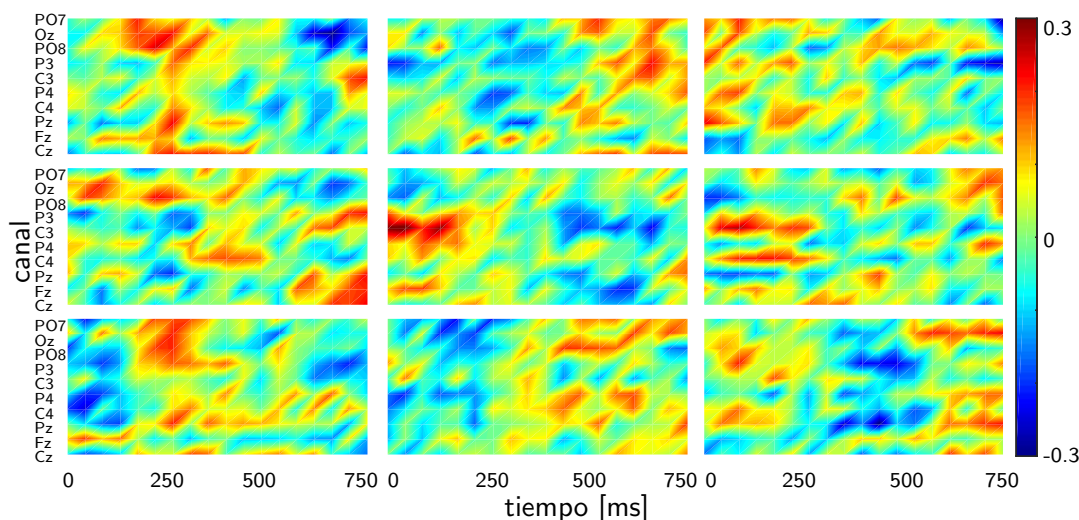


Figura 6.4: Representación espacio-temporal de los pesos de nueve unidades ocultas de un AEP entrenado.

La Figura 6.4 muestra los pesos  $W_{\Theta}$  de la capa de codificación de nueve neuronas de un AEP entrenada con los datos reales del sujeto A. Los pesos fueron reordenados de vectores a matrices para poder visualizar diferentes canales en diferentes filas y diferentes muestras temporales en diferentes columnas. Los valores de los pesos están dados por el mapa de colores, donde el verde representa cero y el azul y el rojo representan valores negativos y positivos extremos, respectivamente. Como era de esperar, las neuronas parecen detectar patrones característicos espacio-temporales relacionados con las diferentes clases. Por ejemplo, algunas neuronas enfatizan el área temporal cerca del P300, mientras que otras hacen lo contrario. Algunas franjas horizontales también indican la importancia relativa de la información proporcionada por los diferentes canales.

Para evaluar la robustez del AEP respecto a variaciones en el número de muestras de entrenamiento, se ajustó el modelo con subconjuntos reducidos de la base de datos simulada. El número total de muestras se varió entre 20000 (todas las muestras disponibles) y 2700 muestras. El desempeño se degradó en menos del 10 %, lo que sugiere que el modelo se desempeñaría relativamente bien en escenarios donde la cantidad de ejemplos de entrenamiento es relativamente pequeña. Esto probablemente se deba a las regularizaciones y restricciones aplicadas [105].

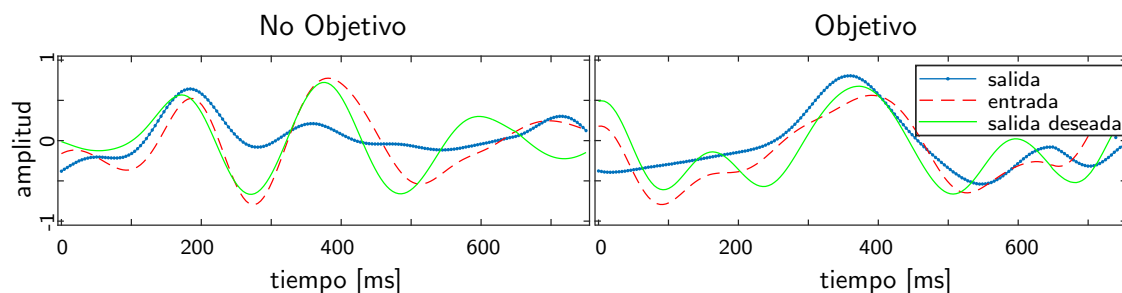


Figura 6.5: Salidas con sus correspondientes entradas y salidas deseadas para el AEP (solo canal Pz).

## 6.6. Estimación de potenciales relacionados con eventos

Para obtener una primera estimación de las capacidades de reconstrucción de la red, entrenamos y evaluamos los resultados del AEP, utilizando la base de datos sintética. Dado que las formas de onda son difíciles de evaluar visualmente si sólo se utilizan 18 muestras temporales por canal, realizamos el proceso utilizando datos no submuestreados con 180 muestras temporales por canal, obteniendo un total de 1800 características en las capas de entrada y salida. La AEP utilizada para obtener las salidas mostradas en la Figura 6.5 tenía los mismos hiperparámetros usados en la Sección 6.5 (ver Tabla 6.1), excepto por el número de unidades ocultas, que se ajustó a la mitad de la dimensión de los patrones de entrada.

La Figura 6.5 muestra las características de las salidas obtenidas con el AEP correspondientes al canal Pz. Se muestran también sus correspondientes entradas y salidas deseadas para una muestra objetivo y uno no objetivo. En el caso de este AEP la salida deseada se calculó promediando tres señales post-estímulo. Debido a la presencia de la onda P300, las muestras objetivo comúnmente tienen amplitudes mayores que sus contrapartes no objetivo. La Figura 6.5 muestra cómo esta diferencia, que podría ser útil para propósitos de clasificación, es resaltada por el AEP.

