Informe Final

Proyecto de grado
Compresión de Electroencefalogramas

Lindsay Ramos
Marianela Carbone

Tutores
Álvaro Martín - Federico Lecumberry - Ignacio Ramirez

Instituto de Computación
Facultad de Ingeniería - Universidad de la República
Montevideo - Uruguay

7 de mayo de 2014

Agradecimientos

Es conmovedor para nosotras culminar esta etapa de mucho esfuerzo y dedicación, lo cual no hubiera sido posible sin el apoyo de muchas personas a lo largo de este proceso.
Nos parece importante agradecer a nuestros tutores Álvaro Martín, Ignacio Ramirez y Federico Lecumberry por estar presentes en todo momento, orientarnos a buscar mejores soluciones a los problemas que nos fueron surgiendo y aportando ideas durante el trascurso del proyecto.
Queremos agradecer y dedicar este trabajo a nuestros familiares y amigos por su apoyo incondicional a lo largo de nuestra carrera universitaria, brindándonos siempre una voz de aliento para seguir adelante.

Resumen

La compresión de señales *EEG* surge a partir de la necesidad de almacenar o transmitir grandes volúmenes de datos, principalmente en aplicaciones médicas.
El objetivo de nuestro trabajo consistió en analizar y evaluar métodos de compresión de señales *EEG*. Según dicho análisis se optó por implementar y evaluar diferentes variantes del algoritmo presentado en [1].
En el desarrollo del algoritmo fueron utilizadas técnicas de predicción (*AR adaptativo* y *AR no adaptativo*) para estimar cada valor de la señal original en función de los valores anteriores, obteniendo así un error de predicción, el cual fue corregido mediante técnicas de *cancelación de sesgo (BC)* y finalmente codificado mediante codificación condicional aplicada a códigos de *Golomb*.
Luego se analizaron diferentes variantes en los parámetros del algoritmo, destacándose los resultados obtenidos en la capacidad de compresión pues con las señales analizadas se obtuvo en promedio más de un 60 % de compresión. Sin embargo, no se obtuvieron los resultados esperados al utilizar la técnica de *BC* dado que no mejora significativamente el error de predicción y como consecuencia la compresión.
Con respecto a los contextos de codificación utilizados, se observó que aumentando la cantidad de contextos de codificación no se obtiene una mejora significativa en la compresión.
A pesar de estos resultados, se obtuvo una mayor compresión con el algoritmo implementado que con el compresor *Zip*[2]. Se destaca también que el algoritmo desarrollado puede ser adaptado para utilizarse tanto en contextos *online* como *offline*.

Palabras clave Electroencefalograma, Compresión, Modelo Autorregresivo, Cancellation Bias, Código de Golomb.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Problema

Los electroencefalogramas (*EEG*) son un método no invasivo para medir las ondas cerebrales de una persona. Son utilizados en diversos campos, tales como la medicina, como herramienta de detección de enfermedades cerebrales, como por ejemplo epilepsia, hasta los sistemas *Brain-Computer Interface (BCI),* donde se utilizan para detectar estímulos cerebrales y utilizarlos para comunicarse con un dispositivo. Para la obtención de los *EEG* se utilizan electrodos colocados sobre el cuero cabelludo para detectar las señales que resultan de la actividad neuronal sincronizada dentro del cerebro. La actividad eléctrica cerebral se representa en una gráfica de voltaje (ubicado en el eje y) contra el tiempo (ubicado en el eje x) como muestra la figura 1.1.1.
La compresión de señales *EEG* surge a partir de la necesidad de almacenar o transmitir grandes volúmenes de datos, principalmente en aplicaciones médicas. Se debe tener en cuenta que se suele utilizar varios electrodos que registran simultáneamente información de diferentes zonas del cerebro, generalmente durante varias horas, lo cual demanda una gran capacidad de almacenamiento. Esta situación exige el uso de técnicas *eficientes* de compresión de datos.

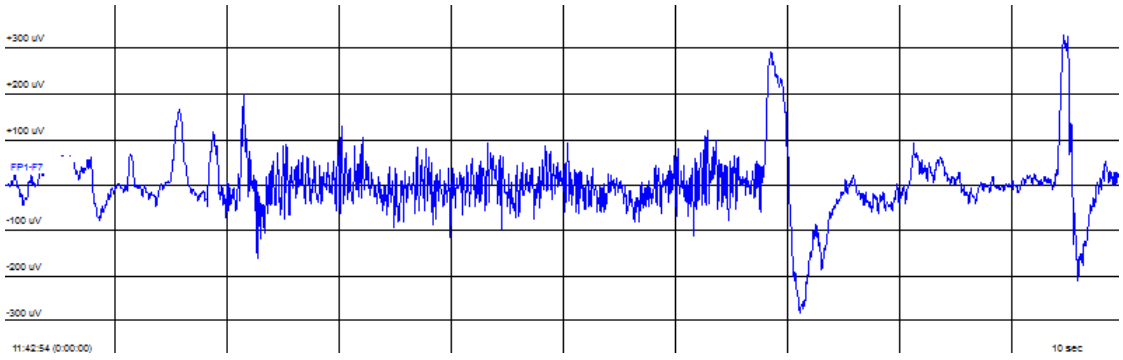


Figura 1.1.1: Gráfico de un *EEG* (voltaje vs tiempo)

1.2. Objetivos

Los principales objetivos del proyecto fueron:

- Realizar un análisis y evaluación de diferentes métodos de compresión de señales *EEG*.
- Realizar la implementación de algunos de los algoritmos de compresión analizados.
- Proponer y experimentar variantes de estos algoritmos.

1.3. Organización del Documento

El presente documento se encuentra organizado de la siguiente manera. El capítulo 2 presenta una breve descripción del estado del arte, incluyendo conceptos generales utilizados, descripción de la obtención y almacenamiento de datos *EEG*, y algunos de los formatos de almacenamiento más comúnmente utilizados. Además se describe detalladamente el algoritmo de compresión a partir del cual realizaremos nuestra contribución. Un análisis más detallado de algunos de las métodos evaluados para la compresión de señales *EEG* se encuentra en el apéndice A.
En el capítulo 3 se describe en detalle la implementación del compresor realizada como parte del proyecto, el proceso de desarrollo, y los problemas encontrados a lo largo del mismo. Nuestra implementación incluye diversas variantes del algoritmo descrito en [1]; algunas de estas variantes corresponden a definiciones dejadas expresamente abiertas en [1] y otras corresponden a nuevas ideas surgidas durante el desarrollo del proyecto.
En el capítulo 4 se describe un resumen de los experimentos realizados junto con los resultados obtenidos.
Por último, en el capítulo 5 se realiza un resumen de las conclusiones recabadas durante el proyecto y se identifican algunas líneas de trabajo a futuro.

Capítulo 2

Compresión de electroencefalogramas

2.1. Obtención de Datos

Para la obtención de datos se utilizan electrodos que se colocan generalmente en el cuero cabelludo humano de acuerdo con la recomendación de la *Federación Internacional de Sociedades de Electroencefalografía y Neurofisiología Clínica* llamada *Sistema Internacional 10-20*, ilustrada en la figura 2.1.1. Este sistema asegura que se coloquen los electrodos sobre las mismas áreas, independientemente del tamaño de la cabeza. Se considera que un electrodo es un canal que emite una señal *EEG*.

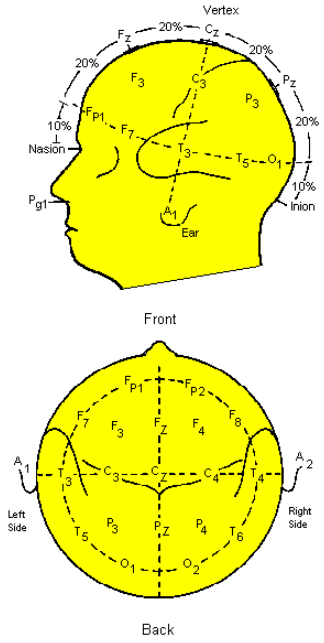


Figura 2.1.1: Posiciones convencionales de los electrodos según el *Sistema Internacional 10-20* para la medición de señales en un *EEG* multi-canal.

Las señales de *EEG* en principio son *analógicas*, es decir continuas en el tiempo y en amplitud. Mediante un proceso de digitalización se convierte una señal de *EEG* analógica en una señal digital, que toma valores discretos en tiempo discreto. En la figura 2.1.2 se muestran las tres etapas básicas de dicho proceso.

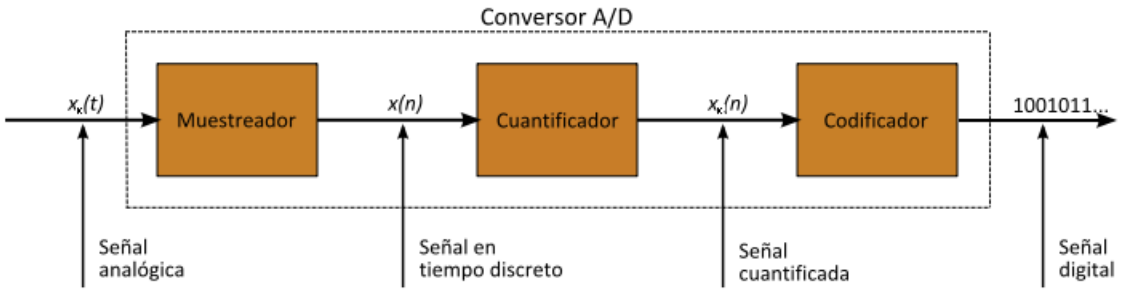


Figura 2.1.2: Proceso de digitalización de una señal de *EEG* analógica a una señal digital.

Durante el proceso de digitalización se mide el nivel de tensión de cada una de las muestras $x(n)$, obtenidas en el proceso de muestreo (muestreador), y se les atribuye un valor finito (discreto) de amplitud (cuantificador). En este momento, la señal analógica se convierte en una señal cuantificada, $x_k(n)$, que toma valores en un conjunto finito preestablecido. Luego el codificador transformará cada $x_k(n)$ en una sucesión de ceros y unos con un largo determinado. El resultado final del codificador se representará con el conjunto $\{x_k\}$.
La señal digital $\{x_k\}$ resultado del *Convertidor A/D* es la que finalmente se almacena en diferentes formatos digitales.

Alguno de los formatos más comunes de almacenamiento de *EEG* son:

- Archivos *CNT*
- General Data Format (*GDF*)
- European Data Format (*EDF*)

Ninguno de estos formatos incluye algún tipo de compresión de datos.

Los *archivos CNT* [3] contienen una forma binaria de los datos producidos por algunos dispositivos de control de Neurofisiología, como un electroencefalograma. El formato de un archivo *CNT* esencialmente contiene tres partes: Cabecera, Datos y Pie de página. La cabecera contiene información básica como por ejemplo el número de canales, en los datos se almacenan las muestras de cada canal y el pie de página contiene la tabla de eventos que indican donde ocurrieron los eventos en el archivo *CNT*.

El *General Data Format* [4] está constituido por cinco secciones, entre ellas se especifican tres secciones de encabezados una sección de datos y una sección con la tabla de eventos. El primer encabezado es fijo y contiene datos del paciente, el número de señales etc. El segundo encabezado es variable y contiene información específica de cada canal. El tercer encabezado contiene información opcional que se necesite almacenar en cierto experimento. La sección de datos contiene las muestras de cada canal. La última sección contiene la información de los eventos para cada canal y momento que ocurrieron.

El *European Data Format* [5] es un formato estándar para almacenar señales biológicas y físicas multicanal, y define dos registros principales:

Registro de encabezado: es una sección de longitud variable que indica datos del paciente y características técnicas de las señales grabadas. Algunos de los datos de interés contenidos en el registro de encabezado son: el número de canales (que indica la cantidad de electrodos de los que se tomaron muestras), la duración de un registro de datos (en segundos) y el número de muestras en cada registro de datos. A partir de estos últimos dos valores se puede calcular la frecuencia de muestreo.

Registro de datos: contiene épocas de duración fija consecutivas del registro poligráfico, que es obtenido simultáneamente de un electroencefalograma con otros parámetros fisiológicos como la activación muscular, el ritmo cardíaco, la respiración y los movimientos oculares. Cada muestra se representa con un entero de 2 bytes en complemento a 2.

2.2. Etapas de Compresión

Los algoritmos de compresión de señales digitales de *EEG* constan en general de tres etapas características como se muestran en la figura 2.2.1.

Figura 2.2.1: Etapas identificadas en el proceso de compresión de señales *EEG*.

2.2.1. Transformación

La etapa de *Transformar* consiste en aplicar un algoritmo que transforma la señal digital original $\{x_k\}$, para llevarla a una forma más apropiada para las etapas posteriores.
Los algoritmos de compresión de señales *EEG*, de acuerdo al algoritmo de transformación que utilizan para comprimir, se pueden agrupar en algoritmos basados en técnicas de predicción, o basados en transformadas. Entre los algoritmos basados en transformadas se destacan variantes de las *wavelets* (*WT*) [6], utilizadas por ejemplo en [7, 8] , *wavelets packets* (*WTP*) [9, 10] y la transformada de *KLT* [11], que se utiliza por ejemplo en [12], transformadas a espacio de frecuencias [13] y bancos de filtros [14], ambas utilizadas en [15]. También se puede transformar la señal a una matriz (2-D), de forma de utilizar algoritmos para compresión de imágenes como en [16].
Algunos algoritmos basados en predicción son: variantes de *modelos autorregresivos* (*AR*) [1], predicción aplicando el Polinomio de Interpolación de Newton con Diferencias Divididas [17], utilizado en [18] y predicción basada en redes neuronales [19] utilizado en [20].

El algoritmo que se propone en [1], que tomamos como iniciativa en este proyecto, se basa en técnicas de *AR*. En este caso, la transformación consiste en restar a cada muestra de la señal de entrada una estimación de su valor calculado en función de muestras anteriores. En [1] se utiliza un predictor lineal, que estima el valor de la muestra actual mediante una combinación lineal de una cantidad predefinida de muestras anteriores.

2.2.2. Cuantificación

La etapa de *Cuantificar* consiste en representar en forma aproximada la señal transformada. Para ello, toda la gama de valores que pueden tomar las muestras se divide en intervalos, y a todas las muestras que caen dentro de un intervalo se les asigna el mismo valor.
Esta etapa en general es usada en compresores con pérdida, en los cuales la señal descomprimida puede no coincidir exactamente con la original. Notar que esta etapa de cuantificación es diferente a la del *Conversor A/D* mencionado en la sección 2.1.
Una de las técnicas de cuantificación que se utiliza en compresión de *EEG* está basada en la selección de un valor umbral, donde dicho valor se determina dependiendo de la calidad que se quiera obtener de la señal recuperada luego de la compresión. Luego se toma la salida de la etapa de transformación y si dicha salida queda por debajo del umbral determinado se le aplica una cuantificación determinada y si queda por encima del umbral se aplica otra cuantificación. Variantes de este tipo de cuantificador, son utilizadas en [9, 20, 10].

2.2.3. Codificación

La etapa de *Codificar* consiste en aplicar un mapeo invertible del conjunto de señales transformadas, y eventualmente cuantificadas, al conjunto de secuencias binarias. A diferencia de la codificación realizada en el *Conversor A/D* de la sección 2.1, en estos casos se busca asignar secuencias binarias cortas a las señales que se estiman más probables que ocurran y (necesariamente) secuencias más largas a las que se estiman menos probables. Por ejemplo, cuando la etapa de transformación está basada en predicción, los valores pequeños de las diferencias entre el valor estimado y el valor real se codifican con pocos bits, mientras que errores de predicción grandes, que son en general menos usuales se codifican con más bits.

Algunas de las técnicas de codificación que se utiliza en compresión de *EEG* son: variantes de códigos de *Huffman* [21], utilizados en [12, 1], codificación *Run-length* [22], utilizada en [9, 15, 10] y *codificación aritmética* (*AC*) [23] utilizada en [8, 16, 20].

