Codecs Para Compressão de EEG's

1st Miguel Marques
Dept. Engenharia Informática
(Faculdade de Ciências e tecnologias)
Universidade de Coimbra
Viseu, Portugal
miguelpmmarques@gmail.com

2nd Paulo Cardoso
Dept. Engenharia Informática
(Faculdade de Ciências e tecnologias)
Universidade de Coimbra
Coimbra, Portugal
paulotomas14@gmail.com

3rd João Nunes
Dept. Engenharia Informática
(Faculdade de Ciências e Tecnologias)
Universidade de Coimbra
Coimbra, Portugal
joao_fnunes@hotmail.com

***Abstract*— Neste artigo apresentaremos e discutiremos métodos para compressão de Encefalografias, o panorama tecnológico actual relativo aos Codecs utilizados para comprimir estes sinais e as diferentes metodologias utilizadas em cada módulo, ou seja, em cada fase de funcionamento do Codec, considerando principalmente os procedimentos mais eficientes. Debruçar-nos-emos especialmente sobre métodos de compressão não destrutivos, compressões estas que satisfaçam as exigências de organizações e profissionais de saúde (dada a falta de legislação na área em inúmeros países, e a importância de cada sample no diagnóstico global, considera-se que não devem ser ignorados quaisquer dados sem a consulta de um especialista), e que ainda assim permitam diminuir substancialmente os custos de armazenamento e transmissão de dados, garantindo o seu funcionamento em aparelhos de especificações limitadas. Aprofundaremos sobre uma colecção de algoritmos que considerámos mais pertinentes, elaborando sobre a sua estrutura, vantagens e desvantagens, e analisando os resultados obtidos e apresentando uma discussão sobre algoritmos de predicção não lineares. Por fim, delinearemos uma estratégia futura para o estudo continuado deste tema.**

Keywords: Electroencefalogramas, Codec.

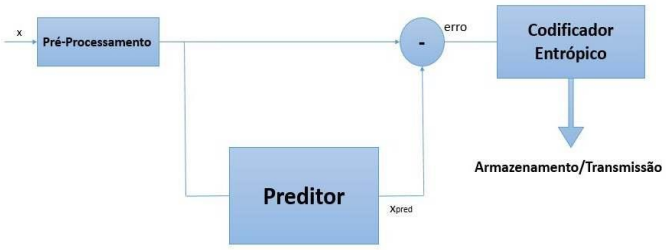
I. INTRODUÇÃO

Teoria de Informação é aplicada em compressão de dados em inúmeros campos tecnológicos. O esquema CODEC (Coder/Decoder) é utilizado para as mais variadas fontes de informação: vídeos, imagens, músicas e outros diversos. No contexto dos Electroencefalogramas, os equipamentos existentes no mercado e na maioria das instalações de saúde pública revelam-se insuficientes para armazenar e/ou transmitir grandes volumes de dados. Numa fase de desenvolvimento anterior da medicina contemporânea, determinou-se que uma análise visual dos exames EEG era insuficiente para elaborar um diagnóstico fiável do estado de saúde do utente. Segue assim que o tratamento desta informação por via computacional é a melhor abordagem. Consideremos o exemplo de um homem adulto comum que realizará este exame [1], a frequência do seu Encefalograma variará entre 0.1 e 100 Hertz. Desta forma, um mínimo de 200 Hertz será necessário para fazer a medição. Considerando um nível de quantização frequentemente utilizado de 16 bits por amostra, um EEG de 10 canais realizado durante um período de 24 horas, teremos que o espaço de armazenamento necessário para guardar este sinal seria de aproximadamente 346 Mb, espaço este que a grande generalidade dos microcontroladores comerciais tem dificuldade em comportar.

Mostra-se então que esta temática não se situa simplesmente na esfera do hipotético, mas sim numa necessidade transversal a variadas instituições de saúde.

De uma forma geral, existe pouca investigação neste ramo de exames complementares de diagnóstico. Assim, este artigo foi elaborado com o intuito de reunir as melhores técnicas de compressão utilizadas nos variados estudos consultados (ver Referências para mais informações), e determinar em que pé está a comunidade científica relativamente a esta temática.

Um algoritmo baseado num codec pode ser considerado como um algoritmo de compressão sem perdas se o método de compressão não alterar a forma dos dados originais, isto é, antes da compressão e após a descompressão os ficheiros obtidos devem ser os mesmos. Simplificadamente, um codec é constituído por três passos principais: uma fase de pré-processamento, uma de predição e uma fase da codificação, todas elas abordadas em maior detalhe ao longo deste artigo, apresentadas no seguinte esquema [2]:



II. CRITÉRIOS DE COMPRESSÃO

Existe um conjunto diverso de critérios para avaliar a eficiência dos métodos de compressão na literatura concernente a este tema. De uma forma sucinta, existem três factores preponderantes: a taxa de compressão, a eficiência do algoritmo e a complexidade do algoritmo.

A taxa de compressão, tal como é compreendida no ramo da teoria da informação, e definida em [3], entende-se como:

$$\mathrm{TC} = \frac{L_{orig} - L_{comp}}{L_{orig}} \times 100\ \%$$

Em que L_{orig} significa o comprimento do sinal original, ou seja, o número de bits necessários para representar a fonte original, e L_{comp} o comprimento do sinal comprimido.
Podemos também considerar esta fórmula como uma taxa de bits por segundo, alternativa esta mais útil em considerações de transmissão de dados em tempo real, para cálculo da banda larga necessária. Define-se como:

$$\mathrm{TC} = \frac{L_{comp} \times \mathrm{f}}{N}$$

Em que f se entende como a frequência da amostragem do sinal e N como o número de amostras do sinal.

Optámos pela primeira equação ao longo deste artigo para exprimir os resultados que encontrámos na nossa pesquisa.

A eficiência e a complexidade dos algoritmos são temas algo mais complexos, mas para os efeitos deste artigo, optámos por aplicar todos os algoritmos aos vários sinais sempre com a mesma máquina, um computador portátil com um processador Intel Core i5 7300HQ e 8 GB de memória RAM, registando os tempos de cada compressão, para efectuar uma breve comparação dos valores obtidos um pouco mais adiante. Os intervalos de tempo para a execução completa de cada Codec foram também registados e tabelados, de modo a obter resultados mais tangíveis e comparáveis nesta secção do trabalho.