En la Figura 6.5 también podemos observar que la red ignora los cambios rápidos en las entradas, comportándose en este sentido como un filtro pasa-bajas. Para examinar más a fondo este fenómeno alimentamos con ruido blanco gaussiano, de una amplitud similar a la de las señales originales, a un AEP entrenado, y calculamos la densidad espectral de potencia (PSD, del inglés Power Spectral Density) de las salidas. Para cada canal, los promedios de la PSD de 500 sali-

das diferentes se representan en la Figura 6.6. Hay que tener en cuenta que las transformaciones producidas por el AEP no son lineales, por lo que estas estimaciones espectrales no representan necesariamente su respuesta de frecuencia. Sin embargo proporcionan evidencia de que la ANN tiene en cierta medida un comportamiento similar a de filtros pasa bajas. La energía del PSD de las salidas por debajo de 10 Hz es despreciable para todos los canales (las frecuencias de corte se ubican aproximadamente en los  $9 \pm 1$  Hz), lo cual es razonable ya que las salidas deseadas de la red eran promedios coherentes de señales filtradas con una frecuencia de corte de 12 Hz, y el proceso de promediado atenúa las frecuencias más altas.

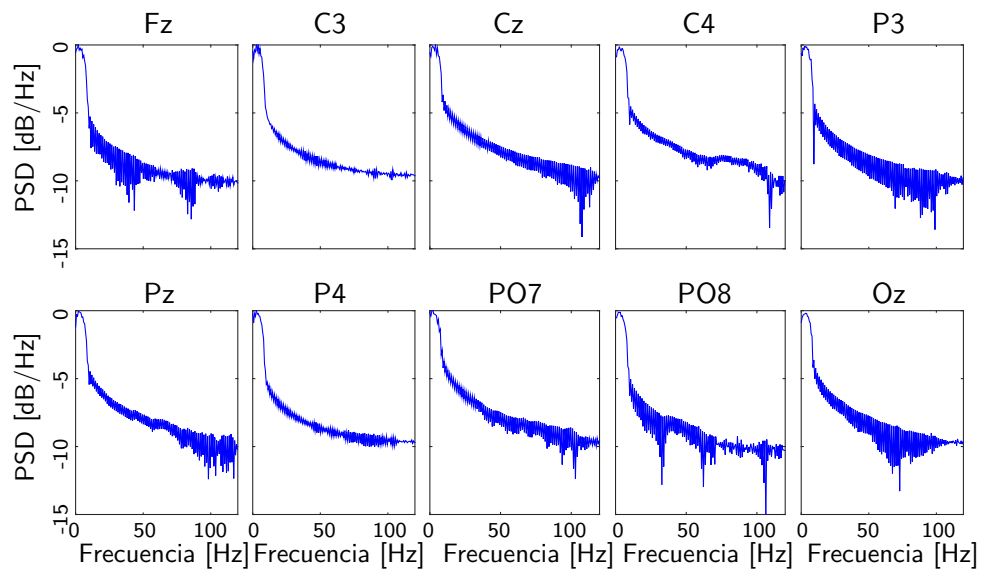


Figura 6.6: Estimación de la PSD de las salidas del AEP para cada canal cuando se lo alimenta con ruido blanco.



---

# Capítulo 7

## Conclusiones

En este capítulo se presentan la discusión y conclusiones junto con las propuestas de líneas de trabajo futuro.

### 7.1. Discusión

Se propuso en esta tesis el AEP, el cual puede considerarse una variante de AE, aplicable a señales donde patrones repetitivos pequeños están enmascarados por ruido aditivo.

La red fue evaluada utilizando tanto señales reales como señales generadas artificialmente. Los resultados obtenidos muestran que la red es eficaz para generar representaciones útiles, ya que permitió aumentar el rendimiento en tareas de clasificación para los distintos clasificadores evaluados. Utilizando la ANN propuesta como parte de un sistema completo incluyendo las etapas de clasificación junto con la extracción de características se obtienen resultados comparables con los de otras técnicas del estado del arte.

Aunque, el AEP fue evaluado utilizando un problema de clasificación binario de nuestro interés, puede ser utilizado también para problemas multiclase. Esto fue contemplado desde el planteo de la red al utilizar funciones softmax en lugar de funciones logísticas para la salida de clasificación.

La idea de fondo del AEP de mejorar la SNR de la salida, en contraposición con la idea detrás del DAE de corromper la entrada puede ser aplicada en formas alternativas. Pueden utilizarse ANN análogas al AEP en cualquier problema donde exista alguna técnica que permita mejorar la representación del patrón original, pero que no pueda ser utilizada en tiempo de operación. De este modo

podría entrenarse el sistema para aproximar el proceso de limpieza de ruido y/o extracción de características en tiempo de operación.

Hay que resaltar que para el entrenamiento de los modelos se utilizaron algoritmos de optimización generales. Los resultados probablemente pueden ser mejorados utilizando algoritmos diseñados específicamente para el entrenamiento de las redes propuestas.

Aunque el elevado número de hiperparámetros dota al algoritmo propuesto con mucha flexibilidad, también puede dificultar el entrenamiento de la red, evitando así que los usuarios sin experiencia obtengan buenos resultados. Por esta razón se propone un método sistemático para ajustar la arquitectura de la red y establecer sus hiperparámetros. Para el caso de la utilización de otro tipo de datos se proporcionaron algunas pautas para ajustar la estrategia propuesta.

