Universidad de la República

Proyecto de Grado de Ingeniería en Computación

Compresión multicanal sin pérdida de electroencefalogramas

Autores:
Ignacio Capurro
Eugenio Rovira

Supervisores:
Federico Lecumberry
Álvaro Martín
Ignacio Ramirez

Núcleo de Teoría de la Información
Facultad de Ingeniería

24 de mayo de 2014

Resumen

Este trabajo propone un algoritmo de compresión sin pérdida para electroencefalogramas (EEG) multicanal haciendo uso de modernas herramientas de teoría de la información (como codificación universal y predicción universal) así como también herramientas de procesamiento de señales, combinadas con métodos más simples para explotar las redundancias espaciales y temporales que se presentan comúnmente en los EEG. El algoritmo de compresión es aplicado a tres diferentes bases de datos públicas de EEG. El algoritmo multicanal logra tasas de compresión superiores a las observadas al comprimir canales individuales por separado, y también supera el estado del arte actual en compresión sin pérdida de EEG multicanal.

Índice general

Índice de figuras

Índice de cuadros

Índice de algoritmos

Notación y abreviaciones

General

$\bar{x}$	Vector de predicciones, pág. 5.
ϵ	Vector de errores de la predicción primaria, pág. 5.
$\hat{\epsilon}$	Vector de errores de la predicción ajustada, pág. 9.
b	Tira de bits, pág. 11.
$c \in \{1, \ldots, M\}$	Canal, pág. 5.
F	Cantidad de archivos de una base de datos, pág. 19.
k	Cantidad de agrupaciones utilizadas en una configuración, pág. 6.
L	Conjunto de canales líderes, pág. 7.
LCM	Largo de código medio, pág. 15.
M	Cantidad de canales de un archivo, pág. 5.
N	Cantidad de muestras de un vector de muestras de un canal, pág. 5.
S	Conjunto de agrupaciones de canales asociadas a los líderes, pág. 7.
$v(c)$	Vector de un canal c, para los vectores definidos, pág. 5.
v_t	Vector en el tiempo t, para los vectores definidos, pág. 5.
x	Vector de muestras, pág. 4.
agrupación	Conjunto de canales asociados en base a un canal líder, pág. 2.
AR(p)	Modelo autorregresivo de orden p, pág. 2.
configuración	Partición del conjunto de canales en agrupaciones, pág. 2.
líder	Canal de una agrupación que luego es usado para codificar los restantes canales en base a él., pág. 2.

K-means

$\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$	Conjunto de puntos de referencia, pág. 7.

μ Centroide de un conjunto de vectores de errores de predicción, pág. 7.

ρ Variación obtenida en la primera iteración del algoritmo, pág. 7.

σ Factor de acercamiento, pág. 7.

τ Parámetro para la condición de parada, pág. 7.

K-medoids

ϕ Costo de la configuración en distancia ℓ_1, pág. 9.

O Canales no líderes, pág. 9.

Golomb

$\bar{N}$ Cantidad de valores codificados hasta el momento, pág. 10.

$\bar{N}^-$ Cantidad de valores negativos codificados hasta el momento, pág. 10.

A Acumulado del valor absoluto de los valores codificados hasta el momento, pág. 10.

m Orden de la codificación de Golomb, pág. 10.

r Valor mapeado con rice, pág. 11.

Capítulo 1

Introducción

El procesamiento de señales en la medicina es un campo de investigación de rápido crecimiento que está produciendo aplicaciones cada vez más sofisticadas para la medicina de hoy en día [1–7]. En neurobiología, los electroencefalogramas (EEG) son un método no invasivo utilizado para medir las ondas cerebrales de una persona, que se aplica en un extenso número de campos de estudio en donde se intentan entender las funciones del cerebro tanto en salud como en enfermedad. Algunos de estos campos son el estudio de la epilepsia o el desorden de sueño. Desde su descubrimiento por Berger [8], muchas actividades de investigación se han centrado en cómo extraer información útil acerca de las condiciones del cerebro basándose en distintas características de las señales de EEG. Muchas aplicaciones requieren la adquisición, almacenamiento y procesamiento automático de estas señales durante largos períodos de tiempo [1, 4, 9–13]. Por ejemplo, para algunos estudios de epilepsia requieren monitoreos de 24 horas. Para tener una idea del espacio requerido por estos registros, el rango de frecuencia normal para una señal de EEG de un adulto está entre 0.1–100Hz, por lo cual, por el teorema de Nyquist-Shannon, se requiere una frecuencia de muestreo al menos de 200Hz. Para un EEG muestreado a esta frecuencia, con una resolución de 16 bits por muestra, 10 canales, y una duración de 24 horas, se requieren aproximadamente 330 MB de espacio de almacenamiento. En otros estudios más modernos, por ejemplo medir la función y/o la capacidad cognitiva [14], son necesarias frecuencias de muestreo de 1000Hz, y más de 31 electrodos, haciendo que una lectura de 24 horas pueda ocupar más de 5 GB de espacio de almacenamiento. Además, para diagnosticar enfermedades y para medir la efectividad de un tratamiento, el proceso de análisis normalmente toma un período muy largo de tiempo. Ya que cada muestra de una señal de EEG es importante, y no se puede distorsionar ninguna sin consulta de expertos, el almacenamiento de señales de EEG a largo plazo se tiene que hacer sin ninguna pérdida de información. El interés de la comunidad biomédica en la compresión de señales de EEG esta motivado por esta gran cantidad de datos involucrados en la recopilación de información de EEG que requiere más memoria para el almacenamiento y ancho de banda para la transmisión [1, 4, 9–13].

En muchas situaciones donde se capturan señales de EEG es necesaria la transmisión inalámbrica de los datos a un dispositivo remoto. Por ejemplo, un paciente que tenga electrodos colocados en su cuero cabelludo durante varios días y necesite que su EEG sea transmitido a su dispositivo móvil [15]. En estos casos, el hardware que transmite los datos de forma inalámbrica debe tener un bajo consumo de energía para poder funcionar ininterrumpidamente durante el tiempo deseado. Comprimir el EEG, disminuye el consumo de energía del transmisor, ya que la tasa de transferencia de bits es menor. Sin embargo, si el algoritmo de compresión de los datos es de alta complejidad computacional, esto se traduce en que el circuito que lo implemente tenga un mayor

consumo de energía. Por ello, es necesario que el algoritmo de compresión logre altas tasas de compresión y que además sea de baja complejidad computacional.

Para lograr tasas de compresión altas es necesario explotar la mayor cantidad posible de redundancias en los datos. Para ello se pueden aprovechar correlaciones intra-canal, como por ejemplo las correlaciones temporales, así como las correlaciones inter-canal, es decir las correlaciones espaciales.

La compresión eficiente de la señal del EEG es una tarea difícil debido a la aleatoriedad inherente en la señal. Esto hace que sea difícil obtener altas tasas de compresión con métodos de compresión sin pérdidas [9]. Una revisión de las técnicas de compresión aplicadas a las señales de EEG se ha descrito en [9]. Compresores sin pérdida de dos etapas que involucran predictores se han reportado en [4, 9, 13]. El esquema general en este tipo de codificadores consiste en calcular una predicción de cada muestra de la señal en función de muestras que ya han sido codificadas, y codificar la diferencia entre la predicción y el valor real.

El codificador sin perdida que exponemos tiene un esquema de codificación de dos etapas involucrando predictores. En la etapa de predicción se utilizaron dos predictores, el primero explota la relación entre las muestras de un canal a lo largo del tiempo (redundancia temporal) y el segundo utiliza la redundancia entre las muestras de los diferentes canales para un mismo tiempo de muestreo (redundancia espacial).

Nuestro predictor temporal está basado en modelos autorregresivos independientes para cada canal, los cuales son ampliamente utilizados [16] para señales de EEG. Estos modelos especifican que la muestra del siguiente instante de tiempo de un canal depende linealmente de las muestras que la preceden inmediatamente en el tiempo. La cantidad de muestras de las que depende la predicción de la siguiente muestra está determinada por el orden del modelo autorregresivo; el término AR(p) se utiliza para hacer referencia a un modelo autorregresivo de orden p. Los resultados experimentales reportados en la sección 3.3 muestran que órdenes cercanos a 3 son suficientes para obtener buenas predicciones con estos modelos y órdenes mayores no producen mejoras significativas.

El segundo predictor utiliza la similitud de los errores de predicción resultantes de aplicar el predictor temporal a los canales, buscando agrupar canales con errores de predicción similares. En cada agrupación de canales se busca un canal *líder* para luego calcular la diferencia del error de predicción resultado del predictor temporal para este canal con el de los canales restantes en la agrupación. A una partición del conjunto de canales en agrupaciones, cada una con un líder, le llamamos *configuración*. La búsqueda de configuraciones se hace en base a un algoritmo de agrupamiento basado en k-means [17] y k-medoids [18], utilizando como distancia entre canales las distancias ℓ_2 y ℓ_1 respectivamente.

El número de agrupaciones utilizado en una configuración es fijado experimentalmente observando que para las 3 bases de datos utilizadas (indistintamente de la cantidad de canales) se obtienen los mejores resultados con un número similar de agrupaciones. Si bien las configuraciones son distintas para cada lectura en la cual son calculadas vemos que elegir una configuración global por base de datos da buenos resultados.

Una vez calculados los errores de predicción (temporales y espaciales) para todos los canales se pasa a la etapa de codificación. En esta etapa utilizamos un codificador de Golomb [19] independiente para cada canal.

Los resultados obtenidos con el compresor son prometedores, se lograron ganancias de hasta 15% en la tasa de compresión utilizando ambos predictores por sobre utilizar solo el predictor temporal. Las comparaciones con otros métodos de compresión también son satisfactorias. Cabe remarcar entre estas la comparación con otros métodos de compresión multicanal de estado del

arte [20], en donde para la misma base de datos las tasas de compresión obtenidas con nuestro método son mejores.

El resto del documento está organizado de la siguiente manera. En el capítulo 2 detallamos nuestro algoritmo de codificación y decodificación. En el capítulo 3 presentamos experimentos realizados junto con sus resultados y observaciones. Por último en el capítulo 4 damos nuestras conclusiones y trabajo futuro.

Capítulo 2

Descripción del algoritmo

En este capítulo describimos nuestro algoritmo de codificación, el cual puede descomponerse conceptualmente en los siguientes pasos:

1. Cálculo de los errores de predicción para cada vector de muestras de un canal en base a predictores lineales independientes para cada canal. Esta primera etapa de predicción nos permite explotar la autocorrelación de las señales en el tiempo.
2. Agrupación de canales por similitud de vectores de errores de predicción, y definición de un canal líder de cada agrupación. Estas agrupaciones son utilizadas para que los canales con vectores de errores de predicción similares puedan ser codificados en base al líder su agrupación correspondiente, con el objetivo de aprovechar la correlación entre canales.
3. Codificación del vector de errores de predicción de los canales distinguidos como líderes y de la diferencia entre los vectores de errores de predicción de cada uno de los canales restantes con respecto al líder de su respectiva agrupación.

En las siguientes subsecciones de este capítulo describimos detalladamente cada uno de estos pasos.

2.1 Predictores lineales

Para modelar la señal captada por un canal, $x = (x_1, x_2, \ldots, x_N)$, haciendo uso de la redundancia en el tiempo, una de las técnicas más utilizadas consiste en usar un modelo autoregresivo de orden p, AR(p), en el cual la muestra en un instante t, x_t, es el resultado de sumar ruido (un valor aleatorio) a una estimación

$$x_t = \bar{x}_t + \epsilon_t \,, \quad t > p \,. \tag{2.1}$$

La estimación de la muestra x_t está dada por

$$\bar{x}_t = \sum_{i=1}^{p} a_i x_{t-i} \,, \quad t > p \,, \tag{2.2}$$

donde $a_1 ... a_p$ son *coeficientes* del modelo AR(p).

Si los coeficientes no son conocidos de antemano, éstos pueden estimarse de diversas maneras a partir de las muestras observadas. Una de éstas es resolviendo una instancia del problema de mínimos cuadrados. Este consiste en obtener los parámetros $a_1 ... a_p$ tal que minimicen el error cuadrático acumulado

$$\sum_{t=p+1}^{N} \epsilon_t^2, \tag{2.3}$$

donde N es la cantidad de muestras del canal. Para solucionar este problema existen técnicas como descomposición QR o descomposición en valores singulares SVD [21]. Otra forma de obtener los coeficientes, es resolviendo las ecuaciones de Yule-Walker, a las cuales se llega asumiendo hipótesis de estacionariedad de los datos [22]. Dichas ecuaciones pueden ser resueltas mediante técnicas como Levinson-Durbin [23] o Burg [24]. Éstas son utilizadas en diversas áreas tales como finanzas, geofísica y pronóstico del tiempo. Dichas técnicas son de menor complejidad computacional que las técnicas mencionadas para resolver el problema de mínimos cuadrados. En nuestro caso elegimos el método de Burg utilizando la implementación detallada en [25] [1].

Cabe remarcar que en una aplicación real, dado que las primeras p muestras son necesarias para realizar la primera predicción, al menos p muestras deben ser codificadas con algún método alternativo al principio de cada canal, para que el decodificador pueda realizar el camino inverso al de codificación, esto es sumar a la estimación $\bar{x}_t$ el error ϵ_t para obtener el valor original x_t. En nuestro caso optamos por extender las definiciones (2.2) y (2.1) con $\bar{x}_t = 0$ para $t \leq p$, de modo que ϵ_t coincide con x_t para $t \leq p$. En otras palabras, las primeras p muestras de cada canal se codifican literalmente.

2.2 Agrupación de canales

De aquí en mas utilizaremos la notación $x(c)$ con $c \in \{1, 2, \ldots, M\}$ para representar al vector de muestras del canal c, donde M es la cantidad de canales. Para aprovechar la correlación espacial entre canales, los agrupamos por similitud de los errores de predicción. Con este fin, usamos como medida de distancia entre dos canales a y b el cuadrado de la norma euclídea, ℓ_2, de la diferencia entre los vectores de errores de predicción,

$$d_{\ell_2}(\epsilon(a), \epsilon(b)) \triangleq \sum_{t=1}^{N} (\epsilon_t(a) - \epsilon_t(b))^2, \tag{2.4}$$

donde $\epsilon_t(c)$ denota el vector de errores de predicción del canal c en el tiempo t. Otras medidas fueron evaluadas, como la entropía empírica de orden cero de las diferencias, considerando la secuencia $x_1(c), x_2(c), ..., x_N(c)$ como una sucesión de variables aleatorias independientes, pero empíricamente observamos que la definida en (2.4) dio los mejores resultados.