III. Métodos de pré-processamento

Em ciência computacional, um pré-processador é considerado um programa que processa os dados que recebe, sendo que a sua emissão será utilizada como input de outro programa (o nosso preditor). Diz-se, então, que o seu output se trata de uma forma pré-processada dos dados originais.

Esta fase de compressão da fonte caracteriza-se principalmente como sendo uma fase de melhoramento do sinal. Elimina-se algum ruído existente, deteta-se a existência de zonas vazias (porções estas que aumentam o comprimento do sinal, mas que não precisam de ser enviadas), e detetam-se coincidências entre canais de amostras de múltiplos canais como é o caso de uma música em formato stereo ou até mesmo os nossos EEGs.

Após alguma pesquisa sobre o assunto, e considerando a definição bastante abrangente de pré-processamento, compreendemos que a fase de predição do sinal, é, por si, uma fase de pré-processamento. Assim, fazemos rapidamente a distinção entre as duas fases: a primeira será por nós referenciada como pré-processamento daqui em diante; a segunda por predição do sinal, ainda que o output de ambas se trate de uma versão pré-processada do sinal original.

Nas várias comparações efectuadas ao longo deste artigo, os sinais já fornecidos se encontravam num estado pré-processado e normalizado aquando da aplicação dos nossos algoritmos sobre estes.

IV. Preditores

Na fase de predição do sinal não ocorre uma compressão no sentido estrito da palavra. Ao invés disso, são implementadas estratégias para manipular o sinal, de modo a diminuir ao máximo a sua entropia menos elevada, isto é, moldar o sinal de forma a obter uma distribuição estatística fortemente centrada em zero (distribuição gama). Tendo em conta que a nossa fase de codificação está assente na utilização de códigos entrópicos, diminuir a distribuição estatística do sinal antes da fase de codificação aumentará substancialmente a compressão conseguida.

No que toca à pesquisa e levantamento de diversos procedimentos específicos de pré-processamento na literatura relevante, destacamos os seguintes métodos, relevantes mas normalmente utilizados em conjunto com os restantes métodos que aqui iremos explicitar: Transformadas de Wavelet e Fast Fourier Transform. Sendo o método das transformadas de Walevet o mais utilizado das duas, obtendo geralmente os melhores resultados [5].

A. Cadeias de Markov

Este tipo de preditor é ser implementado sobre a hipótese de que o sinal tem propriedades Markovianas e que é gerado sobre uma fonte modelada com uma cadeia de Markov. [2] Assumindo o modelo de Markov de ordem 1, são estimadas todas as probabilidades condicionais do tipo:

P ($X_n = a_i \mid X_{n-1} = a_j$), em que X_n é o estado seguinte e o X_{n-1} é o estado atual.

Seja X uma variável discreta que retira valores aleatórios do alfabeto finito (A = $\{a_1, ..., a_n\}$) de uma dada fonte. É criada então uma matriz de ocorrências que conta as incidências de $X_n = a_i$ procedido de $X_{n-1} = a_j$, na posição ij da matriz.

$$\prod_{ij} \approx \frac{F_{ij}}{\sum_k F_{kj}}$$

Seja Π uma matriz de transição de uma cadeia de Markov, cujos elementos constituem uma matriz de n por n, de valores estocásticos não negativos. Deste modo, o somatório de cada uma das linhas da matriz será igual a 1. X_n pode ser previsto a partir de E [$X_n \mid X_{n-1}$]. Para calcular o sucessor de a_j, minimizando o erro do mínimo quadrado, fazemos o seguinte somatório:

$$_{succ}(a_j) = \sum_k k \prod_{kj}$$

Para qualquer a_j que pertença ao alfabeto.

B. Preditores lineares

De um modo geral, os preditores lineares baseiam-se fortemente numa ideia de que a dependência amostral presente no sinal é quantificável, e assim, numa situação limite, na qual o valor de uma amostra é estatisticamente dependente do valor da amostra anterior, poderíamos prever sem erros todas as amostras do nosso sinal, conhecendo esta dependência. Contudo, essa proposição raramente se verifica. Mas ainda que a dependência estatística não seja total, isto não significa que os valores das amostras sejam independentes. Existe sempre um nível de dependência presente, e, nos fenómenos naturais este valor é elevado. Assim, estes preditores estimam o valor da amostra em questão, recorrendo aos valores das p amostras anteriores:

$$x_{pred}(k) = \sum_i^p a_i \; x(k-i)$$

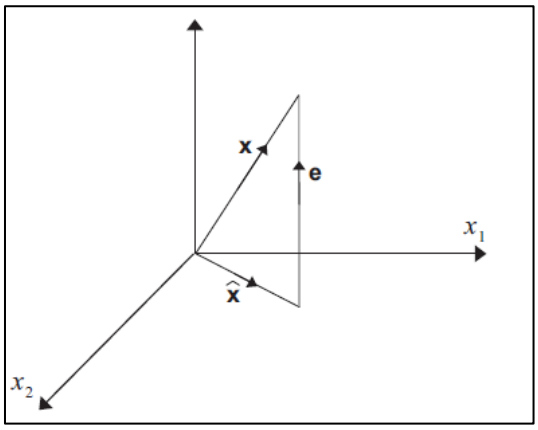
Com a_i o coeficiente considerado no método preditivo escolhido, como por exemplo, o método da codificação diferencial, um método simples, e por isso com pouco custo computacional, frequentemente utilizado, no qual estimamos que o valor da nossa amostra actual será igual ao valor da amostra anterior. Esta estimativa não está muito longe da realidade, tendo em conta que os fenómenos naturais não são caracterizados por alterações muito bruscas no sinal. Considerando uma janela suficientemente pequena, em vez de considerar o sinal de um ponto de vista macroscópico, esta estimativa é perfeitamente aceitável.