2.3. Algoritmo de compresión basado en contextos

A continuación se describe en detalle el algoritmo de compresión basado en contextos [1] y en el cual nos basamos para realizar nuestra contribución. Se destacan las siguientes etapas para la compresión:

Transformación El esquema de predicción de la etapa de transformación, está basado en un *modelo autorregresivo (AR)*. Este modelo trata a la señal *EEG* como una serie de tiempo $\{x_k\}$ donde cada muestra (un número entero de 16 bits), x_k, se estima en función de las P muestras anteriores como:

$$\hat{x}_k = \sum_{i=1}^{P} a_i x_{k-i} \; con \; k \geq P, \tag{2.3.1}$$

siendo P y $\{a_i\}_{i=1,\ldots,P}$, el orden y los coeficientes respectivamente del modelo *AR*. Dado que cada muestra x_k es un número entero, se denota como $\hat{x}_k^q$ al entero más cercano de la muestra predicha $\hat{x}_k$, siendo *q(.)* la función de redondeo aplicada a $\hat{x}_k$.
La diferencia $e_k = x_k - \hat{x}_k^q$ es un entero que representa el error de predicción para la muestra x_k. Los coeficientes del modelo pueden ser inicializados con un conjunto de valores fijos o pueden utilizarse algoritmos que adaptan dichos coeficientes a la señal procesada.
La estimación preliminar de $\hat{x}_k^q$ se refina posteriormente mediante una técnica conocida como *cancelación de sesgo* (*BC*) . Dicha técnica define, para cada muestra x_k, un contexto $C = (d_1, d_2, d_3, d_4, d_5)$, donde

$d_i = signo(x_{k-i} - x_{k-i-1})$, $1 \leq i \leq P-1$. Asociado a cada contexto C, se mantiene una estimación entera del sesgo $\bar{\varepsilon}(C)$ (media muestral) obtenida a partir de la estimación de las muestras que ocurren en el contexto C, que se suma a la predicción de la muestra x_k para calcular una nueva estimación de la muestra $\tilde{x}_k^q$:

$$\tilde{x}_k^q = \hat{x}_k^q + \bar{\varepsilon}(C) \tag{2.3.2}$$

Codificación En ésta etapa se realiza la codificación de los errores de predicción luego de aplicar *BC,* dicho error $\tilde{e}_k$ es un entero y se define como:

$$\tilde{e}_k = x_k - \tilde{x}_k^q, \tag{2.3.3}$$

Con el fin de codificar los errores de predicción de manera eficiente, se necesita un modelo que capture la estructura de los mismos. Este paso se refiere a menudo como el *modelado de error*. Dicho modelado está basado en estados de condicionamiento, que se denominan *contextos de codificación*. El *contexto de codificación* en una muestra x_k se define a partir de la función $A(s)$, donde $A(.)$ es un cuantificador escalar no uniforme de N niveles y s es el promedio ponderado de los errores de predicción anteriores definido como:

$$s = |e_{k-1}| + |e_{k-2}| + 0{,}75|e_{k-3}| + 0{,}75|e_{k-4}| + 0{,}5|e_{k-5}| + 0{,}5|e_{k-6}|, \tag{2.3.4}$$

En [1] no se especifica de qué forma se calculan los intervalos del cuantificador escalar de N niveles.
En cada *contexto de codificación* se recolectan las estadísticas de los errores de predicción ocurridos en el mismo. De esta forma se obtiene una distribución empírica del error de predicción para luego aplicar *Huffman* [21]. El código de *Huffman* es un código óptimo si se conoce la distribución de probabilidad en el receptor. En caso contrario, se tiene que describir la distribución mediante contadores que indiquen cuantas veces ocurre cada error en un contexto. Luego, el decodificador, utilizando los contadores, puede reconstruir la distribución de probabilidad empírica y decodificar cada muestra. Estos contadores deben ser codificados y trasmitidos junto con la codificación de las muestras, lo cual implica utilizar una cantidad de bits extra como parte de la codificación.

Capítulo 3

Desarrollo de variantes de un compresor basado en contextos

3.1. Descripción de la Solución

En este capítulo se describen las características del compresor de *EEG* desarrollado. Nuestro trabajo consistió en implementar y probar diferentes variantes del algoritmo presentado en [1] y descripto en la sección 2.3. Algunas de las variantes se corresponden con decisiones que se dejan expresamente abiertas en [1] y otras son mejoras propuestas durante el desarrollo del proyecto. Se describen estas variantes en las etapas de transformación y codificación.

3.1.1. Transformación

En ésta etapa se implementó un predictor lineal *AR adaptativo* y un predictor lineal *AR no adaptativo,* de orden *P* configurable. El parámetro configurable *P*, así como otros parámetros que se describen más adelante, se incluyen en un encabezado del archivo comprimido (detallado en el apéndice B) para que estén accesibles al decodificador. La diferencia entre los predictores lineales se da en que los coeficientes del modelo *AR adaptativo* se estiman progresivamente, mientras que los coeficientes del modelo *AR no adaptativo* se mantienen fijos con los valores iniciales que se le asigne. Los valores iniciales de los coeficientes del predictor lineal *AR* son trasmitidos en el encabezado del archivo comprimido.
Existen varios métodos para el cálculo de los coeficientes, entre ellos el de *Levinson-Durbin* [24]. Éste método fue uno de los utilizados para calcular los coeficientes iniciales del modelo tanto para el modelo *AR adaptativo* como para el *AR no adaptativo*.
Para la estimación de los coeficientes del modelo *AR adaptativo* se utilizó el método *Least mean squares (LMS*). El objetivo del algoritmo *LMS* es el de adaptar progresivamente los coeficientes de un predictor lineal de modo de minimizar el error cuadrático medio:

$$< e^2 >= \frac{1}{d}\sum_{k=1}^{d} e_k^2 = \frac{1}{d}\sum_{k=1}^{d} (x_k - a^T x_{k-P}^{k-1})^2, \tag{3.1.1}$$

donde a es el vector de coeficientes del modelo *AR,* a^T es la transpuesta de a, $x_i^j = (x_i, x_{i+1} ... x_j)^T$, y d es la cantidad total de muestras a procesar.

Luego de estimar la muestra x_k a partir de las P muestras anteriores y de los coeficientes $a(k-1)$ calculados en función de x_1^{k-1}, se calculan los coeficientes $a(k)$ para estimar la próxima muestra. Para resolver este problema se realiza un descenso por gradiente. Llamamos $J_k(a)$ al error cuadrático medio obtenido hasta la posición k, con coeficientes a,

$$J_k(a) = \frac{1}{k}\sum_{r=1}^{k}(x_r - a^T x_{r-P}^{r-1})^2 \tag{3.1.2}$$

y actualizamos los coeficientes de la siguiente forma:

$$a(k) = a(k-1) - \eta\,\nabla J_k(a(k-1)), \tag{3.1.3}$$

donde $0 < \eta < 1$ es un parámetro que controla la convergencia del algoritmo. Desarrollando se obtiene:

$$a(k) = a(k-1) - \eta\,\frac{2}{k}\sum_{r=1}^{k} e_r x_{r-P}^{r-1} = a(k-1) - \eta\,\frac{2}{k}\,w(k), \tag{3.1.4}$$

donde la estadística $w(k) = \sum_{r=1}^{k} e_r x_{r-P}^{r-1}$ se calcula de forma incremental como $w(k) = w(k-1) + e_k x_{k-P}^{k-1}$.
El valor η es calculado empíricamente como:

$$\eta = n\,/\,var(x_1^d), \tag{3.1.5}$$

donde n es un valor configurable del algoritmo y $var(x_1^d)$ es la varianza empírica de toda la señal (x_1^d) a comprimir. El valor de n debe ser menor a 10^{-4} (valor determinado empíricamente) y se transmite en el encabezado del archivo comprimido para ser utilizado por el decodificador, al igual que la combinación del tipo de predictor lineal junto con los coeficientes iniciales que fueron utilizados en la compresión.

Posteriormente se aplica el método de *BC* para corregir la estimación de la muestra obtenida por el predictor lineal *AR* utilizado. Los contextos utilizados al aplicar *BC* son los mencionados en la sección 2.3. Este último método también se puede configurar de forma que se pueda elegir si aplicar o no *BC*, dicha elección se trasmite en el encabezado del archivo comprimido.

3.1.2. Codificación

En [1] se utiliza un código de *Huffman* adaptado a la distribución empírica de los errores de predicción (lo cual implica codificar la distribución de probabilidad), como se mencionó en la sección 2.3. Observando empíricamente que los valores absolutos de los errores de predicción parecen ser bien modelados por una distribución geométrica [25], y tomando como referencia el estándar de compresión sin pérdidas de imágenes, *JPEG-LS* [26], decidimos implementar una codificación de *Golomb.*
La codificación óptima (en el sentido de minimizar el largo de código esperado) de enteros no negativos sorteados con una distribución geométrica se puede realizar mediante un código de *Golomb*. Estos códigos dependen de un parámetro entero, $m\,(m \geq 1)$ llamado orden del código.
El código de *Golomb* de orden m para un entero no negativo z se define como:

$$G_m(z) = binary_m(z \bmod m) \cdot unary(z \operatorname{div} m), \tag{3.1.6}$$

donde $z \bmod m$ y $z \operatorname{div} m$ son el resto y la parte entera del cociente z/m respectivamente, $binary_m(j)$ es la representación binaria de j como entero no negativo de $\lceil log\,m \rceil$ bits y $unary(j)$ es la representación unaria de j, $\underbrace{00...,0}_{j}1$.

Dado m y $G_m(z)$ se puede decodificar de forma única el valor z. Un caso particular y muy utilizado de los *códigos de Golomb* se obtiene cuando $m = 2^k$ para algún entero $k \geq 0$, en cuyo caso el cómputo de $G_m(z)$ es elemental.

El *código de Golomb* fue diseñado para codificar secuencias de números enteros no negativos. Sin embargo, existe una extensión del mismo para considerar los números enteros negativos, que consiste en mapear de forma única y reversible los valores a un valor positivo antes de ser codificados.
El mapeo es el siguiente:

$$M(\varepsilon) = \begin{cases} 2\varepsilon & \varepsilon \geq 0 \\ -2\varepsilon - 1 & \varepsilon < 0 \end{cases} \tag{3.1.7}$$

El resultado del mapeo intercala valores negativos y positivos en la secuencia $0, -1, 1, -2, 2...$. Los resultados de aplicar $M(\varepsilon)$ son los que finalmente se codifican cuando se utiliza la extensión del *código de Golomb* original.
El valor óptimo del parámetro k para codificar enteros aleatorios distribuidos depende del parámetro q de la distribución geométrica $P_\theta(q)$ definida como:

$$P_\theta(q) = (1-\theta)\,\theta^q \; 0 < \theta < 1, \tag{3.1.8}$$

siendo $m = 2^k$ el único número entero (positivo) que satisface:

$$\theta^m + \theta^{m+1} \leq 1 < \theta^m + \theta^{m-1} \tag{3.1.9}$$

Si este parámetro q no es conocido, puede estimarse k a partir de N' muestras como:

$$k = min\{k' \,|\, 2^{k'} \,.\, N' \geq A\}, \tag{3.1.10}$$

donde A es la suma acumulada de las magnitudes de los errores de predicción de las N' muestras anteriores.

En cada contexto de codificación (ver sección 2.3), se mantienen estadísticas de los errores que pertenecen al mismo, las cuales se van recolectando y actualizando a lo largo del procesamiento del vector de errores, y se corresponden con las variables A *y* N'.
Para obtener el contexto c al que pertenece cada error de predicción e_k, se utiliza la función $A(s)$ definida en la sección 2.3, donde cada error tiene asociado un valor s calculado según (2.3.4) a partir de los errores de predicción $e_{k-1}...e_{k-6}$. El valor de los N niveles utilizados por la función $A(s)$ es un parámetro configurable del algoritmo.
Para el cálculo de los umbrales utilizados por la función $A(s)$ se realiza una estimación de la función acumulativa empírica, ordenando los s de menor a mayor y formando un vector ordenado s_{ord}. Los $N+1$ umbrales quedan determinados por:

$$umbral(i) = \begin{cases} 0 & i = 0 \\ s_{ord}(\lfloor \frac{d}{N} \rfloor * i - 1) & 0 < i < N \\ s_{ord}(d-1) & i = N \end{cases}, \tag{3.1.11}$$

donde d representa la cantidad total de muestras a comprimir. Utilizando esta definición de umbrales se logra que todos los contextos tengan aproximadamente la misma cantidad de muestras. Los umbrales y el valor de N utilizados en la codificación son trasmitidos en el encabezado del archivo comprimido. La salida de la etapa de codificación es un archivo (con extensión *.EEGzip*) que contiene las siguientes dos secciones principales:

Registro de encabezado: es una sección de longitud variable, donde se incluyen parámetros del algoritmo y el cabezal del archivo .edf original.

Registro de datos: es una sección de longitud variable, que contiene la codificación de cada muestra resultante del *código de Golomb*.

El formato del archivo comprimido se describe con más detalle en el apéndice B.

Para verificar el correcto funcionamiento del algoritmo de compresión de *EEG* y de las diferentes variantes del mismo, se implementó un módulo de decodificación del archivo comprimido. El mismo consiste en tomar como entrada el archivo comprimido (por el módulo de codificación) y decodificar muestra a muestra para generar nuevamente la señal digital original y almacenarla en un nuevo archivo .edf.

3.2. Proceso de Desarrollo

En el desarrollo del proyecto se pueden identificar varias etapas. La primera etapa se centró en la investigación sobre conceptos generales de compresión y el estado del arte en compresión de *EEG*, donde posteriormente se realizó una evaluación de los métodos encontrados seleccionando el más adecuado para su implementación.
La segunda etapa identificada consistió en la implementación de un prototipo del método seleccionado, para luego evaluar su funcionamiento y proponer posibles mejoras al mismo, cambiando eventualmente alguna parte. El prototipo fue desarrollado en *Matlab* para contar con mejores herramientas gráficas y realizar, de forma temprana, experimentos que permitieron evaluar el correcto funcionamiento del algoritmo. Entre los experimentos realizados se destaca el estudio de las estadísticas de error para identificar si era posible ajustar las distribuciones empíricas del error absoluto a distribuciones geométricas, de forma de poder utilizar una codificación de *Golomb* en lugar de *Huffman*. Los resultados de éste y otros experimentos realizados se detallan en la sección 4.3.
En la tercera etapa identificada, se implementó en lenguaje *C* el método seleccionado y las variantes realizadas al mismo. Esto permitió mejorar el rendimiento de los experimentos con mayores volúmenes de datos.
En esta etapa se definieron los requisitos principales necesarios para el desarrollo de la solución:

- El código desarrollado sea portable para los sistemas operativos *Windows* y *Linux*.
- Exista un parámetro que permita guardar las variables intermedias generadas por el compresor.
- Que los parámetros relevantes de la compresión sean configurables en un archivo con un formato determinado.

La estructura de la solución implementada consta de los siguientes tres módulos:

- ***Acceso a Datos:*** Este módulo se encarga de la manipulación de las señales a comprimir y del archivo comprimido.
  - Archivos EDF: Sub-módulo encargado de la manipulación de los archivos *EDF*.
  - Archivos comprimidos: Sub-módulo encargado de la manipulación de los archivos comprimidos.
- ***Codificador:*** Este módulo se encarga de la compresión de las señales.

  - Predicción: Sub-múdulo encargado de la implementación de la etapa de transformación definida en la sección 3.1.1.
  - Codificación: Sub-múdulo encargado de la implementación de la etapa de codificación definida en la sección 3.1.2.

- ***Decodificador:*** Este módulo se encarga de la descompresión de las señales comprimidas.

  - Deco-Predicción: Sub-múdulo encargado de implementar el método inverso de la etapa de trasformación.
  - Decodificación: Sub-múdulo encargado de implementar el método inverso de la etapa de codificación.

Luego de la implementación de cada módulo, se realizaron pruebas de forma independiente e integrada de los mismos para asegurar su correcto funcionamiento.
Al finalizar la implementación del compresor y descompresor en *C*, se realizó el diseño de los experimentos descritos en el apéndice C. Finalizada la etapa de diseño de los experimentos se implementaron los mismos en lenguaje *C* para luego realizar el análisis de los resultados obtenidos.