Dentro de cada agrupación definimos un líder para codificar todos los canales de dicha agrupación en base a él. A un conjunto de agrupaciones que particiona el conjunto de todos los canales conjuntamente con una definición de un líder para cada agrupación le llamamos *configuración*.

El algoritmo utilizado para definir una configuración consiste en dos etapas. En la primera se aplica k-means [17], que es un algoritmo de agrupamiento iterativo. Esta agrupación inicial se

[1] Este código esta públicamente disponible en el siguiente repositorio git https://github.com/RhysU/ar

refina en una segunda etapa aplicando k-medoids [26], que es otro algoritmo de agrupamiento similar a k-means cuya diferencia es que toma puntos (en nuestro caso vectores de errores de predicción) como centros y trabaja con norma ℓ_1. El uso de la norma ℓ_2 por sobre la norma ℓ_1 en la primera etapa está motivado por el hecho de que la primera es una norma diferenciable y por lo tanto nos asegura la obtención de máximos locales al contrario de la norma ℓ_1 que no es diferenciable y corremos el riesgo de caer en puntos singulares. Sin embargo, como describimos más adelante en la sección 3.2, el largo de codificación de los errores de predicción, que es lo que nos interesa minimizar, depende de la norma ℓ_1. Con esto en mente nuestra estrategia será agrupar primero los puntos en base a la norma ℓ_2 y luego refinar esta agrupación con la norma ℓ_1.

K-means es un método iterativo que intenta resolver el problema de hallar k particiones o grupos de un conjunto de n puntos, de manera que cada uno de estos pertenezca al grupo cuyo centroide está a menor distancia con respecto al punto en cuestión.

Específicamente, dado $\{\epsilon(1), \epsilon(2), \ldots, \epsilon(M)\}$, conjunto de vectores de errores de predicción, k-means construye una partición de k conjuntos de canales, $S = \{S_1, S_2, ..., S_k\}$, a fin de minimizar la suma de las distancias al cuadrado de cada punto con respecto al centroide del grupo al que pertenece

$$S = \text{argmin}_s \sum_{i=1}^{k} \sum_{c \in S_i} d_{\ell_2}(\epsilon(c), \mu_i), \tag{2.5}$$

donde μ_i es el centroide de $\{\epsilon(s) : s \in S_i\}$, que se calcula como

$$\mu_i = \frac{1}{|S_i|} \sum_{c \in S_i} \epsilon(c). \tag{2.6}$$

Presentamos el pseudogódigo del algoritmo k-means utilizado en la primera etapa del cálculo de una configuración en el algoritmo 2.2.1. Este recibe como entradas los vectores de errores de predicción de los canales, la cantidad de agrupaciones k que queremos retornar, un factor τ utilizado en la condición de parada y el factor de acercamiento σ utilizado para la actualización de los puntos λ. El algoritmo comienza tomando un subconjunto aleatorio Λ de tamaño k de los vectores de errores de predicción, como puntos de referencia iniciales. Luego se itera sobre los puntos de E en el ciclo de la línea 2 para actualizar este conjunto de puntos Λ. En cada iteración se busca, en la línea 5, el punto de Λ, λ_i, más cercano al punto a de E. Luego, en la línea 6, se actualiza λ_i haciendo una combinación convexa de proporción σ entre su valor original y el valor de a. En la primera iteración se guarda la variación obtenida entre los puntos de Λ actualizados con los puntos originales (líneas 8 a 10), a la cual llamaremos ρ. Luego, en cada iteración el criterio de parada consiste en comparar la variación actual con una fracción de ρ (línea 11). Una vez que tenemos estos puntos calculados, los cuales, si el algoritmo convergiera al óptimo global coincidirían con los μ de la ecuación (2.5), debemos construir las configuraciones. Para nuestro caso necesitamos elementos del conjunto E como líderes y para ello tomamos los más cercanos en distancia ℓ_2 a cada uno de los puntos de Λ (línea 13). Luego tenemos que asociar a estos líderes los elementos restantes del conjunto E en base a sus distancias con los elementos de Λ

(línea 14).

Algoritmo 2.2.1: k-means con distancia ℓ_2

entrada: $E : \{\epsilon(1), \epsilon(2), \ldots, \epsilon(M)\}$
k : Número de particiones.
τ : Factor de parada.
σ : Parámetro para el ajuste iterativo de los puntos de referencia.
salida : $L = \{l_1, l_2, ..., l_k\}$: conjunto de líderes. $S = \{S_1, S_2, ..., S_k\}$: conjunto de agrupaciones.

1 $\Lambda \leftarrow$ Subconjunto aleatorio de E de tamaño k como puntos de referencia iniciales.
2 **repeat**
3 $\quad \hat{\lambda}_i \leftarrow \lambda_i \, , \forall i = 1 \ldots k$
4 $\quad$ **foreach** $a \in \{1, \ldots, M\}$ **do**
5 $\quad\quad i \leftarrow \text{argmin}_i\{d_{\ell_2}(\epsilon(a), \lambda_i) : i \in \{1, \ldots, k\}\}$
$\quad\quad$ `// Actualizar` λ `en base a su distancia con` $\epsilon(a)$ `y al parámetro` σ`.`
6 $\quad\quad \lambda_i \leftarrow \lambda_i + (\epsilon(a) - \lambda_i)\sigma$
7 $\quad$ **end**
8 $\quad$ **if** *primera iteración* **then**
9 $\quad\quad \rho \leftarrow \sum_{i=1\ldots k} |\hat{\lambda}_i - \lambda_i|$
10 $\quad$ **end**
11 **until** $\sum_{i=1\ldots k} |\hat{\lambda}_i - \lambda_i| < \rho/\tau$;
$\quad$ `// Definir los líderes y sus agrupaciones.`
12 **foreach** $i \in \{1, ..., k\}$ **do**
13 $\quad l_i \leftarrow \text{argmin}_{l_i} d_{\ell_2}(\epsilon(l_i), \lambda_i), l_i \in \{1, ..., M\}$
14 $\quad S_i \leftarrow \{a : a \in \{1, ..., M\}, i = \text{argmin}_i\{d_{\ell_2}(\epsilon(a), \lambda_i) : i \in \{1, \ldots, k\}\}$
15 **end**

En la etapa de inicialización del algoritmo 2.2.1 vemos que el conjunto de puntos de referencia, Λ, se elige de forma aleatoria, y siendo este un problema no convexo, el algoritmo puede no converger al óptimo global de la función de costo definida en (2.5), sino que puede estancarse en óptimos locales.

Podemos ver el problema con un ejemplo representado en la figura 2.1. Consideremos un conjunto de n puntos que pueden separarse en 4 subconjuntos, cada uno contenido en una esfera de radio δ, dispuestos en una línea y separados entre sí por una distancia d mucho mayor que δ.

Una posible elección aleatoria de los puntos de referencia iniciales de Λ podría ser la indicada por puntos oscuros en la figura 2.1.

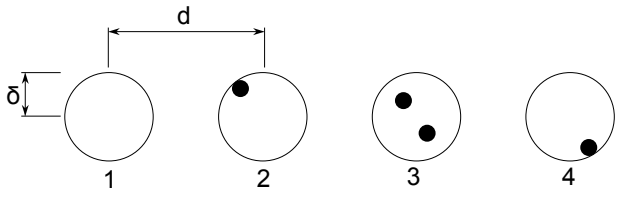


FIGURA 2.1: Ejemplo de un estado inicial de los puntos de referencia del algoritmo k-means.

Es claro que la solución óptima global comprende un punto de referencia en el centro de cada esfera, sin embargo, en la figura 2.2 vemos que la posición de los puntos de referencia en k-means (previo a la definición de los líderes y las agrupaciones) es otra.

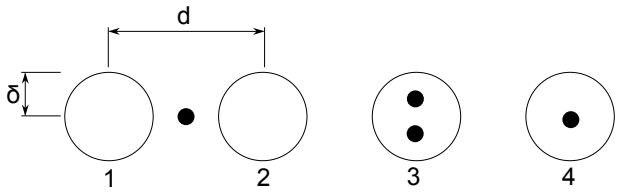


FIGURA 2.2: Estado de los puntos de referencia del algoritmo k-means previo a la definición de los líderes y las agrupaciones, tomando como puntos de referencia iniciales los de la figura 2.1.

El punto de referencia que inicialmente se encontraba en la esfera 4 quedó ubicado en el centro de la misma, los puntos de referencia de la esfera 3 se quedaron en la misma y cambiaron poco su posición, pero el de la esfera 2 quedó posicionado entre las esferas 1 y 2.

Esto se debe a que en la elección aleatoria inicial de los puntos de referencia, la esfera 1 quedó demasiado lejos de los mismos, tanto que k-means no es capaz de acercarle un punto de referencia.

Utilizamos k-medoids (ver algoritmo 2.2.2) inicializándolo con el conjunto de líderes hallado con k-means. Este algoritmo tiene como entrada el conjunto de canales líderes y el conjunto de canales no líderes. Al comienzo del mismo se inicializan dos nuevos conjuntos de canales líderes y no líderes, $\hat{L}$ y $\hat{O}$, con los conjuntos de entrada, se calcula el costo de la configuración inicial, ϕ, asociando los canales no líderes a sus líderes mas cercanos y calculando la suma de las distancias entre ellos (línea 3). Luego en el ciclo de las líneas 4 a 17, los canales líderes son intercambiados por canales no líderes (líneas 8 y 9). El costo de la configuración resultante luego de cada intercambio es calculado (línea 10) y comparado con el de la mejor configuración hasta el momento (línea 11). Si este costo es menor, se toma esta nueva configuración como la mejor. El criterio de parada se da cuando luego de una iteración los líderes, $\hat{L}$, se mantienen incambiados. Es claro ver que este algoritmo siempre termina ya que si no lo hiciera significaría que existen

infinitas configuraciones posibles, lo cual no es así ya que ambos conjuntos L y O son finitos.

Algoritmo 2.2.2: k-medoids con distancia ℓ_1

entrada: L conjunto de canales líderes.
O conjunto de canales de no líderes.
salida : $\hat{L}$ y $\hat{O}$ resultantes.

```
1  L̂ ← L
2  Ô ← O
3  φ ← Σ_{o∈O} min{d_{ℓ1}(ε(o), ε(l)) : l ∈ L}
4  repeat
5  |  L_prev = L̂
6  |  foreach l ∈ L̂ do
7  |  |  foreach o ∈ Ô do
8  |  |  |  O' ← (O\{o}) ∪ {l}
9  |  |  |  L' ← (L\{l}) ∪ {o}
10 |  |  |  φ' ← Σ_{o'∈O'} min{d_{ℓ1}(ε(o'), ε(l')) : l' ∈ L'}
11 |  |  |  if φ' < φ then
12 |  |  |  |  L̂ ← L'
13 |  |  |  |  Ô ← O'
14 |  |  |  |  φ ← φ'
15 |  |  |  end
16 |  |  end
17 |  end
18 until L_prev = L̂;
```

La distancia con norma ℓ_1 es definida como

$$d_{\ell_1}(\epsilon(a), \epsilon(b)) \triangleq \sum_{t=1}^{N} |\epsilon_t(a) - \epsilon_t(b)|. \tag{2.7}$$

2.3 Codificación de los canales

Una vez definida una configuración, se calcula la resta entre cada vector de errores de predicción (calculados en base al modelado AR(p)) y el vector de errores de predicción del líder de la respectiva agrupación. El resultado es un nuevo vector para cada canal, a, definido como

$$\hat{\epsilon}(a) = \begin{cases} \epsilon(a), & \text{si } a = b \\ \epsilon(a) - \epsilon(b), & \text{en otro caso,} \end{cases} \tag{2.8}$$

en donde b es el líder de la agrupación a la cual pertenece a.

El vector $\hat{\epsilon}(a)$ puede interpretarse como el vector de errores de predicción que se obtiene en el canal al ajustar la predicción primaria, $\bar{x}$, sumándole el error de predicción $\epsilon(b)$ cometido en el canal vecino b, que es de esperar que tenga un comportamiento similar al de a.

En los vectores $\hat{\epsilon}$, observamos empíricamente que los valores absolutos están distribuidos de tal forma que se puede ajustar muy bien una distribución geométrica. Esto es de mucha utilidad, ya que sabemos que los códigos de Golomb [19] son óptimos para este tipo de distribuciones, en

el sentido de minimizar el largo de código esperado. La codificación de Golomb de orden m de un entero j es definida como

$$G_m(j) = \text{binaria}_m(j \mod m) \oplus \text{unaria}(j \div m), \tag{2.9}$$

donde el número natural positivo m es un parámetro del código, $\oplus$ es la operación de concatenación de tiras de bits, $binaria_m(x)$ devuelve la representación binaria de x como entero sin signo de $\lceil \log_2(m) \rceil$ bits y $unaria(y)$ es la representación unaria de y, que consta de y ceros seguidos de un uno.

Una parte fundamental de la codificación con códigos de Golomb es la determinación del parámetro m, que, por razones de eficiencia de cómputo, suele tomarse como una potencia de 2. Si el parámetro que define la distribución de probabilidad que gobierna los datos es conocido, el valor óptimo de m se puede hallar analíticamente. Si este parámetro es desconocido a priori, como en nuestro caso, el valor de m se estima a partir de los datos. Esto se hace mediante cálculos con estadísticas acumuladas a lo largo de la codificación de un canal. Éstas estadísticas son la suma acumulada de los valores absolutos de los errores de predicción (A), la cantidad de muestras codificadas hasta el momento ($\bar{N}$) y la cantidad de muestras negativas codificadas hasta el momento ($\bar{N}^-$).