Segue então, poderemos calcular um valor aproximado da nossa amostra próxima do valor real. No entanto, isto não é

suficiente para uma predição dita não destrutiva. Por si só, os métodos preditivos lineares são considerados destrutivos. Contudo, calculando o erro associado a cada amostra, e enviando esse erro com o nosso sinal estimado, poderemos reconstruir exatamente o sinal sem perdas.
Desta forma, o erro calculado será:

$$e(k) = x(k) - x_{pred}(k) \leftrightarrow x(k) = e(k) + x_{pred}$$

Enviando para o nosso codificador o erro associado a cada amostra, numa fase posterior de descompressão será possível reconstruir o sinal original sem perdas.
Neste artigo, escolhemos concentrar-nos no preditor conhecido como LPC. Este preditor linear utiliza os conceitos algébricos de matrizes de correlação e recursividade de Levison-Durbin para calcular os coeficientes a utilizar. A principal estratégia utilizada é minimizar o erro quadrático para obter os melhores coeficientes possíveis: E[$e_N{}^f$ (n) $x*$ (n-i)]=0 para 1<=i<=N, com "N" o nosso número de amostras, "E" a esperança matemática, "*e"* o nosso erro, e "*x"* o nosso conjunto de amostras. De acordo com o princípio da ortogonalidade [6], a nossa estimativa, p, resultará num erro médio quadrático mínimo, de entre todas as estimativas lineares possíveis, quando o erro de estimação for ortogonal às N variáveis aleatórias.
Teorema da ortogonalidade esquematizado:



Seguidamente, consideremos: x um vector, x=[.$x_i, \dots, x_{i-n}$], que representa as n amostras aleatórias. Para estas queremos calcular a=[.$a_i, \dots, a_{i-n}$], que serão os nossos coeficientes, e x*=[. $x*_i, \dots, x*_{i-n}$], as nossas n amostras anteriores observadas em concreto no nosso sinal. Assim, e após alguns passos de cálculo matricial, obtemos:

$$\underbrace{\begin{bmatrix} E[\,|x_1|^2] & E[x_1x_2^*] & \dots & E[x_1x_N^*] \\ E[x_2x_1^*] & E[\,|x_2|^2] & \dots & E[x_2x_N^*] \\ \vdots & \vdots & \ddots & \vdots \\ E[x_Nx_1^*] & E[x_Nx_2^*] & \dots & E[\,|x_N|^2] \end{bmatrix}}_{\mathbf{R}} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix}}_{\mathbf{a}} = \underbrace{-\begin{bmatrix} E[x_1x^*] \\ E[x_2x^*] \\ \vdots \\ E[x_Nx^*] \end{bmatrix}}_{-\mathbf{r}}.$$

Uma equação do tipo Ax=b, que pode ser resolvida através do algoritmo de eliminação de Gauss, a grande custo computacional, ou através da recursividade de Levinson-Durbin (muito menos intensiva para os mesmos recursos).

Como já explicámos anteriormente, um algoritmo com base num preditor apenas pode ser considerado sem perdas se for também enviado o valor do erro de predição, para que a reconstrução do sinal possa ser feita.

Para a posterior fase de descompressão necessitaremos apenas de enviar os erros da predição efectuada, e os coeficientes calculados, de modo a obter o sinal original, como está explicado em [6]. Elaborando:

Consideremos os coeficientes calculados: A= $a_0Z^0+a_1Z^1+\cdots+a_NZ^N$, e o nosso vector de erros obtidos, *E*. Para obter o nosso sinal previsto através da função filter, desprezaremos o primeiro valor de coeficiente, que corresponderá sempre a 1. Ainda assim, admitamos que D= -A. Teremos então, após alguns passos de cálculo, $\frac{X}{E} = \frac{1}{D}$, com X o sinal original. Assim, sabemos que a equação $X = \frac{E}{D}$ nos fornecerá o sinal original, dado que necessitamos apenas dos coeficientes calculados e dos erros obtidos (ambos conhecidos), para reconstruir a matrix X.

C. Redes Neuronais Artificiais

As redes neuronais são caracterizadas por estruturas paralelas de tamanho considerável, grau elevado de interligações, computação a alta velocidade, mapeamento não linear e auto-organização, o que faz destas óptimos preditores e excelentes para problemas de compressão [1]. Dentro deste tipo de preditor temos vários tipos, como as SLP, MLP, EN, entre outras.
Vejamos por exemplo [7] a aplicação de uma rede neuronal recursiva com uma arquitetura de 4-6-1, em que os símbolos a ser codificados são dados por:

$$d(t) = x^t - f(x^{t-1}, \dots, x^{t-4}, h_1^{(t-1)}, \dots, h_6^{(t-1)}, o_1^{(t-1)})$$

Em que x^t representa o valor atual dado como input do sinal no instante t, $d(t)$ é a diferença entre o valor atual e o valor preditado, em que $f(\dots)$ é o output da rede neuronal. Por fim, h e o são respectivamente a camada escondida e a camada do output.

V. CODIFICAÇÃO

Na última fase do processo, são utilizados principalmente os métodos de codificação de Huffman, Aritméticos, códigos LZ78 e LZW [9]. Estes dois primeiros métodos baseiam-se nos seguintes processos genéricos: identificação dos padrões de bits que ocorrem mais frequentemente numa dada fonte e padrões que ocorrem com menor frequência, que serão codificados com mais bits do que os padrões mais frequentes, sendo estes substituídos por códigos menos extensos. Nos últimos dois métodos estamos perante técnicas que se baseiam em dicionários, que, resumidamente, consistem na codificação das sequências em índices, recorrendo a um dicionário que armazena estas sequências de símbolos. É de notar que codificações LZ77 e LZ78 são assimétricas, pelo que a codificação é bastante mais complexa que a descompressão. Algoritmos, LZW, pelo contrário, não apresentam esta característica, sendo necessário um menor envio de informação, dado que o

algoritmo de descompressão elabora também o dicionário em tempo real, à medida que processa novos valores.
Por fim, apontamos que, na literatura a que tivemos acesso, grande parte destes algoritmos implementam uma camada de Run-Length Encoding (RLE), que permite que qualquer sequência de caracteres repetidos possa ser abreviada. No entanto, e como precisaremos mais à frente, dadas as características específicas do nosso sinal, esta estratégia pode revelar-se infrutífera.