3.3. Problemas encontrados

A lo largo del proyecto, se presentaron dificultades en las diferentes etapas. A continuación se nombran algunos de los problemas junto con la solución encontrada:

- Las pruebas en Matlab del prototipo demoraban varios minutos con señales de más de 100.000 muestras, lo cual dificultó el análisis de los resultados para señales con muchas muestras. Este problema se eliminó al realizar la implementación en lenguaje *C*.

- Al implementar la solución final en lenguaje *C*, tuvimos dificultades para utilizar las operaciones brindadas por la librería edflib [27], ya que no cumplían con nuestros requerimientos para reconstruir la señal al formato edf original. Dado que la librería es de código abierto, pudimos adaptar la misma a los requerimientos del proyecto aunque no fue algo trivial.

- Una de las dificultades principales fue la familiarización con las señales *EEG*. A raíz de ésto surgieron problemas en su manipulación. Por citar un ejemplo, al realizar la conversión de las señales al formato edf distorsionamos la señal, a raíz de ello se estaba trabajando con datos incorrectos y conclusiones erróneas. Otro problema en la manipulación fue que realizábamos los experimentos con señales de epilepsia, que luego tuvimos que descartar porque contenían mucho ruido de red eléctrica.

Capítulo 4

Experimentos y Resultados

4.1. Introducción

En este capítulo se describen las pruebas realizadas sobre el compresor y descompresor de señales *EEG*. En la sección 4.2 se detallan los repositorios de señales que se utilizaron para hacer las pruebas. En la sección 4.3 se muestra el resultado obtenido en cada uno de los experimentos y se derivan conclusiones de los mismos. El apéndice C incluye una especificación detallada de cada experimento. El programa implementado manipula señales en formato edf, quedando abierta la extensión a otros formatos. La compresión de las señales fue realizada para un solo canal.

4.2. Base de datos de señales EEG

Las bases de datos con las que realizamos las pruebas fueron las siguientes:

- *BD1*: Señales *EEG* muestreadas a 100Hz de personas con problemas de sueño (`http://www.physionet.org/physiobank/database/sleep-edf/`): Esta base de datos contiene archivos con dos canales de señales *EEG*, entre otros datos del experimento del paciente. Dentro de estos archivos, los nombrados como sc*, pertenecen a personas sanas grabadas durante 24 horas en su vida diaria normal. Los nombrados como st*, pertenecen a personas sanas que tenían problemas para conciliar el sueño, las cuales fueron grabadas 12 horas en el hospital. Para los experimentos, se tomaron las señales de *EEG* de cada uno de los archivos de la base de datos. Estos archivos fueron convertidos en archivos .edf para poder ser comprimidos por nuestro codificador.
- *BD2*: Señales *EEG* muestreadas a 160Hz de personas realizando movimientos y luego imaginando los mismos (`http://www.physionet.org/pn4/eegmmidb/`): Esta base de datos contiene más de 1500 registros de señales *EEG* de dos minutos de duración, de personas realizando diferentes tareas motoras e imaginarias. Los archivos proporcionados están en formato *EDF+*. Para los experimentos, se tomó al azar un solo canal de algunos de los archivos *EEG* proporcionados por ésta base de datos, los cuales fueron convertidos en archivos *EDF*.
- *BD3*: Señales *EEG* muestreadas a 1000Hz de personas reconociendo imágenes (`http://sccn.ucsd.edu/~arno/fam2data/data/`): Esta base de datos contiene un conjunto de archivos registrados para 14 pacientes, cuya tarea fue el reconocimiento de fotografías naturales presentadas brevemente. Los

archivos proporcionados están en formato *CNT*. Para los experimentos, se tomó al azar un sólo canal de algunos de los archivos *EEG* proporcionados por ésta base de datos, los cuales fueron convertidos en archivos *EDF*.

La conversión a *EDF* de todos los archivos de diferentes formatos, fue realizada con la herramienta *EEGLAB* [28]. Dicha herramienta de código abierto para *Matlab*, proporciona la manipulación de señales *EEG* entre otros formatos.

4.3. Análisis de Resultados

En esta sección se describen y discuten los resultados obtenidos en los experimentos realizados.
Un experimento interesante realizado fue observar si las distribuciones empíricas del error de predicción luego de aplicar *BC* (valor absoluto) se ajustaban a distribuciones geométricas en cada contexto, de forma de poder utilizar una codificación de *Golomb* en lugar de *Huffman*. Para el estudio de las estadísticas de error se tomaron señales representativas de alguna de las bases de datos mencionadas en la sección 4.2 y se recolectaron los errores de predicción en cada contexto 2.3. A partir de dichos errores se calcularon las distribuciones empíricas por contexto para visualizar si las mismas se ajustaban a una distribución geométrica. Los parámetros utilizados para realizar la predicción fueron: $P = 6$, *AR* no adaptativo con coeficientes iniciales calculados con *Levinson-Durbin* y aplicando *BC*. En general se observó que las distribuciones empíricas del error de predicción en valor absoluto, se pueden ajustar a una distribución geométrica de parámetro $\theta = 0.1$ y $\theta = 0.8$ (dependiendo del tipo de señal). Dado que los códigos de *Golomb* son óptimos cuando los valores a codificar se aproximan a una distribución geométrica, consideramos la posibilidad de utilizar códigos de *Golomb* en lugar de *Huffman*. Alguno de los resultados obtenidos se muestran en las figuras 4.3.1 y 4.3.2.

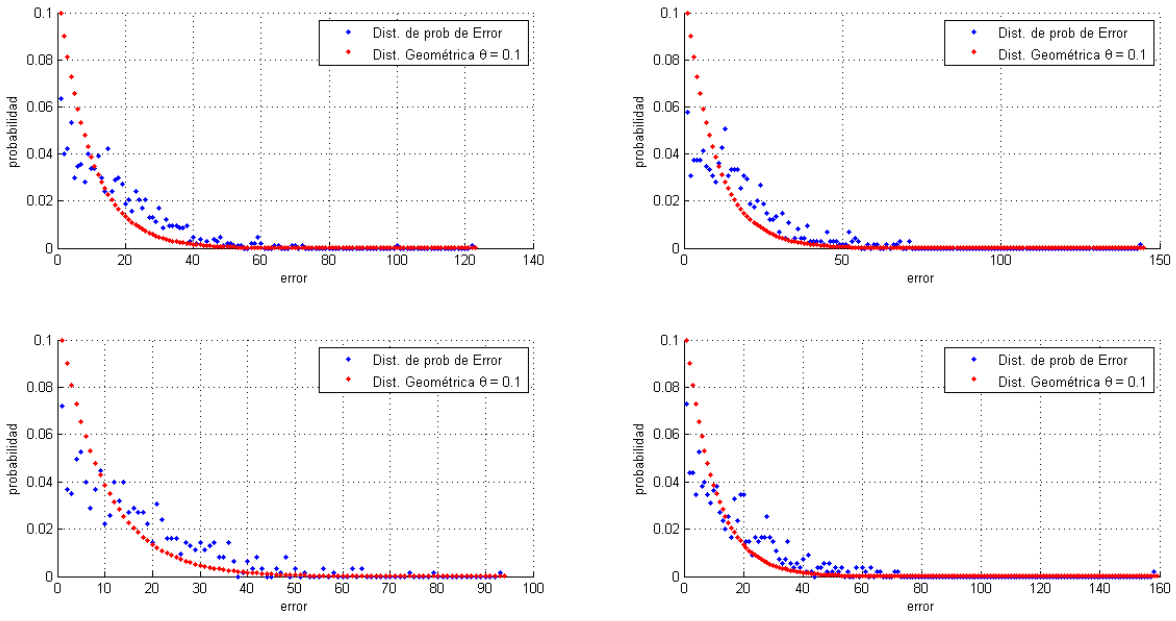


Figura 4.3.1: Distribución de probabilidad del error de predicción vs distribución geométrica de parámetro $\theta = 0{,}1$ en cuatro contextos distintos para una señal de la base de datos *BD2*.

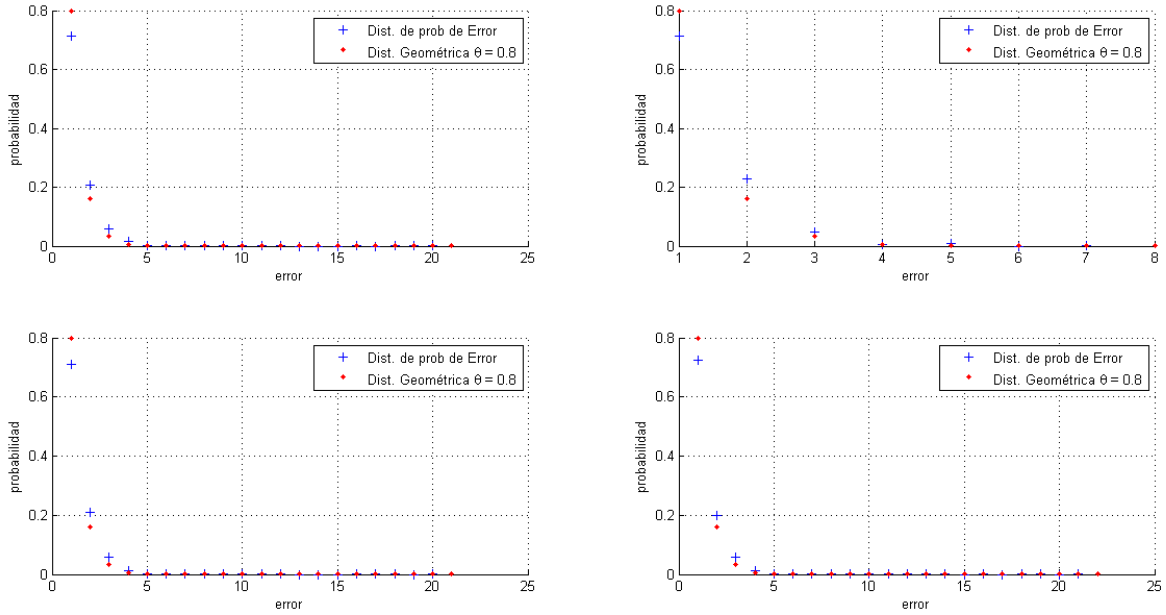


Figura 4.3.2: Distribución de probabilidad del error de predicción vs distribución geométrica de parámetro $\theta = 0{,}8$ en cuatro contextos distintos para una señal de la base de datos *BD3*.

Otro experimento interesante realizado en el desarrollo del proyecto fue estimar el impacto sobre el largo de código al utilizar códigos de *Golomb*, en lugar de *Huffman* como se propone originalmente en [1].
Para estimar el largo del código de *Huffman* de forma teórica se utilizó la fórmula (4.3.1):

$$largoHuffman = \sum_{c=1}^{c=N} d(c)\,\lceil e(c)\rceil + t(c)\log_2(d(c)), \tag{4.3.1}$$

donde $d(c)$, $e(c)$ y $t(c)$ son la cantidad de muestras, la entropía empírica y el tamaño del alfabeto (cantidad de símbolos distintos del alfabeto) en el contexto c respectivamente. Los contextos c fueron calculados como se mencionó en la sección 2.3.
La entropía empírica se calcula a partir de la probabilidad empírica de los errores de predicción:

$$e(c) = -\sum_{er=-E}^{er=E} p(er)\log_2(p(er)),\ p(er) > 0, \tag{4.3.2}$$

donde $p(er)$ es la probabilidad empírica calculada como $p(er) = \#ocurr_c(er)/max\#ocurr(c)$, siendo $\#ocurr_c(er)$ la cantidad de ocurrencias del error er en el contexto c, $max\#ocurr(c)$ la cantidad de ocurrencias totales de errores en el contexto c y E es la cantidad posible de errores sin signo.

Para el cálculo del largo de código de *Golomb* se utilizó el tamaño del archivo final codificado sin contar el cabezal detallado en el anexo B.
El experimento consistió en aplicarle el algoritmo de compresión a 7 señales de diferentes frecuencias y recolectar la información necesaria para calcular el largo de código de *Huffman*. Los parámetros utilizados

en el algoritmo fueron: $P = 6$, *AR no adaptativo* con coeficientes iniciales calculados con *Levinson-Durbin*, $N = 4$ y sin aplicar *BC*. Los resultados obtenidos se muestras en el cuadro 4.1.

	Algoritmo codificación	
Señal	**Huffman (KB)**	**Golomb (KB)**
1	6,215.79	6,020
2	5,176.55	4,850
3	4,953.07	4,686
4	5,470.66	5,206
5	4,710.52	4,448
6	6,362.82	5,999
7	2,110.36	2,555

(a) Señales de *BD1*.

	Algoritmo codificación	
Señal	**Huffman (KB)**	**Golomb (KB)**
1	17.85	15
2	15.88	14
3	17.95	15
4	14.94	13
5	14.85	13
6	14.89	13
7	15.71	14

(b) Señales de *BD2*.

	Algoritmo codificación	
Señal	**Huffman (KB)**	**Golomb (KB)**
1	83.32	79
2	81.81	72
3	69.73	62
4	70.07	62
5	55.91	60
6	57.12	60
7	57.16	60

(c) Señales de *BD3*.

Cuadro 4.1: Largo de código de *Huffman* vs *Golomb* para señales de las bases de datos *BD1*, *BD2*, *BD3*.

Como se puede observar en el cuadro 4.1 en general los largos de código de *Golomb* son inferiores a los largos de código de *Huffman*. Por este motivo y considerando que el código de *Golomb* es menos complejo que el código *Huffman*, se optó por utilizar en la etapa de codificación un código de *Golomb* en lugar de un código de *Huffman*.

El objetivo de los demás experimentos realizados, y cuyos resultados se detallan a continuación, fue probar diferentes variantes en los parámetros del algoritmo implementado y evaluar su comportamiento con respecto a la compresión de las señales. Ademas discutir si es posible utilizar el algoritmo implementado en contextos *online* y *offline*, considerando que en un contexto *online* la compresión de la señal *EEG* se realiza en tiempo real, donde desde el momento que se obtiene cada muestra hasta que se codifica la misma no se conoce la señal en su totalidad. En un contexto *offline* la compresión se realiza conociendo la totalidad de la señal.
Para los experimentos fueron seleccionadas 7 señales de *EEG* de diferente tipo de actividad y frecuencia, cuya procedencia fue descrita en la sección 4.2. Se consideró la fórmula (4.3.3) para el cálculo de la tasa de bits por muestra (*tbm*):

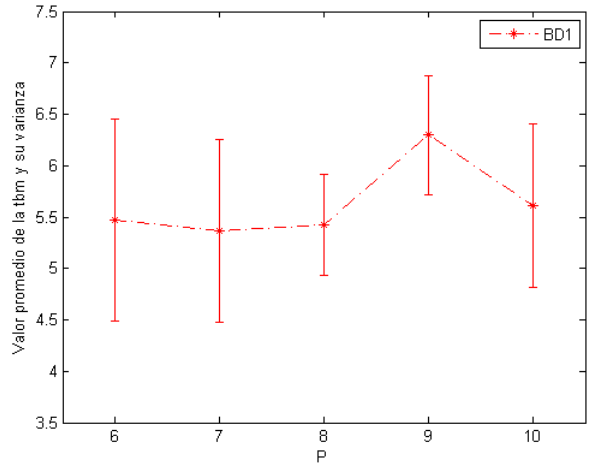
$$tbm = \frac{T_c\,(bit)}{d}, \tag{4.3.3}$$

donde T_c es el tamaño del archivo comprimido en bits y d representa la cantidad de muestras que contiene la señal a comprimir.