Dado que los errores de predicción pueden ser tanto positivos como negativos, para utilizar los códigos de Golomb (que están definidos para números naturales) utilizamos el mapeo de Rice [27], definido como

$$\text{rice}(x) = \begin{cases} 2x, & \text{si } x \geq 0 \\ -2x - 1, & \text{si } x < 0. \end{cases} \tag{2.10}$$

Este mapeo lleva los valores negativos a valores positivos impares, y los positivos a valores pares. El mapeo inverso es

$$\text{rice}^{-1}(y) = \begin{cases} y \div 2, & \text{si } y \mod 2 = 0 \\ -(y+1) \div 2, & \text{si } y \mod 2 = 1. \end{cases} \tag{2.11}$$

Siguiendo el esquema de codificación utilizado en LOCO-I [28] y también mencionado en [29], introducimos un mapeo de Rice alternativo para el caso en donde $m = 1$ y la frecuencia de aparición de valores negativos es mayor que la de no negativos,

$$\text{rice}_{\text{alt}}(x) = \text{rice}(-x - 1). \tag{2.12}$$

Observar que cuando m = 1 solo se utiliza codificación unaria para Golomb. Como puede verse en el cuadro 2.1, el mapeo alternativo cambia la prioridad de los número negativos, acortando el largo de código de los mismos en 1 con respecto al mapeo de Rice original.

El algoritmo 2.3.1 utilizado para la codificación, recibe como entrada un vector de errores de predicción $\hat{\epsilon}$. Al comienzo, en la línea 2, se inicializan las estadísticas A, $\bar{N}$ y $\bar{N}^-$ que se utilizan para calcular secuencialmente el exponente, q, del orden de Golomb potencia de 2, es decir, $m = 2^q$. Luego para cada muestra en el vector de entrada se define el mapeo a utilizar en base al valor de q y las estadísticas de la cantidad de muestras negativas y muestras totales vistas hasta el momento (líneas 4 a 8). Una vez seleccionado el mapeo, la muestra es mapeada (línea 9) y se codifica utilizando la codificación de Golomb de orden m (línea 11). Al final de la codificación

Valor anterior a Rice	0	-1	1	-2	2	-3	3	-4
Valor anterior a Rice alternativo	-1	0	-2	1	-3	2	-4	3
Valor mapeado	0	1	2	3	4	5	6	7
Largo de código para $m = 1$	1	2	3	4	5	6	7	8

CUADRO 2.1: Ejemplo del mapeo de Rice junto con el mapeo alternativo para el caso en que los errores de predicción son mayoritariamente negativos y $m = 1$.

Algoritmo 2.3.1: codificación de Golomb

entrada: vector de errores de predicción $\hat{\epsilon} = (\hat{\epsilon}_1, \hat{\epsilon}_2, ..., \hat{\epsilon}_N)$
salida : b tira de bits

```
1  b ← ∅
2  A ← 0, N̄⁻ ← 0, N̄ ← 0, q ← 0 // Inicializar estadísticas
3  foreach i ∈ {1, ..., N} do
4  |  if (q = 0) and (2N̄⁻ > N̄) then // Definir tipo de mapeo
5  |  |  x' ← −ε̂_i − 1
6  |  else
7  |  |  x' ← ε̂_i
8  |  end
9  |  r ← rice(x')
10 |  m ← 2^q
11 |  b ← b ⊕ binaria_m(r mod m) ⊕ unaria(r ÷ m)
12 |  if ε̂_i < 0 then // Actualizar estadísticas y recalcular q
13 |  |  N̄⁻ ← N̄⁻ + 1
14 |  end
15 |  N̄ ← N̄ + 1, A ← A + |ε̂_i|, q ← ⌈log_2(A/N̄)⌉
16 end
```

de cada muestra las estadísticas y el exponente del orden del código de Golomb son actualizados (líneas 12 a 15).

El algoritmo 2.3.2 de decodificación realiza el proceso inverso al del algoritmo 2.3.1 de codificación.

Algoritmo 2.3.2: decodificación de Golomb

entrada: b tira de bits
salida : vector de errores de predicción $\hat{\epsilon} = (\hat{\epsilon}_1, \hat{\epsilon}_2, ..., \hat{\epsilon}_N)$

$A \leftarrow 0$, $\bar{N}^- \leftarrow 0$, $\bar{N} \leftarrow 0$, $q \leftarrow 0$ `// Inicializar las estadísticas`
for $i \leftarrow 1, 2, ..., N$ **do**
  $m \leftarrow 2^q$
  $d_1 \leftarrow \text{binaria}_m^{-1}(b)$
  $d_2 \leftarrow \text{unaria}^{-1}(b)$
  $r \leftarrow md_2 + d_1$
  $x' \leftarrow rice^{-1}(r)$
  if $(q = 0)$ ***and*** $(2\bar{N}^- > \bar{N})$ **then** `// Definir tipo de mapeo`
    $\hat{\epsilon}_i \leftarrow -x' - 1$
  else
    $\hat{\epsilon}_i \leftarrow x'$
  end
  if $\hat{\epsilon}_i < 0$ **then** `// Actualizar estadísticas y recalcular q`
    $\bar{N}^- \leftarrow \bar{N}^- + 1$
  end
  $\bar{N} \leftarrow \bar{N} + 1$, $A \leftarrow A + |\hat{\epsilon}_i|$, $q \leftarrow \lceil \log_2(\frac{A}{\bar{N}}) \rceil$
end

El algoritmo 2.3.3 resume nuestro esquema de codificación. El algoritmo recibe como entradas datos a codificar y una configuración. Esta configuración puede determinarse como se describió en la sección 2.2, mediante una lectura previa de los datos a codificar, o puede haberse determinado a priori en una etapa de entrenamiento analizando señales adquiridas previamente.

Algoritmo 2.3.3: Codificación.

entrada: conjunto de vectores de muestras $\{x(1), x(2), \ldots, x(M)\}$,
agrupaciones $S = \{S_1, S_2, ..., S_k\}$ con el conjunto de sus
respectivos líderes $L = \{l_1, l_2, ..., l_k\}$.
salida : b tira binaria

$b \leftarrow \emptyset$
$b \leftarrow b \oplus$ cod. binaria de la configuración (L y S)
foreach $i \in \{1, \ldots, k\}$ **do**
  calcular los coeficientes para el modelo AR(p) mediante Burg para $x(l_i)$
  $\bar{x}(l_i) \leftarrow$ predicción de $x(l_i)$ utilizando modelo AR(p)
  $\epsilon(l_i) \leftarrow x(l_i) - \bar{x}(l_i)$
  $b \leftarrow b \oplus$ cod. binaria de los coeficientes del modelo AR(p)
  $b \leftarrow b \oplus$ cod. de Golomb de $\epsilon(l_i)$
  foreach $s \in S_i \backslash \{l_i\}$ **do**
    calcular los coeficientes para el modelo AR(p) mediante Burg para $x(s)$
    $\bar{x}(s) \leftarrow$ predicción de $x(s)$ utilizando modelo AR(p)
    $\epsilon(s) \leftarrow x(s) - \bar{x}(s)$
    $\hat{\epsilon}(s) \leftarrow \epsilon(s) - \epsilon(l_i)$
    $b \leftarrow b \oplus$ cod. binaria de los coeficientes del modelo AR(p)
    $b \leftarrow b \oplus$ cod. de Golomb de $\hat{\epsilon}(s)$
  end
end

El esquema de decodificación se presenta en el algoritmo 2.3.4.

Algoritmo 2.3.4: Decodificación.

entrada: b tira binaria
salida : $\{x(1), x(2), \ldots, x(M)\}$ conjunto de vectores de muestras

```
L ← decodificar los líderes de las agrupaciones desde la tira binaria b
S ← decodificar las agrupaciones de canales desde la tira binaria b
foreach i ∈ {1, ..., k} do
    decodificar los coeficientes del modelo AR(p) de la tira binaria b
    decodificar ε(l_i) mediante Golomb de la tira binaria b
    x̄(l_i) ← predicción de x(l_i) utilizando modelo AR(p) con los coeficientes
    decodificados
    x(l_i) ← x̄(l_i) + ε(l_i)
    foreach s ∈ S_i \ {l_i} do
        decodificar los coeficientes del modelo AR(p) de la tira binaria b
        decodificar ε̂(s) mediante Golomb de la tira binaria b
        ε(s) ← ε̂(s) + ε(l_i)
        x̄(s) ← predicción de x(s) utilizando modelo AR(p) con los coeficientes
        decodificados
        x(s) ← x̄(s) + ε(s)
    end
end
```

Nuestro codificador realiza dos pasadas sobre el archivo que contiene la señal, una durante el algoritmo de Burg en la cual se recolectan estadísticas y se calculan los coeficientes del modelo autorregresivo AR(p) y otra en la cual se codifica la señal. Tener la señal completa en memoria nos permite acceder al archivo en disco una sola vez. Sin embargo, esto trae dos problemas: alta latencia, dado que se precisa haber leído hasta la ultima muestra para poder empezar a codificar, y un elevado requerimiento de cantidad de memoria cuando el tamaño de la señal es grande. Para atenuar estas limitaciones, proponemos modificar los algoritmos 2.3.3 y 2.3.4 y dividir las señales de entrada en bloques de tamaño fijo preestablecido.

La principal modificación es correr el algoritmo como si cada bloque fuera una señal independiente con la diferencia de que las estadísticas de los codificadores de Golomb son globales, lo que significa que se comparten entre corridas. Notar que los coeficientes de los modelos AR(p) para cada canal son recalculados en los distintos bloques, por lo que deben ser guardados. La decodificación es similar.

Cabe mencionar un detalle no menor. A la hora de poner en práctica el algoritmo, para poder aplicar el modelado AR(p) necesitamos las primeras p muestras de los canales. En el algoritmo sin bloques estas muestras se guardan al comienzo para cada canal y son leídas en la decodificación. En este caso, para no tener que escribir estas muestras iniciales por cada bloque la mejor opción es mantener las últimas p muestras del bloque anterior (ya procesadas en el bloque anterior) y concatenarlas al principio del bloque actual. Esto agrega cierta complejidad al algoritmo pero ahorra bits de codificación.

Capítulo 3

Resultados experimentales

En este capítulo presentamos los experimentos realizados para estudiar el desempeño del codificador especificado en el capítulo 2, y los resultados de éstos. También presentamos experimentos realizados que nos permitieron ajustar parámetros del mismo (orden del modelo AR, cantidad de agrupaciones, etc).

3.1 Bases de datos

Para nuestros experimentos utilizamos como datos de entrada, EEG provenientes de tres bases de datos distintas, detalladas en el cuadro 3.1.

Además de diferir en el número de canales y la frecuencia de muestreo, como se puede apreciar en el cuadro 3.1, estas bases de datos comprenden diferentes tipos de actividades realizadas durante la captura de los datos, las cuales son bastante diversas, y son explicadas con más detalle en el apéndice B.

3.2 Orden del modelo AR

Uno de los pasos más importantes en la aplicación de un modelado AR como el de la sección 2.1 es la determinación del orden. Dado que un modelo AR de un determinado orden p es un caso particular de un modelo AR de orden mayor a p, el mínimo error de predicción cuadrático medio disminuye mientras el orden del modelo crece. Sin embargo, un incremento en el orden implica que debe describirse una mayor cantidad de coeficientes y también un mayor costo computacional. Por lo tanto, decidir cuál es el "mejor" orden del modelo no es trivial y puede impactar sobre el desempeño del sistema, tanto en la tasa de compresión como en el rendimiento computacional.

El orden del modelo AR en aplicaciones BCI es estudiado en [30], en donde el óptimo en base al criterio de información de Akaike [31] para EEG se determinó que era de aproximadamente 10 con frecuencias menores a 64Hz. Dicho criterio nos da una medida del equilibrio entre la bondad de ajuste del modelo y la complejidad del mismo. En [32], el modelo óptimo con el mismo criterio para modelar EEG encontrado fue 5 con frecuencias de 100Hz. Teniendo estos órdenes en cuenta realizamos un experimento en el cual utilizamos un archivo de cada base y calculamos el *largo de código medio* (LCM) por muestra para cada uno de los canales de dichos archivos variando el

Base de datos	Frecuencia (Hz)	Canales	Pacientes	Archivos por paciente	Duración (s)
Neuroscan	1000	31	14	25	206
PhysioNet	160	64	100	14	120
BCI Competition	1000	118	4	1	3000

CUADRO 3.1: Características de las bases de datos utilizadas

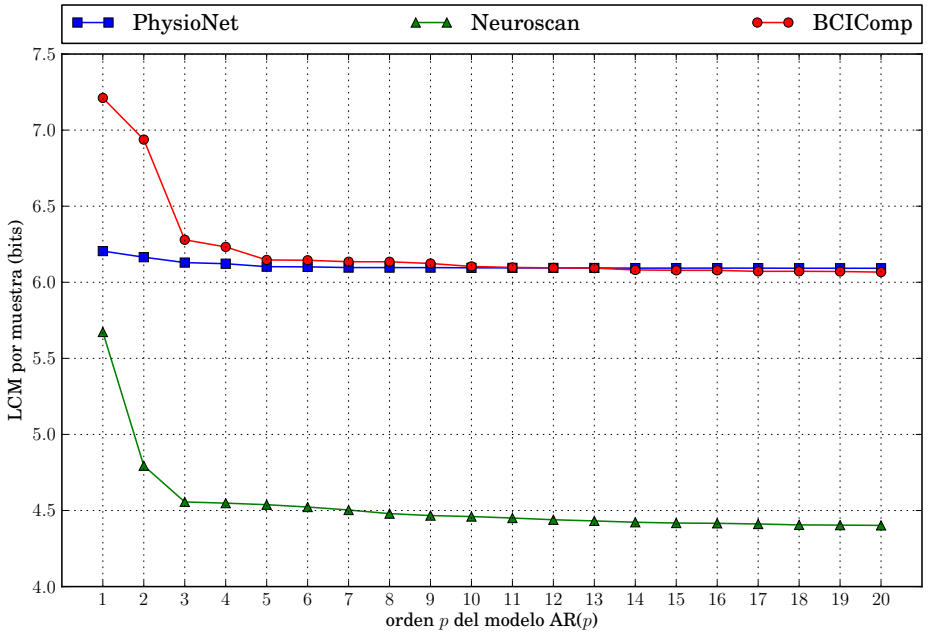


FIGURA 3.1: LCM por muestra para un archivo de cada base de datos en función del orden del modelo AR. A partir del orden 3, el LCM por muestra no disminuye significativamente.

orden del modelo AR. En la figura 3.1 mostramos el promedio de los resultados para cada base de datos.