VI. Especificidades dos EEG's

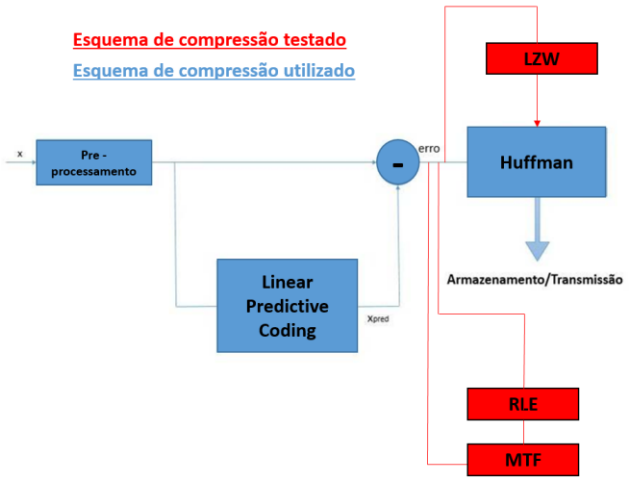
Um Electroencefalograma é um sinal de monitorização de potenciais electrónicos corticais medidos ao longo do escalpe. São utilizados para diagnosticar mortes cerebrais, casos de epilepsia, e outras patologias. São sinais considerados largamente não estacionários e não lineares [8], fortemente afectados por ruído proveniente de barulhos musculares e movimentos oculares e espasmos faciais. Ao contrário de outros tipos de exames desta natureza, os EEG's têm a dificuldade acrescida de captar os sinais eléctricos relevantes através do crânio dos pacientes de forma não evasiva, acrescentando assim à complexidade e custos do equipamento, e aumentando também o ruído captado. São sinais com um nível de aleatoriedade elevado em toda a sua extensão, ainda que exibam comportamentos que provam existir uma correlação espacial e temporal entre as suas amostras, dadas janelas suficientemente pequenas de amostragem. Deste modo, abordagens que aproveitam esta dependência produzem os melhores resultados. O método de vectores de quantização [4] é de sublinhar, aproveitando a correlação espacial (atingindo taxas de compressão de 62%) sendo este um método derivado de Redes Neuronais, apresentadas anteriormente. Para aproveitar a dependência temporal, destacamos o método de Predição linear com inputs atrasados (60.2% de taxa de compressão).

Destacamos também métodos MVAR[10], e a utilização de uma combinação de pré-processamento de Wavelet com uma fase de predição assente em redes neuronais, preservando os coeficientes de Wavelet para utilizar no treino da nossa rede neuronal[11].

Passamos também a explicar este conceito de dependência temporal e espacial. Entende-se por dependência temporal que uma amostra verificada no passado partilha uma relação matemática com os dados recolhidos no presente e os dados futuros. Por outro lado, dependência espacial significa semelhantemente que um dados recolhidos por elétrodos em proximidade terão também uma relação matemática, isto é, uma amostra recolhida por um elétrodo colocado na região parietal do crânio terá uma correlação com uma amostra recolhida por um elétrodo colocado na mesma região, mas não terá uma relação com uma amostra recolhida na glabela.

VII. Codec Sugerido

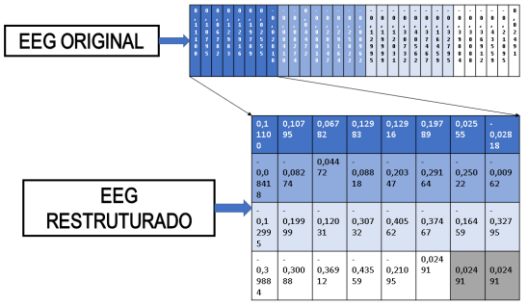
Para abordar este desafio, escolhemos utilizar o programa MATLAB, dado que é uma ferramenta poderosa na área da análise de dados. Antes de partir para qualquer codificação, teremos primeiro de proceder à quantização do nosso sinal para 16 bits. Feito isto, escolhemos aplicar um preditor linear, a dinâmica dos quais já explicitamos anteriormente.



(Esquema da fase de compressão do codec utilizado)

O preditor que escolhemos para este campo do trabalho foi a função pré-existente do MATLAB "lpc", que recebe como parâmetros de entrada o nosso sinal, e a ordem do preditor, isto é, a quantidade de valores e coeficientes, n, que vamos considerar para efectuar a nossa estimativa. No entanto, esta aplicação colocou um problema. Como referimos, os Electroencefalogramas são largamente considerados não estacionários e não dependentes na sua globalidade. Nestes, a dependência estatística apenas se verifica para janelas mais pequenas, e, assim sendo, não poderemos considerar estes mesmos coeficientes para prever os valores de uma janela que já não obedece à dependência estatística considerada. Deste modo, teremos então que voltar a estimar os novos coeficientes para a janela respectiva, e elaborar a nossa predição para a nova janela com base nestes. Após uma análise do algoritmo, compreendemos que a função "lpc" está preparada para considerar cada coluna de uma matriz de sinal como sinais separados, calculando os coeficientes para cada um deles.

Por isso, dividimos o nosso vector de sinal em k janelas de tamanho j e colocámos estes valores numa matriz, cada coluna da qual correspondendo a uma janela do sinal original. Deste modo, conseguimos calcular os diferentes coeficientes de cada uma destas janelas, e, aplicando a função "filter", obtemos a nossa estimativa. Desta calculamos o nosso erro, e, então, voltamos a colocar a matriz de resultados na sua forma vectorial, bem como a matriz de coeficientes e de erros. Por fim, quantizamos os erros obtidos, tal como havíamos feito anteriormente para o sinal original, e, assim sendo, o nosso sinal está pronto para entrar na fase de codificação.



(Para a função lpc é enviada a transposta destra matriz esquematizada)

Antes de aplicar os codificadores entrópicos, escolhemos testar algumas possibilidades para comprimir ainda mais o sinal e diminuir a sua entropia. Foram estas: algoritmos Run Length Encoding, Move-to-Front e Burrows-Wheeler.

Contudo, os resultados obtidos não foram rentáveis. Dada a elevada aleatoriedade estatística do nosso sinal, o overhead adicionado nos algoritmos RLE excede largamente o ganho introduzido. No entanto, colocámos a hipótese de que a aplicação de uma camada MTF seguida de uma RLE fosse capaz de colmatar este defeito.