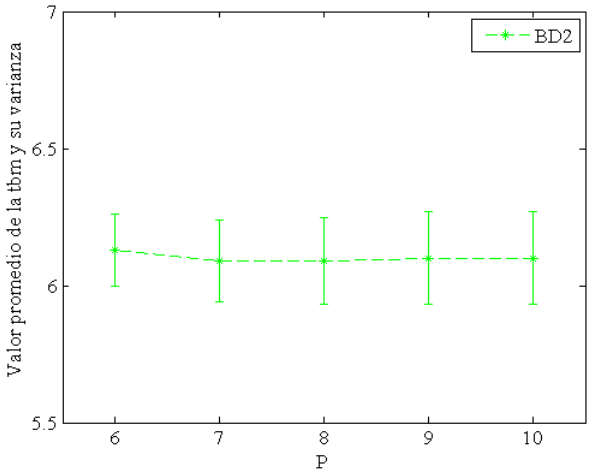
La primera variación realizada al algoritmo de compresión fue el orden del *modelo AR* (P). El experimento (C.1 del apéndice C) consistió en determinar con qué orden del *modelo AR adaptativo* se obtiene una mayor compresión de las señales *EEG*. Los valores evaluados fueron de $P = 6$ a $P = 10$. Para cada tipo de señal se calculó el promedio de la *tbm* de las señales comprimidas con los diferentes valores de P. Las varianzas calculadas para dichos promedios indican que no es significativa ninguna cifra después de la coma en los valores de compresión obtenidos. Por lo tanto la diferencia entre comprimir las señales con $P = 6$ y con P mayores a 6 es menor a un bit por muestra. No es posible concluir que algún valor de P sea mejor que otro, pero sí se puede decir que no se observaron mejoras significativas al comprimir las señales con un orden de *AR* mayor a 6.
En el cuadro 4.2 y figura 4.3.3 se indica para cada juego de datos y valor de P, cuál fue el promedio de la *tbm* (Prom.) junto con la varianza (Var.) obtenida.

P	**6**		**7**		**8**		**9**		**10**	
Base	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**
BD1	5.47	0.98	5.37	0.89	5.42	0.49	6.30	0.58	5.61	0.80
BD2	6.13	0.13	6.09	0.15	6.09	0.16	6.10	0.17	6.10	0.17
BD3	2.58	0.03	2.58	0.04	2.56	0.03	2.55	0.03	2.54	0.03

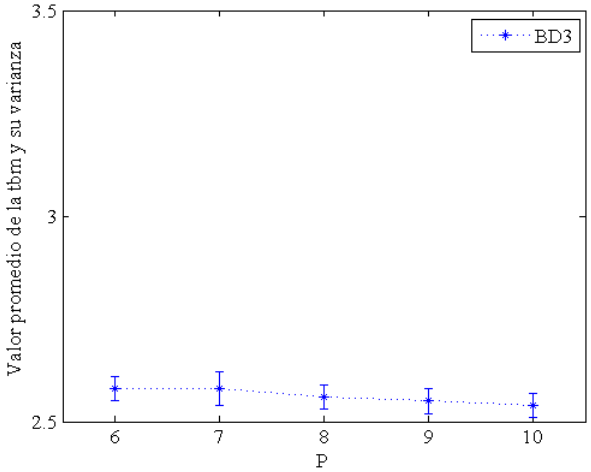
Cuadro 4.2: Promedio y varianza de la *tbm* de los distintos tipos de señales, comprimidos con los diferentes valores de P.



(a) Señales de *BD1*.



(b) Señales de *BD2*.



(c) Señales de *BD3*.

Figura 4.3.3: Promedio y varianza de la *tbm* para los diferentes tipos de señales comprimidas variando el valor de *P*. Donde cada segmento representa el intervalo de confianza del promedio ubicado en el punto medio del mismo.

Para los casos en que se utiliza el predictor *AR adaptativo* se debe determinar el valor del parámetro n , factor que determina η, quien controla la convergencia del algoritmo *LMS*. A partir de los experimentos (C.2 y C.3 del apéndice C) realizados se determinó que para lograr la convergencia del algoritmo *AR adaptativo,* se debe utilizar un valor de n menor a 10^{-4}.
Se observó que para las señales de *BD1* se obtuvo un menor error de predicción en promedio con $n = 10^{-5}$que con $n = 10^{-4}$. Aunque este comportamiento es razonable considerando la naturaleza sumamente variable de este tipo de señales (contiene cambios bruscos cuando las personas pasan de estar dormidas a despiertas), esta disminución del n no mejora el error de predicción promedio en los demás tipos de señales analizados. Por lo tanto se considera que el valor $n = 10^{-4}$ es un valor razonable para utilizar en todos los casos.

Otra de las variantes que fue evaluada para el algoritmo es la estimación de las muestras a comprimir con diferentes tipos de predictores y coeficientes iniciales. El objetivo del experimento (C.4 del apéndice C) fue determinar qué combinación entre el tipo de predictor (*AR adaptivo o no adaptivo*) y coeficientes iniciales es más eficiente en cada contexto (*online* / *offline*).
Las diferentes combinaciones entre tipos de predictores y coeficientes iniciales analizadas fueron:

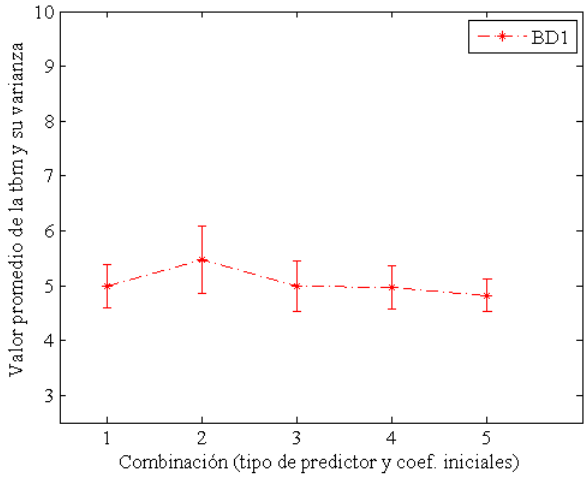
1. *AR adaptativo* con coeficientes iniciales calculados con *Levinson-Durbin*.
2. *AR adaptativo* con coeficientes iniciales iguales a 0.
3. *AR adaptativo* con coeficientes iniciales iguales al promedio de los coeficientes finales obtenidos de un conjunto de señales de entrenamiento.
4. *AR no adaptativo* con coeficientes calculados con *Levinson-Durbin*.
5. *AR no adaptativo* con coeficientes iniciales iguales al promedio de los coeficientes finales obtenidos de un conjunto de señales de entrenamiento.

Para cada tipo de señal, se calculó el promedio de la *tbm* de las señales comprimidas con las diferentes combinaciones mencionadas.
En principio, el algoritmo no se podría utilizar en modo *online* con ninguna de las combinaciones mencionadas anteriormente, pues el cálculo de los umbrales de codificación definido en la sección 3.1.11 requiere conocer toda la señal, al igual que el cálculo del parámetro η definido en la sección 3.1.1, cuando se utiliza *AR adaptativo.* Sin embargo, si los umbrales y el parámetro η son predefinidos en base a señales de entrenamiento (u otro método) de forma de evitar recorrer la señal más de una vez, las combinaciones 2, 3 y 5 se podrían utilizar en contextos *online*. Además se observó que la combinación 2 para los diferentes tipos de señales comprimidas, presenta un promedio de *tbm* y varianza mayor que todas las demás combinaciones. Este comportamiento se debe a que al comenzar con coeficientes iniciales en 0 el modelo *AR adaptativo* demora más tiempo en adaptarse, esto implica que el error de predicción en el comienzo de la compresión es grande y por ende hace que aumente el tamaño de la codificación.
Una desventaja de utilizar las combinaciones 1 y 4 es que el algoritmo no se podría utilizar para contextos *online*, pues el cálculo de los coeficientes iniciales con *Levinson-Durbin* requiere recorrer inicialmente toda la señal. Por lo tanto, desde el punto de vista de una aplicación en tiempo real del algoritmo, las combinaciones 3 y 5 tienen la ventaja de que no necesitan conocer toda la señal como sí lo necesitan las combinaciones 1 y 4. Entre ellas, la combinación 3 en general produce una mayor compresión que la 5. La combinación 4 es la que da mejores resultados de compresión en la mayoría de los casos, pero no es posible utilizarla en contextos *online* como se mencionó anteriormente.

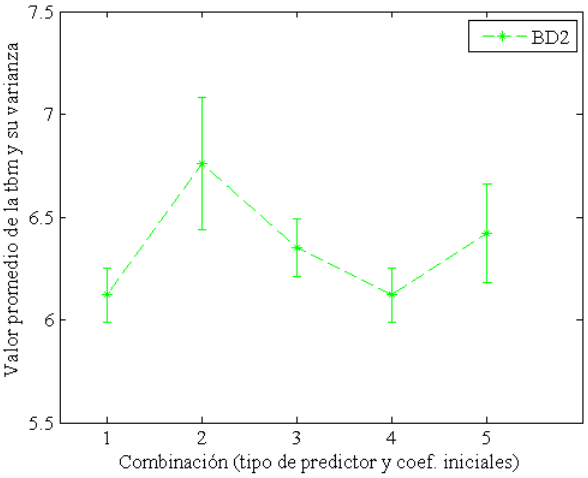
En el cuadro 4.3 y figura 4.3.4 se indica para cada combinación y juego de datos, cuál fue el promedio de la *tbm* (Prom.) junto con la varianza (Var.) obtenida.

Combinación	**1**		**2**		**3**		**4**		**5**	
Base	***tbm***	**Var**	***tbm***	**Var**	***tbm***	**Var**	***tbm***	**Var**	***tbm***	**Var**
BD1	4.99	0.39	5.47	0.61	4.98	0.46	4.97	0.40	4.82	0.29
BD2	6.12	0.13	6.76	0.32	6.35	0.14	6.12	0.13	6.42	0.24
BD3	2.54	0.03	7.14	2.11	3.67	0.64	2.54	0.03	4.36	0.88

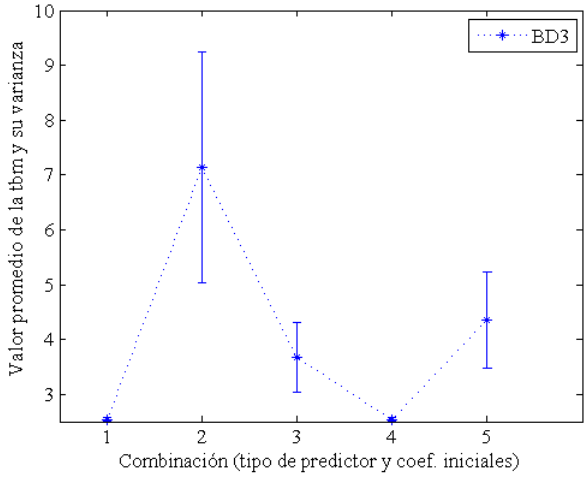
Cuadro 4.3: Promedio y varianza de la *tbm* de los distintos tipos de señales, comprimidos con los diferentes tipos de predictores y coeficientes iniciales.



(a) Señales de *BD1*.



(b) Señales de *BD2*.



(c) Señales de *BD3*.

Figura 4.3.4: Promedio y varianza de la *tbm* para los diferentes tipos de señales comprimidas variando los tipos de predictores y coeficientes iniciales mencionados en la sección 4.3. Donde cada segmento representa el intervalo de confianza del promedio ubicado en el punto medio del mismo.

Otra de las variantes que fue evaluada para el algoritmo, es si aplicar o no *BC*. El objetivo de los experimentos (C.5 y C.6 del apéndice C) fue analizar si aplicando *BC* mejora la compresión. Para ello en primer lugar se analizó si el promedio de los errores de predicción utilizando *BC* se aproxima más a 0 que el promedio de los errores de predicción sin aplicar *BC*. En segundo lugar se calculó para cada tipo de señal, el promedio de la *tbm* de las señales comprimidas aplicando y no *BC*. Se observó que si se aplica *BC* en general tanto el promedio como la varianza de errores es menor que si no se aplica *BC,* sin embargo la diferencia no es significativa y no se obtienen mejores resultados en la compresión. Por lo tanto aplicar *BC* aumenta el costo computacional del algoritmo y no mejora la compresión para las señales analizadas.
En el cuadro 4.4 se muestran los resultados obtenidos de los promedios y varianzas de los errores de predicción sin utilizar *BC* (Prom. y Var. de e_j) y utilizando *BC* (Prom. y Var. de $\tilde{e_j}$).

Errores	e_j		$\tilde{e_j}$	
Señal	**Prom.**	**Var.**	**Prom.**	**Var.**
1	-0.131	237.103	-0.121	232.435
2	0.115	38.750	0.114	37.418
3	0.107	33.458	0.053	33.183
4	-0.128	88.308	-0.105	85.668
5	0.024	25.728	0.002	25.521
6	-0.048	363.492	-0.172	353.307
7	-6.245	370.307	-4.660	308.276

(a) Señales de *BD1*.

Errores	e_j		$\tilde{e_j}$	
Señal	**Prom.**	**Var.**	**Prom.**	**Var.**
1	-0.088	442.381	-0.056	441.260
2	0.032	137.174	0.011	136.616
3	-0.316	106.953	-0.304	107.070
4	-0.211	112.574	-0.153	111.880
5	-0.744	259.723	-0.574	259.530
6	0.195	175.840	0.096	176.473
7	0.175	120.935	0.105	121.232

(b) Señales de *BD2*.

Errores	e_j		$\tilde{e_j}$	
Señal	**Prom.**	**Var.**	**Prom.**	**Var.**
1	0.1768	2.5863	0.1768	2.5866
2	-0.1270	2.0207	-0.1270	2.0208
3	0.0113	1.1932	0.0112	1.1933
4	-0.0525	1.7234	-0.0523	1.7231
5	0.0215	1.2108	0.0222	1.2095
6	0.0290	0.8265	0.0290	0.8266
7	0.0039	0.8226	0.0039	0.8228

(c) Señales de *BD3*.

Cuadro 4.4: Promedios y varianzas de los errores de predicción sin utilizar *BC* (Prom. y Var. e_j) y utilizando *BC* (Prom. y Var. $\tilde{e_j}$) para los juegos de datos *BD1*, *BD2* y *BD3*.

En el cuadro 4.5 se indica para cada juego de datos cuál fue el promedio de *tbm* (Prom.) y varianza (Var.) si se aplica o no *BC*.

BC	**Sin BC**		**Con BC**	
Base	**Prom.**	**Var.**	**Prom.**	**Var.**
BD1	5.156	0.477	5.170	0.649
BD2	6.182	0.135	6.181	0.136
BD3	2.541	0.030	2.541	0.030

Cuadro 4.5: Promedio y varianza de la *tbm* de los distintos tipos de señales, comprimidas con y sin *BC*.

La última variante evaluada para el algoritmo es la cantidad de contextos N utilizados en la codificación. El objetivo de los experimentos (C.7 y C.8 del apéndice C) fue analizar con qué cantidad de contextos se obtiene una mayor compresión utilizando codificación condicional. En primer lugar se analizó si utilizando 8 contextos ($N = 8$) para la codificación condicional mejora la compresión. Para dicho análisis se calculó para cada tipo de señal el promedio del parámetro k de *Golomb*, de las señales comprimidas en los diferentes contextos. Dado que las señales utilizadas para los experimentos contienen entre si información diferente, no es posible concluir en conjunto si con 8 contextos se obtiene una mejor codificación condicional para todas las señales.

Para las señales de *BD1* se observa que el parámetro k de *Golomb* varía entre un contexto y otro, o sea que existe una discriminación entre las distintas distribuciones de error, por lo tanto utilizar contextos de codificación realmente ayuda a reducir el costo de la compresión. Para las señales de *BD2* se observó que la variación de k entre un contexto y otro no es significativa, por lo que no sería ideal contar con 8 contextos de codificación en estos casos. Por último se observó que para las señales de *BD3* la utilización de 8 contextos de codificación no mejora la compresión.

En el cuadro 4.6 se indica para cada contexto y las bases de datos *BD1* y *BD2*, cual fue el promedio del orden k de *Golomb* en cada uno de los contextos de codificación. En el caso de la *BD3*, el orden de *Golomb* promedio es 1 con varianza prácticamente nula en todos los contextos.