Para estimar el LCM por muestra asociado al vector de errores de predicción con correlación espacial de un canal c, calculamos la entropía diferencial empírica de una distribución de Laplace [1] como

$$\hat{h}(c) = \log_2(2\theta) + \log_2(e), \tag{3.1}$$

donde θ se define como

$$\theta = \frac{\sum_{t=1}^{N} |\hat{\epsilon_t}(c) - \mu|}{N}, \tag{3.2}$$

siendo μ la mediana del vector de error de predicción con correlación espacial para el canal c y N es la cantidad de muestras del canal.

[1] La distribución de Laplace, o doble exponencial, puede pensarse como una versión continua de la distribución geométrica a dos lados que usamos para modelar los errores de predicción, en el sentido de que la parte entera de una variable exponencial tiene distribución geométrica.

En la figura 3.1 puede observarse como para órdenes pequeños (1 a 3) la ganancia en LCM por muestra es considerable al aumentar el orden del modelo, mientras que para órdenes mayores a 3 esta ganancia es despreciable. Por esta razón, de aquí en más a los errores de predicción del modelo AR los calcularemos en base a un modelo de orden 3.

3.3 Número de agrupaciones

Luego de haber determinado un orden del modelo AR que permita hacer un buen aprovechamiento de la correlación temporal intra-canal, la pregunta fundamental a respondernos es:

> ¿Qué tanto se pueden aprovechar las correlaciones espaciales entre canales utilizando el método de agrupación definido en la sección 2.2?

Para ver esto corrimos el algoritmo de agrupación para todos los archivos de un individuo en particular en cada base datos, (con el ajuste detallado en la sección 3.4) utilizando una cantidad variable de agrupaciones, de 1 a M.

Con esto obtuvimos, para cada archivo de prueba, M configuraciones distintas para las cuales calculamos el LCM por muestra resultante de codificar los vectores de errores de predicción con dichas configuraciones, definidos en (2.8).

Los resultados del experimento pueden verse en las gráficas de la figura 3.2, en donde para cada base de datos graficamos los LCM por muestra obtenidos para cada archivo así como también el promedio de estos. En estas podemos observar que las curvas que describen los distintos archivos son similares, pero poseen distintos desplazamientos en el eje de las ordenadas. Esto quiere decir que, para las distintas actividades realizadas por el individuo, el comportamiento de las señales al variar el número de agrupaciones es similar.

Cabe remarcar que los valores al final de cada serie, es decir los largos de código medio por muestra resultantes de utilizar M agrupaciones, son equivalentes a no utilizar agrupaciones y codificar los errores de predicción de canales únicamente mediante modelado AR. A partir de la figura 3.2, podemos apreciar que realizar agrupaciones de canales siempre produce un LCM por muestra menor a no realizar ninguna. Lo anterior se puede notar más fácilmente en la gráficas del promedio de bits por muestra para cada base de datos. Este es el primer indicio de que podemos aprovechar las correlaciones espaciales entre canales aplicando este método.

Además de la diferencia entre los valores de LCM por muestra para $k = 1$ y para $k = M$, en las gráficas podemos observar que, en todos los casos, aumentar la cantidad de agrupaciones hace disminuir los LCM hasta llegar a un valor mínimo, para luego aumentar de manera monótona.

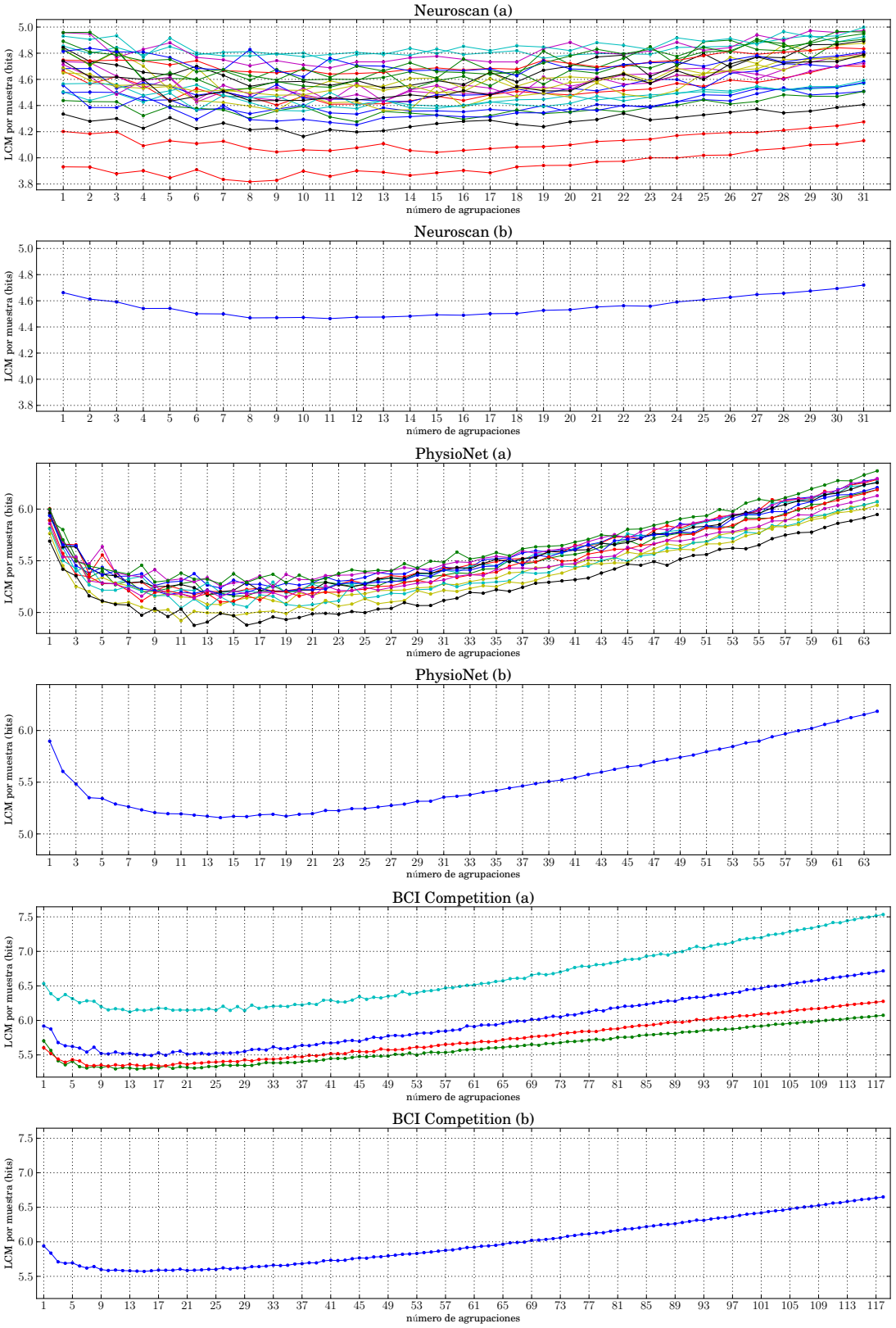


FIGURA 3.2: (a) Bits por muestra de cada archivo de la base de datos en función del número de agrupaciones. (b) Bits por muestra promedio en función del número de agrupaciones para la base de datos.

Base de datos	LCM por muestra sin agrupaciones	LCM por muestra mejor agrupación	% Mejora	Número de agrupaciones
Neuroscan	4.72	4.46	5.7%	11
PhysioNet	6.19	5.16	19.9%	14
BCI Competition	6.65	5.57	19.3%	15

Cuadro 3.2: LCM por muestra y cantidad de agrupaciones que lo minimiza para cada base de datos.

Podemos ver algunos resultados de forma numérica en el cuadro 3.2, donde se muestra el LCM por muestra obtenido sin agrupación de canales y el obtenido con la agrupación que minimiza este LCM. El cuadro también muestra con qué cantidad de agrupaciones se alcanza el mínimo y cuál es el porcentaje de mejora de una configuración con respecto a la otra.

Para las tres bases de datos, las cuales corresponden a experimentos donde distintos individuos realizaron actividades distintas, durante distintos períodos de tiempo, y donde los datos fueron adquiridos a distintas frecuencias de muestreo, el número de agrupaciones de los canales que permite llegar a un valor mínimo de bits por muestra está en un entorno de 12 ± 2. Este número parece no depender de la cantidad de canales que están siendo agrupados, la cual varía significativamente en las distintas bases de datos utilizadas (31 en Neuroscan, 64 en PhysioNet y 118 en BCI Competition).

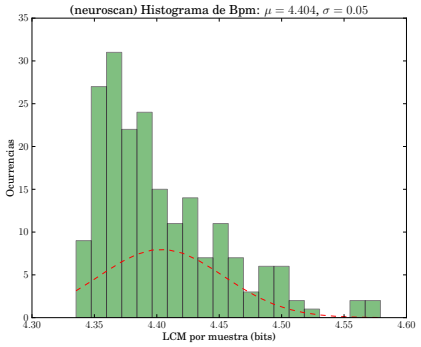
Si observamos en la figura 3.2 las gráficas correspondientes a los bits por muestra en promedio por cada base de datos, la diferencia entre los alcanzados con $k = 12$ y el mínimo, en cada caso, es de centésimas de bit por muestra. Por lo tanto, en la práctica, utilizar conjuntos de 12 agrupaciones de canales nos permite lograr un muy buen aprovechamiento de las correlaciones espaciales entre los canales simultáneamente en las tres bases de datos con las que experimentamos. En otras condiciones, sin embargo, los resultados podrían ser diferentes.

3.4 Ajuste del algoritmo de agrupamiento

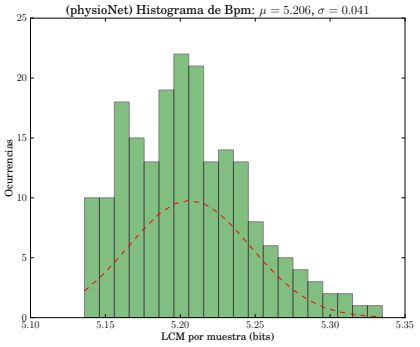
Para evaluar la estabilidad del algoritmo de agrupamiento propuesto, lo corrimos 200 veces para un archivo de cada base de datos y calculamos los LCM por muestra resultantes de codificarlo con cada una de las 200 configuraciones que obtuvimos en cada corrida.

Como podemos ver en la figura 3.3, en los tres casos la desviación estándar empírica es menor o igual a 0.05 bits por muestra, es decir que en promedio una configuración va a dar como resultado una codificación cuyo LCM por muestra va a estar a menos de una décima de la media, lo cual nos dice que el algoritmo es relativamente estable.

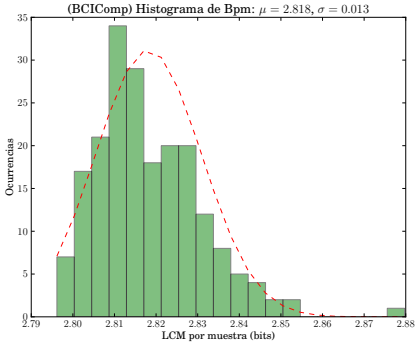
En los histogramas de la figura 3.3 también podemos observar que aproximadamente en 40 de las 200 corridas (la quinta parte), se alcanzaron los valores más pequeños. Luego, como nuestro objetivo es minimizar el LCM, y como el algoritmo de agrupamiento puede ser corrido en una primera instancia de entrenamiento analizando señales adquiridas previamente, sin formar parte del codificador, podemos permitirnos ejecutar nuestro algoritmo de agrupamiento 5 veces en vez de una sola en nuestros experimentos, y elegir la configuración que resulte en mínimo LCM. Para el tipo de señales evaluadas en nuestros experimentos sabemos que, con alta probabilidad dicha configuración va dar como resultado una codificación cuyos bits por muestra se encuentran a centésimas de la mejor configuración alcanzable con nuestro algoritmo.



(a) Neuroscan



(b) PhysioNet



(c) BCI Competition

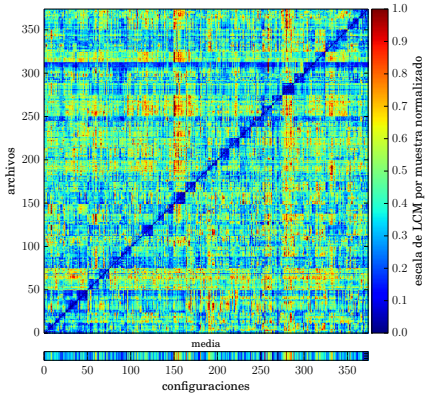
FIGURA 3.3: Histogramas de los largos de código medio por muestra resultantes de aplicar el algoritmo de agrupamiento 200 veces para un archivo de cada base de datos. Las líneas rojas muestran la forma de la densidad de probabilidad de una variable aleatoria con distribución normal $N(\mu, \sigma^2)$.

3.5 Desempeño de las agrupaciones

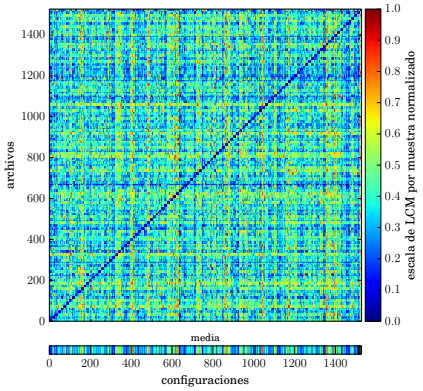
Sabiendo que al utilizar 12 agrupaciones se obtienen buenos resultados, nuestro siguiente objetivo es observar cómo se comportan las agrupaciones obtenidas para los demás archivos de la base de datos, con el fin de obtener una configuración fija para cada base de datos.

Para ver lo anterior realizamos un experimento en donde, para cada archivo de cada base de datos, calculamos una configuración para cada uno de ellos con nuestro algoritmo de agrupamiento, y aplicamos la misma a todos los archivos de la correspondiente base de datos. El resultado, para cada base de datos, es una matriz de F x F valores de LCM, donde F es la cantidad de archivos de cada base. En la figura 3.4 pueden verse estos resultados donde cada matriz se muestra como una imagen con una escala de colores asociada al rango de valores que toman sus entradas. Los LCM por muestra fueron normalizados para cada fila de la matriz, es decir que cada LCM por muestra se dividió entre la suma de los LCM por muestra de la fila correspondiente de la matriz, y los píxeles azules se corresponden con los bits por muestra más bajos obtenidos, mientras que los rojos se corresponden con los más altos.