Porém, os algoritmos Move-to-Front encontrados mostraram-se pouco eficientes e fiáveis, sendo pouquíssimos sequer certificados pela empresa desenvolvedora da linguagem em questão. Por isto, decidimos elaborar o nosso próprio algoritmo MTF:

*A.*Algoritmo Move-to-Front

O algoritmo move-to-front é um método para codificar dados com o propósito de melhorar a performance do codificador entrópico, diminuído a entropia do sinal, efetuando um mapeamento adaptativo entre símbolos existentes no alfabeto, já definido previamente. Algoritmo revela-se funcional para codificação em tempo real, partindo do pressuposto que o alfabeto conhecido não será mudado, sendo o fluxo de informação transmitida entre o emissor e o receptor sincronizado graças ao seu funcionamento que será explicado desde já:

1. É definido o alfabeto dos dados que pretendemos transmitir.
2. É lido o símbolo que predemos enviar e consequentemente enviado o índice da sua posição no alfabeto.
3. É Procurado esse mesmo símbolo no alfabeto e colocado no início do nosso alfabeto. O nosso alfabeto é, assim, sempre atualizado, a não ser que o símbolo a enviar já esteja na primeira posição. Nessa situação é enviado o valor 1 e passa-se ao símbolo seguinte.

É de notar que este codificador pode ser implementado em qualquer fase da predição, mas é bem mais benéfico quando usado imediatamente antes do codificador entrópico, no nosso caso Huffman.

Contudo, novamente, e apesar da acrescentada complexidade algorítmica, o sinal quase duplicou o seu tamanho, com poucos ganhos à vista. Testamos ainda assim a codificação MTF isolada, e chegamos às mesmas conclusões. A aleatoriedade e a reduzida dependência estatística dos dados estudados não permite uma boa compressão, dado que muitos destes algoritmos foram elaborados para comprimir dados cujos alfabetos são relativamente pequenos, e que, logicamente, se tratam de dados maiores redundâncias que podem ser exploradas para comprimir os dados com ganhos substanciais, algo que não acontece no nosso algoritmo. Para explicar este conceito, consideremos um ficheiro de texto codificado seguindo o paradigma ASCII. Tendo este alfabeto apenas 256 caracteres, e conhecendo o funcionamento da língua inglesa, sabemos que não só existirão bastantes amostras repetidas, como também estas repetições se verificarão em padrões ao longo do sinal (sílabas, por exemplo). Assim, um algoritmo de RLE produzirá bastantes ganhos. Por outro lado, o nosso EEG, não estacionário e não dependente, ainda que passível de conter amostras de valores repetidos, não terá o mesmo tipo de repetição de padrões. Por isso, não são comprimidos valores suficientes por cada overhead adicionado para produzir ganhos. No caso da amostra fornecida nos materiais do trabalho, o sinal comprimido quase duplica o seu tamanho em relação ao original.

Desta forma, partimos para os algoritmos de Burrows-Wheeler disponíveis, mas deparámo-nos com uma biblioteca reduzida de funções pouco confiáveis. As poucas dignas de testes revelaram-se demasiado complexas. Os seus outputs resultavam num tempo de compressão médio, por aplicação posterior da codificação de Huffman, de mais de duas horas. Concluímos assim que os ganhos no espaço ocupado pelos ficheiros comprimidos, versus os ficheiros originais, no intervalo de 2% a 3%, não compensavam em qualquer situação prática, pelo tempo dilatado que demoravam a efectuar. É de apontar que a elevada demora tem uma explicação muito plausível. As funções encontradas operavam de forma a obter e produzir dados em forma de ficheiros de texto, csv e outros, ou seja, armazenamento em discos rígidos, o que se traduz em extensos tempos de leitura, e espera mais elevada do que seria necessário, caso os dados estivessem carregados em memória RAM, como acontece com os restantes algoritmos testados.

Por isso, partimos para a próxima fase do nosso trabalho. Testámos o preditor linear considerado para o maior número de janelas e ordens de preditor possível, Obtendo resultados extremamente promissores e em concordância com a literatura em variados aspectos. Testámos este preditor para ordens de predição entre 1 (preditor diferencial) e ordem igual ao tamanho do sinal. Para cada um destes valores, testámos janelas desde 10 valores até ao comprimento do sinal, filtrando os dados por valores de entropia menores que 12 (sendo que a entropia original do sinal era de 14,1927), obtendo assim, pelo menos, um ganho de três bits por cada amostra. Fizemos também uma filtragem por valores de q, quantum, obtidos aquando da quantização do sinal. Um valor pequeno de quantum implica menores perdas na fase de quantização do sinal, e portanto uma compressão mais próxima da compressão sem perdas. O mínimo escolhido para o valor de quantum foi 1, dado que a maioria dos valores obtidos para os erros se encontravam no intervalo [0,10].

Obtivemos então 3185 amostras que correspondiam às características procuradas. Apontámos, que de entre todos estes valores, todos eles se situam na gama de ordem de preditor de 1 até 13, valores estes suportados pelas conclusões encontradas em [4].

Apresentamos, seguidamente, alguns dos melhores valores obtidos:

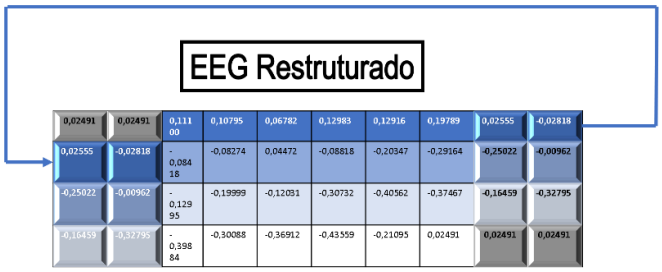
Ordem	*Entropia*	*Quantum*	*Compressão*	*Tempo*	*Janela*
3	**4.1869**	**0.0855**	**3,78:1**	**0.5592**	*528*
6	**7.3448**	**0.0098**	**2,16:1**	**1.2064**	*274*
6	**4.6935**	**0.0584**	**3,39:1**	**0.6228**	*728*
6	**4.1927**	**0.0828**	**3,78:1**	**0.7059**	*753*
9	**4.8574**	**0.0593**	**3,28:1**	**1.5012**	*1230*
10	**4.7468**	**0.0696**	**3,35:1**	**2.6538**	*1901*

Os valores de compressão acima enunciados foram obtidos através da codificação de Huffman, utilizando as funções pré-definidas de MATLAB "huffmandict", para elaborar o dicionário e "huffmanenco" para produzir o valor final.