## 7.2. Aportes

Durante el proceso de desarrollo de esta tesis se publicaron los siguientes trabajos:

- “Coherent averaging estimation autoencoders applied to evoked potentials processing”. Iván E. Gareis, Leandro D. Vignolo, Ruben D. Spies, Hugo L. Rufiner. *Neurocomputing*, ISSN 0925-2312 - 2017.
- “Open Access database of EEG signals recorded during imagined speech”. G. Pressel-Coreto, I. E. Gareis, H. L. Rufiner. 12th International Symposium on Medical Information Processing and Analysis (SIPAIM) - 2016
- “Detección de potenciales evocados relacionados a eventos en interfaces cerebro-computadora mediante transformada wavelet”. V. Peterson, Y. Atum, F. Jauregui, I. E. Gareis, R. C. Acevedo, H. L. Rufiner. *Revista Ingeniería Biomédica*, Volumen 7, Numero 14, pág. 50–58 - 2013.
- “Extracción de características en Interfaces cerebro computadoras basadas en transformaciones ortogonales: resultados preliminares”. V. Peterson, Y. Atum, F. Jauregui, I. E. Gareis, H. L. Rufiner, R. C. Acevedo. Proc. 3rd Chilean Meeting on Biomedical Engineering (JCIB 2012) - sep 2012.

También se realizó durante este trabajo la co-dirección de dos tesinas de grado relacionadas con la temática:



- “Implementación de un framework para la construcción de redes neuronales con aprendizaje profundo. Caso de aplicación: clasificación de señales cerebrales”. Alumno: L. Ferrado. Director: H. L. Rufiner. Co-director: I. E. Gareis. Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral - 2017.
- “Diseño y elaboración de una base de datos pública de registros electroencefalográficos orientados a la clasificación de habla imaginada”. Alumno: G. Pressel-Coreto. Director: H. L. Rufiner. Co-director: I. E. Gareis. Facultad de Ingeniería - Universidad Nacional de Entre Ríos - 2016.

## 7.3. Trabajo futuro

En esta sección se presentan algunas líneas de trabajo futuro relacionadas con esta tesis.

### 7.3.1. Autocodificadores de estimación de promedios coherentes apilados

Dependiendo de las características particulares de cada variante de AE, el algoritmo de apilamiento básico descrito en la Sección 4.9.1 tiene que ser ligeramente modificado. En el caso del AEP apilado, el proceso de promediado sobre la representación de características proporcionada por una capa de codificación previa no necesariamente mejora la SNR, debido justamente a la presencia de la transformación no lineal de codificación. Por esto se propone establecer como salida deseada promedios coherentes de los patrones de entrada en el proceso de apilamiento del AEP. Sin embargo, a medida que añadimos capas se hace progresivamente más difícil para las capas de decodificación para volver al espacio original. Para resolver este problema usamos las capas de decodificación previamente entrenadas, por lo que cada capa de decodificación sólo necesita interpretar la transformación realizada por su capa de codificación correspondiente.

En la Figura 7.1 se muestra un esquema del entrenamiento de un AEP apilado. Durante el entrenamiento de la  $l$ -ésima capa solo se ajustan los parámetros de  $f(\cdot; \Theta^l)^l$  y  $g(\cdot; \Phi^l)^l$ , mientras que los de todas las capas anteriores (i.e.  $\Theta^p$  y  $\Phi^p$  con  $p < l$ ) permanecen fijos. El error de reconstrucción es retropropagado a través de las capas de decodificación previamente entrenadas hasta las capas del AEP que

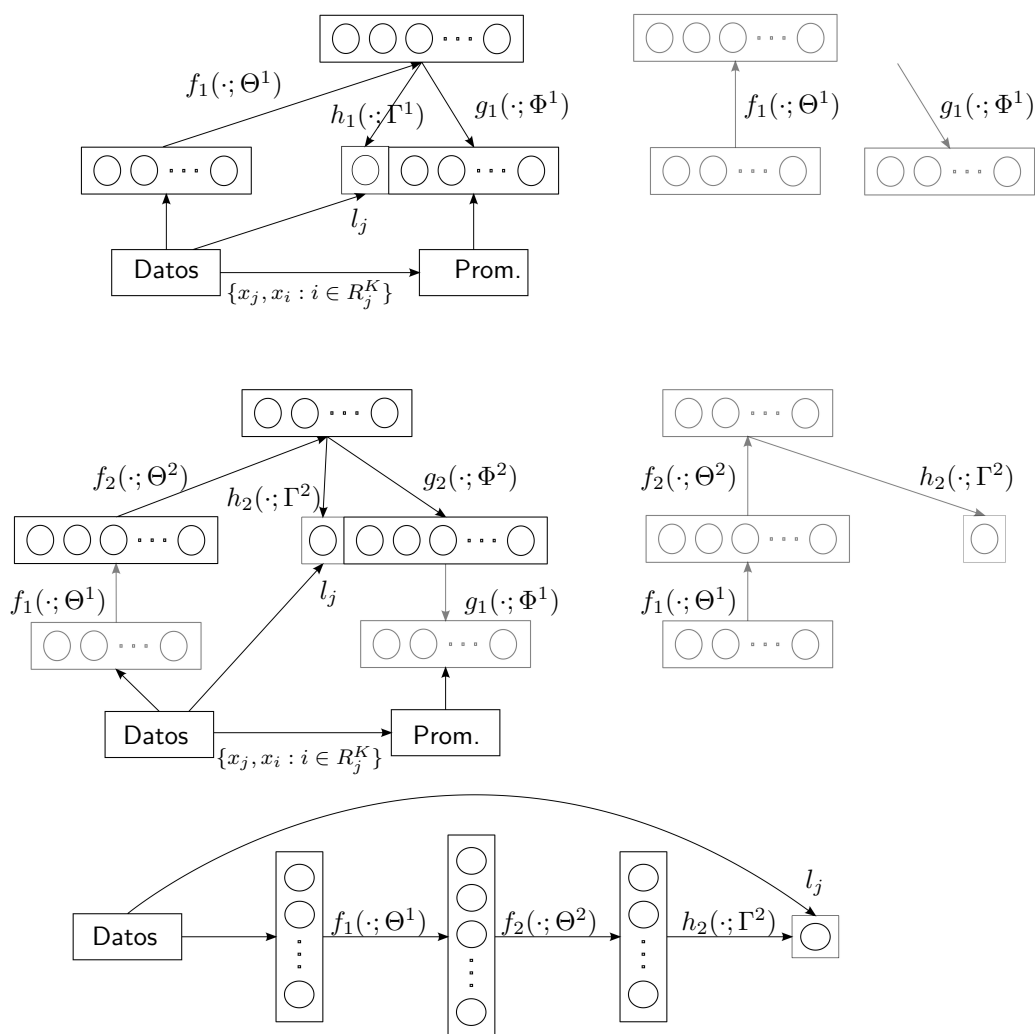


Figura 7.1: Esquema del entrenamiento de AEP apilado.

está siendo ajustado. Después del proceso de entrenamiento, todas las capas de decodificación se descartan.

