UNIVERSIDADE FEDERAL DO ABC

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DA INFORMAÇÃO

Interface Cérebro-Computador Explorando Métodos para Representação Esparsa dos Sinais

Vinícius Ormenesse

Orientador: Prof. Dr. Ricardo Suyama

Santo André - SP, 2018

Vinícius Ormenesse

Interface Cérebro-Computador Explorando Métodos para Representação Esparsa dos Sinais

Dissertação apresentada como parte dos requisitos para a obtenção do título de Mestre em Engenharia da Informação pela Universidade Federal do ABC.
Orientador: Prof. Dr. Ricardo Suyama

Universidade Federal do ABC

Engenharia da Informação

Santo André - SP, 2018

Sistema de Bibliotecas da Universidade Federal do ABC
Elaborada pelo Sistema de Geração de Ficha Catalográfica da UFABC
com os dados fornecidos pelo(a) autor(a).

Ormenesse, Vinícius
Interface Cérebro-Computador Explorando Métodos para Representação Esparsa dos Sinais / Vinícius Ormenesse. — 2018.

70 fls. : il.

Orientador: Ricardo Suyama

Dissertação (Mestrado) — Universidade Federal do ABC, Programa de Pós-Graduação em Engenharia da Informação, Santo André, 2018.

1. Esparsidade do Sinal. 2. Sinais Neurais. 3. Processamento de Sinais. 4. Brain Computer Interface. 5. Aprendizado não supervisionado. I. Suyama, Ricardo. II. Programa de Pós-Graduação em Engenharia da Informação, 2018. III. Título.

Agradecimentos

A presente dissertação de mestrado não poderia ter chegado ao seu fim sem que houvesse apoio de várias pessoas.

Em primeiro lugar, não posso deixar de agradecer ao meu orientador, Professor Doutor Ricardo Suyama, por toda a paciência, empenho e sentido prático com que sempre me orientou neste trabalho.

Desejo igualmente agradecer pelo apoio incondicional a toda a minha família, a minha mãe, Mirian, ao meu pai, Arnaldo, ao meu irmão, Ivan, a minha cunhada, Mayara, aos meus tios, primos e a minha avó.

Não esqueço de agradecer, também, inúmeros colegas de trabalho que entenderam e me ajudaram a conciliar todas as minhas tarefas diárias.

Resumo

Uma interface cérebro-computador (BCI) é projetada para que se consiga, de modo efetivo, fornecer uma via alternativa de comunicação entre o cérebro do usuário e o computador. Sinais captados por meio de eletrodos, tipicamente posicionados no escalpo do indivíduo, são previamente processados para que haja eliminação de ruídos externos. A partir daí, diversas técnicas para processamento de sinais são utilizadas para posteriormente classificar os sinais registrados e realizar a tradução do estado mental do usuário em um comando específico a ser executado pelo computador. No presente trabalho são utilizadas técnicas de representação esparsa dos sinais para a extração de características relevantes para classificação dos mesmos, com intuito de aumentar a robustez e melhorar o desempenho do sistema. Para a extração de sinais esparsos, foram utilizados algoritmos de criação de dicionários, a partir dos quais é possível obter uma representação esparsa para todo o subespaço de sinal. No trabalho foram utilizados 5 diferentes algoritmos de criação de dicionário: Método de direções ótimas (MOD), K-SVD, RLS-DLA, LS-DLA e Aprendizado de dicionário Online (ODL). A classificação dos sinais foi realizada com o método de k vizinhos mais próximos ($k - NN$). Os resultados obtidos com a abordagem de representação esparsa foram comparados com os resultados do BCI Competition IV dataset 2a. Para o primeiro colocado da competição foi obtido, em termos do coeficiente kappa, uma acurácia de 0.57 enquanto que no trabalho utilizando os métodos esparsos, obteve-se, em coeficiente kappa, uma acurácia de 0.90. Em comparação obteve-se um ganho de 0.33 de acurácia, onde se deduz que o uso de sinais esparsos pode ser benéfico para o difícil problema de se projetar uma interface cérebro computador.