Contexto	1		2		3		4		5		6		7		8	
Señal	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.
1	2.83	0.14	3.00	$2x10^{-4}$	3.00	10^{-4}	3.55	0.247	4.00	10^{-4}	4.00	$2x10^{-5}$	4.00	10^{-5}	5.00	10^{-4}
2	1.00	0	1.33	0.221	2.00	0.001	2.00	0.001	2.00	10^{-4}	2.00	0.001	3.00	10^{-4}	3.00	10^{-5}
3	1.00	0.001	1.03	0.027	1.18	0.148	1.60	0.239	2.00	$2x10^{-6}$	2.00	$4x10^{-6}$	2.92	0.073	3.00	$2x10^{-5}$
4	1.11	0.094	2.00	10^{-5}	2.01	0.006	2.33	0.223	3.00	0.0002	3.00	10^{-5}	3.82	0.149	4.00	$3x10^{-5}$
5	1.00	10^{-4}	1.00	10^{-4}	1.00	$3x10^{-4}$	1.00	$4x10^{-4}$	1.00	0.005	2.00	$3x10^{-5}$	2.00	$2x10^{-4}$	3.00	$2x10^{-5}$
6	2.35	0.228	3.00	10^{-4}	3.01	0.006	3.88	0.108	4.00	10^{-5}	4.00	$2x10^{-5}$	5.00	$3x10^{-5}$	5.00	$2x10^{-5}$
7	2.00	0.001	2.00	0.001	2.00	0.002	2.74	0.194	3.00	10^{-5}	0	0	3.00	0.001	4.77	0.734

(a) Señales de *BD1*

Contexto	1		2		3		4		5		6		7		8	
Señal	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.
1	4.00	0.005	4.00	0.004	4.00	0.006	4.00	0.005	4.00	0.004	4.00	0.005	4.62	0.245	5.00	0.006
2	3.00	0.002	3.00	0.005	3.00	0.006	3.00	0.004	3.00	0.002	3.15	0.130	4.00	0.007	4.00	0.010
3	3.00	0.004	3.00	0.004	3.01	0.008	3.00	0.004	3.00	0.006	3.04	0.044	3.02	0.022	3.95	0.046
4	3.01	0.010	3.00	0.003	3.01	0.011	2.99	0.009	3.00	0.003	3.00	0.008	3.00	0.004	3.72	0.208
5	4.00	0.008	3.99	0.010	4.00	0.004	4.00	0.008	3.99	0.010	4.00	0.009	4.00	0.005	4.12	0.124
6	4.00	0.004	3.97	0.033	3.99	0.015	3.98	0.023	4.00	0.004	3.97	0.031	4.00	0.004	4.00	0.005
7	3.00	0.006	3.01	0.011	3.09	0.082	3.02	0.024	3.01	0.018	3.00	0.002	3.04	0.047	3.99	0.022

(b) Señales de *BD2*

Cuadro 4.6: Promedio y varianza del orden *k* de *Golomb* de los distintos tipos de señales en cada uno de los contextos de codificación, para las bases de datos *BD1* y *BD2*.

Luego se analizó con qué cantidad de contextos de codificación se obtiene una mayor compresión. Los valores de contextos evaluados fueron $N = 1, 2, 3, 4, 6, 8, 16$ *y* 32 . Para cada tipo de señal se calculó el promedio de la *tbm* de las señales comprimidas con los diferentes valores de *N*. En general se observó que aumentando la cantidad de contextos de codificación no se obtiene una mejora significativa en la compresión, pues la diferencia de las *tbm* es menor a un bit en en casi todos los casos tomando en cuenta la varianza calculada.

Como se mencionó anteriormente, por la diversidad de las señales utilizadas, no es posible afirmar con qué cantidad de contextos se obtiene mejor compresión. Sin embargo buscando un punto intermedio al utilizar la codificación condicional, se observó que con 4 contextos de codificación (tomando en cuenta el promedio de *tbm* y su varianza) se obtienen resultados similares en cuanto a la compresión y computacionalmente es menos costoso que utilizar más de 8 contextos.

Por otra parte, para las señales de *BD3* no se logra una mejora significativa en la compresión para ningún valor de *N*.

En el cuadro 4.7 se indica para cada valor de *N* y juego de datos, cuál fue el promedio y varianza de *tbm* .

N	1		2		3		4		8		16		32	
Base	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**
BD1	5.22	0.50	5.10	0.44	5.05	0.39	5.05	0.40	5.03	0.40	5.03	0.40	5.02	0.40
BD2	6.12	0.12	6.11	0.12	6.11	0.13	6.11	0.13	6.12	0.13	6.16	0.13	6.22	0.13
BD3	2.54	0.03	2.54	0.03	2.54	0.03	2.54	0.03	2.54	0.03	2.54	0.03	2.55	0.03

Cuadro 4.7: Promedio y varianza de la *tbm* de los distintos tipos de señales, comprimidas con los diferentes valores de *N*.

Por último se realizó una comparación entre el algoritmo implementado y el compresor *Zip* [2], para determinar con qué algoritmo se obtiene una mayor compresión. En este caso se calculó el promedio de las tasas de compresión de cada tipo de señal para el algoritmo implementado y para el compresor *Zip*. Se consideró la fórmula (4.3.4) para el cálculo de la tasa de compresión (TC):

$$TC = \frac{TAC}{TAO}, \tag{4.3.4}$$

donde TAC es el tamaño en bits del archivo comprimido y TAO es el tamaño en bits del archivo original.

Se observó que para todas las señales consideradas en el experimento (C.9 del apéndice C) se obtiene una menor TC, y por lo tanto mayor compresión, con el algoritmo implementado que con el compresor *Zip*. Además se observó que para las señales de *BD1* se obtiene un mayor porcentaje de compresión que para las señales de *BD2*. Esto se debe a que las señales de *BD1* contienen varios episodios donde la persona está durmiendo, con lo cual no existe mucha variación en la señal aumentando la compresión de la misma.
En el cuadro 4.8 se indica para cada juego de datos, cuál fue el promedio de la TC para el algoritmo implementado (Prom.TC) y el promedio de la TC para el compresor *Zip* (Prom.TC Zip). Además se muestra el porcentaje de compresión para ambos casos (% Comp.):

Base	**Prom. TC**	**%Comp.**	**Prom. TC Zip**	**% Comp.**
BD1	0.32	68.41	0.44	55.65
BD2	0.38	62.31	0.57	42.55
BD3	0.16	84.14	0.31	69.32

Cuadro 4.8: Promedio de la *TC* y porcentaje de compresión para todos los tipos de señales comprimidas con el algoritmo implementado y con el compresor *Zip*.

Otra de las observaciones que se pueden mencionar luego de analizar los resultados de todos los experimentos, es que con las señales de *BD3* en general se obtiene una mayor compresión en comparación con las demás, el análisis de esto queda fuera del alcance de este trabajo.

Capítulo 5

Conclusiones y trabajo a futuro

5.1. Conclusiones

La motivación principal de este proyecto fue analizar y evaluar diferentes métodos de compresión de señales de *EEG*. De los trabajos estudiados, fue seleccionado un artículo para realizar un prototipo, el cual nos permitió evaluar el funcionamiento de las técnicas utilizadas por el mismo. A medida que se analizaron cada una de las etapas del algoritmo seleccionado, se agregaron y evaluaron variantes sobre las mismas, las cuales conllevaron a tomar decisiones sobre lo que se dejó expresamente abierto en [1] y en la utilización de otras técnicas que aprovechan mejor las características de las señales. A partir de este análisis, se detectó que en general los métodos de compresión utilizan 3 etapas principales (transformación, cuantificación y codificación).
En lo que respecta a las técnicas analizadas en este proyecto y considerando las señales analizadas es importante destacar:

- La técnica de predicción *AR* utilizada permite obtener una buena estimación de la muestra original si se considera una buena elección de los coeficientes iniciales, dado que los mismos influyen de forma relevante en el desempeño del predictor. No obstante, el método de adaptación de coeficientes que utilizamos (*LMS*) no disminuye significativamente el error de predicción y consecuentemente no aumenta la compresión.

- La técnica de *cancelación de sesgo* no presentó mejoras relevantes en la reducción del error de predicción. En consecuencia el uso de dicha técnica no presenta mejoras en la compresión de *EEG* y aumenta el costo computacional del algoritmo.

- La codificación condicional, según [1], logra una discriminación entre las distintas distribuciones de errores por contexto, aumentando la compresión. Sin embargo, los resultados de incorporar dicha técnica en nuestras pruebas no proporcionaron ventajas significativas al respecto.

Luego de analizar los resultados experimentales, considerando las variantes sobre el algoritmo implementado, se obtuvieron resultados alentadores en cuanto a la compresión del algoritmo sobre las señales utilizadas. Por otra parte, el algoritmo presenta mejores resultados de compresión que el compresor *Zip*.
El algoritmo de compresión fue implementado de forma modular, donde cada módulo se corresponde con cada una de las etapas de compresión definidas. Esto permite sustituir fácilmente la etapa de transformación

y codificación por otros métodos y de esta manera poder evaluar otras alternativas del algoritmo.

Surgieron varias dificultades a lo largo del proyecto. Entre ellas se destaca la dificultad de encontrar bases de datos de señales *EEG* de personas realizando diferentes actividades a frecuencias de muestreo distintas y la falta de experiencia con la manipulación de señales, causando que la duración del proyecto se prolongara más tiempo de lo esperado.
A pesar de las dificultades que se fueron presentado, el proyecto nos pareció muy interesante pues es un área de investigación que contribuye en la manipulación de las señales *EEG*, mejorando entre otras cosas, la calidad y facilidad de los estudios médicos de los pacientes.

Esperamos que tanto la investigación como los resultados de este proyecto formen parte de un antecedente para futuros proyectos de compresión de *EEG.*
En este sentido, identificamos algunas líneas posibles de trabajo a futuro que detallamos a continuación:

- Calcular los coeficientes del modelo *AR adaptativo* en forma batch, de manera de adaptar los mismos en segmentos de EEG.
- Entrenar los umbrales con varias señales y luego dejarlos fijos, de modo de evitar recorrer nuevamente la señal para el cálculo de los mismos.
- Incluir otros formatos de señales como entrada para poder comprimirlos (edf+, .rec,.mat).
- Lograr que el compresor se pueda utilizar de forma *online*, es decir que solo con una pasada por la señal original se pueda comprimir y descomprimir la señal.
- Analizar en los casos que se utilice *AR adaptativo* si el valor de η del método *LMS* puede ser calculado independientemente de la varianza de toda la señal a comprimir, para poder ser utilizado en un contexto *online*.
- Analizar por qué la técnica de *BC* no mejora el error de predicción.

Apéndice A

Estado del Arte

A continuación se presenta un resumen del estado del arte en lo que a trabajos de compresión de señales EEG se refiere.

Descripción de los métodos de compresión de señales EEG

Lossless Multi-channel EEG Compression

En éste trabajo [12] se expone un método de compresión sin pérdida de señales EEG multi-canal. Dónde se explota la redundancia entre los canales adyacentes de señales de *EEG*, utilizando la transformación *IntKLT* (cuantificar los coeficientes obtenidos por la transformada *Karhunen-Loeve (KLT)* [11]). El método de compresión se realiza en cuatro etapas. En la primera etapa se aprovecha la correlación entre las muestras para trasmitir menor cantidad de bit, ya que lo que se trasmite es la diferencia entre canales adyacentes. En la segunda etapa se cuantifican los coeficientes obtenidos por *KLT*, obteniendo una compresión sin pérdida de la señal. En la tercera etapa se realiza una retroalimentación de los residuos de cada salida. En la cuarta etapa se realiza la codificación de la entropía mediante *Huffman* [21]. El estudio e implementación de éste método es interesante, ya que para la comprensión se aprovecha la correlación entre canales pero tiene un costo computacional importante en la implementación de la transformada *IntKLT*.

An accuracy aware low power wireless EEG

En éste trabajo [7] se expone un método de compresión y adaptación de señales *EEG* multi-canal para ser trasmitidos de forma inalámbrica con un bajo consumo energético. El método analiza el contenido de las señales de *EEG* de forma independiente y luego lleva a cabo la compresión adaptativa. La compresión adaptativa se refiere a que dependiendo del tipo de información que se vaya procesando, se determina el contenido de la misma y se ve que tipo de compresión se realiza. Si es una señal que contiene mucha información se comprimirá menos que las que no aportan mucha información. Si un pico no se detecta dentro de un marco o marcos vecinos, los datos se comprimen y, eventualmente se transmitirán. Cuando un pico se detecta en el marco (o marcos vecinos) los datos se transmiten directamente, produciendo un resultado de alta calidad.

Algoritmo basado en wavelet Packets para la compresión de señales EEG

En éste trabajo [9] se expone un método de compresión de señales *EEG* que se basa en la *Transformada Wavelets Packets* (WPT)[29]. La señal *EEG* se descompone mediante la WPT y los coeficientes obtenidos se someten a un proceso de comparación con umbrales, aquellos cuyos valores absolutos resultan menores que el umbral son igualados a cero, preservándose los demás. Los coeficientes distintos de cero son sometidos a una *cuantificación escalar uniforme* [30] y luego codificados mediante el algoritmo *Run-length*[22]. El método propuesto tiene un bajo costo computacional y es fácil de programar.

Compresión de señales electroencefalográficas epilépticas y normales

En este trabajo [15] se analiza el comportamiento de la compresión de señales que contienen episodios epilépticos, comparado con otras que no lo contienen. Se establece un grupo de parámetros de calidad y de compresión para establecer la comparación. Como resultado se obtiene una mejor calidad y una mayor compresión cuando la señal contiene episodios epilépticos. Las etapas son, descomposición de la señal, cuantificación y codificación sin pérdida. El método está enfocado a la epilepsia y obtiene mejores resultados cuando la señal contiene episodios epilépticos.

Compressed Sensing Framework for EEG Compression

En este trabajo [31] no se realiza una compresión de la señal, sino que se basa en utilizar diccionarios de *Gabor* [32], los cuales toman características de la señal original y se consideran dichas características como la señal "comprimida".

A High-Performance Lossless Compression Scheme for EEG Signals Using Wavelet Transformand Neural Network Predictors

Éste trabajo [8] utiliza una combinación de dos transformaciones sobre las señales *EEG*. Primero se aplica una *Transformada Wavelet [6]* a la señal *EEG* original, luego con los coeficientes generados se entrenan los predictores de una Red Neuronal implementada. Finalmente los residuos de error son codificados usando un codificador de entropía combinacional [33]. Básicamente se generan los coeficientes generados con la *Transformada Wavelet* y los mismos se utilizan para entrenar a los predictores de las redes neuronales implementados [34]. La compresión sin pérdida está asegurada debido a que la señal de error decodificada junto con la señal predicha recupera la señal original sin perder la información de diagnóstico.

A two-dimensional approach for lossless EEG compression

Éste trabajo [16] presenta una técnica simple de pre-procesamiento, donde se dispone de la señal del *EEG* en forma de una matriz (2-D) antes de la compresión. Se utiliza un codificador de dos etapas para comprimir la matriz de *EEG*, con una etapa de codificación con pérdidas *SPIHT [35]* y una etapa residual de codificación (codificación aritmética).
En la técnica de pre-procesamiento, la señal de *EEG* se fracciona en segmentos de igual longitud, donde cada segmento es una fila de la matriz, donde las filas impares se llenan directamente, y las filas pares son llenadas en forma inversa. Además se utiliza un esquema de codificación de dos etapas, en la primera etapa de compresión con pérdida se utiliza *SPIHT* y en la segunda etapa codificación aritmética para codificar los residuos.
La idea es transformar la señal en una "imagen" o matriz 2-D y luego aplicar la codificación para comprimir la matriz.

Algoritmo de Compresión de EEG mediante Predicciones usando el Polinomio de Interpolación de Newton con Diferencias Divididas

En este trabajo [18] se propone un algoritmo de compresión con pérdida de bajo costo computacional, basado en la Predicción mediante el uso del *Polinomio de Interpolación de Newton con Diferencias Divididas* (*PINDD*). Se utiliza la aproximación *PINDD* dado que las señales *EEG* son variables en el tiempo, las mismas no se pueden aproximar mediante una función periódica, pero es posible establecer funciones específicas para pequeños intervalos de tiempo. La idea del algoritmo es, si la señal puede ser interpolada mediante *PINDD* entonces la misma no se trasmite. Se define un *Error de Predicción* como la diferencia entre la señal original y la señal interpolada. Luego se define un *Error Permitido* y se debe cumplir que el error de predicción debe ser menor que el error permitido para interpolar. Si es mayor entonces la muestra será trasmitida pero no comprimida. Se transmitirá además un dato que señalizará las muestras que no pudieron ser interpoladas dentro del *EEG* comprimido.
El algoritmo presentado posee muy bajo costo computacional y poca perdida de información.