### 7.3.2. Habla imaginada

Recientemente algunos investigadores han propuesto un nuevo paradigma de BCI denominado habla imaginada<sup>1</sup>. En este último el sujeto debe imaginar que pronuncia palabras o comandos pero sin emitir sonidos ni realizar movimientos. A pesar de esta restricción, se producen en la corteza motora procesos similares a cuando se pronuncia realmente la palabra. Estas señales pueden ser capturadas

<sup>1</sup>del inglés “imagined speech”.

mediante la EEG. Este fenómeno implica un problema de gran dificultad y aún abierto en la actualidad [19].

El uso de habla imaginada para la implementación de BCI se encuentra en sus primeras etapas. Las estrategias que se pueden desarrollar para utilizar estas señales todavía no están estudiadas a fondo y las tasas de reconocimiento que se obtienen están apenas encima de las tasas de decisión al azar [106]. Para avanzar en el desarrollo de estos sistemas uno de los problemas principales es la falta de bases de datos públicas con registros de EEG de habla imaginada. Esto dificulta la generación de nuevos algoritmos de clasificación, así como la comparación y validación de las propuestas realizadas por los distintos grupos de investigación.

Para afrontar el problema de reconocimiento de patrones en el contexto del habla imaginada durante el desarrollo de la presente tesis se realizó la adquisición de una base de datos propia con registros de varios sujetos en condiciones experimentales controladas. Se diseñó un protocolo de adquisición para el registro de la señal de EEG mediante 6 canales (F3, F4, C3, C4, P3 y P4) bajo dos modalidades, habla pronunciada y habla imaginada. Se empleó un diccionario compuesto por dos categorías de palabras: las 5 vocales del español y palabras que indican comandos (arriba, abajo, derecha, izquierda, adelante y atrás). Para la implementación del protocolo diseñado en este proyecto se desarrolló una aplicación sobre la plataforma BCI2000 que permite la presentación de las palabras, seguido de una ventana de 4 segundos para la imaginación o la pronunciación de la misma. Esta aplicación posibilita también el registro de la señal de audio sincronizada con la de EEG. El protocolo diseñado se practicó sobre 15 individuos de entre 24 y 28 años de edad, todos voluntarios, sanos e hispanohablantes (argentinos nativos), registrándose 50 repeticiones por palabra, de las cuales 10 correspondían a la modalidad pronunciada y el resto a la imaginada.

Se evaluó el desempeño del algoritmo propuesto por Torres et al. [94]. Los resultados obtenidos fueron superiores al azar en varios sujetos para 6 clases, lo que proporciona indicios de que existe información en la señal de EEG acerca de la identidad de la palabra imaginada. A partir de estos datos propios y los correspondientes resultados de referencia, se planea evaluar otros métodos de representación y aprendizaje. En primera instancia se ensayaran métodos en sus formas estándar y posteriormente, contando ya con un mejor entendimiento de los datos y del problema, se propondrán nuevos métodos *ad-hoc*.



---

# Acrónimos

Los acrónimos corresponden en general a la versión inglesa utilizada en la bibliografía del área.

**AE** Autoencoder (autocodificador). 36, 38, 39, 42, 43, 46–48, 50, 73, 75

**AEP** Autocodificadores de Estimación de Promedios coherentes. 46–55, 57–59, 63–71, 73, 75

**ANN** Artificial Neural Network (red neuronal artificial). 3, 8, 25, 31–36, 39–46, 48, 52, 53, 63, 71, 73

**AR** Autorregresivos. 59, 60

**BCI** Brain Computer Interface (interfaz cerebro-computadora). 1–3, 5–9, 15–19, 21, 44, 45, 47, 59, 61, 66, 76, 77

**BFGS** Broyden-Fletcher-Goldfarb-Shanno. 24

**BP** Backpropagation (retropropagación). 40–43

**BSS** Blind Source Separation (separación ciega de fuentes). 14

**CA** Coherent Averaging (promediación coherente). 14, 15, 46–48, 50

**CAE** Contractive Autoencoder (autocodificador contractivo). 39

**CAR** Common Average Reference (referencia promedio común). 7, 11

**CG** Conjugate Gradient (gradiente conjugado). 24, 64

**CNN** Convolutional Neural Networks (redes neuronales convolucionales). 44, 45, 68

- CNV** Contingent Negative Variation (variación contingente negativa). 13
- CSP** Common Spatial Patterns (patrones espaciales comunes). 7
- DAE** Denoising Autoencoder (autocodificador de limpieza de ruido). 38, 39, 45, 46, 48, 73
- DBN** Deep Belief Network (red de creencia profunda). 42, 45
- DL** Deep Learning (aprendizaje profundo). 31, 40, 42, 45
- ECoG** Electrocorticografía. 8
- EEG** Electroencefalografía. 1, 3, 6–12, 18, 19, 21, 45, 59–61, 77
- ELA** Esclerosis Lateral Amiotrófica. 16
- EOG** Electrooculografía. 7
- ERD** Event Related Desynchronization (desincronización relacionada con eventos). 17, 18
- ERN** Error Related Negativity (negatividad relacionada a errores). 13
- ERP** Event Related Potentials (potenciales relacionados con eventos). 12–14, 19, 20, 45, 60–62, 67
- ERS** Event Related Synchronization (sincronización relacionada con eventos). 17, 18
- FIR** Finite Impulse Response (respuesta finita al impulso). 62
- fMRI** Functional Magnetic Resonance Imaging (resonancia magnética nuclear funcional). 9
- fNIRS** Functional Near-Infrared Spectroscopy (espectroscopía del infrarrojo cercano funcional). 9
- GA** Genetic Algorithm (algoritmo genético). 8
- HMI** Human Machine Interface (interfaz hombre-máquina). 16