Palavras-chave: Sinais neurais, esparsidade do sinal, processamento de sinais, Brain Computer Interface (BCI), interface cérebro computador.

Abstract

A brain computer interface (BCI) is designed to effectively translate commands thought by human individuals into commands that a computer can effectively understand. Electrical impulses generated from the brain sculp are recorded from a device called an electroencephalograph and are preprocessed for elimination of external noise. From there, several techniques for signal processing are used to later classify the signals obtained by the electroencephalograph. In this work, techniques for sparse representation of signals are used for feature extraction, in order to increase robustness and system performance. For the extraction of sparse signals, five different dictionary learning algorithms were used, being able to produce a basis capable of represensing the entire signal subspace. In this work, 5 different dictionary learning algorithms were used: Method of Optimal Directions (MOD), K-SVD, Recursive Least Square Dictionary Learning (RLS-DLA), Least Square Dictionary Learning (LS-DLA) and Online Dictionary Learning (ODL). For the classification task, the k-NN method was used. The simulation results obtained with this approach were compared with the best BCI Competition IV dataset 2a results. For the first place in the competition, an accuracy of 0.57 was obtained, in terms of the kappa coefficient, whereas in the work using the sparse methods, a kappa coefficient of 0.90 was obtainned, improving accuracy in 0.33 accuracy was obtained, which indicates that the use of sparse signals may be beneficial to the difficult problem of designing a brain computer interface.

Keywords: Neural Signals, Sparse signals, signal processing, Brain Computer Interface (BCI).

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BCI	*Brain Computer Interface* - Interface cérebro-computador
CSP	*Common Spatial Pattern* - Padrões espaciais comuns
EEG	Eletroencefalografia
HMM	*Hidden Markov Model* - Modelo oculto de Markov
ICA	*Independent Component Analysis* - Análise de Componentes Independentes
JAD	*Joint Approximate Diagonalization*
LARS	*Least Angles Regressions*
LDA	*Linear Discriminant Analysis* - Análise linear discriminante
LS-DLA	*Least Squares Dictionary Learning Algorithm*
M-CSP	*Multiclass Common Spatial Pattern*
MLP	*Multilayer Perceptron* - Perceptron multi-camada
MOD	*Method of Optimal Directions*
NBC	*Naive Bayes Classifier* - Classificador Naive Bayes
OMP	*Orthogonal Matching Pursuit*
PCA	*Principal Component Analysis* - Análise de componentes principais
RLS-DLA	*Recursive Least Squares Dictionary Learning Algorithm*
SVD	*Singular Value Decomposition*
SVM	*Support Vector Machine*

Notação Utilizada

Ao longo da dissertação, letras maiúsculas (M,N,P,O,...) são utilizadas para designar matrizes, enquanto letras minúsculas são utilizadas para designar vetores coluna. Dessa forma, uma matriz M pode ser descrita em termos de suas colunas, i.e., $M = [M_{:,1}|...|M_{:,n}]$, onde o termo ':' representa todos os índices daquela coluna. Os coeficientes de um vetor são referenciados utilizando letras minúsculas (m,n,o,p,...), por exemplo, $v = (v_1, ..., v_m)^T$. A notação 0 é utilizada para denotar um vetor nulo (todas suas componentes são nulas), e seu comprimento é deduzido do contexto. Por exemplo, $M = 0$ se refere à uma matriz nula.

Dado um vetor com um único índice $i = i_1$ ou uma sequência de índices $i = (\underline{i}_1, ..., \underline{i}_k)$ pode-se denotar uma matriz $M_i = [M_{:,i_i}|...|M_{:,i_k}]$ como sendo uma submatriz que contém as respectivas colunas indexadas por i, na ordem em que se aparece no vetor i. Usa-se a notação $M_{i,j}$, para se referir a um algorismo da matriz M, com i respectivo à linhas e j respectivo à colunas. Letras minúsculas sublinhadas se referem à algarismos, ou seja, $\underline{a}$ se refere, por exemplo, à $\underline{a} = 1$.