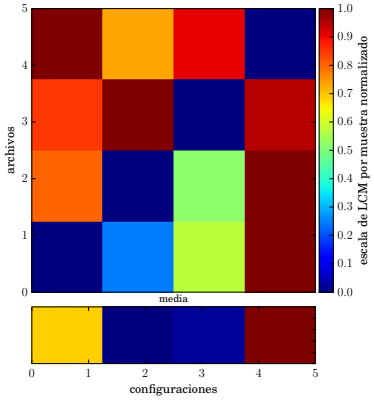
En la diagonal de las gráficas de la figura 3.4 se encuentran los resultados de utilizar para cada archivo la configuración que minimiza el LCM para ese mismo archivo, lo cual explica el color azul que la distingue claramente del resto de la imagen. Esto es aún más notorio en la figura



(a) Neuroscan



(b) PhysioNet



(c) BCI Competition

FIGURA 3.4: Matriz de LCM por muestra resultante de aplicar todas las configuraciones a todos los archivos de cada base de datos. Los valores se encuentran normalizados por fila. Las diagonales azules indican que, para cada archivo, la configuración que produce la menor cantidad de bits por muestra, en general es la obtenida con el algoritmo de agrupamiento para el propio archivo. El rectángulo inferior a cada matriz indica el desempeño en media de cada configuración.

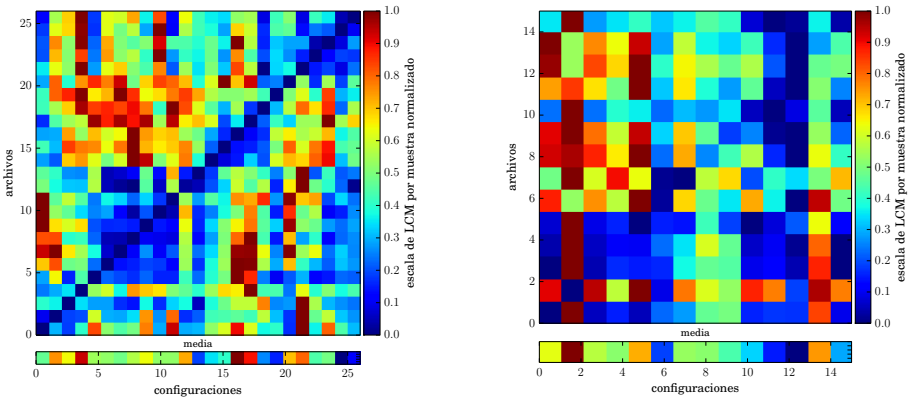
3.4 donde se grafica el LCM por muestra para los archivos de un solo individuo de cada base de datos. Como se ve en la escala, los LCM por muestra fueron normalizados con

$$\text{norm}(f) = \left\{ \frac{(x - \min(f))}{\max(f) - \min(f)} : x \in f \right\}, \tag{3.3}$$

donde f es un vector fila, $\min(f)$ y $\max(f)$ representan el valor mínimo y máximo de la fila f.

Para tener una idea de la magnitud de las diferencias de la figura 3.4, en el cuadro 3.3 se muestra el mayor y el menor LCM por muestra entre todas las combinaciones de configuraciones y archivos, es decir, los valores extremos en las matrices mostradas en las figuras. En el cuadro 3.4 se ve el mínimo LCM por muestra promedio, es decir el resultante de utilizar la mejor configuración, y el máximo (resultante de utilizar la peor configuración) para cada base de datos.

Al ver las diferencias entre la mejor y peor configuración global en el cuadro 3.4, y las desviaciones estándar de aplicar el algoritmo de agrupamiento vistas en la figura 3.3, notamos que las



(a) Individuo CBA de Neuroscan

(b) Individuo S010 de PhysioNet

FIGURA 3.5: Las matrices indican la misma información que en la figura 3.4 pero sobre subconjuntos de archivos correspondientes a un individuo de Neuroscan, y un individuo de PhysioNet.

Base de datos	Mejor LCM por muestra obtenido para un archivo (bits)	Peor LCM por muestra obtenido para un archivo (bits)
Neuroscan	3.62	5.04
PhysioNet	2.70	7.90
BCI Competition	5.29	6.30

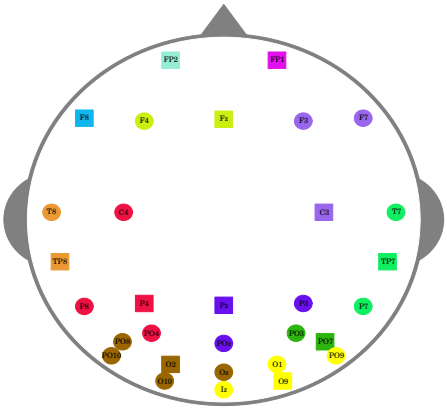
CUADRO 3.3: LCM por muestra para los mejores y peores pares de configuraciones y archivo de cada base de datos

Base de datos	Mejor configuración (bits)	Peor configuración (bits)
Neuroscan	4.58	4.82
PhysioNet	5.23	5.51
BCI Competition	5.64	5.67

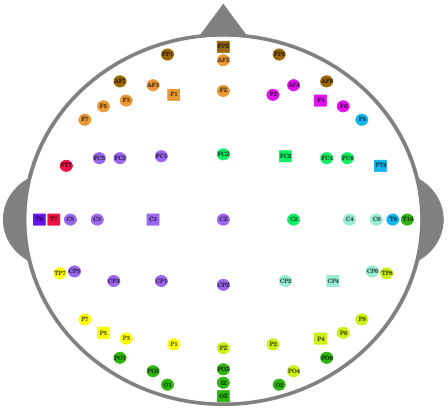
CUADRO 3.4: LCM por muestra para las mejores y peores configuraciones aplicadas a todos los archivos de cada base de datos en media

variaciones de nuestro algoritmo de agrupamiento son de una magnitud 5 veces más pequeña que las variaciones entre las mejores y peores configuraciones (ver cuadro 3.4). Esta observación no es cierta para la base de datos de BCI Competition, en donde la diferencia entre la desviación estándar de nuestro algoritmo de agrupamiento es de una magnitud poco menor a 2 veces más pequeña que la variación entre las 2 configuraciones; esto puede atribuirse al conjunto reducido de archivos para los cuales realizamos el experimento de la sección 3.4. Estas magnitudes, si bien son pequeñas, no son despreciables, y justifican el ajuste realizado en 3.4.

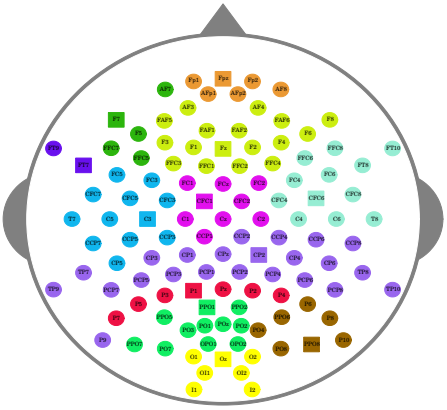
En la figura 3.6 se puede observar la disposición de los electrodos así como la mejor configuración obtenida para cada base de datos sacadas del cuadro 3.4. En los dibujos cada color representa una agrupación (pueden observarse 12 colores diferentes) y los nodos marcados con un cuadrado son los líderes de cada agrupación. En estas figuras puede apreciarse que las agrupaciones tienen una fuerte componente espacial, y que están agrupadas en zonas especificas del cerebro.



(a) Mejor configuración para Neuroscan



(b) Mejor configuración para PhysioNet

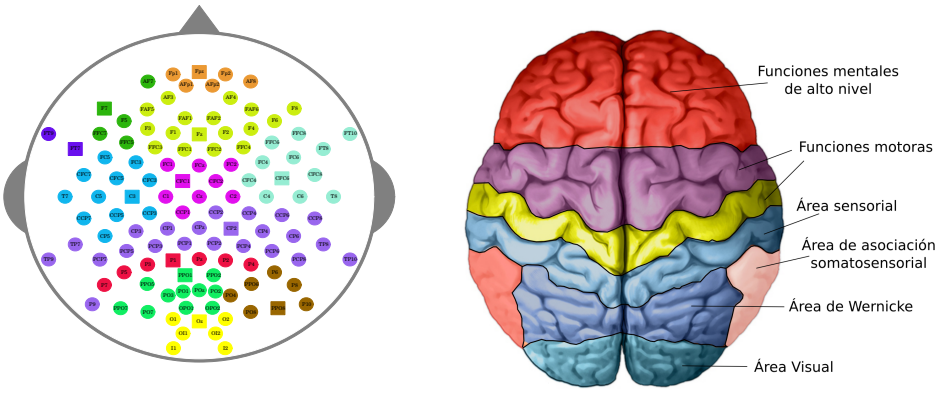


(c) Mejor configuración para BCI Competition

FIGURA 3.6: Mejores configuraciones obtenidas para cada base de datos. Cada color representa una agrupación, y el nodo marcado con un cuadrado es el líder de la misma.

En la imagen 3.7 se puede ver una comparación entre las diferentes áreas funcionales del cerebro y la mejor configuración obtenida para BCI Competition. Esta comparación nos deja la impresión de que existe una relación entre las agrupaciones de electrodos que utilizamos para explotar las correlaciones espaciales y las diferentes áreas funcionales del cerebro.

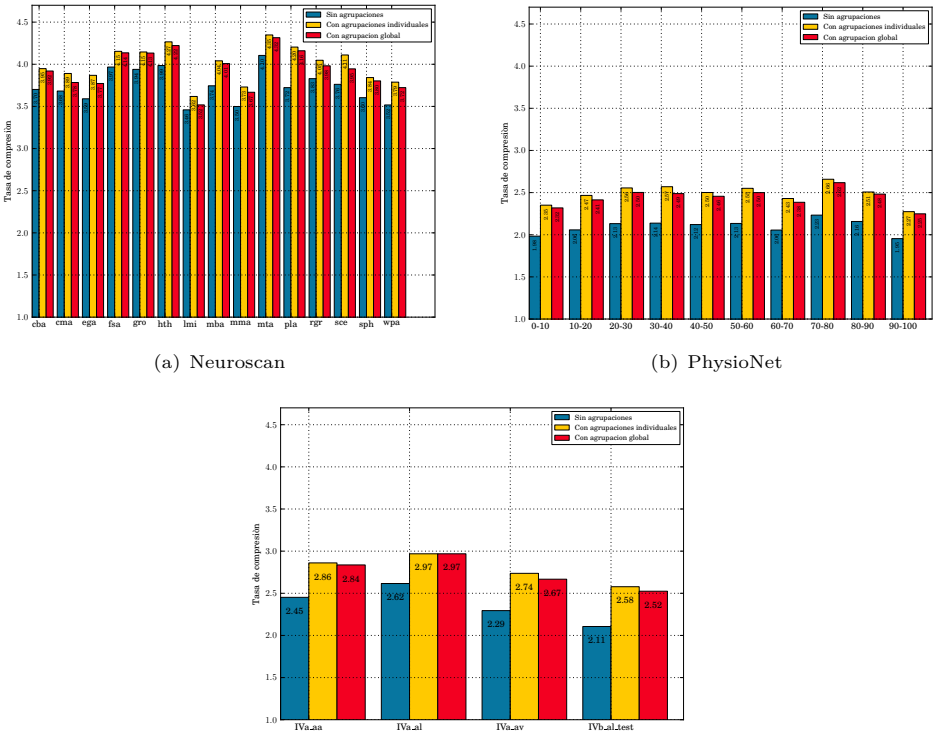
Con este experimento concluimos que es factible predefinir una única configuración para cada base de datos y utilizarla en lugar de calcularla cada vez, mediante el algoritmo de agrupamiento para cada archivo a comprimir. Esto repercute en una mayor velocidad de compresión y evita una recorrida extra del archivo, abriendo la posibilidad de comprimir secuencialmente. Hay que tener en cuenta, sin embargo, que es necesario utilizar un conjunto de archivos de entrenamiento previo para hallar una configuración adecuada.



(a) Mejor configuración para BCI Competition

(b) Anatomía y áreas funcionales del cerebro

FIGURA 3.7: Comparación entre la localización de las agrupaciones para una configuración y las áreas funcionales del cerebro.



(a) Neuroscan

(b) PhysioNet

(c) BCI Competition

FIGURA 3.8: Media de la tasa de compresión obtenida para cada individuo de cada base de datos utilizando cada una de las diferentes técnicas.

3.6 Desempeño del compresor

Para medir el nivel de compresión sobre las bases de datos utilizamos una implementación en C++ del algoritmo definido el el capitulo 2, utilizando el tamaño en bits de los archivos codificados (haciendo uso de la codificación de Golomb definida en el algoritmo 2.3.1 del Capitulo 2) en lugar de la aproximación utilizando la entropía diferencial empírica de una distribución de Laplace. Cabe mencionar que se utilizo la técnica de bloques de 100.000 muestras (estudiamos este parámetro con mas detalle en el experimento 3.9) para realizar la compresión en la practica, y en el tamaño de los archivos se toma en cuenta el cabezal definido en el apéndice C.1.

Para cada base de datos comprimimos todos los archivos de ellas de tres maneras diferentes:

1. Codificar cada canal independientemente, lo cual equivale a utilizar una configuración con M agrupaciones.
2. Utilizar una configuración calculada para cada archivo individualmente.
3. Utilizar una configuración global obtenida anteriormente para cada base de datos.

En la figura 3.8 puede observarse la tasa de compresión media obtenida para cada individuo de cada base de datos utilizando cada una de las tres técnicas. Podemos notar que comprimir utilizando las configuraciones por archivo es en todos los casos mejor, pero sin embargo utilizar las “mejores” configuraciones (obtenidas en el experimento 3.5) da una tasa de compresión media ligeramente inferior lo cual nos da la pauta de que es posible encontrar una configuración que funciona simultaneamente bien para todos los archivos de una base.