- 
- ICA** Independent Component Analysis (análisis de componentes independientes). 7
- ISI** Inter Stimulus Interval (intervalo inter estímulo). 20
- ITE** Infartos del Tronco Encefálico. 16
- ITR** Information Transfer Rate (tasa de transferencia de información). 2
- KLD** Kullback-Leibler Divergence (divergencia de Kullback-Leibler). 37
- KNN** k-Nearest Neighbors (k-vecinos más cercanos). 27, 28, 57, 58, 62, 66, 67
- LBFGS** Limited Memory Broyden-Fletcher-Goldfarb-Shanno (Broyden-Fletcher-Goldfarb-Shanno de memoria limitada). 57
- LDA** Linear Discriminant Analysis (análisis discriminante lineal). 8
- LDB** Local Discriminant Bases (bases discriminantes locales). 68
- LFP** Local Field Potential (potencial de campo local). 8, 10
- LMEC** Lesiones de la Médula Espinal Cervical. 16
- LPC** Linear Predictive Coding (codificación predictiva lineal). 59
- LSTM** Long Short Term Memory (gran memoria de corto plazo). 42
- MEG** Magnetoencefalografía. 9, 13
- MLP** Multi Layer Perceptron (perceptrón multi-capas). 8
- PCA** Principal Components Analysis (análisis de componentes principales). 7, 8
- PSD** Power Spectral Density (densidad espectral de potencia). 70, 71
- PSP** Postsynaptic Potentials (potenciales postsinápticos). 10
- RBM** Restricted Boltzman Machine (red de Boltzman restringida). 42
- ReLU** Rectified Linear Unit (unidad lineal rectificadora). 32, 53

- RNN** Recursive Neural Networks (redes neuronales recursivas). 42
- SAE** Sparse Autoencoder (autocodificador ralo). 37, 49
- SCP** Slow Cortical Potentials (potenciales corticales lentos). 19
- SF** Softmax Function (función softmax). 35
- SMAE** Smoothing Autoencoder (autocodificador de suavizado). 39
- SMR** Sensorimotor Rhythms (ritmos sensorimotores). 17, 18, 45
- SNN** Softmax Artificial Neural Network (red neuronal artificial softmax). 35, 36, 49, 57, 62, 64-67
- SNR** Signal to Noise Ratio (relación señal a ruido). 1, 6, 14, 15, 20, 46, 47, 61, 66, 73, 75
- SR1** Symmetric Rank 1 (rango simétrico 1). 24
- SSVEP** Steady State Visual Evoked Potentials (potenciales evocados visuales de estado estacionario). 18
- SVM** Support Vector Machine (máquina de soporte vectorial). 8, 27, 28, 57, 58, 62, 66, 67
- VAE** Variational Autoencoder (autocodificador variacional). 39
- WP** Wavelet Packet (paquete de onditas). 68



---

## Bibliografía

- [1] Xiu An, Deping Kuang, Xiaojiao Guo, Yilu Zhao, y Lianghua He. A Deep Learning Method for Classification of EEG Data Based on Motor Imagery. En *Intelligent Computing in Bioinformatics*, n<sup>o</sup> 8590 en Lecture Notes in Computer Science, págs. 203–210. Springer International Publishing, 2014. ISBN 978-3-319-09329-1 978-3-319-09330-7.
- [2] Jorge Aunon, Clare McGillem, y Donald Childers. Signal processing in evoked potential research: averaging and modeling. *Critical reviews in bio-engineering*, 5(4):323—367, 1981.
- [3] Ali Bashashati, Mehrdad Fatourehchi, Rabab K. Ward, y Gary E. Birch. A survey of signal processing algorithms in brain-computer interfaces based on electrical brain signals. *Journal of Neural Engineering*, 4(2):R32–57, 2007. ISSN 1741-2560.
- [4] Gerhard Bauer, Franz Gerstenbrand, y Erik Rumpl. Varieties of the locked-in syndrome. *Journal of neurology*, 221(2):77–91, 1979.
- [5] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. En *Neural Networks: Tricks of the Trade*, págs. 437–478. Springer, 2012.
- [6] Yoshua Bengio, Aaron Courville, y Pierre Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.
- [7] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

- 
- [8] Luigi Bianchi, Lucia Rita Quitadamo, Girolamo Garreffa, Gian Carlo Cardarilli, y Maria Grazia Marciani. Performances evaluation and optimization of brain computer interface systems in a copy spelling task. *IEEE Transactions on neural systems and rehabilitation engineering*, 15(2):207–216, 2007.
- [9] Guangyu Bin, Xiaorong Gao, Yijun Wang, Bo Hong, y Shangkai Gao. Vep-based brain-computer interfaces: time, frequency, and code modulations. *IEEE Computational Intelligence Magazine*, 4(4), 2009.
- [10] Benjamin Blankertz, Klaus-Robert Müller, Dean Krusienski, Gerwin Schalk, Jonathan Wolpaw, Alois Schlögl, Gert Pfurtscheller, José del R. Millán, Michael Schröder, y Niels Birbaumer. The BCI competition. III: Validating alternative approaches to actual BCI problems. *IEEE transactions on neural systems and rehabilitation engineering: a publication of the IEEE Engineering in Medicine and Biology Society*, 14(2):153–159, 2006. ISSN 1534-4320.
- [11] John Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. En *Neurocomputing*, págs. 227–236. Springer, 1990.
- [12] Anne-Marie Brouwer y Jan Van Erp. A tactile P300 Brain-Computer Interface. *Frontiers in neuroscience*, 4:19, 2010.
- [13] Anne-Marie Brouwer, Jan Van Erp, Dirk Heylen, Ole Jensen, y Mannes Poel. Effortless passive bcis for healthy users. En *International Conference on Universal Access in Human-Computer Interaction*, págs. 615–622. Springer, 2013.
- [14] Clemens Brunner, Giuseppe Andreoni, Luigi Bianchi, Benjamin Blankertz, Christian Breitwieser, Shinichiro Kanoh, Christian Kothe, Anatole Lécuyer, Scott Makeig, Jürgen Mellinger, Paolo Perego, Yann Renard, Gerwin Schalk, Putu Susila, Bastian Venthur, y Gernot Müller-Putz. BCI software platforms. En *Towards Practical Brain-Computer Interfaces*, págs. 303–331. Springer, 2012.
- [15] Hubert Cecotti y Axel Graser. Convolutional Neural Networks for P300 Detection with Application to Brain-Computer Interfaces. *IEEE Transac-*