Sumário

1 Introdução

Qualquer forma natural de comunicação requer nervos periféricos e músculos para a realização do mesmo. Um processo de movimentação muscular, por exemplo, começa com a intenção de se movimentar; a intenção de movimentação ativa, em algumas áreas, um complexo processo no cérebro que envia sinais através dos nervos até o músculo envolvido que realiza o movimento desejado [Graimann, Allison e Pfurtscheller 2010].

O BCI (do inglês, Brain Computer Interface em português, interface cérebro-computador) oferece uma alternativa para tal comunicação e controle entre o cérebro e o músculo, captando os sinais diretamente do cérebro do indivíduo (de maneira invasiva ou não) e os processa diretamente no computador. Para isso, em linhas gerais, um sistema de BCI deve ser capaz de amostrar sinais provenientes do cérebro, processá-los e associar os padrões observados a comandos que podem ser utilizados para controlar, por exemplo, um computador ou um braço mecânico.

A captura da atividade neural do indivíduo, pode ser realizada de maneira invasiva, com por exemplo através da Eletroencefalografia Intracraniana, na qual os eletrodos são implantados no interior da caixa craniana, ou de maneira não-invasiva, por meio de registros de EEG com eletrodos posicionados no escalpo do indivíduo. Em geral, a captura dos sinais utilizando técnicas invasivas oferecem uma visão mais detalhada da atividade cerebral, permitindo, assim, identificar com maior precisão as regiões nas quais há uma atividade mais elevada (melhor resolução espacial). Em contrapartida, a implantação dos eletrodos requer uma intervenção cirúrgica, o que pode limitar a aplicação da BCI [Allison, Wolpaw e Wolpaw 2007].

O processamento dos sinais coletados deve ser capaz de ressaltar e/ou extrair características relevantes que permitam a classificação correta do estado mental do indivíduo. A etapa de processamento final do sistema consiste em realizar a tradução das características extraídas para um comando específico, realizando corretamente a classificação dos padrões neurais observados. Aplicações de BCI encontradas na literatura incluem o controle de um computador para tarefas cotidianas, como realizar pesquisa na internet, escrever documentos , etc; controle de robôs e até mesmo controles de equipamentos de locomoção como cadeiras de rodas [Allison, Wolpaw e Wolpaw 2007].

Embora o BCI, em geral, conte com diversos canais de EEG para coletar a informação sobre a atividade cerebral, processar corretamente estes sinais ainda é uma tarefa muito complexa e pouco assertiva, principalmente quando se considera a utilização de métodos não invasivos para captura dos sinais. Além disso, outras dificuldades encontradas

no desenvolvimento das interfaces relacionam-se com o fato de que cada indivíduo possui padrões cerebrais próprios, que também podem se alterar com o tempo. Dessa forma, o sistema precisa ser flexível o bastante para se adaptar a diferentes usuários e contar com algoritmos eficientes para a seleção e classificação das características mais relevantes para identificação da intenção do usuário.

Assim, o presente trabalho propõe a utilização de uma nova técnica para ser utilizada junto ao pré-processamento do sinal. A técnica proposta é a utilização de vetores esparsos, que podem eventualmente colaborar para a classificação dos sinais esparsos. Os vetores esparsos são criados a partir de um dicionário que, em teoria, representa todo o subespaço de sinais existente nas amostras já obtidas do EEG e portanto, são capazes retratar o sinal da maneira mais esparsa possível, auxiliando assim, na classificação dos sinais em melhorando o desempenho geral do BCI.