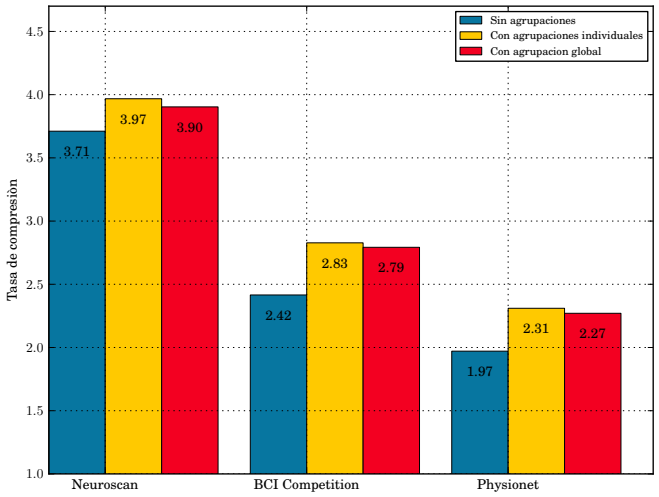


FIGURA 3.9: Media de la tasa de compresión obtenida para cada base de datos utilizando cada una de las diferentes técnicas.

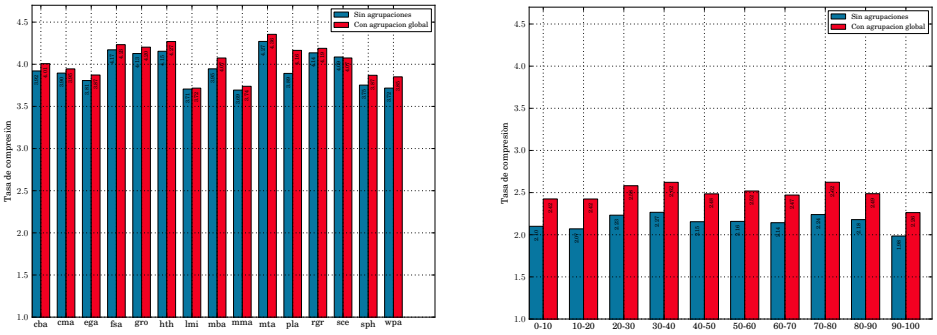
Base de datos	Canales	Ganancia de configuración por archivo sobre canales independientes (%)	Ganancia configuración fija sobre canales independientes (%)
Neuroscan	31	6.55	4.87
PhysioNet	64	14.72	13.22
BCI Competition	118	16.94	15.29

CUADRO 3.5: Ganancias de relación de compresión obtenidas para las diferentes técnicas y para cada base de datos en relación a codificar los canales independientemente.

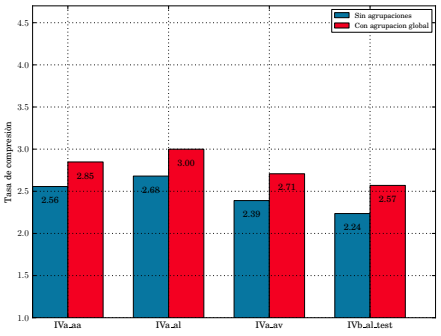
En la figura 3.9 tenemos la tasa de compresión media obtenida para cada base de datos y cada una de las técnicas. Aquí podemos ver de manera más global la ganancia obtenida de aplicar configuraciones sobre realizar la compresión de cada canal independientemente, así como la poca diferencia entre usar una configuración fija o una específica para cada archivo.

En el cuadro 3.5 podemos apreciar los porcentajes de ganancia, es decir cuan superior es una tasa de compresión media que otra, en relación a la compresión obtenida con las técnicas de agrupación sobre codificar los canales independientemente. Parece haber una relación entre la cantidad de canales utilizados y la ganancia en compresión, mientras más canales mejor es la ganancia obtenida con esta técnica.

Una interrogante a la hora de comparar las ganancias de utilizar configuraciones es saber si relamente se están aprovechando las correlaciones que no pueden ser captadas codificando los canales independientemente, o si solo se están aprovechando correlaciones que podrían ser captadas por un modelo AR(p) de mayor orden. Para ver esto utilizamos un predictor temporal con un modelo AR(8) ya que como se ve en la figura 3.1 desde este orden en adelante la mejora en relación al LCM es mínima. En las figuras 3.10 y 3.11 pueden apreciarse los resultados de las corridas del compresor sin configuraciones y con la mejor configuración global hallada en la sección 3.5. Si bien lo correcto hubiera sido volver a calcular esas configuraciones con los nuevos canales codificados con el modelo AR(8), los resultados muestran que estas configuraciones siguen funcionando bien.



(a) Neuroscan (b) PhysioNet



(c) BCI Competition

FIGURA 3.10: Media de tasa de compresión obtenida para cada individuo de cada base de datos utilizando cada una de las diferentes técnicas.

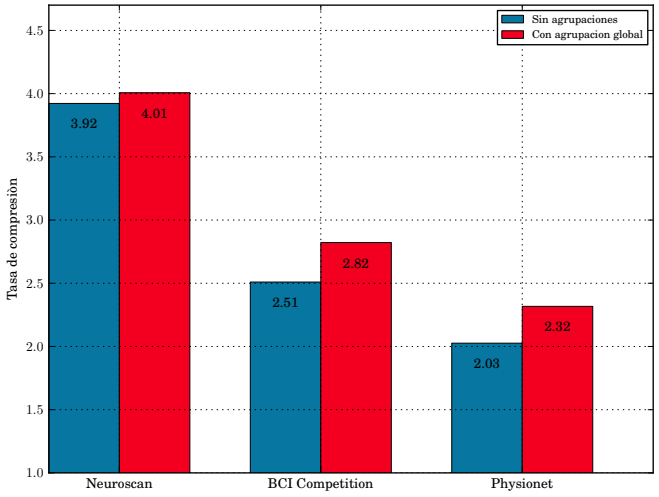


FIGURA 3.11: Media de tasa de compresión obtenida para cada base de datos utilizando cada una de las diferentes técnicas.

En el cuadro 3.6 podemos ver que el porcentaje de ganancia de utilizar las configuraciones fijas sobre AR(8) y AR(3) son bastante parecidos, salvo en el caso de Neuroscan. También confirmamos los resultados de la sección 3.2, es decir que (sin utilizar las configuraciones espaciales) la ganancia de pasar de un modelo AR(3) a un modelo AR(8) es baja.

Base de datos	Ganancia AR(8) sobre AR(3)	Ganancia de utilizar el predictor espacial sobre el temporal de AR(3)	Ganancia de utilizar el predictor espacial sobre el temporal de AR(8)
Neuroscan	5.36 %	4.87 %	2.30 %
PhysioNet	3.59 %	13.22 %	12.35 %
BCI Competition	2.96 %	15.29 %	14.29 %

CUADRO 3.6: Ganancias de relación de compresión obtenidas para las diferentes técnicas y para cada base de datos en relación a solo codificar realizando los errores de predicción de canales mediante modelado AR

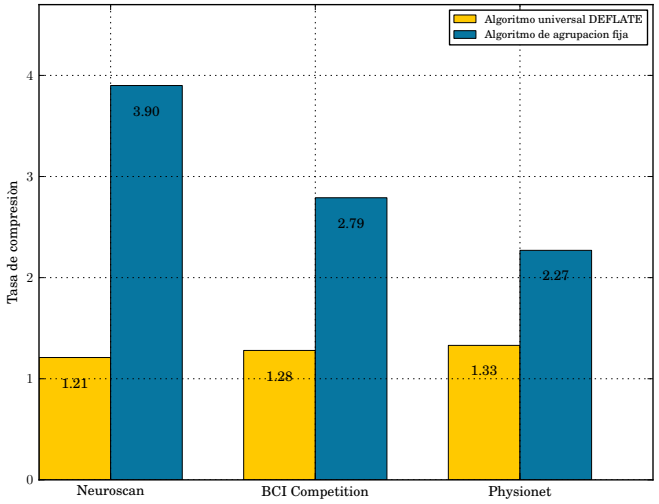


FIGURA 3.12: Tasa de compresión media obtenida para cada base de datos utilizando el algoritmo universal DEFLATE y nuestro algoritmo de agrupamiento con agrupaciones fijas por base de datos.

3.7 Comparación con un algoritmo de compresión sin pérdida universal

Decidimos comparar el desempeño de nuestro compresor sin pérdida de EEG con un algoritmo universal, es decir un algoritmo de compresión de uso general, que no está específicamente diseñado para EEG. Para ello, generamos archivos binarios crudos a partir de las bases de datos, de modo de eliminar cabezales innecesarios, y poder comprimir solo a partir de la misma información que consideramos a la hora de utilizar nuestro compresor. Esta es la cantidad de canales M y el número total de muestras por canal N de cada archivo, los cuales almacenamos como enteros (32 bits), y por supuesto las muestras, las cuales fueron almacenadas con la resolución que tenían en los archivos originales. Para comprimir los archivos utilizamos la herramienta gzip, que está basada el algoritmo de compresión universal LZ77 [33].

Para nuestras pruebas utilizamos el parámetro de compresión máxima que ofrece la herramienta (gzip -9). Los resultados pueden verse en el cuadro 3.7.

Comparando con el desempeño de nuestro compresor, para las bases de datos utilizadas, nuestro compresor es superior. Los valores mínimos y máximos no están muy lejos del promedio en cada caso, y las desviaciones estándar son pequeñas.

Base de datos	CR promedio	Desv. estándar	Mínimo	Máximo
Neuroscan	1.21	0.03	1.13	1.28
BCI Competition	1.28	0.01	1.27	1.30
PhysioNet	1.33	0.16	1.09	1.84

CUADRO 3.7: Promedio, desviación estándar, mínimo y máximo de tasa de compresión obtenidos con el algoritmo universal DEFLATE para cada base de datos.

Base de datos	Tasa compresión Dauwels	Tasa de compresión obtenida
PhysioNet*	2.14	2.22

CUADRO 3.8: Comparación de la tasa de compresión media obtenida con nuestro algoritmo contra la obtenida en [20]. Ambos algoritmos utilizaron el mismo juego de datos como entrada.

3.8 Comparación con el estado del arte

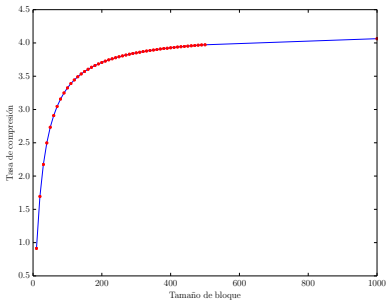
Decidimos comparar nuestro algoritmo con los resultados presentados recientemente en [20], para los mismos juegos de datos (un subconjunto de los archivos de la base PhysioNet). Las tasas de compresión se pueden ver en el cuadro 3.8.

Los resultados obtenidos son buenos, ya que muestran que la tasa media de compresión de nuestro algoritmo es superior.

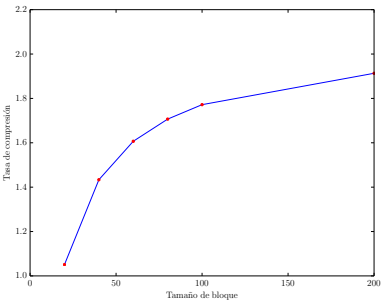
3.9 Tamaño de los bloques de muestras

Un parámetro de nuestro codificador es el tamaño de bloque. Considerando la gran cantidad de muestras de los archivos de la base de datos BCI Competition [34] decidimos darle a este parámetro el valor fijo igual a 100.000, es decir que los algoritmos de codificación 2.3.3 y decodificación 2.3.4 tienen como entrada y salida vectores de muestras de este tamaño, salvo el último bloque cuyo tamaño es menor o igual que los demás. En este experimento estudiamos el efecto que tiene la variación de este parámetro en la tasa de compresión para un archivo de cada base de datos.

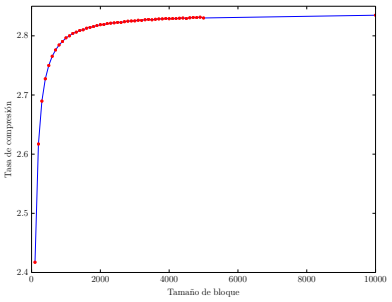
En la figura 3.13 podemos ver las tasas de compresión obtenidas para bloques muy pequeños, mientras que en la figura 3.14 podemos ver el resultado global de variar los bloques hasta su valor máximo (N). Observamos que cuando los bloques son pequeños la tasa de compresión es baja, y a medida que se aumenta el tamaño de los mismos (menos bloques pero de mayor tamaño), la tasa va subiendo hasta cierto punto. Es razonable que para tamaños de bloque bajos las tasas de compresión sean bajas, ya que un tamaño de bloque bajo implica que se utilice una gran cantidad de bloques, teniendo que describir para cada uno de ellos un nuevo conjunto de coeficientes del modelo AR(p). Esto causa que el archivo comprimido sea grande y por lo tanto, que la tasa de compresión sea baja. Por otro lado cuando los bloques son muy grandes (pocos bloques) las tasas de compresión tienen mayor variabilidad.



(a) Neuroscan

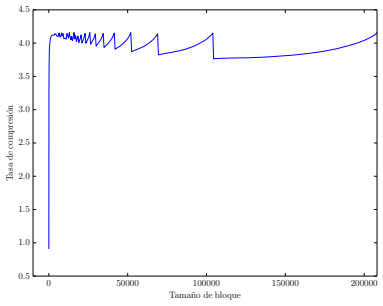


(b) PhysioNet

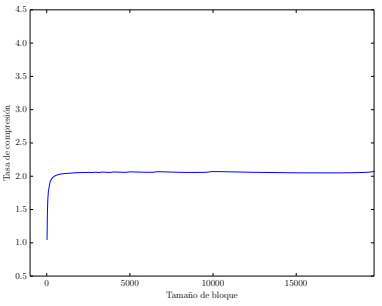


(c) BCI Competition

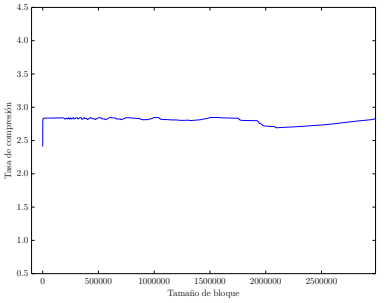
FIGURA 3.13: Tasa de compresión en función de tamaños de bloques pequeños para un archivo de cada base de datos.