No entanto, procurámos diminuir ao máximo o erro da nossa predição, e, para este propósito, procuramos uma nova estratégia para lidar com este desafio.

Enunciamos então o nosso método:

Considerando a estratégia anterior de divisão do nosso sinal em janelas mais pequenas e restruturação do mesmo numa matriz constituída por estas janelas, decidimos adicionar uma pequena colecção de valores da janela anterior a cada coluna (consideramos o número destes valores adicionados a ordem do nosso preditor), como mostra o seguinte esquema:



Enviamos então esta nova matriz para o preditor, e, depois de obtida a nossa estimativa, truncamos os novos valores adicionados e colocamos de novo a nossa predição em forma vectorial, calculando os erros e avaliando a nova entropia.

No entanto, esta nova estratégia revelou-se infrutífera. Apresentamos os resultados, selecionados de entre os melhores de 2937 resultados com uma entropia aceitável:

Ordem	*Entropia*	*Quantum*	*Compressão*	*Tempo*	*Janela*
3	**4.4306**	**0.0936**	**3,58:1**	**0.6973**	*398*
6	**7.3913**	**0.0950**	**3.61:1**	**0.6464**	*728*
9	**7.7836**	**0.0094**	**2.05:1**	**2.0653**	*1044*
9	**4.4701**	**0.0965**	**3,54:1**	**1.3066**	*1230*

Como podemos verificar, não só os resultados da compressão foram mais baixos, como também os valores do quantum são mais elevados. Isolamos o resultado obtido para o valor de janela 728 e ordem 6, valor este presente em ambas as tabelas apresentadas, e que comprova as afirmações anteriores. No entanto, estes dados recolhidos são contrários às previsões que havíamos feito, e à solidez do raciocínio e dos conhecimentos matemáticos por detrás da sua implementação. Concluímos assim que a discrepância entre dados recolhidos e os resultados espectáveis se prende com uma pequena limitação dos processos utilizados para os obter. Passamos a explicar.

Para proceder a esta implementação dentro dos mesmos moldes dos algoritmos anteriores, ou seja, colocando todos os dados em forma matricial, temos então também de completar a primeira linha desta nova matriz, de modo a que o algoritmo lpc a aceite, e efectue as previsões necessárias. De outra maneira, o algoritmo não é capaz de calcular os coeficientes desejados. Assim, escolhemos estimar que estes valores a inserir antes das primeiras amostras do nosso sinal, e recorrendo também aos nossos conhecimentos de processos matemáticos naturais, poderiam ser estimados com alguma veracidade serem iguais aos valores das últimas amostras. No entanto, de modo a conseguir colocar estes dados em forma matricial originalmente havíamos também efectuado algumas aproximações, nomeadamente, no caso de o sinal original não ser divisível pelo tamanho das janelas, estimámos que os valores extra a inserir no sinal poderiam ser iguais ao último valor, desprezando assim estes valores na compressão do sinal. No entanto, esta estimativa entra em conflito com a nova aproximação que teremos de efectuar, pelo que nos vimos impossibilitados de aplicar esta estratégia sem alterar substancialmente o método original. No entanto, a única implementação possível para poder fazer a estimativa correta, sem comprometer o rigor científico que nos predispusemos a seguir, implicava substituir a nosso algoritmo de predição de um baseado em prever uma matriz inteira de dados, para um baseado em prever cada janela individualmente num ciclo "for". No entanto, este método aumentava substancialmente os tempos médios de compressão do nosso sinal, e talvez mais importante, este aumento de tempo tornava impossível a recolha massificada dos dados entrópicos a analizar. Para deixar esta limitação bem clara, explicaremos. Recolher a entropia de uma determinada janela dada uma certa ordem de predição não sofreu um aumento substancial, passado de um tempo médio de poucas fracções de segundo para dois segundos. No entanto, este aumento revela-se impeditivo à recolha de milhares de amostras. Optámos assim por não implementar este método.

Posteriormente, decidimos testar uma compressão dos mesmos dados com codificações aritméticas e algoritmos de Lempel-Ziz. No entanto, as funções existentes de MATLAB para codificação aritmética e Lempel-Ziv apenas se revelaram funcionais para pequenos vectores de valores inteiros (no intervalo [0:255]), tendo sido elaboradas com o intuito de comprimir somente fontes de tipos muito específicos, nomeadamente imagens e texto ASCII. Assim sendo, aplicar estes algoritmos na nossa fonte revelou-se impossível. No entanto, não dispostos a sacrificar o rigor do nosso projecto, procedemos no sentido de elaborar o nosso próprio algoritmo LZW. Escolhemos este, ao invés de algoritmos aritméticos, dado que, em toda a nossa pesquisa, uma concordância nas conclusões retiradas pelos diferentes investigadores parecia inevitável: Huffman, tendo em conta o alfabeto extenso das Encefalografias, não devendo por isso ser utilizado agrupamento de símbolos de modo a não elevar a complexidade algorítmica para níveis incomportáveis, produz os melhores rácios de compressão na generalidade dos casos. Deste modo procedemos para a elaboração do nosso algoritmo LZ, optando pela variante Lempel-Ziv-Welch, visto que esta é maioritariamente considerada pela comunidade tecnológica um melhoramento face às outras duas variantes.

Após a realização de variados testes com os outros algoritmos LZW encontrados (para sinais por estes comportados), verificamos a validade do algoritmo por nós construído. Assim, passámos à fase seguinte, e aplicámos este algoritmo à nossa Encefalografia. Em termos de eficiência computacional apontamos que o algoritmo criado não é tão eficiente quanto uma função nativa de MATLAB. Porém, os tempos obtidos para a aplicação desta ferramenta não excederam os três minutos na grande maioria dos casos. Para

janelas não óptimas, este valor não excedeu os sete minutos. Consideramos então a aplicação deste algoritmo viável.