- tions on Pattern Analysis and Machine Intelligence*, 33(3):433–445, 2011. ISSN 0162-8828.
- [16] Dan Cireşan, Alessandro Giusti, Luca Gambardella, y Jürgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. En *International Conference on Medical Image Computing and Computer-Assisted Intervention*, págs. 411–418. Springer, 2013.
- [17] Corinna Cortes y Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [18] Janis Daly y Jane Huggins. Brain-computer interface: Current and emerging rehabilitation applications. *Archives of physical medicine and rehabilitation*, 96(3):S1–S7, 2015.
- [19] Charles DaSalla, Hiroyuki Kambara, Makoto Sato, y Yasuharu Koike. Single-trial classification of vowel speech imagery using common spatial patterns. *Neural Networks*, 22(9):1334–1339, 2009.
- [20] George Dawson. A summation technique for detecting small signals in a large irregular background. *The Journal of Physiology*, 115(1):2p–3p, 1951. ISSN 0022-3751.
- [21] George Dawson. A summation technique for the detection of small evoked potentials. *Electroencephalography and Clinical Neurophysiology*, 6:65–84, 1954. ISSN 0013-4694.
- [22] Bruce Dobkin. Brain-computer interface technology as a tool to augment plasticity and outcomes for neurological rehabilitation. *The Journal of Physiology*, 579(Pt 3):637–642, 2007. ISSN 0022-3751.
- [23] R. P. W. Duin. Prtools version 3.0: A matlab toolbox for pattern recognition. En *Proc. of SPIE*, pág. 1331. 2000.
- [24] Heinz Werner Engl, Martin Hanke, y Andreas Neubauer. *Regularization of inverse problems*, tomo 375. Springer Science & Business Media, 1996.
- [25] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, y Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

- [26] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, y Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.
- [27] Lawrence Ashley Farwell y Emanuel Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, 70(6):510–523, 1988.
- [28] Reza Fazel-Rezai y Kamyar Abhari. A region-based P300 speller for brain-computer interface. *Canadian Journal of Electrical and Computer Engineering*, 34(3):81–85, 2009. ISSN 0840-8688.
- [29] Gary Garcia-Molina, Tsvetomira Tsoneva, y Anton Nijholt. Emotional brain-computer interfaces. *International journal of autonomous and adaptive communications systems*, 6(1):9–25, 2013.
- [30] Gene Golub, Michael Heath, y Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.
- [31] Bernhard Graimann, Brendan Allison, y Gert Pfurtscheller. Brain-computer interfaces non-invasive and invasive technologies. 2011.
- [32] Alex Graves, Abdel-rahman Mohamed, y Geoffrey Hinton. Speech recognition with deep recurrent neural networks. En *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, págs. 6645–6649. 2013. ISSN 1520-6149.
- [33] Christoph Guger, Shahab Daban, Eric Sellers, Clemens Holzner, Gunther Krausz, Roberta Carabalona, Furio Gramatica, y Guenter Edlinger. How many people are able to control a P300-based brain-computer interface (BCI)? *Neuroscience Letters*, 462(1):94 – 98, 2009. ISSN 0304-3940.
- [34] Mårten Gulliksson y Per-Åke Wedin. Analyzing the nonlinear L-curve. *Technical Report, Sweden: Department of Computer Science, University of Umeå.*, 1998.
- [35] William W Hager y Hongchao Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

- 
- [36] Uno Hämarik, Reimo Palm, y Toomas Raus. A family of rules for parameter choice in tikhonov regularization of ill-posed problems with inexact noise level. *Journal of Computational and Applied Mathematics*, 236(8):2146–2157, 2012.
- [37] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. 2010. Tech. Rep. UTML TR 2010-003.
- [38] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.
- [39] Geoffrey Hinton, Simon Osindero, y Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- [40] Geoffrey Hinton y Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. ISSN 0036-8075.
- [41] Kurt Hornik, Maxwell Stinchcombe, y Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
- [42] Xiaojuan Jiang, Yinghua Zhang, Wensheng Zhang, y Xian Xiao. A novel sparse auto-encoder for deep unsupervised learning. En *2013 Sixth International Conference on Advanced Computational Intelligence (ICACI)*, págs. 256–261. 2013.
- [43] Suwicha Jirayucharoensak, Setha Pan-Ngum, y Pasin Israsena. EEG-Based Emotion Recognition Using Deep Learning Network with Principal Component Based Covariate Shift Adaptation. *The Scientific World Journal*, 2014:e627892, 2014. ISSN 2356-6140.
- [44] Valer Jurcak, Daisuke Tsuzuki, y Ippaita Dan. 10/20, 10/10, and 10/5 systems revisited: their validity as relative head-surface-based positioning systems. *Neuroimage*, 34(4):1600–1611, 2007.