Deste modo, a estrutura da dissertação segue da seguinte maneira: No Capítulo 2 são discutidos os blocos fundamentais de um sistema de interface cérebro-máquina, apresentando os algoritmos utilizados para aprendizagem e seus métodos. No Capítulo 3, são apresentados os conceitos que fundamentam o problema de representação esparsa e os algoritmos utilizados para aprendizado de dicionário. No Capítulo 4 é apresentada a proposta de utilização da representação esparsa no problema de classificação de padrões em BCI. Os resultados são discutidos no Capítulo 5, e as conclusões finais e perspectivas para trabalhos futuros, apresentados, finalmente, no Capítulo 6.

2 Interfaces Cérebro-Computador

Um sistema BCI deve prover uma via alternativa de comunicação para o indivíduo, de maneira que seja possível associar determinadas características presentes nos sinais a comandos específicos utilizados para controlar dispositivos.

Para isso, o sistema, em geral, conta com ao menos 4 blocos funcionais, conforme ilustrado na Figura 1: Aquisição dos sinais, pré-processamento, extração de características e classificação [Alamdari et al. 2016].

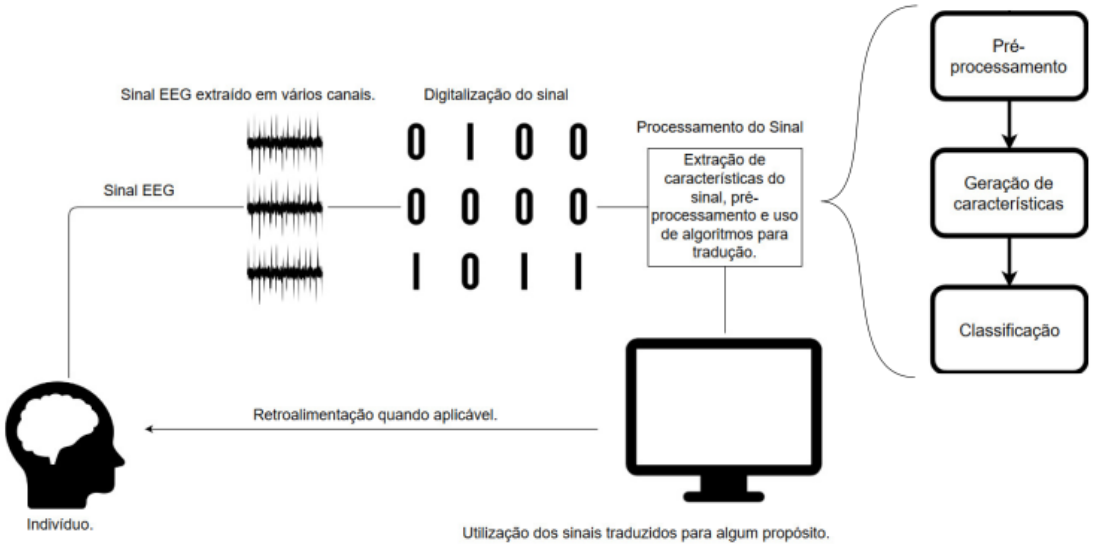

Figura 1: Estrutura geral de um sistema BCI.

No presente capítulo são abordados os principais conceitos relacionados a sistemas BCI, descrevendo algumas das abordagens propostas na literatura e apresentando os blocos fundamentais presentes nas interfaces.

2.1 Paradigmas Existentes em BCI

Os sinais cerebrais envolvem uma grande variedade de fenômenos que se relacionam a diferentes tarefas cognitivas, e muitas destas relações ainda não são completamente compreendidas pelos pesquisadores na neurociência.

Há, entretanto, certos comportamentos peculiares observados nos sinais neuronais que podem ser explorados para o desenvolvimento da interface cérebro-computador, uma

vez que a intenção do usuário é capaz de modular determinados tipos de sinais neuronais, que são detectáveis por meio de técnicas avançadas de processamento. Dentre os diferentes tipos de sinais já estudados, há ao menos três abordagens que vem sendo adotadas na construção de sistemas BCI: Potenciais Evocados Visualmente, Potenciais Evocados P300 e Potenciais senso-motores [Nicolas-Alonso e Gomez-Gil 2012].