Após a sua utilização na nossa fonte, as entropias obtidas aumentaram. Esta ocorrência era previsível, dado que os ganhos obtidos com este tipo de compressão se baseiam não no melhoramento da entropia do sinal, mas sim na diminuição no número de amostras através da identificação de padrões no sinal em toda a sua extensão e consequente registo no dicionário. Assim, apesar do aumento da entropia, o número de amostras diminuiu substancialmente. Desta forma, para que qualquer ganho seja obtido, a diminuição do comprimento do sinal após a aplicação do LZW terá de compensar pelo aumento relativo da entropia. Todavia, e conhecendo a fraca dependência amostral do nosso sinal, após a codificação de Huffman, e tendo em conta a acrescida complexidade algorítmica, os ganhos obtidos revelaram-se marginais, tendo, inclusivamente, alguns dos valores de compressão obtidos piorado com esta nova abordagem (são estes os casos para os quais a entropia, após a predição do sinal, ainda se mostrava relativamente alta, e portanto, a aplicação do LZW acabava por não compensar por este aumento).

Mostramos então os valores obtidos para uma aplicação de LZW. Consideraremos um caso no qual a entropia obtida na fase de predição se revelou baixa, e outro para o qual esta era relativamente alta:

	Ordem	*Janela*	*Compressão*	*Tempo*	*Entropia*
Huff	**6**	**728**	**3,39:1**	**1,2s**	**4,69**
LZW+Huff	**6**	**728**	**3,40:1**	**218,8 s**	**12,43**
Huff	**6**	**274**	**2,16:1**	**1,3s**	**7,35**
LZW+Huff	**6**	**274**	**2,07:1**	**472,1s**	**11,88**

VII. Análise e interpretação de resultados

Examinando os valores apresentados ao longo deste artigo, e também a generalidade dos valores obtidos, demasiado extensos para colocar neste pequeno artigo, concluímos que a aplicação de um preditor linear, seguida por uma camada de codificação de Huffman obtém, na maioria dos casos testados, uma taxa de compressão até 73,5%, ou compressão de 3,78:1 (com o nosso sinal original normalizado, pré-processado, e quantizado para 16 bits).

Para todos os dados fornecidos, concluímos que a melhor ordem para este tipo de sinais para o preditor considerado (lpc) é 12 (de entre 3185 amostras com uma boa entropia obtida, a maior porção delas, cerca de 817 resultados, foram obtidos com uma ordem de predição 12), conclusões estas suportadas por [4].

Concluímos também que, embora o algoritmo LZW utilizado produza alguns ganhos, numa secção muito considerável dos dados, os ganhos relativamente a uma compressão de Huffman simples são algo negligenciáveis, principalmente em situações de transmissão de dados, nas quais a diferença entre tempos de compressão(poucos segundos para uma compressão de Huffman normal, versus minutos para uma compressão LZW+Huffman) seriam impeditivas para uma transmissão fluida de dados, algo que, atendendo ao contexto temporalmente sensível de envio de dados médicos, não pode ser ignorado, sendo assim a melhor opção uma compressão simples de Huffman. Por outro lado, caso o nosso objectivo seja uma compressão de dados para armazenamento, a aplicação do algoritmo LZW pode revelar-se útil, caso o tempo expedido possa ser considerado negligenciável. É também de apontar que o algoritmo LZW por nós elaborado poderia facilmente ser implementado noutra linguagem de programação, por exemplo linguagens de baixo nível, algo que diminuiria substancialmente os tempos de compressão obtidos.

VIII. Hipotetizando: Redes Neuronais e cadeias de Markov

Numa fase prévia da elaboração deste projeto, na qual já possuíamos algumas bases relativamente às características de Eletroencefalogramas, mas na qual ainda não tínhamos aplicado os nossos próprios métodos nem os nossos próprios testes sobre o sinal, considerámos elaborar uma predição não linear para além do método linear apresentado. Ulteriormente, já tendo mais alguma experiência, compreendemos as limitações deste modelo, escolhendo aprofundar ao máximo a predição linear.

Porém, optámos por estudar estas diferentes aplicações, consultando uma colecção vasta de artigos sobre o assunto. Explicamos, portanto, o porquê da não aplicabilidade prática de alguns destes métodos não lineares.

Comecemos então pelos preditores baseados em cadeias de Markov. Tal como havíamos enunciado anteriormente, a aplicação de um preditor deste tipo pressupõe que o sinal em questão tem propriedades Markovianas. Logo, à partida, conseguimos perceber que, caso esta premissa não se verifique (em encefalogramas raramente acontece), este preditor não poderá obter bons resultados. Analisando este conceito sob uma perspetiva mais teórica, podemos concluir que, sendo o eletroencefalograma um sinal tão claramente estocástico, a dependência estatística desta fonte mostra-se praticamente nula, matematicamente falando. Considerando, desta maneira, a repartição do sinal em janelas de tamanho variável, com X e Y duas janelas distintas do mesmo sinal com um tamanho aleatório, mas idêntico entre X e Y, consideremos a sua entropia conjunta:

$$H(X,Y) = H(X) + H(Y|X)$$

Caso os valores de ambas as amostras estejam próximas da independência estatística, o valor de $H(Y|X)$ será praticamente igual a $H(Y)$, pelo que a predição não beneficiará muito desta pequena dependência. De facto, considerando um EEG na sua plenitude, esta ocorrência constata-se, excluindo o caso em que as janelas consideradas são pequenas o suficiente para que a linearidade se verifique. Assim, conclui-se que este tipo de sinais não tem, de facto, propriedades Markovianas, a não ser talvez nestas pequenas janelas. Contudo, aplicando os preditores lineares e preditores baseados em cadeias de Markov ao mesmo sinal dividido nestas pequenas janelas, os preditores lineares obtêm, na generalidade dos casos, maiores ganhos.

Logo, podemos concluir que o método das cadeias de Markov, mesmo sendo um ótimo preditor de sinais com uma dependência estatística comum, não será, de longe, o mais conveniente a ser utilizado no nosso caso.

Relativamente às redes às redes neuronais, fortemente inspiradas no sistema nervoso central dos animais, capazes do chamado “machine learning”, e também de reconhecimento de alto nível dos mais variados padrões,

são retiradas conclusões semelhantes, ainda que por diferentes razões.