- 
- [45] Samira Ebrahimi Kahou, Christopher Pal, Xavier Bouthillier, Pierre Froumenty, Çağlar Gülçehre, Roland Memisevic, Pascal Vincent, Aaron Courville, Yoshua Bengio, Raul Chandias Ferrari, et al. Combining modality specific deep neural networks for emotion recognition in video. En *Proceedings of the 15th ACM on International conference on multimodal interaction*, págs. 543–550. ACM, 2013.
- [46] Diederik P Kingma y Max Welling. Auto-encoding variational bayes. En *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014. 2013.
- [47] Alex Krizhevsky, Ilya Sutskever, y Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. En *Advances in neural information processing systems*, págs. 1097–1105. 2012.
- [48] Andrea Kübler, Boris Kotchoubey, Thilo Hinterberger, Nimr Ghanayim, Juri Perelmouter, Margarete Schauer, Christoph Fritsch, Edward Taub, y Niels Birbaumer. The thought translation device: a neurophysiological approach to communication in total motor paralysis. *Experimental brain research*, 124(2):223–232, 1999.
- [49] Yann LeCun y Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.
- [50] Yann LeCun, Yoshua Bengio, y Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [51] Steven Lemm, Benjamin Blankertz, Thorsten Dickhaus, y Klaus-Robert Müller. Introduction to machine learning for brain imaging. *NeuroImage*, 56(2):387 – 399, 2011. ISSN 1053-8119. Multivariate Decoding and Brain Reading.
- [52] Deng Li y Yu Dong. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387, 2014. ISSN 1932-8346.
- [53] Junhua Li, Zbigniew Struzik, Liqing Zhang, y Andrzej Cichocki. Feature learning from incomplete eeg with denoising autoencoder. *Neurocomputing*, 165:23–31, 2015.

- 
- [54] Kongming Liang, Hong Chang, Zhen Cui, Shiguang Shan, y Xilin Chen. Representation Learning with Smooth Autoencoder. En *Computer Vision – ACCV 2014*, nº 9004 en Lecture Notes in Computer Science, págs. 72–86. Springer International Publishing, 2014.
- [55] Ralph Linsker. *An application of the principle of maximum information preservation to linear systems*. IBM Thomas J. Watson Research Division, 1988.
- [56] Steven Luck. *An introduction to the event-related potential technique*. MIT press, 2014.
- [57] John Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, 1975.
- [58] Jaakko Malmivuo y Robert Plonsey. *Bioelectromagnetism: Principles and Applications of Bioelectric and Biomagnetic Fields*. Oxford University Press, 1995. ISBN 978-0-19-505823-9.
- [59] Qirong Mao, Ming Dong, Zhengwei Huang, y Yongzhao Zhan. Learning salient features for speech emotion recognition using convolutional neural networks. *IEEE Transactions on Multimedia*, 16(8):2203–2213, 2014.
- [60] Dennis McFarland, William Sarnacki, y Jonathan Wolpaw. Electroencephalographic (EEG) control of three-dimensional movement. *Journal of Neural Engineering*, 7(3):036007, 2010.
- [61] Dennis McFarland y Jonathan Wolpaw. Brain-Computer Interface Operation of Robotic and Prosthetic Devices. *Computer*, 41(10):52–56, 2008. ISSN 0018-9162.
- [62] Clare McGillem, Jorge Aunon, y Donald Childers. Signal processing in evoked potential research: applications of filtering and pattern recognition. *Critical Reviews in Bioengineering*, 6(3):225—265, 1981.
- [63] Marvin Minsky y Seymour Papert. *Perceptrons: an introduction to computational geometry (expanded edition)*. 1988.
- [64] Jennifer Müller y Samuli Siltanen. *Linear and nonlinear inverse problems with practical applications*. Computational Science and Engineering. Society for Industrial and Applied Mathematics, 2012.

- [65] Christa Neuper y Gert Pfurtscheller. Motor imagery and erd. *Event-related desynchronization*, 6:303–325, 1999.
- [66] Luis Fernando Nicolas-Alonso y Jaime Gomez-Gil. Brain computer interfaces, a review. *Sensors*, 12(2):1211–1279, 2012.
- [67] Femke Nijboer, Adrian Furdea, Ingo Gunst, Jürgen Mellinger, Dennis J. McFarland, Niels Birbaumer, y Andrea Kübler. An auditory brain–computer interface (bci). *Journal of Neuroscience Methods*, 167(1):43 – 50, 2008. ISSN 0165-0270. Brain-Computer Interfaces (BCIs).
- [68] Anton Nijholt, Danny Plass-Oude Bos, y Boris Reuderink. Turning shortcomings into challenges: Brain–computer interfaces for games. *Entertainment Computing*, 1(2):85–94, 2009. ISSN 1875-9521.
- [69] Robert Oostenveld y Peter Praamstra. The five percent electrode system for high-resolution EEG and ERP measurements. *Clinical neurophysiology*, 112(4):713–719, 2001.
- [70] Victoria Peterson, Rubén Acevedo, Hugo Rufiner, y Ruben Spies. Local discriminant wavelet packet basis for signal classification in brain computer interface. En *Anales del VI Congreso Latinoamericano de Ingeniería Biomédica (CLAIB 2014)*, pág. 279. 2014.
- [71] Gert Pfurtscheller. Event-related synchronization (ers): an electrophysiological correlate of cortical areas at rest. *Electroencephalography and clinical neurophysiology*, 83(1):62–69, 1992.
- [72] Gert Pfurtscheller y A Aranibar. Evaluation of event-related desynchronization (erd) preceding and following voluntary self-paced movement. *Electroencephalography and clinical neurophysiology*, 46(2):138–146, 1979.
- [73] Nicolas Pinto, David Doukhan, James J DiCarlo, y David D Cox. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Comput Biol*, 5(11):e1000579, 2009.
- [74] John Polich. Updating P300: an integrative theory of P3a and P3b. *Clinical neurophysiology*, 118(10):2128–2148, 2007.