Os Potenciais evocados visualmente (VEPs, do inglês, *Visual Evoked Potentials*) são respostas decorrentes da modulação da atividade cerebral que ocorrem no córtex visual a partir de um estímulo visual [Nicolas-Alonso e Gomez-Gil 2012]. Por exemplo, uma pessoa que foca sua atenção em um estímulo visual que pisca a uma frequência de 15Hz, terá um aumento no registo de atividade neuronal na região do córtex visual na frequência de 15Hz - uma resposta denominada de SSVEP (do inglês, *steady state visual evoked potentials*). Caso existam diferentes estímulos visuais, piscando em frequências distintas, é possível inferir para qual estímulo a pessoa está focando sua atenção (correspondendo, por exemplo, a um comando específico a ser realizado pela interface) observando em qual frequência ocorre um aumento da atividade neuronal, sendo assim possível interpretar a intenção do usuário [Nicolas-Alonso e Gomez-Gil 2012][Hwang, Lee e Lee 2017].

Outro exemplo de potencial evocado que tem sido bastante utilizado em BCI são os potenciais P300, que correspondem a picos positivos nas ondas cerebrais captadas após a exposição do usuário a estímulos audiovisuais ou somatossensoriais específicos [Nicolas-Alonso e Gomez-Gil 2012]. Estes picos normalmente aparecem alguns milissegundos após os estímulos serem apresentados ao usuário, e, em geral, quanto menor a probabilidade do estímulo (i.e., mais raro é a ocorrência do estímulo), maior será o pico da onda detectada.

O exemplo emblemático de uso do P300 em interfaces cérebro-computador é o chamado *P300 speller*, cuja ideia é ilustrada no diagrama apresentado na Figura 2. A operação da interface ocorre da seguinte forma: O usuário, sentado a frente de uma tela contendo os diferentes caracteres, deve focar sua atenção à letra que deseja digitar; As linhas e colunas da matriz de caracteres piscam de maneira aleatória, e quando a linha/coluna da matriz da letra na qual o usuário tem sua atenção focada se acende, ou se destaca na tela, o estímulo visual gera uma onda P300 (um pico aproximadamente 300 ms após a linha/coluna ter sido acesa); O sistema, sendo capaz de detectar a ocorrência de uma onda P300, é capaz de inferir a linha/coluna selecionada pelo usuário, e assim a letra a ser digitada [Brunner et al. 2011]. Muitos BCI que utilizam ondas P300 não são treinados, pois os estímulos por si só já se sobressaem aos sinais normalmente gerados. Entretanto, à medida em que estes estímulos diferentes se tornam normais ou mais presentes, os picos captados ficam menores, reduzindo a acurácia do sistema [Hwang, Lee e Lee 2017].

Por fim, outra abordagem frequente para BCI, utilizada também no presente trabalho, busca analisar sinais nas bandas de frequências que estão entre 7 e 30 Hz (ondas

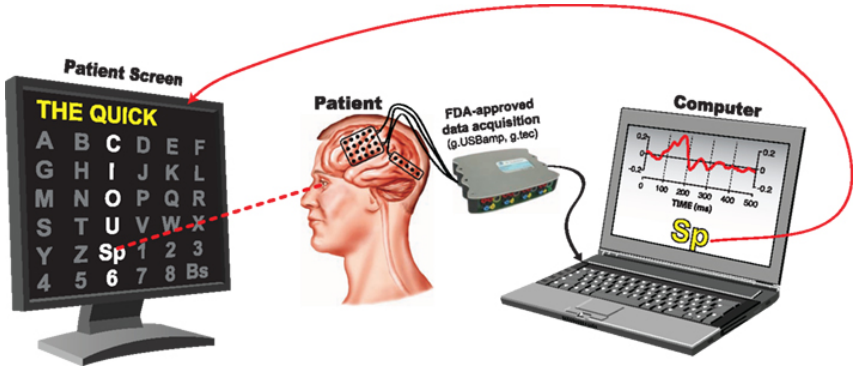

Figura 2: BCI baseada em potenciais evocados P300 (extraído de [Brunner et al. 2011].