Comecemos então por explicar os limites práticos da aplicação deste sistema. Exploraremos duas possíveis implementações práticas de uma rede neuronal neste projecto:

No primeiro caso, procederíamos do mesmo modo que procedemos para os preditores lineares. Isto é, dividiríamos o nosso sinal em pequenas janelas, e forneceríamos estas janelas aos nós de entrada da nossa rede neuronal, de modo a que esta fosse capaz de reconhecer os padrões e a dependência presentes em cada uma destas janelas. No entanto, esta abordagem teria um grande senão. Após a aplicação da nossa rede a um sinal diferente, toda a aprendizagem efectuada por esta, relativamente ao primeiro sinal, seria perdida. Com cada novo sinal a nossa rede neuronal voltaria à estaca zero, perdendo assim uma das maiores vantagens destes preditores, nomeadamente o melhoramento de cada predição com o processamento sucessivo de sinais da mesma espécie.

Passemos, portanto, a uma segunda implementação, na qual forneceríamos uma base de dados inteira à nossa rede neuronal, para a treinar de modo a que a limitação do caso anterior não se verificasse. No entanto, a quantidade de amostras presente na grande maioria das bases de dados internacionais (consideremos a base de dados da Physionet, considerada uma das mais reputáveis), normalmente no intervalo de 500 a 600 amostras (obtidas nas mesmas condições; outros sets de amostras têm diferentes condições), não é considerada suficiente para treinar uma rede neuronal. Assim sendo, para testar esta hipótese, teríamos a necessidade de estabelecer algumas parcerias com diversas instituições, de modo a poder aceder a maiores bases de dados, normalmente não disponíveis ao comum utilizador da internet.

Por outro lado, e como já foi mencionado enumeras vezes ao longo deste artigo, o carácter aleatório deste sinal faz com que o número de padrões seja escasso, algo que não abona a favor desta implementação.

Logo, por estes motivos principalmente não procedemos à implementação concreta destes dois preditores na nossa investigação.

IX. Conclusão

Ao longo deste artigo estabelecemos diversas comparações e retirámos enumeras conclusões sobre o tema em causa, a grande maioria destas, senão todas elas, em concordância com a literatura concernente a este tema. Aproveitamos, ainda assim, para abrir caminho para trabalho futuro, desenvolvendo sobre as maiores limitações deste projecto e estratégias que tomaríamos para as colmatar.

Tendo em conta que a nossa fase de predição e compressão, os sinais apenas têm em conta um canal, (dos dez ou mais canais que normalmente constituem este tipo de sinal), sabemos que apenas estamos a ter em consideração a dependência temporal entre as amostras, não podendo aproveitar a dependência espacial inter-canal que também os caracteriza. Caso contrário, e especialmente para canais correspondentes a receptores em relativa proximidade poderíamos calcular a informação mútua entre esses canais, e, dependendo do seu resultado, poderíamos então proceder à implementação de uma rede neuronal, dado o maior número de padrões existentes entre estes diferentes canais. Utilizaríamos então uma toolbox já implementada em MATLAB, nomeada de EEGLAB, para melhor visualizar e compreender os dados com que fossemos confrontados.

Experimentarímos também com preditores lineares de inputs atrasados (delayed inputs), e exploraríamos as fases de pré-processamento de sinais. Exploraríamos estratégias mais complexas, como por exemplo “Vectores de Quantização” ou modelos “MVAR”.

Por outro lado, prepararíamos a nossa candidatura a um site de base de dados internacional como a Berlin Brain-Computer Interface (instituição esta

Que exige algumas informações aos interessados em aceder aos dados disponíveis), de modo a obtermos bases de dados mais alargadas para proceder à implementação de redes neuronais na fase de predição do sinal.

Experimentaríamos também com algoritmos de Golomb-Rice para a fase de codificação do sinal, e tentaríamos implementar a nossa própria fase de pré-processamento.

Procederíamos também de modo a optimizar ao máximo as funções LZW e MTF por nós criadas, e completaríamos esta linha de funções elaborando as suas análogas de descompressão.

Por fim, pesquisaríamos e tentaríamos executar tantas estratégias alternativas quanto fossemos capazes de implementar.

X. Reconhecimentos

Os autores deste artigo gostariam de agradecer ao Professor Doutor Paulo de Carvalho, professor catedrático do Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologias da Universidade de Coimbra, pela sua orientação e incentivo, e paciência no esclarecimento das nossas muitas dúvidas.

XI. Referências

[1] N. Sriraam, “A High-Performance Lossless Compression Scheme for EEG Signals Using Wavelet Transform and Neural Network Predictors”
[2] M. Brito, “Compressão de ECGs. Uma abordagem baseada na regularidade do sinal”
[3] T. M. Cover and J. A. Thomas, “Elements of Information Theory”
[4] Antoniol, G., and Tonella, P. (1997). “EEG data compression techniques”
[5] M. Akin, “Comparison of Wavelet Transform and FFT Methods in the Analysis of EEG Signals”
[6] P.P.Vaidyanathan, “TheTheoryofLinearPrediction”
[7] R. Battiti, A. Sartori, G. Tecchiolli, P. Tonella, and A. Zorat, “Neural compression: An integrated application to EEG signals”
[8] Darius Birvinskas, Vacius Jusas, Ignas Martisius, and Robertas Damasevicius, “Fast DCT Algorithms for EEG Data Compression in Embedded Systems”
[9] Khalyd Sayood, “Introduction to Data Compression”, third edition
[10] Laxmi Shaw, Daleef Rahman, and Aurobinda Routray, “Highly Efficient Compression Algorithms for Multichannel EEG”
[11] Bashar A. Rajoub, “An Efficient Coding Algorithm for the Compression of ECG Signals Using the Wavelet Transform”
[12] Allen Gersho and Robert M. Gary, “Vector Quantization and Signal Compression”
[13] David J.C. MacKay, “Information Theory, Inference, and Learning Algorithms”
[14] Zoran Fejzo, “Lossless Mutli-Channel Audio Codec”
[15] Markus J. Bauer, Anthony James Cox, Giovanna Rosone and Dirk Evers, “Methods and Systems for Data Analysis Using the Burrows Wheeler Transform”
[16] Young Han Nam, Seop Hyeong Park, Tae Kyoon Ha, Yun Ho Jeon, “Preprocessing of Digital Audio Data for Mobile Audio Codecs”