- [75] Yann Renard, Fabien Lotte, Guillaume Gibert, Marco Congedo, Emmanuel Maby, Vincent Delannoy, Olivier Bertrand, y Anatole Lécuyer. Openvibe: An open-source software platform to design, test, and use brain–computer interfaces in real and virtual environments. *Presence: teleoperators and virtual environments*, 19(1):35–53, 2010.
- [76] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, y Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. En *Proceedings of the 28th international conference on machine learning (ICML-11)*, págs. 833–840. 2011.
- [77] Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.
- [78] Otto Rompelman y H. Ros. Coherent averaging technique: A tutorial review Part 1: Noise reduction and the equivalent filter. *Journal of Biomedical Engineering*, 8(1):24–29, 1986. ISSN 0141-5425.
- [79] David Rumelhart, Geoffrey Hinton, y Ronald Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [80] Marc Saab. Basic concepts of surface electroencephalography and signal processing as applied to the practice of biofeedback. *Biofeedback*, 36(4), 2008.
- [81] Naoki Saito, Brons M Larson, Bertrand Bénichou, y Bruno Olshausen. Sparsity vs. statistical independence from a best-basis viewpoint. En *Proc. SPIE*, tomo 4119, págs. 474–486. 2000.
- [82] Mathew Salvaris y Francisco Sepulveda. Visual modifications on the p300 speller bci paradigm. *Journal of neural engineering*, 6(4):046011, 2009.
- [83] Saeid Sanei y Jonathon A Chambers. *EEG signal processing*. John Wiley & Sons, 2013.
- [84] Gerwin Schalk, Dennis McFarland, Thilo Hinterberger, Niels Birbaumer, y Jonathan Wolpaw. BCI2000: a general-purpose brain-computer interface (BCI) system. *IEEE Transactions on biomedical engineering*, 51(6):1034–1043, 2004.

- [85] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. ISSN 0893-6080.
- [86] Mark Schmidt. minfunc: unconstrained differentiable multivariate optimization in matlab. <http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html>, 2005.
- [87] Eric Sellers, Dean Krusienski, Dennis McFarland, Theresa Vaughan, y Jonathan Wolpaw. A P300 event-related potential brain–computer interface (BCI): the effects of matrix size and inter stimulus interval on performance. *Biological psychology*, 73(3):242–252, 2006.
- [88] SJM Smith. EEG in neurological conditions other than epilepsy: when does it help, what does it add? *Journal of Neurology, Neurosurgery & Psychiatry*, 76(suppl 2):ii8–ii12, 2005.
- [89] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, y Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [90] André Stuhlsatz, Christine Meyer, Florian Eyben, Thomas Zielke, Günter Meier, y Björn Schuller. Deep neural networks for acoustic emotion recognition: raising the benchmarks. En *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, págs. 5688–5691. IEEE, 2011.
- [91] Shravani Sur, VK Sinha, et al. Event-related potential: An overview. *Industrial psychiatry journal*, 18(1):70, 2009.
- [92] Yaniv Taigman, Ming Yang, MarcÁurelio Ranzato, y Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, págs. 1701–1708. 2014.
- [93] Kazuo Tanaka, Kazuyuki Matsunaga, y Hua Wang. Electroencephalogram-Based Control of an Electric Wheelchair. *IEEE Transactions on Robotics*, 21(4):762–766, 2005. ISSN 1552-3098.
- [94] Alejandro Torres-García, Carlos Reyes-García, Luis Villaseñor-Pineda, y Juan Ramírez-Cortés. Análisis de señales electroencefalográficas para la

- clasificación de habla imaginada. *Revista Mexicana de Ingeniería Biomédica*, 34(1):23–39, 2013.
- [95] George Townsend, Brandon LaPallo, Chadwick Boulay, Dean Krusienski, Gerald Frye, Christopher Hauser, Neil Schwartz, Theresa Vaughan, Jonathan Wolpaw, y Eric Sellers. A novel P300-based brain–computer interface stimulus presentation paradigm: Moving beyond rows and columns. *Clinical Neurophysiology*, 121(7):1109 – 1120, 2010. ISSN 1388-2457.
- [96] Jose Antonio Urigüen y Begoña Garcia-Zapirain. EEG artifact removal—state-of-the-art and guidelines. *Journal of neural engineering*, 12(3):031001, 2015.
- [97] Anirudh Vallabhaneni, Tao Wang, y Bin He. Brain—Computer Interface. En *Neural Engineering*, págs. 85–121. Springer, 2005.
- [98] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, y Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. En *Proceedings of the 25th international conference on Machine learning*, págs. 1096–1103. ACM, 2008.
- [99] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, y Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.
- [100] Curtis Vogel. Non-convergence of the l-curve regularization parameter selection method. *Inverse problems*, 12(4):535, 1996.
- [101] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, y Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.
- [102] Jonathan Wolpaw, Niels Birbaumer, Dennis McFarland, Gert Pfurtscheller, y Theresa Vaughan. Brain-Computer Interfaces for Communication and Control. *Clinical neurophysiology*, 113(6):767–791, 2002.
- [103] Jonathan Wolpaw y Elizabeth Winter Wolpaw. *Brain-computer interfaces: principles and practice*. OUP USA, 2012.

- [104] Stephen Wright y Jorge Nocedal. Numerical optimization. *Springer Science*, 35:67–68, 1999.
- [105] Yu Zhang, Guoxu Zhou, Jing Jin, Qibin Zhao, Xingyu Wang, y Andrzej Cichocki. Sparse bayesian classification of EEG for brain-computer interface. *IEEE Transactions on Neural Networks and Learning Systems*, 27(11):2256–2267, 2016.
- [106] Shunan Zhao y Frank Rudzicz. Classifying phonological categories in imagined and articulated speech. En *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, págs. 992–996. 2015. ISSN 1520-6149.

**Doctorado en Ingeniería**  
**mención Señales Sistemas e Inteligencia Computacional**

Título de la obra:

**Algoritmos bioinspirados para la implementación de Interfaces Cerebro Computadoras**

Autor: Iván Emilio Gareis

Lugar: Santa Fe, Argentina

Palabras Clave:

Potenciales Relacionados con Eventos,  
Interfaces Cerebro Computadora,  
Redes Neuronales Artificiales,  
Problemas Multiobjetivo,  
Promediación Coherente,  
Autocodificadores.