Lossless compression of electroencephalographic (EEG) data

En el trabajo [36] se examina una técnica de dos etapas de compresión sin pérdida utilizando la decorrelación entre los puntos de la señal *EEG* y la codificación de la entropía de la señal resultante. Para realizar la primera etapa se manejan 2 alternativas, un filtro de coeficiente fijo o un filtro de mínimos cuadrados recursivo. Para realizar la segunda etapa se utiliza codificación aritmética para codificar los datos. Para descomprimir se utiliza inversamente el filtro utilizado en la compresión.

Performance Evaluation of Neural Network and Linear Predictors for Near-Lossless Compression of EEG Signals

En este artículo [20] se realiza una comparación de la performance de compresión casi sin pérdida de señales *EEG* obtenida utilizando redes neuronales y los predictores lineales. Se utilizan tres predictores de redes neuronales, de una sola capa de perceptrón (*SLP*) [37], perceptrón multicapa (*MLP*)[38], y la red de Elman (*ES*) [39]. También se utilizan dos predictores lineales, el modelo autorregresivo (*AR*) donde para estimar los parámetros se usa el método de *Levinson-Durbin* [24], y filtros FIR [40]. Para todos los predictores se aplica cuantificación uniforme al error de predicción obtenido de la diferencia entre la muestra original y la predicha.
La compresión de la señal de *EEG* se realiza en dos etapas. En la primer etapa se utiliza un predictor de compresión sin perdida y en la segunda etapa un codificador de la entropía. Para recuperar la señal, el receptor suma el error de predicción a la señal predicha obteniendo de la muestra actual.
Luego de realizar una comparación entre todos los algoritmos propuestos, el predictor *SLP* obtiene el mejor resultado y se determina un ahorro de bits por muestra con respecto a los demás métodos evaluados.

A wavelet-packets based algorithm for EEG signal compression

El objetivo de este trabajo [10] es desarrollar un algoritmo eficiente para la compresión con pérdida de *EEG*. La idea del algoritmo es segmentar la señal *EEG* y descomponerla a través de paquetes wavelet (*WP*) [6]. Luego se les aplica una umbralización a los coeficientes de la descomposición de *WP* y aquellos que tienen valores absolutos por debajo de un umbral establecido son eliminados. Los coeficientes restantes están debidamente cuantificados y codificados utilizando un esquema de codificación run-length. La señal comprimida *EEG* puede ser recuperada por un proceso inverso.

El algoritmo tiene un buen equilibrio entre lo que comprime (Compression Ratio, *CR* [41]) y a calidad de la señal reconstruida (Percent Root-mean squared Difference, *PRD* [41]), además la descompresión es bastante buena, es fácilmente programable y tiene un bajo costo computacional.
Una desventaja de este algoritmo es que pierde información al realizar la compresión y existe un problema con el cálculo de los umbrales.

Context-Based Lossless and Near-Lossless Compression of EEG Signals

En este articulo [1] se propone tres etapas principales en la compresión de una señal *EEG*, codificación predictiva, contextos basados en cancelación de sesgo y codificación condicional. En cada etapa se analizan y comparan varios algoritmos existentes para resolver lo referente a dicha etapa.

Apéndice B

Formato archivo comprimido

En esta sección se proporciona una descripción general de cada una de las tres secciones principales que constituyen el formato de los archivos comprimidos como se muestra en la figura B.0.1. Las secciones están separadas por un marcador *M*, el cual indica donde comienza y finaliza cada sección del archivo comprimido.

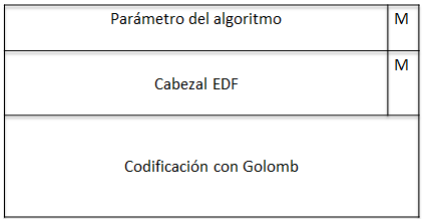


Figura B.0.1: Secciones del archivo comprimido.

En la sección "Parámetros del algoritmo" se incluyen los parámetros configurados en el codificador y además los parámetros que necesita el decodificador para decodificar la señal. En el cuadro B.1 se muestran todos los parámetros de dicha sección, los mismos están separados por un espacio.

Largo (byte)	**Parámetro**	**Descripción**
4	P	Orden del modelo *AR*
4	N	Cantidad de contextos de codificación.
8	n	Indicador de convergencia (*AR adaptativo*).
P x 8	a	Coeficientes del modelo *AR*.
P x 4	Xjs	Primeras P muestras (no codificadas).
N+1 x 8	umb	N+1 umbrales.
4	tipo Predictor	Indicador del tipo de predictor y coeficientes iniciales.
4	c/s BC	Indicador de si aplicar o no *BC*.
4	debug	Indica si se graban o no archivos con variables de interés.

Cuadro B.1: Formato de la sección "Parámetros del algoritmo" del archivo comprimido.

En la sección “Cabezal EDF” se incluye el cabezal del archivo .edf (a comprimir) sin codificar, el mismo presenta el formato de *EDF* [5].
En la sección “Codificación con Golomb” se incluye la codificación realizada para toda la señal mediante el *código de Golomb.*

Apéndice C

Diseño de Experimentos

El diseño de pruebas se centró en responder las siguientes preguntas, sobre cada parámetro configurable del algoritmo y sobre la capacidad de compresión del mismo con respecto a otro tipo de compresor.

Parámetro P (orden del modelo AR): ¿Con qué orden del modelo *AR adaptativo* se obtiene una mayor compresión utilizando como coeficientes iniciales los calculados con *Levinson-Durbin*?.

Parámetro n (convergencia AR adaptativo): ¿A partir de qué valor de *n* el algoritmo *AR adaptativo* converge?. ¿Con qué valor *n* el algoritmo *AR adaptativo* mejora su adaptación?.

Parámetro Tipo de predictor(combinación tipo de predictor y coef. iniciales): ¿Que combinación entre los tipos de predictores y coeficientes iniciales es más eficiente en cada contexto (online / offline)? Siendo las combinaciones posibles las siguientes:

1. *AR adaptativo* con coeficientes iniciales calculados con *Levinson-Durbin*.
2. *AR adaptativo* con coeficientes iniciales iguales a 0.
3. *AR adaptativo* con coeficientes iniciales iguales al promedio de los coeficientes finales obtenidos de un conjunto de señales de entrenamiento.
4. *AR no adaptativo* con coeficientes calculados con *Levinson-Durbin*.
5. *AR no adaptativo* con coeficientes iniciales iguales al promedio de los coeficientes finales obtenidos de un conjunto de señales de entrenamiento.

Parámetro BC(Indicador de si aplicar o no *BC*): ¿Disminuye el error de predicción utilizando *Bias Cancellation*? ¿Se obtiene una mayor compresión utilizando Bias Cancellation?

N(contextos de codificación): ¿Con $N = 8$ se mejora la compresión? ¿Con qué cantidad de contextos de codificación se obtiene una mayor compresión?

Zip ¿El algoritmo de compresión implementado, utilizando los parámetros óptimos seleccionados de los experimentos anteriores, obtiene una mayor compresión que el compresor *Zip* [2]?.

Algunas de las consideraciones importantes para la ejecución de los experimentos:

1. Los tipos de señales a utilizar para los siguientes experimentos son:

- *BD1: EEG* de personas con problemas de sueño (100 Hz).
- *BD2: EEG* de personas realizando movimiento y luego imaginando los mismos (160 Hz).
- *BD3*: *EEG* de personas reconociendo imágenes (1000 Hz).

C.1. Experimento 1

Para este experimento se utilizaron los siguientes parámetros:

- P = variable
- N = 8
- n = 0.0001 (el valor de *n* se utiliza para calcular el parámetro que controla la convergencia del algoritmo AR, $\eta = n/varianza(datosdelaseñal)$)
- tipo Predictor = *AR adaptativo* con coeficientes iniciales calculados con *Levinson-Durbin*.
- BC = SI (aplicar Bias Cancellation)

<table>
<tr><td>Experimento</td><td>Busca responder: ¿Con qué orden del modelo AR adaptativo se obtiene una mayor compresión utilizando como coeficientes iniciales los calculados con Levinson-Durbin?.</td></tr>
<tr><td>Objetivo</td><td>Determinar el orden del modelo AR adaptativo que mejora la compresión utilizando como coeficientes iniciales los calculados con Levinson-Durbin.</td></tr>
<tr><td>Procedimiento</td><td>1. Recolectar 7 señales de cada juego de datos:
 a) Para cada base:
  1) Comprimir las señales utilizando el predictor AR adaptativo con coeficientes iniciales calculados con Levinson-Durbin y $P = 6, 7, .., 10$.
  2) Obtener el promedio de las tasas de bits por muestra (tbm) para cada valor de P.
2. Observar como influye el parámetro P para cada tipo de señal.</td></tr>
<tr><td>Resultado</td><td>Resultados de la tbm promedio y su varianza, obtenidos para los diferentes valores de P:
<table>
<tr><th>P</th><th colspan="2">6</th><th colspan="2">7</th><th colspan="2">8</th><th colspan="2">9</th><th colspan="2">10</th></tr>
<tr><th>Base</th><th>Prom.</th><th>Var.</th><th>Prom.</th><th>Var.</th><th>Prom.</th><th>Var.</th><th>Prom.</th><th>Var.</th><th>Prom.</th><th>Var.</th></tr>
<tr><td>BD1</td><td>5.47</td><td>0.98</td><td>5.37</td><td>0.89</td><td>5.42</td><td>0.49</td><td>6.30</td><td>0.58</td><td>5.61</td><td>0.80</td></tr>
<tr><td>BD2</td><td>6.13</td><td>0.13</td><td>6.09</td><td>0.15</td><td>6.09</td><td>0.16</td><td>6.10</td><td>0.17</td><td>6.10</td><td>0.17</td></tr>
<tr><td>BD3</td><td>2.58</td><td>0.03</td><td>2.58</td><td>0.04</td><td>2.56</td><td>0.03</td><td>2.55</td><td>0.03</td><td>2.54</td><td>0.03</td></tr>
</table></td></tr>
</table>

C.2. Experimento 2

Para este experimento se utilizaron los siguientes parámetros:

- P = el seleccionado en el experimento C.1 ($P = 6$)
- N = 8
- n = 0.0001 (el valor de n se utiliza para calcular el parámetro que controla la convergencia del algoritmo AR, $\eta = n/varianza(datosdelaseñal)$)
- tipo Predictor = *AR adaptativo* con coeficientes iniciales calculados con *Levinson-Durbin*.
- BC = SI (aplicar Bias Cancellation)

Experimento	Busca responder: ¿A partir de qué valor n el algoritmo *AR adaptativo* converge?
Objetivo	Determinar el valor n para que el algoritmo *AR adaptativo* converja.
Procedimiento	1. Recolectar 7 señales de cada juego de datos: *a*) Para cada base: 1) Comprimir las señales utilizando el predictor *AR adaptativo* con coeficientes iniciales calculados con *Levinson-Durbin* y con 3 valores diferentes de $n\,(10^{-4}, 10^{-3}, 10^{-2})$. 2. Observar para qué valor de n el algoritmo converge.
Resultado	Con $n = 10^{-2}$ y $n = 10^{-3}$ el predictor *AR adaptativo* con coeficientes iniciales calculados con *Levinson-Durbin* no converge. Con valores de n menores a 10^{-3} el algoritmo converge.

C.3. Experimento 2.1

Para este experimento se utilizaron los siguientes parámetros:

- P = el seleccionado en el experimento C.1 ($P = 6$)
- N = 8
- n = 0.0001 (el valor de *n* se utiliza para calcular el parámetro que controla la convergencia del algoritmo AR, $\eta = n/varianza(datosdelaseñal)$)
- tipo Predictor = *AR adaptativo* con coeficientes iniciales calculados con *Levinson-Durbin*.
- BC = SI (aplicar Bias Cancellation)

Experimento	Busca responder: ¿Con qué valor de *n* el predictor *AR adaptativo* mejora su adaptación?
Objetivo	Determinar el valor $n\,(10^{-5}\,o\,10^{-4})$, para el cuál el predictor *AR adaptativo* obtenga una mejor estimación de la señal predicha con respecto a la señal original, por lo tanto con que *n* se obtiene menor error de predicción en promedio.
Procedimiento	1. Recolectar 7 señales de cada juego de datos: *a*) Para cada base: 1) Comprimir las señales utilizando el predictor *AR adaptativo* con coeficientes iniciales calculados con *Levinson-Durbin* y con los valores de *n* 10^{-5} y 10^{-4}. 2) Obtener para cada señal comprimida, el promedio de errores obtenidos para cada *n*. 2. Observar para qué valor de *n* el predictor *AR adaptativo* obtiene una mejor estimación de la señal (errores de predicción menores en promedio).

Resultado

Resultados de los promedios de los errores de predicción, obtenidos para los diferentes valores de n:

n	**0.00001**		**0.0001**	
Señal	**Prom.**	**Var.**	**Prom.**	**Var.**
1	0.05	222.04	0.21	727.10
2	0.23	276.06	0.16	97.28
3	-0.05	25.88	-0.05	33.40
4	-0.12	108.06	-0.02	4,090.09
5	0.03	26.31	0.0001	88.43
6	0.02	287.19	0.02	724.84
7	0.02	64.79	-0.19	181.44

(a) Señales de *BD1*.

n	**0.00001**		**0.0001**	
Señal	**Prom.**	**Var.**	**Prom.**	**Var.**
1	-0.07	445.50	-0.17	462.88
2	0.03	137.21	0.03	137.24
3	-0.32	106.94	-0.32	107.00
4	-0.20	112.66	-0.20	113.27
5	-1.25	259.80	-1.12	264.07
6	0.18	176.24	0.20	176.13
7	0.16	121.01	0.19	120.99

(b) Señales de *BD2*.

n	**0.00001**		**0.0001**	
Señal	**Prom.**	**Var.**	**Prom.**	**Var.**
1	-0.021	2.58	-0.020	2.58
2	0.020	2.01	0.009	2.03
3	0.007	1.19	-0.096	2.14
4	-0.016	1.70	-0.032	1.71
5	0.029	1.21	0.058	1.24
6	0.034	0.83	0.022	0.84
7	0.002	0.82	-0.006	0.83

(c) Señales de *BD3*.

C.4. Experimento 3

Para este experimento se utilizaron los siguientes parámetros:

- P = el seleccionado en el experimento C.1 ($P = 6$)
- N = 8
- n = el seleccionado en el experimento C.3 (en los casos que se utiliza *AR adaptativo*)
- tipo Predictor = variable
- BC = SI (aplicar Bias Cancellation)

Experimento	Busca responder: ¿Que combinación entre los tipos de predictores y coeficientes iniciales es más eficiente en cada contexto (online / offline)?
Objetivo	Determinar que combinación entre los tipos de predictores y coeficientes iniciales es más eficiente en cada contexto (online / offline).
Procedimiento	1. Recolectar 7 señales de cada juego de datos: *a*) Para cada base: 1) Comprimir cada señal utilizando cada uno de los siguientes predictores y coeficientes iniciales: a' *AR adaptativo* con coeficientes iniciales calculados con *Levinson-Durbin*. b' *AR adaptativo* con coeficientes iniciales iguales a 0. c' *AR adaptativo* con coeficientes iniciales iguales al promedio de los coeficientes finales obtenidos de un conjunto predeterminado de señales. d' *AR no adaptativo* calculados con *Levinson-Durbin*. e' *AR no adaptativo* iguales al promedio de los coeficientes finales obtenidos de un conjunto predeterminado de señales. 2) Analizar de forma cualitativa la complejidad computacional para cada predictor. 3) Analizar el cálculo de los coeficientes iniciales necesarios para ejecutar cada predictor en los contextos online / offline. 4) Obtener el promedio de las tasas de bits por muestra para cada predictor. 2. Comparar para cada tipo de señal compresión de los diferentes predictores. 3. El predictor es seleccionado considerando las características analizadas en los puntos 1a1, 1a2 y 1a3.