(a) Neuroscan



(b) PhysioNet



(c) BCI Competition

FIGURA 3.14: Tasa de compresión en función del tamaño de bloque para un archivo de cada base de datos. Se puede ver una mayor variabilidad en las tasas a medida que aumenta el tamaño de bloque.

Capítulo 4

Conclusiones y trabajo futuro

En este capítulo hacemos un repaso de los resultados obtenidos e identificamos también posible trabajo futuro en base al conocimiento adquirido.

4.1 Conclusiones

El algoritmo propuesto cumple de manera satisfactoria la compresión sin pérdida de electroencefalogramas multicanal explotando las correlaciones temporales así como las espaciales. Este algoritmo es superior a métodos de compresión universal como DEFLATE, y también obtiene mejores tasas de compresión que otros algoritmos de estado del arte como los presentados en [20].

Utilizar bloques en la implementación del compresor fue una buena decisión de diseño para no tener que esperar a tener todo un canal almacenado en memoria para correr el algoritmo de estimación del modelo autoregresivo offline (Burg en nuestro caso) y así poder utilizar menos recursos. Adicionalmente, desde el punto de vista de modelado, la fragmentación de la señal en bloques brinda cierta capacidad de adaptación del algoritmo a potenciales cambios en el comportamiento estadístico de la señal.

En efecto, los métodos no adaptivos requieren que la señal sea estacionaria, es decir que sus características estadísticas no varíen con el tiempo. Este no es usualmente el caso de los EEG, que, por ejemplo, tienen un espectro de frecuencia variable en el tiempo [35]. Para este tipo de señales en general es beneficioso utilizar modelos adaptivos, por lo que una posible variante sería probar con modelos que puedan adaptarse secuencialmente, muestra a muestra, en lugar de en bloques.

Con respecto a nuestro método de agrupación de canales para aprovechar correlaciones espaciales, para todas las bases de datos, utilizar al menos una agrupación disminuyó el LCM. Esto deja en claro que el método de agrupación espacial funciona y es un componente importante para agregar a los errores de predicción de canales mediante modelado AR(p) dada su baja complejidad computacional.

Si bien parecía lógico que el número de agrupaciones que minimizara el LCM resultante fuera dependiente de la cantidad de canales del archivo, para las bases de datos estudiadas el número de agrupaciones que lo hizo fue independiente de la misma, resultando en media en un valor en un entorno de 12 ± 2 y con una varianza pequeña. Esto podría deberse a componentes biológicas del cerebro y a los tipos de experimentos realizados. Esta observación nos sirvió para poder fijar

esta cantidad de agrupaciones a la hora de correr el algoritmo de agrupamiento para las bases de datos.

Existe una agrupación global con buen rendimiento para poder codificar todos los archivos pertenecientes a una base de datos. Al ver las disposiciones de los canales en las agrupaciones parece clara una fuerte influencia de la ubicación espacial de los canales en las configuraciones, dado que en general los canales cercanos espacialmente se encuentran en la misma agrupación. Esto es algo que vale la pena seguir investigando en un futuro, y una posible estrategia sería basar el algoritmo de agrupamiento solamente en el posicionamiento espacial de los canales. Una comparación entre la localización espacial de las agrupaciones y las áreas funcionales del cerebro puede verse en la figura 3.7.

4.2 Posible trabajo futuro

Una buena alternativa a estudiar en el futuro es usar algoritmos RLS [36] (Recursive Least Squares) para hacer el cálculo de los parámetros. Dichos algoritmos son secuenciales, y nos dan la habilidad de calcular los parámetros muestra a muestra (de forma adaptiva). Los algoritmos presentados en [36] tienen la propiedad de ser recursivos en el orden, lo que significa que empiezan calculando el modelo de orden 1, luego el modelo de orden 2 se calcula en base al modelo de orden 1 y así sucesivamente hasta un orden p arbitrario. Esto último nos da la opción de poder ajustar los diferentes modelos en base al rendimiento que van teniendo a lo largo de la codificación, tal como se propone en [37]. La clave de nuestro algoritmo de agrupamiento es la resta del vector de errores de predicción del canal líder de una agrupación a los vectores de errores de predicción de los demás canales de la agrupación correspondiente, pero no deja de ser la resta entre dos vectores, por lo tanto lo importante es la asociación entre pares. Esto puede generalizarse y estudiar qué sucede a la hora de asignar a cada canal un "líder", en otras palabras podríamos encontrar un árbol donde cada nodo está asociado a un canal, la raíz es el único canal al cual no se le realiza una resta y los demás canales son restados contra su nodo padre comenzando desde las hojas. Nuestra codificación no es otra cosa que un caso particular de lo anterior en donde tenemos un árbol de profundidad 1 por cada agrupación de canales, es decir que los nodos son hojas o raíz. Dicho árbol podría obtenerse usando un algoritmo para hallar árboles de cubrimiento mínimo con la distancia física entre los electrodos que adquirieron las señales de cada canal como función de costos. Una cuestión no menor seria hallar el nodo raíz adecuado. Esta idea es puesta en práctica en otro proyecto de mayor alcance dentro del cual toma parte este trabajo [38]. En nuestra codificación actual, podemos expresar el error de predicción de la muestra x_t de un canal c_1 no líder como el error de predicción en el tiempo t basado únicamente en muestras anteriores del propio canal, menos el error de predicción de un canal c_2 líder de una agrupación (este también basado únicamente en muestras suyas), lo cual podemos expresar como

$$\epsilon_t(c_1) = (x_t(c_1) - \sum_{i=1}^{p} a_i x_{t-i}(c_1)) - (x_t(c_2) - \sum_{i=1}^{p} b_i x_{t-i}(c_2)), \tag{4.1}$$

esto podemos reordenarlo y expresarlo de la forma

$$\epsilon_t(c_1) = x_t(c_1) - (\sum_{i=1}^{p} a_i x_{t-i}(c_1) - x_t(c_2) + \sum_{i=1}^{p} b_i x_{t-i}(c_2)). \tag{4.2}$$

Es claro que para predecir la muestra $x_t(c_1)$ estamos utilizando las muestras

$$x_{t-1}(c_1), ..., x_{t-p}(c_1), x_t(c_2), x_{t-1}(c_2), ..., x_{t-p}(c_2),$$

pero estamos multiplicando $x_{t-1}(c_1), ..., x_{t-p}(c_1)$ por coeficientes que minimizan el error cuadrático medio únicamente para c_1, $x_t(c_2)$ tiene el coeficiente constante 1 y estamos multiplicando $x_{t-1}(c_2), ..., x_{t-p}(c_2)$ por coeficientes que minimizan el error cuadrático medio únicamente para c_2. Por lo tanto, podríamos generalizar esto y calcular $\epsilon_t(c_1)$ de la forma

$$\epsilon_t(c_1) = x_t(c_1) - (\sum_{i=1}^{p_1} u_i x_{t-i}(c_1) + \sum_{i=p_1+1}^{p_2} u_i x_{t-(p_1+1)-i}(c_2)). \tag{4.3}$$

Así, para predecir la muestra $x_t(c_2)$ estamos utilizando las p_1 muestras anteriores de c_1, la muestra presente del canal c_2 y las $p_2 - 1$ muestras anteriores de c_2 pero con los coeficientes que minimicen el error cuadrático medio.

Lo difícil ahora sería encontrar el canal c_2 que "funcione mejor" con c_1. Para esto podemos tomar ventaja del hecho de que sabemos que hay una fuerte correlación espacial entre canales cercanos (figura 3.6) y tomar por ejemplo a c_2 como el canal más cercano espacialmente a c_1. Al igual que en la generalización anterior, podríamos calcular un árbol de cubrimiento mínimo de los canales, basándonos en las distancias espaciales entre los mismos y calcular las predicciones siguiendo el mismo. Notar que es necesario un árbol para evitar dependencias circulares, ya que para predecir un canal estamos utilizando la muestra del tiempo presente de otro.

En el capitulo 1 establecimos la importancia de que la compresión de EEG fuera sin pérdida, sin embargo, en una situación donde se presente un requerimiento de *casi sin pérdida*, se puede mejorar la tasa de compresión con respecto al caso *sin pérdida* sin agregar mucha complejidad al algoritmo. Como se explica en [28], el método asegura que cada muestra d va a cumplir $|d - \hat{d}| \leq \delta$ para un δ arbitrario no negativo, siendo $\hat{d}$ el valor decodificado de d. El método es simple; consiste en cuantizar los errores de predicción en intervalos cerrados de tamaño $2\delta+1$ y centros $i \times (2\delta+1)$ con $i \in \mathbb{N}$

El codificador codifica los errores de predicción cuantizados, para lo cual se requiere en general menos bits que para describir completamente dicho error, y el decodificador utiliza el centro de cada intervalo como aproximación del valor original.

Apéndice A

Normas de posicionamiento de electrodos

El conocimiento de la posición de los electrodos es muy importante a la hora de encontrar una configuración espacial que aproveche la correlación de canales mas próximos pertenecientes a una zona en particular.

La International Federation of Societies for Electroencephalography and Clinical Neuro-physiology recomienda el posicionamiento convencional (llamado 10–20) para 21 electrodos, como se muestra en la figura A.1. Para posicionar un numero mayor de electrodos usando el sistema convencional planteado el resto de los electrodos son colocados entre los anteriores a distancias equidistantes. Por ejemplo C1 se posiciona entre C3 y Cz. Ejemplos de diferentes posicionamientos pueden verse en las figuras A.1, A.2 y A.3.

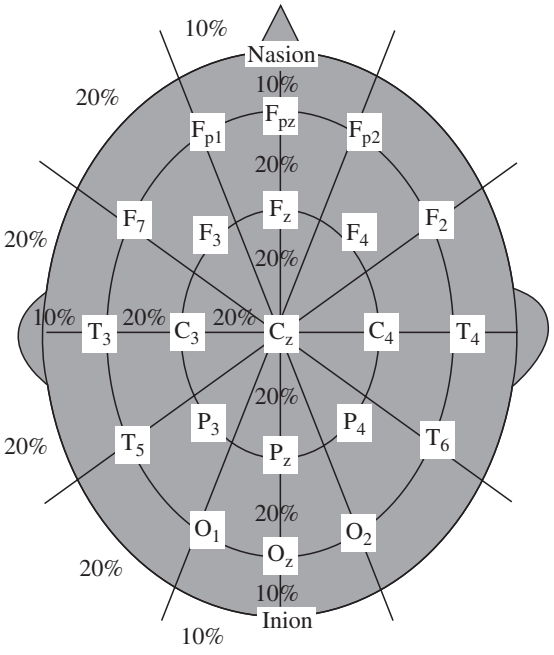


Figura A.1: Posicionamiento convencional de 21 electrodos mediante la norma 10–20

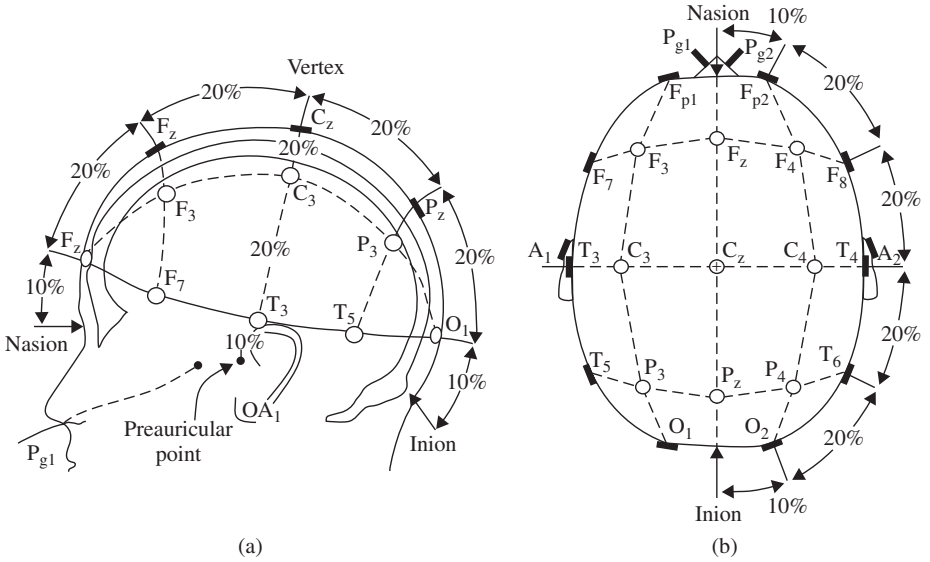


FIGURA A.2: Representación de las medidas tridimensionales de la norma 10-20.

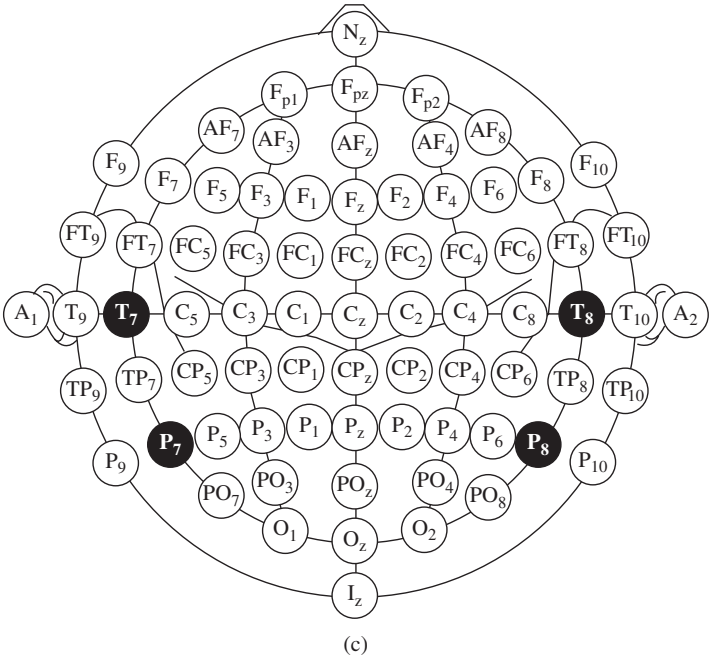


FIGURA A.3: Diagrama mostrando el posicionamiento de 74 electrodos con la norma 10–20 Diagrama mostrando el posicionamiento de 74 electrodos con la norma 10–20 incluyendo los electrodos de referencia.