<table>
<tr><td>Resultado</td><td>Análisis cualitativo de la complejidad computacional para cada predictor:
<table>
<tr><th>Nro. Predictor</th><th>Procesamiento en cada señal a predecir</th></tr>
<tr><td>1a1a</td><td>1. Cálculo de los coeficientes iniciales mediante Levinson-Durbin.
2. Estimación de cada muestra a partir de las P muestras anteriores y de los coeficientes estimados en el paso anterior.
3. Cálculo de los nuevos coeficientes a partir de los coeficientes anteriores.
4. Cálculo de la varianza de las muestras para el cálculo de n utilizado en el algoritmo de adaptación.
5. Cálculo de umbrales para los contextos de la codificación condicional.
Se realizan 4 pasadas por cada señal a predecir.</td></tr>
<tr><td>1a1b</td><td>1. Estimación de cada muestra a partir de las P muestras anteriores y de los coeficientes estimados en el paso anterior.
2. Cálculo de los nuevos coeficientes a partir de los coeficientes anteriores.
3. Cálculo de la varianza de las muestras para el cálculo de n utilizado en el algoritmo de adaptación.
4. Cálculo de umbrales para los contextos de la codificación condicional.
Se realizan 3 pasada por cada señal a predecir.</td></tr>
<tr><td>1a1c</td><td>1. Cálculo de los coeficientes iniciales a partir de ciertas señales diferentes previamente seleccionadas.
2. Estimación de cada muestra a partir de las P muestras anteriores y de los coeficientes estimados en el paso anterior.
3. Cálculo de los nuevos coeficientes a partir de los coeficientes anteriores.
4. Cálculo de umbrales para los contextos de la codificación condicional.
Se realiza 3 pasada por cada señal a predecir.</td></tr>
</table></td></tr>
</table>

	1*a*1*d*	1. Cálculo de los coeficientes iniciales mediante Levinson-Durbin. 2. Estimación de cada muestra a partir de las *P* muestras anteriores y coeficientes fijos calculados inicialmente. 3. Cálculo de umbrales para los contextos de la codificación condicional. Se realizan 3 pasada por cada señal a predecir.
	1*a*1*e*	1. Cálculo de los coeficientes iniciales a partir de ciertas señales diferentes previamente seleccionadas. 2. Estimación de cada muestra a partir de las *P* muestras anteriores y coeficientes fijos calculados inicialmente. 3. Cálculo de umbrales para los contextos de la codificación condicional. Se realiza 2 pasadas por cada señal a predecir.

Resultados obtenidos de las bit por muestra para los diferentes tipos de predictores:

Combinación	**1**		**2**		**3**		**4**		**5**	
Base	***tbm***	**Var**	***tbm***	**Var**	***tbm***	**Var**	***tbm***	**Var**	***tbm***	**Var**
BD1	4.99	0.39	5.47	0.61	4.98	0.46	4.97	0.40	4.82	0.29
BD2	6.12	0.13	6.76	0.32	6.35	0.14	6.12	0.13	6.42	0.24
BD3	2.54	0.03	7.14	2.11	3.67	0.64	2.54	0.03	4.36	0.88

C.5. Experimento 4

Para este experimento se utilizaron los siguientes parámetros:

- P = el seleccionado en el experimento C.1 ($P = 6$)
- N = 8
- n = No aplica
- tipo Predictor = el seleccionado en el experimento C.4 (*AR no adaptativo* calculados con *Levinson-Durbin*.)
- BC = SI (aplicar Bias Cancellation)

Experimento	Busca responder: ¿Disminuye el error de predicción utilizando Bias Cancellation?
Objetivo	Determinar si aplicando *BC* disminuye el error de predicción.
Procedimiento	1. Recolectar 7 señales de cada juego de datos: *a*) Para cada base: 1) Calcular el promedio de los errores de predicción. 2) Calcular el promedio de los errores de predicción corregidos mediante *BC*. 2. Analizar para cada tipo de señal, si el promedio de los errores de predicción utilizando *BC* está más cerca de 0 que el promedio de los errores de predicción sin corregir.

Resultado

Resultados obtenidos de los promedios y varianzas de los errores de predicción sin utilizar *BC* (e_j Prom. y Var.) y utilizando *BC* ($\tilde{e_j}$ Prom. y Var.):

Errores	e_j		$\tilde{e_j}$	
Señal	**Prom.**	**Var.**	**Prom.**	**Var.**
1	-0.131	237.103	-0.121	232.435
2	0.115	38.750	0.114	37.418
3	0.107	33.458	0.053	33.183
4	-0.128	88.308	-0.105	85.668
5	0.024	25.728	0.002	25.521
6	-0.048	363.492	-0.172	353.307
7	-6.245	370.307	-4.660	308.276

(a) Señales de *BD1*.

Errores	e_j		$\tilde{e_j}$	
Señal	**Prom.**	**Var.**	**Prom.**	**Var.**
1	-0.088	442.381	-0.056	441.260
2	0.032	137.174	0.011	136.616
3	-0.316	106.953	-0.304	107.070
4	-0.211	112.574	-0.153	111.880
5	-0.744	259.723	-0.574	259.530
6	0.195	175.840	0.096	176.473
7	0.175	120.935	0.105	121.232

(b) Señales de *BD2*.

Errores	e_j		$\tilde{e_j}$	
Señal	**Prom.**	**Var.**	**Prom.**	**Var.**
1	0.1768	2.5863	0.1768	2.5866
2	-0.1270	2.0207	-0.1270	2.0208
3	0.0113	1.1932	0.0112	1.1933
4	-0.0525	1.7234	-0.0523	1.7231
5	0.0215	1.2108	0.0222	1.2095
6	0.0290	0.8265	0.0290	0.8266
7	0.0039	0.8226	0.0039	0.8228

(c) Señales de *BD3*.

C.6. Experimento 5

Para este experimento se utilizaron los siguientes parámetros:

- P = el seleccionado en el experimento C.1 ($P = 6$)
- N = 8
- n = No aplica
- tipo Predictor = el seleccionado en el experimento C.4 (*AR no adaptativo*, coeficientes calculados con *Levinson-Durbin*.)
- BC = variable (0 si no se quiere aplicar BC y 1 en caso contrario)

<table>
<tr><td>Experimento</td><td>Busca responder: ¿Se obtiene una mayor compresión utilizando Bias Cancellation?</td></tr>
<tr><td>Objetivo</td><td>Determinar si aplicando Bias Cancellation se mejora compresión.</td></tr>
<tr><td>Procedimiento</td><td>1. Recolectar 7 señales de cada juego de datos:
a) Para cada base:
1) Comprimir la señal utilizando el predictor seleccionado en la prueba C.4 y sin aplicar Bias Cancellation.
2) Comprimir la señal utilizando el predictor seleccionado en la prueba C.4 y aplicando Bias Cancellation.
3) Obtener el promedio de las tasas de bits por muestra de cada tipo de señal para los puntos i. e ii..
2. Analizar para cada juego de datos, la tasa de bits por muestra promedio utilizando o no BC y teniendo en cuenta cualitativamente la complejidad computacional.</td></tr>
<tr><td>Resultado</td><td>Resultados obtenidos del promedio de las tasas de bits por muestra:
<table>
<tr><th>BC</th><th colspan="2">Sin BC</th><th colspan="2">Con BC</th></tr>
<tr><th>Base</th><th>Prom.</th><th>Var.</th><th>Prom.</th><th>Var.</th></tr>
<tr><td>BD1</td><td>5.16</td><td>0.48</td><td>5.17</td><td>0.65</td></tr>
<tr><td>BD2</td><td>6.18</td><td>0.13</td><td>6.18</td><td>0.14</td></tr>
<tr><td>BD3</td><td>2.54</td><td>0.03</td><td>2.54</td><td>0.03</td></tr>
</table></td></tr>
</table>

C.7. Experimento 6

Para este experimento se utilizaron los siguientes parámetros:

- P = el seleccionado en el experimento C.1 ($P = 6$)
- N = 8
- n = No aplica
- tipo Predictor = el seleccionado en el experimento C.4 (*AR no adaptativo*, coeficientes calculados con *Levinson-Durbin*.)
- BC = el seleccionado en el experimento C.6

Experimento	Busca responder: ¿Con $N = 8$ se mejora la compresión?
Objetivo	Determinar si con $N = 8$ se obtienen diferentes valores del parámetro 'k' de Golomb en cada contexto.
Proced.	1. Recolectar 7 señales de cada juego de datos: *a*) Para cada base: 1) Comprimir cada señal utilizando el predictor seleccionado en la prueba C.4, aplicar o no BC según la pruebaC.6 2) Obtener el promedio del parámetro 'k' de Golomb en cada uno de los N contexto. 2. Registrar en una tabla el promedio del parámetro 'k' de Golomb en cada uno de los N contextos para cada señal, frecuencia y tipo. 3. Analizar los resultados obtenidos observando cómo varía el 'k' en cada contexto y observar la cantidad de contextos a utilizar para mejorar la compresión.

Resultado

Resultados obtenidos del promedio del parámetro 'k' de Golomb por contexto:

Base de datos BD1

Contexto	1		2		3		4		5		6		7		8	
señal	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.
1	2.83	0.14	3.00	$2x10^{-4}$	3.00	10^{-4}	3.55	0.247	4.00	10^{-4}	4.00	$2x10^{-5}$	4.00	10^{-5}	5.00	10^{-4}
2	1.00	0	1.33	0.221	2.00	0.001	2.00	0.001	2.00	10^{-4}	2.00	0.001	3.00	10^{-4}	3.00	10^{-5}
3	1.00	0.001	1.03	0.027	1.18	0.148	1.60	0.239	2.00	$2x10^{-6}$	2.00	$4x10^{-6}$	2.92	0.073	3.00	$2x10^{-5}$
4	1.11	0.094	2.00	$5x10^{-5}$	2.01	0.006	2.33	0.223	3.00	$2x10^{-4}$	3.00	10^{-5}	3.82	0.149	4.00	$3x10^{-5}$
5	1.00	10^{-4}	1.00	10^{-4}	1.00	$3x10^{-4}$	1.00	$4x10^{-4}$	1.00	0.005	2.00	$3x10^{-5}$	2.00	$2x10^{-4}$	3.00	$2x10^{-5}$
6	2.35	0.228	3.00	10^{-4}	3.01	0.006	3.88	0.108	4.00	10^{-5}	4.00	$2x10^{-5}$	5.00	$3x10^{-5}$	5.00	$2x10^{-5}$
7	2.00	0.001	2.00	0.001	2.00	0.002	2.74	0.194	3.00	10^{-5}	0.00	0	3.00	0.001	4.77	0.734

Base de datos BD2

Contexto	1		2		3		4		5		6		7		8	
señal	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.
1	4.00	0.005	4.00	0.004	4.00	0.006	4.00	0.005	4.00	0.004	4.00	0.005	4.62	0.245	5.00	0.006
2	3.00	0.002	3.00	0.005	3.00	0.006	3.00	0.004	3.00	0.002	3.15	0.130	4.00	0.007	4.00	0.010
3	3.00	0.004	3.00	0.004	3.01	0.008	3.00	0.004	3.00	0.006	3.04	0.044	3.02	0.022	3.95	0.046
4	3.01	0.010	3.00	0.003	3.01	0.011	2.99	0.009	3.00	0.003	3.00	0.008	3.00	0.004	3.72	0.208
5	4.00	0.008	3.99	0.010	4.00	0.004	4.00	0.008	3.99	0.010	4.00	0.009	4.00	0.005	4.12	0.124
6	4.00	0.004	3.97	0.033	3.99	0.015	3.98	0.023	4.00	0.004	3.97	0.031	4.00	0.004	4.00	0.005
7	3.00	0.006	3.01	0.011	3.09	0.082	3.02	0.024	3.01	0.018	3.00	0.002	3.04	0.047	3.99	0.022

Base de datos BD3

Contexto	1		2		3		4		5		6		7		8	
señal	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.	Prom.	Var.
1	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
2	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
3	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
4	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
5	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
6	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
7	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00

C.8. Experimento 7

Para este experimento se utilizaron los siguientes parámetros:

- P = el seleccionado en el experimento C.1 ($P = 6$)
- N = variable
- n = No aplica
- tipo Predictor = el seleccionado en el experimento C.4 (*AR no adaptativo*, coeficientes calculados con *Levinson-Durbin*.)
- BC = el seleccionado en el experimento C.6

Experimento	Busca responder: ¿Con qué cantidad de contextos de codificación se obtiene una mayor compresión?
Objetivo	Determinar la cantidad de contextos que mejoran compresión.
Procedimiento	1. Recolectar 7 señales de cada juego de datos: *a*) Para cada base: 1) Comprimir la señal utilizando el predictor seleccionado en el experimento C.4, aplicar o no *BC* según el experimento C.6 y con parámetros $N = 1, 2, 3, 4, 8, 16 y 32$. 2) Obtener el promedio de las tasas de bits por muestra para cada N. 2. Analizar para cada tipo de señal, la tasa de bit por muestras promedio para los diferentes valores de N obtenidos.
Resultado	Resultados obtenidos para los diferentes valores de N:

Contexto	**1**		**2**		**3**		**4**		**8**		**16**		**32**	
Base	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**	**Prom.**	**Var.**
BD1	5.22	0.50	5.10	0.44	5.05	0.39	5.05	0.40	5.03	0.40	5.03	0.40	5.02	0.40
BD2	6.12	0.12	6.11	0.12	6.11	0.13	6.11	0.13	6.12	0.13	6.16	0.13	6.22	0.13
BD3	2.54	0.03	2.54	0.03	2.54	0.03	2.54	0.03	2.54	0.03	2.54	0.03	2.55	0.03

C.9. Experimento 8

Para este experimento se utilizaron los siguientes parámetros:

- P = el seleccionado en el experimento C.1 ($P = 6$)
- N = el seleccionado en el experimento C.8 ($N = 4$)
- n = No aplica
- tipo Predictor = el seleccionado en el experimento C.4 (*AR no adaptativo*, coeficientes calculados con *Levinson-Durbin*.)
- BC = el seleccionado en el experimento C.6 ($BC = NO$)

<table>
<tr><td>Experimento</td><td>Busca responder: ¿El algoritmo de compresión implementado, utilizando los parámetros óptimos seleccionados de los experimentos anteriores, obtiene una mayor compresión que el compresor Zip?</td></tr>
<tr><td>Objetivo</td><td>Determinar si el algoritmo de compresión implementado obtiene una mayor compresión que el compresor Zip.</td></tr>
<tr><td>Procedimiento</td><td>1. Recolectar 7 señales de cada juego de datos:
 a) Para cada base:
  1) Comprimir la señal utilizando los parámetros óptimos obtenidos en los experimentos anteriores.
  2) Obtener el promedio de las tasas de compresión.
 b) Comparar los promedios de las tasas de compresión obtenidos para el algoritmo de compresión implementado y las obtenidas con el compresor Zip.</td></tr>
<tr><td>Resultado</td><td>Los resultados de los promedios de TC obtenidos son los siguientes:
<table>
<tr><th>Base</th><th>Prom. TC</th><th>%Comp.</th><th>Prom. TC Zip</th><th>%Comp.</th></tr>
<tr><td>BD1</td><td>0.32</td><td>68.41</td><td>0.44</td><td>55.65</td></tr>
<tr><td>BD2</td><td>0.38</td><td>62.31</td><td>0.57</td><td>42.55</td></tr>
<tr><td>BD3</td><td>0.16</td><td>84.14</td><td>0.31</td><td>69.32</td></tr>
</table></td></tr>
</table>