Apéndice B

Bases de datos de EEG

Neuroscan[1]

En esta base de datos [39], las actividades realizadas por los pacientes fueron las de reconocimiento y categorización de animales o de objetos, y estas se dieron de forma alternada. Para la realización de dichas actividades, a los pacientes se les mostraron imágenes muy variadas. En la actividad de categorización, estos debían indicar si en la imagen había un animal/objeto, mientras que en la actividad de reconocimiento, los pacientes debieron pasar primero por una fase de aprendizaje, para luego ser capaces de reconocer las imágenes aprendidas.

Cada paciente desarrolló 25 actividades, distribuidas en dos días (13 el primer día y 12 el segundo).

La actividad YY (1 hasta 13, o 1 hasta 12 dependiendo del día) realizada por el paciente XXX (iniciales del paciente) en el dia DD (1 o 2) la identificamos por XXXDDffYY.

Los datos se capturaron con 32 electrodos (en realidad 31 mas un electrodo de referencia), siguiendo el estándar 10/20.

La frecuencia de muestreo de los datos es de 1000hz, es decir que cada electrodo realizó 1000 medicines por segundo. Esta frecuencia de muestreo es considerara alta en lo que es la captura de EEG, y esto se debe a que uno de los objetivos del las actividades fue la detección de cierto tipo de eventos de corta duración, como por ejemplo focos de epilepsia.

El análisis ERP en estos datos fue publicado en [39]. Estos datos también fueron usados para generar animaciones dinámicas del cerebro en [40] la cual es la primera publicación en computar la sincronización entre actividades separadas del cerebro utilizando ICA.

PhysioNet[2]

Los archivos de esta base de datos [41, 42] contienen capturas de señales de EEG de 14 tareas realizadas por 109 individuos distintos, es decir son un total de 1526 archivos.

Las dos primeras tareas fueron de referencia y tuvieron un minuto de duración cada una. En la primera los individuos tenían los ojos abiertos y en la segunda cerrados.

[1] http://sccn.ucsd.edu/~arno/fam2data/publicly_available_EEG_data.html
[2] http://www.physionet.org/pn4/eegmmidb/

En algunas de las restantes tareas (todas de aproximadamente 2 minutos de duración) los individuos realizaron distintos movimientos con manos y pies (por ejemplo, al aparecer un objetivo en frente de él, el individuo abría y cerraba uno de sus puños hasta que el objetivo desapareciera), mientras que en otras el individuo solamente imaginaba realizar dichos movimientos.

Las capturas fueron realizadas con 64 electrodos, utilizando el sistema BCI2000, y los mismos estaban ubicados espacialmente siguiendo el estándar 10-10 con modificaciones menores.

La frecuencia de muestreo de los datos es de 160Hz, la cual es significativamente menor que en las otras bases de datos que utilizamos.

Los archivos siguen la nomenclatura SXXXRYY, donde XXX es el número de individuo (de 000 a 109) y XX es el número de tarea (de 00 a 14).

Este conjunto de datos fue creado y contribuido a PhysioNet por los desarrolladores del sistema de instrumentación BCI200, que fue utilizado para hacer estas grabaciones en [42].

BCI Competition[3]

De esta base de datos [34], utilizamos un subconjunto de 4 archivos. Tres de ellos; data_set_IVa_aa_cnt, data_set_IVa_al_cnt y data_set_IVa_av_cnt se corresponden a individuos distintos, de iniciales “AA”, “AL”, y “AV”. Cada archivo contiene 280 ensayos realizados por el individuo, con una duración total de aprox. 50 minutos. En dichos ensayos, el individuo imaginaba querer mover la mano derecha, el pie derecho, o no realizar ningún movimiento. El objetivo de la competencia era poder clasificar lo mejor posible las intenciones del individuo.

En el cuarto archivo utilizado, data_set_IVb_al_test_cnt, el individuo involucrado realizó la mismas actividades, pero con los miembros izquierdos. Como los ensayos fueron de entrenamiento, la duración total fue menor, de aproximadamente 13 minutos.

En todos los casos, la frecuencia de muestreo de los datos es 1000Hz, y los mismos fueron capturados con 118 electrodos siguiendo el estándar internacional 10/20 extendido.

[3] https://www.bbci.de/competition/iii/#data_set_iva

Apéndice C

Detalles de implementación

Descripción del cabezal

Para poder descomprimir el archivo binario son necesarios ciertos parámetros de configuración, estos son guardados en el cabezal que se detalla en el cuadro C.1.

<table>
<tr><td colspan="2">cantidad de canales M [cod. unaria-binaria]</td></tr>
<tr><td colspan="2">tamaño de bloque [cod. unaria binaria]</td></tr>
<tr><td colspan="2">cantidad de bloques [cod. unaria-binaria]</td></tr>
<tr><td colspan="2">tamaño último bloque [cod. binaria con $\lceil \log_2(\text{tamaño de bloque}) \rceil$ bits]</td></tr>
<tr><td colspan="2">orden de autocorrelación [cod. unaria-binaria]</td></tr>
<tr><td colspan="2">bit de configuración [1 bit]</td></tr>
<tr><td rowspan="5">si hay configuración</td><td>cantidad de agrupaciones k [cod. unaria-binaria]</td></tr>
<tr><td>número de canal del 1er líder [cod. binaria con $\lceil \log_2(M) \rceil$ bits]</td></tr>
<tr><td>número de canal del 2do líder [cod. binaria con $\lceil \log_2(M) \rceil$ bits]</td></tr>
<tr><td>$\vdots$</td></tr>
<tr><td>número de canal del k-esimo líder [cod. binaria con $\lceil \log_2(M) \rceil$ bits]</td></tr>
</table>

Cuadro C.1: Estructura del encabezado de los archivos codificados.

La codificación unaria-binaria esta definida como

$$\text{unaria-binaria}(x) = \text{unaria}(\lceil \log_2(x) \rceil) \oplus \text{binaria}(x), \tag{C.1}$$

en donde la parte unaria representa la cantidad de bits que llevará la representación binaria del número.

Una vez escrito el cabezal se pasa a escribir la codificación de los canales. Para cada bloque se escribe al comienzo de la codificación de cada canal los p coeficientes del modelo AR(p). En caso

de estar utilizando una configuración, en el primer bloque, para los canales no lideres, el índice de su canal líder en la lista provista en el cabezal es escrito.

Bibliografía

[1] Sukhwinder Singh, Vinod Kumar, and HK Verma. Adaptive threshold-based block classification in medical image compression for teleradiology. *Computers in Biology and Medicine*, 37(6):811–819, 2007.

[2] Masaomi Takizawa, Shusuke Sone, Kazuhisa Hanamura, and Kazuhiro Asakura. Telemedicine system using computed tomography van of high-speed telecommunication vehicle. *Information Technology in Biomedicine, IEEE Transactions on*, 5(1):2–9, 2001.

[3] Yao-Tien Chen and Din-Chang Tseng. Wavelet-based medical image compression with adaptive prediction. *Computerized medical Imaging and graphics*, 31(1):1–8, 2007.

[4] Nasir Memon, Xuan Kong, and Judit Cinkler. Context-based lossless and near-lossless compression of EEG signals. *Information Technology in Biomedicine, IEEE Transactions on*, 3(3):231–238, 1999.

[5] Julián L Cárdenas-Barrera and Juan Valentin Lorenzo-Ginori. Mean-shape vector quantizer for ECG signal compression. *Biomedical Engineering, IEEE Transactions on*, 46(1):62–70, 1999.

[6] Rajeev Agarwal, Jean Gotman, Danny Flanagan, and Bernard Rosenblatt. Automatic EEG analysis during long-term monitoring in the ICU. *Electroencephalography and clinical Neurophysiology*, 107(1):44–58, 1998.

[7] Rajeev Agarwal and Jean Gotman. Computer-assisted sleep staging. *Biomedical Engineering, IEEE Transactions on*, 48(12):1412–1423, 2001.

[8] Hans Berger. Über das elektrenkephalogramm des menschen. *European Archives of Psychiatry and Clinical Neuroscience*, 87(1):527–570, 1929.

[9] Giuliano Antoniol and Paolo Tonella. EEG data compression techniques. *Biomedical Engineering, IEEE Transactions on*, 44(2):105–114, 1997.

[10] Rajeev Agarwal and Jean Gotman. Long-term EEG compression for intensive-care settings. *Engineering in Medicine and Biology Magazine, IEEE*, 20(5):23–29, 2001.

[11] Natarajan Sriraam and C Eswaran. An adaptive error modeling scheme for the lossless compression of EEG signals. *Information Technology in Biomedicine, IEEE Transactions on*, 12(5):587–594, 2008.

[12] Natarajan Sriraam and C Eswaran. Performance evaluation of neural network and linear predictors for near-lossless compression of EEG signals. *Information Technology in Biomedicine, IEEE Transactions on*, 12(1):87–93, 2008.

[13] Natarajan Sriraam and C Eswaran. Context based error modeling for lossless compression of EEG signals using neural networks. *Journal of Medical Systems*, 30(6):439–448, 2006.

[14] Helen C Sing, Mary A Kautz, David R Thorne, Stanley W Hall, Daniel P Redmond, Dagny E Johnson, Kimberly Warren, Joshua Bailey, and Michael B Russo. High-frequency EEG as measure of cognitive function capacity: a preliminary report. *Aviation, space, and environmental medicine*, 76(Supplement 1):C114–C135, 2005.

[15] Alexander J Casson, Shelagh Smith, John S Duncan, and Esther Rodriguez-Villegas. Wearable EEG: what is it, why is it needed and what does it entail? In *Engineering in medicine and biology society, 2008. embs 2008. 30th annual international conference of the ieee*, pages 5867–5870. IEEE, 2008.

[16] J. J. Wright, R. R. Kydd, and A. A. Sergejew. Autoregression models of EEG. *Biological cybernetics*, 62(3):201–210, 1990.

[17] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

[18] L. Kaufman and P. Rousseeuw. Clustering by Means of Medoids. In Faculty of Mathematics and Informatics; Delft University of Technology, editor, *Reports of the Faculty of Mathematics and Informatics*. Delft : Faculty of Mathematics and Informatics, Netherlands, 1987.

[19] S. W. Golomb. Run-length encodings. *Information Theory, IEEE Transactions on*, 12(3):399, 1966.

[20] Justin Dauwels, Senior Member, K Srinivasan, and M Ramasubba Reddy. Near-Lossless Multichannel EEG Compression Based on Matrix and Tensor Decompositions. 17(3):708–714, 2013.

[21] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[22] James Pardey, Stephen Roberts, and Lionel Tarassenko. A review of parametric modelling techniques for EEG analysis. *Medical engineering & physics*, 18(1):2–11, 1996.

[23] Tom Bäckström. Levinson-Durbin Recursion. In *Linear Predictive Modelling of Speech - Constraints and Line Spectrum Pair Decomposition.* 2004.

[24] J. P. Burg. Maximum entropy spectral analysis. In *Proc. 37th Meeting Soc. Exploration Geophys.*, Oklahoma City, Okla., October 1967.

[25] M. K. Ibrahim. Improvement in the speed of the data-adaptive weighted Burg technique. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(10):1474–1476, October 1987.

[26] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications*, 36(2, Part 2):3336 – 3341, 2009.

[27] Neri Merhav, Gadiel Seroussi, and Marcelo J Weinberger. Coding of sources with two-sided geometric distributions and unknown parameters. *Information Theory, IEEE Transactions on*, 46(1):229–236, 2000.

[28] Marcelo J Weinberger, Gadiel Seroussi, and Guillermo Sapiro. The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS. *Image Processing, IEEE Transactions on*, 9(8):1309–1324, 2000.

[29] Neri Merhav, Gadiel Seroussi, and Marcelo J Weinberger. Optimal prefix codes for sources with two-sided geometric distributions. *Information Theory, IEEE Transactions on*, 46(1):121–135, 2000.

[30] B H Jansen, J R Bourne, and J W Ward. Autoregressive estimation of short segment spectra for computerized EEG analysis. *IEEE transactions on bio-medical engineering*, 28(9):630–8, September 1981.

[31] Hirotogu Akaike. Information theory and an extension of the maximum likelihood principle. In *Breakthroughs in statistics*, pages 610–624. Springer, 1992.

[32] F Vaz, P G De Oliveira, and J C Principe. A study on the best order for autoregressive EEG modelling. *International journal of bio-medical computing*, 20(1-2):41–50, January 1987.

[33] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.

[34] G. Dornhege, B. Blankertz, G. Curio, and K. Muller. Boosting bit rates in noninvasive EEG single-trial classifications by feature combination and multiclass paradigms. 51(6):993–1002, June 2004.

[35] James Pardey, Stephen Roberts, and Lionel Tarassenko. A review of parametric modelling techniques for EEG analysis. *Medical engineering & physics*, 18(1):2–11, 1996.

[36] Peter Strobach. New forms of levinson and schur algorithms. *Signal Processing Magazine, IEEE*, 8(1):12–36, 1991.

[37] A.C. Singer and M. Feder. Universal linear prediction by model order weighting. *Signal Processing, IEEE Transactions on*, 47(10):2685–2699, Oct 1999.

[38] Ignacio Capurro, Federico Lecumberry, Alvaro Martín, Ignacio Ramirez, Eugenio Rovira, and Gadiel Seroussi. Low-complexity, low-latency, real-time, multi-channel, lossless and near-lossless EEG compression. Paper submitted for publication, 2014.

[39] Arnaud Delorme, Guillaume A Rousselet, Marc J.-M Macé, and Michèle Fabre-Thorpe. Interaction of top-down and bottom-up processing in the fast visual analysis of natural scenes. *Cognitive Brain Research*, 19(2):103 – 113, 2004.

[40] Arnaud Delorme, Scott Makeig, Michèle Fabre-Thorpe, and T Sejnowski. From single-trial EEG to brain area dynamics. *Neurocomputing*, 44:1057–1064, 2002.

[41] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[42] G. Schalk, D.J. McFarland, T. Hinterberger, N. Birbaumer, and J.R. Wolpaw. BCI2000: a general-purpose brain-computer interface (BCI) system. 51(6):1034–1043, June 2004.