Apéndice D

Pseudocódigo

D.1. Compresor

```
ComprimirEEG(parametrosIn)
{
  //Cargar la señal en un arreglo de enteros
  data = cargarEDF(parametrosIn->rutaOrigen);
  Si el tipoPredictor es 0 o 2
  {
        //Calcular los coeficientes con el método de Levison-durbin
        a = LPC (parametrosIn->P, data, cantMuestras);
  }

  //Para los tipos de predictores que utilizan adaptación se calcula la varianza.
  Si el tipoPredictor es 0 o 1
  {
        var = varianza(data, cantMuestras)
        n = parametrosIn->n / var
  }
  //se inicializa el vector Xjs con las P muestras iniciales Xjs = data[0..P];

  //si se habilito el modo debug
  if (parametrosIn->debug)
  {
        //Se crean los archivos que contendrán variables de interés
        CrearArchivosDebug(parametrosIn->rutaDestino)
  }
```

```
para cada muestra de data[P .. cantMuestras]
{
      /***** tipoPredictor:
              0-AR adaptativo(con coeficientes inicales de Levinson Durbin)
              1-AR adaptativo(con coeficientes iniciales a ingresar)
              2-AR no adaptativo(con coeficientes inicales de Levinson Durbin)
              3-AR no adaptativo a ingresar.
      ******/

  Si tipoPredictor 0 o 1
  {
      //se aplica AR adaptativo para obtener la muestra predicha (Xp),
      //se calcula el error de predicción (ep) y se actualizan los coeficientes (a)
      [Xp, a, ep] = AR(a, parametrosIn->P, P muestras anteriores, parametrosIn->n)
  }
  Si tipoPredictor 2 o 3
  {
      //se aplica AR no adaptativo para obtener la muestra predicha (Xp)
      //y se calcula el error de predicción (ep)
      [Xp, ep] = AR(a, parametrosIn->P, P muestras anteriores)
  }

  err = ep

  Si se aplica BC
  {
      //Se calcula la corrección del error de predicción (ep)
      e_nioqui = BC(ep, Xp, P muestras anteriores, parametrosIn->P)
      err = e_nioqui
  }

  //se calcula el estado de condicionamiento del ep para obtener
      //los umbrales de codificación
  s = EstadoCondicionamiento(vector de P errores anteriores)
}

//Se calculan los umbrales de codificación a partir de los errores de predicción
umbrales = EstadoCondicionamientoA_S(vector con los ep,s,parametrosIn->N)

//Se crea el archivo comprimido
archivoComprimido = CrearArchivo(paramIn->rutaDestino)

//Se escribe el cabezal del archivo EDF original
//y los parámetros que se trasmitirán al descompresor.
```

```
  para cada muestra de data[P .. cantMuestras]
  {
        //graba en el archivo archivoComprimido la codificación de Golomb
        //de los errores err
        Golomb(archivoComprimido, umbrales, err, N)
  }

  CerrarArchivo(archivoComprimido)
  if (parametrosIn->debug)
  {
        escribirYCerrarArchivosDebug()
  }
}
```

D.2. Descompresor

```
DescomprimirEEG(rutaArchivoComprimido)
{
  archivoCompprimido = AbrirArchivo(rutaArchivoComprimido)
  parametros = LeerParametrosYCabezalEDF(archivoComprimido)

  Si (parametros->debug)
  {
        CrearArchivoDebugDecodif(rutaArchivoComprimido)
  }

  errores = vector de (parametros->P) ceros
  vector_X = (parametros->P) muestras originales iniciales

  Mientras no encuentro (finArchivo)
  {
        //se decodifica el estado de condicionamiento
        s = EstadoCondicionamiento(errores)

        //se obtiene el contexto (cc_s) de codificación
        cc_s = A_S(s, parametros->umbrales, parametros->N)

        //se calcula el parámetro del código de Golomb
        //a partir de los errores del contexto cc_s (A(cc_s)) y su cantidad (N(cc_s))
        k = Calcular_k (cc_s, A(cc_s), N(cc_s))

        //Se decodifica el error de predicción
        finArchivo, err = DecodificarGolomb(archivoComprimido, k, &e_nioqui)
```

```
        Si no es (finArchivo)
        {
            A(cc_s), N(cc_s) = ActualizarContadoresGolomb(err)
            error = err

            //Si se aplica BC, se debe decodificar el error de prediccion
            if(parametros->BC)
            {
                error, X = DecodificarBC(err, vector_X, parametros->P)
            }

            Si tipoPredictor 0 o 1
            {
                //se aplica el decodificador AR adaptativo
                Xp, a = DecodificarAR(a, P, vector_X, parametros->n, error)
            }

            Si tipoPredictor 2 o 3
            {
                //se aplica el decodificador AR no adaptativo
                Xp = DecodificarAR(a, P, vector_X)
            }
        }

        //se obtiene la muestra original X y se agrega al vec_X_originales
        X = error + Xp

        //Agregar el nuevo X a un vector que contiene todos los X decodificados
        Agregar(vec_X_originales, X)
        Actualizar(vector_X, X)
        Actualizar(errores, error)

        Si (paramtros->debug)
        {
          escribirArchivosDebug()
        }
    }

    Si(paramtros->debug)
    {
        CerrarArchivoDebug()
    }

    rutaArchivoOriginal = agregarExtension(rutaArchivoComprimido, ".edf")
    escribirEDF(rutaArchivoOriginal, parametros->edfhdr, vec_X_originales)
}
```

Apéndice E

Funciones principales

Se dispone de 2 funciones principales:

int ComprimirEEG(struct parametros_struct* paramIn)

parametros_struct estructura que contiene los siguientes atributos:

- **P =** indica el orden del modelo AR.
- **N =** cantidad de contexto de codificación.
- **n =** valor que determina la convergencia del algoritmo *AR adaptativo* (para los casos en que se seleccione tipoPredictor = 0 o 1).
- **a =** coeficientes iniciales del modelo *AR* (para los casos en que se seleccione tipoPredictor = 1 o 3).
- **Xjs =** arreglo con las P muestras iniciales de la señal
- **umbrales =** arreglo con los N+1 umbrales calculados para la codificación de la señal, como se indicó en la sección 3.1.11
- **tipoPredictor =** número que indica las siguientes combinaciones:
  - **0** *AR adaptativo* con coeficientes iniciales calculados con *Levinson-Durbin*.
  - **1** *AR adaptativo* con coeficientes iniciales a ingresar.
  - **2** *AR no adaptativo* con coeficientes iniciales calculados con *Levinson-Durbin*.
  - **3** *AR no adaptativo* con coeficientes iniciales a ingresar.
- **BC =** valor que indica si aplicar *BC* (1) o no aplicar *BC* (0).
- **rutaOrigen =** ruta donde se encuentra el archivo EDF original que se va a comprimir.
- **rutaDestino =** ruta donde se encuentra el archivo EDF comprimido cuya extensión es .EEGzip.
- **edfhdr =** contiene los datos del cabezal del archivo EDF original.
- **debug =** valor que indica si se graban archivos con variables de interés (1) o no se graba (0).

Retorna: 0 en caso de que la compresión haya sido exitosa, 1 en caso contrario.

Comprime el archivo *EDF* ubicado en la ruta *rutaOrigen* y tomando en cuanta la selección de los demás parámetros, genera un archivo comprimido con el formato especificado en B (en la misma ubicación de *rutaOrigen*). En caso de que se seleccione entre los parámetros *debug* = 1 los archivos de *debug* generados se guardaran en la misma ruta donde se encuentra el ejecutable.

int DescomprimirEEG(char* ruta)

Ruta: ruta donde se ubica el archivo comprimido.

Retorna: 0 en caso de que la compresión haya sido exitosa, 1 en caso contrario.

Descomprime el archivo ubicado en ruta, generando a partir del ComprimirEEG.

Nomenclatura

AC	Codificación Aritmética
AR	Model autoregressive
BC	Bias Cancellation
BCI	Brain-Computer Interface
CC	Codificación condicional
EEG	Electroencefalograma
KLT	Transformada Karhunen-Loeve
LMS	Least mean squares
WT	Transformada Wavelet
WTP	Transformada Wavelet Packets

Bibliografía

[1] Judit Cinkler Nasir Memon, Xuan Kong. Context-based lossless and near-lossless compression of eeg signals. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=788586&contentType=Journals+%26+Magazines. Accedida 18-Julio-2013.

[2] http://dotnetzip.herobo.com/DNZHelp/Index.html#. Accedida 26-Octubre-2013.

[3] http://eon.wustl.edu/neuroscan.shtml#batch. Accedida 15-Febrero-2014.

[4] El formato gdf guess. http://guess.wikispot.org/The_GUESS_.gdf_format. Accedida 15-Febrero-2014.

[5] Bob Kemp. European data format. http://www.edfplus.info. Accedida 25-Febrero-2014.

[6] Dr. Marcelo Lester. Introducción a la transformada wavelet. http://www.exa.unicen.edu.ar/escuelapav/cursos/wavelets/apunte.pdf. Accedida 18-Julio-2013.

[7] Simeranjit Brar Jeremy R. Tolbert, Pratik Kabali and Saibal Mukhopadhyay. An accuracy aware low power wireless eeg unit with information content based adaptive data compression. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5333943&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F5307844%2F5332379%2F05333943.pdf%3Farnumber%3D5333943. Accedida 19-Octubre-2013.

[8] N. Sriraam. A high-performance lossless compression scheme for eeg signals using wavelet transformand neural network predictors. http://www.hindawi.com/journals/ijta/2012/302581/. Accedida 19-Octubre-2013.

[9] Juan V. Lorenzo-Ginori y Ernesto Rodríguez-Valdivia Julián L. Cárdenas-Barrera. Algoritmo basado en wavelet packets para la compresión de señales electroencefalográficas. http://www.hab2001.sld.cu/arrepdf/00210.pdf. Accedida 19-Octubre-2013.

[10] Juan V. Lorenzo-Ginori Julián L. Cárdenas-Barrera and Ernesto Rodríguez-Valdivia. A wavelet-packets based algorithm for eeg signal compression. http://informahealthcare.com/doi/abs/10.1080/14639230310001636499. Accedida 19-Octubre-2013.

[11] Gustavo Brown Rodríguez. Compresión de imágenes mediante klt. http://iie.fing.edu.uy/ense/asign/codif/material/monografias/2003-02.pdf. Accedida 18-Julio-2013.

[12] Toshihisa Tanaka Yodchanan Wongsawat, Soontorn Oraintara and K. R. Rao. Lossless multi-channel eeg compression. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=

1692909&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1692909. Accedida 19-Octubre-2013.

[13] Transformada discreta de fourier, definición, propiedades, etc. http://centrodeartigos.com/articulos-informativos/article_71576.html. Accedida 18-Julio-2013.

[14] Claudio Estienne. Banco de filtros coseno modulado - implementación polifásica. Accedida 18-Julio-2013, Noviembre 2006.

[15] Julián Cárdenas-Barrera y Fernando Cruz-Roldán Carlos Bazán-Prieto, Manuel Blanco-Velasco. Compresión de señales electroencefalográficas epilépticas y normales. http://rielac.cujae.edu.cu/index.php/rieac/article/viewFile/99/pdf_91. Accedida 19-Octubre-2013.

[16] M. Ramasubba Reddy K. Srinivasan, Justin Dauwels. A two-dimensional approach for lossless eeg compression. http://www.sciencedirect.com/science/article/pii/S174680941100005X. Accedida 19-Octubre-2013.

[17] Jesús García Quesada. Tutorial de análisis numérico. interpolación: Fórmula de newton en diferencias divididas. http://numat.net/tutor/newton.pdf. Accedida 24-Octubre-2013.

[18] E. Velarde Reyes. Algoritmo de compresión de eeg mediante predicciones usando el polinomio de interpolación de newton con diferencias divididas. http://rd.springer.com/chapter/10.1007/978-3-540-74471-9_30. Accedida 19-Octubre-2013.

[19] Santiago Lafon. Monografía sobre redes neuronales - tratamiento estadístico de señales. http://www.fing.edu.uy/iie/ense/asign/tes/materiales/monografias/RedesNeuronales.pdf. Accedida 24-Octubre-2013.

[20] N. Sriraam and C. Eswaran. Performance evaluation of neural network and linear predictors for near-lossless compression of eeg signals. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4358888&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4358888. Accedida 19-Octubre-2013.

[21] José Luis López Fernández. Algoritmo de huffman. http://webs.uvigo.es/joselu/material/Algoritmo%20de%20Huffman.pdf. Accedida 24-Octubre-2013.

[22] Run-length encoding. http://es.wikipedia.org/wiki/Run-length_encoding. Accedida 18-Julio-2013.

[23] Radford M. Neal Ian H. Witten and John G. Cleary. Arithmetic coding for data compression. http://www.stanford.edu/class/ee398a/handouts/papers/WittenACM87ArithmCoding.pdf. Accedida 26-Octubre-2013.

[24] D.F García D. de la Fuente. Modelado de series temporales con métodos en bloque y recursivos. http://www.idescat.cat/sort/questiio/questiiopdf/12.3.1.delafuente.pdf. Accedida 26-Octubre-2013.

[25] http://es.wikipedia.org/wiki/Distribuci%C3%B3n_geom%C3%A9trica. Accedida 26-Octubre-2013.

[26] Jpeg-ls. http://www.jpeg.org/jpeg/jpegls.html. Accedida 18-Julio-2013.

[27] Teunis van Beelen. Edflib is a programming library for c/c++ to read/write edf+/bdf+ files. http://www.teuniz.net/edflib/. Accedida 23-Octubre-2013.

[28] http://sccn.ucsd.edu/eeglab/. Accedida 26-Octubre-2013.

[29] Alfonso Fernández Sarría. Estudio de técnicas basadas en la transformada wavelet y optimización de sus parámetros para la clasificación por texturas de imágenes digitales. Accedida 18-Julio-2013, Febrero 2007.

[30] Rafael Molina. Cunatificación escalar. https://www.google.com.uy/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CC0QFjAA&url=http%3A%2F%2Fdecsai.ugr.es%2Fccd%2Ftransparencias%2F07%2520cuantificacion_escalarx4.pdf&ei=Ec2HUtS3OozGkQewvICQDw&usg=AFQjCNHJSd_NJDAznAsTqTqwZ6QzQnDuZg&sig2=U6Q3oDfb6EELXf-9d4l-LA. Accedida 26-Octubre-2013.

[31] Selin Aviyente. Compressed sensing framework for eeg compresion. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4301243&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4301243. Accedida 19-Octubre-2013.

[32] Aplicación del algoritmo matching pursuit para la estimación del ritmo cardíaco en señales obtenidas mediante un sensor capacitivo. http://zaguan.unizar.es/TAZ/CPS/2010/5429/TAZ-PFC-2010-375.pdf. Accedida 26-Octubre-2013.

[33] http://es.wikipedia.org/wiki/Codificador. Accedida 26-Octubre-2013.

[34] Víctor Hugo Benítez Casma Guillermo Tejada Muñoz, Steven Jesús Zarzosa Chávez. Red neuronal implementada en fpga. http://sisbib.unmsm.edu.pe/bibvirtualdata/publicaciones/electronica/2008_n22/pdf/a03.pdf. Accedida 26-Octubre-2013.

[35] Set partitioning in hierachical trees (spiht). http://zaguan.unizar.es/TAZ/EINA/2012/9707/TAZ-PFC-2012-708_ANE.pdf. Accedida 26-Octubre-2013.

[36] Mingui Sun Neeraj Magotra, Giridhar Mandyam and Wes McCoy. Lossless compression of electroencephalographis (eeg) data. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=541709&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D541709. Accedida 19-Octubre-2013.

[37] Definición de perceptrón. http://es.wikipedia.org/wiki/Perceptr%C3%B3n. Accedida 26-Octubre-2013.

[38] El perceptrón multicapa. http://proton.ucting.udg.mx/~cheko/neu/pdf/perceptron.pdf. Accedida 26-Octubre-2013.

[39] Redes neuronales recurrentes. http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/oropeza_c_ca/capitulo3.pdf. Accedida 26-Octubre-2013.

[40] Judit Cinkler Nasir Memon, Xuan Kong. Diseño de filtros fir. http://ocw.uv.es/ingenieria-y-arquitectura/filtros-digitales/tema_3._diseno_de_filtros_fir.pdf. Accedida 18-Julio-2013.

[41] J. Ignacio Godino-Llorenteb Joaquín Blanco-Velascoa-Carlos Armiens-Aparicioa Francisco López-Ferrerasa Manuel Blanco-Velascoa, Fernando Cruz. On the use of prd and cr parameters for ecg compression. https://www.google.com.uy/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CDoQFjAB&url=http%3A%2F%2Fwww.researchgate.net%2Fpublication%2F7868296_On_the_use_of_PRD_and_CR_parameters_for_ECG_compression%2Ffile%2F3deec51bb26da3a896.pdf&ei=tV2JUrfHA8nKkAfJ3oAQ&usg=AFQjCNHzF3vrAqqqktx4EyVl3oHvPUUVRw&sig2=Bt7scG8DqUx__4YaEogfsQ. Accedida 26-Octubre-2013.