Theta, Alpha e Beta)[Nicolas-Alonso e Gomez-Gil 2012]. Estas ondas estão relacionadas a atividades motoras, estimulação sensorial e imaginação de movimento. Estes potenciais podem gerar dois diferentes tipos de modulações; dessincronização relacionada a um evento (ERD, do inglês *event-related desynchronization*) e sincronização relacionada a um evento (ERS, do inglês *event-related synchronization*) que estão relacionadas a uma supressão da onda correspondente e um aumento na amplitude do sinal, respectivamente[Blankertz et al. 2008][Pfurtscheller et al. 2003]. Entretanto padrões oscilatórios complexos podem ser formados, como a junção de diversas frequências em diferentes estados de sincronização ou dessincronização e podem estar localizadas nas mais diversas áreas do cérebro [Jeannerod 1995][Pfurtscheller e Neuper 2001]. Dessa forma, não é possível associar facilmente uma determinada característica dos sinais observados a uma intenção do usuário (a exemplo do que ocorre nas interfaces baseadas em SSVEP e P300). Então, para o desenvolvimento de uma interface funcional baseada nos ritmos senso-motores, é necessário empregar técnicas adequadas para o processamento e classificação dos sinais,

2.2 Aquisição dos Sinais

Atualmente existem diversas técnicas para estudo da atividade neuronal, e que seriam passíveis de utilização em sistemas BCI, como a Ressonância Magnética Funcional, a Espectroscopia de Infravermelho próximo (do inglês, *Near Infrared Spectroscopy* - NIRS), a Magnetoencefalografia e a Eletroencefalografia. Se considerarmos alguns requisitos desejáveis para a BCI, como:

- Portabilidade: idealmente o sistema deve ser portátil, para facilitar o uso do equipamento;
- Tempo de resposta: qualquer que seja o método de aquisição dos sinais, o tempo de resposta deve ser suficientemente pequeno para que o sistema opere em tempo real;

- Baixo Custo: o equipamento não deve ter custo muito elevado, tendo em vista o uso em aplicações cotidianas, como controlar uma cadeira de rodas ou digitar um texto;

A escolha do método de aquisição dos sinais, em geral, recai sobre a Eletroencefalografia pois melhor atende os requisitos desejados.

As primeiras experiências para registro da atividade neuronal foram realizados por Hans Berger, em 1924, utilizando um galvanômetro para medir a atividade cerebral [Lotte, Bougrain e Clerc 2015], dando origem assim ao EEG. Desde então, uma série de novas técnicas, como a magnetoencefalografia e a ressonância magnética funcional, foram desenvolvidas para o estudo do cérebro, permitindo aos pesquisadores identificar e visualizar regiões específicas do cérebro de maneira bastante precisa. Entretanto, no desenvolvimento de interfaces cérebro-computador, o EEG não-invasivo permanece como técnicas mais popular para a aquisição dos sinais [Lotte, Bougrain e Clerc 2015].

No trabalho pioneiro de Hans Berger [Lotte, Bougrain e Clerc 2015], foram feitas diversas considerações qualitativas relacionando os tipos de sinais observados e o estado mental do indivíduo. Por exemplo, foi constatado que em diversas pessoas em estado de repouso, com olhos fechados, a forma de onda observada no EEG apresentava um ritmo (frequência) de aproximadamente 10Hz, e alterava seu ritmo logo após a abertura dos seus olhos. Tal relação entre o estado mental e determinados ritmos observados nos sinais de EEG deram origem a uma nomenclatura para as ondas observadas que é utilizada até hoje, como por exemplo [Rao 2013]:

- Ondas *Delta*: ondas que possuem frequências que variam de 0,5 a 4 Hz e estão presentes no EEG de bebês ou em indivíduos que estão dormindo;
- Ondas *Theta*: ondas que variam entre 4 e 8 Hz e estão associadas com a sonolência ou inatividade tanto em crianças quanto em indivíduos adultos;
- Ondas *Alpha*: também conhecidas como “ondas de Berger” são ondas que variam entre 8 e 13 Hz e normalmente são encontradas quando o indivíduo está acordado em estado relaxado e de olhos fechados;
- Ondas *Beta*: ondas que possuem frequências entre 13 e 30 Hz e surgem quando o indivíduo está se concentrando;
- Ondas *Gama*: ondas que variam entre 30 e 100 Hz são normalmente associadas à realização de tarefas que envolvem a memória de curta duração e tarefas onde há integração multissensoriais.

A figura 3 ilustra alguns dos ritmos mencionados anteriormente. Note que seria possível explorar tal associação entre os tipos de ondas presentes nos registros do EEG

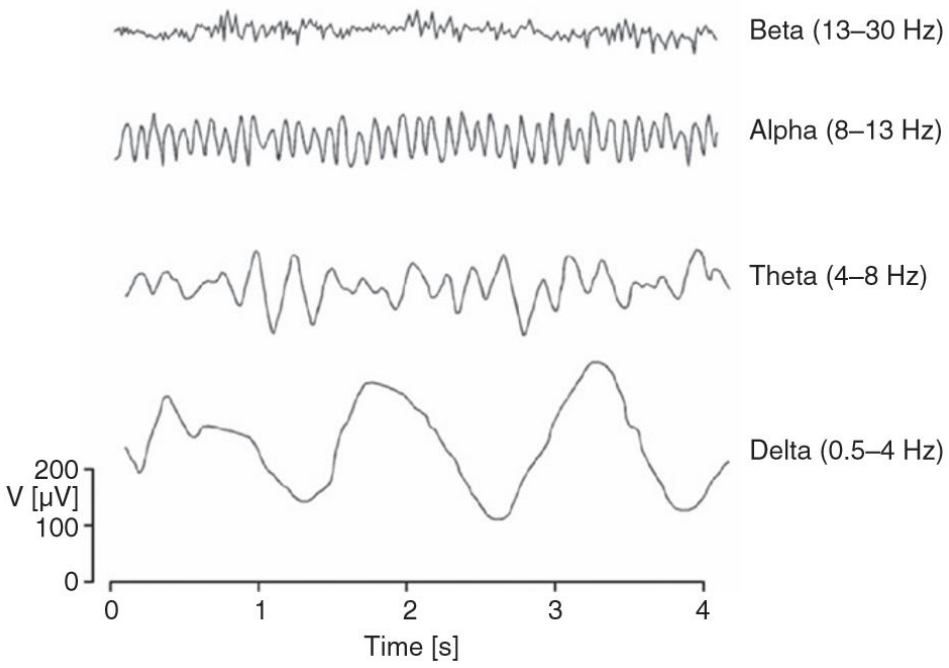

Figura 3: Exemplo de ondas adquiridas pelo EEG [Rao 2013].

com determinadas ações realizadas pelos indivíduos para construir uma interface cérebro-computador rudimentar. Por exemplo, considerando que o indivíduo consiga promover, conscientemente, o aumento ou diminuição das ondas a *Alpha* (e.g., relaxando e fechando os olhos para aumentar as ondas *Alpha*, e permanecendo alerta para diminuir), seria possível projetar um sistema que estima a potência dos sinais na faixa de frequência relacionada às ondas *Alpha* e assim prover um controle binário para acionamento de um dispositivo (por exemplo, comparando a potência dos sinais com um limiar). Na prática, no entanto, tal interface mostra-se bastante limitada, de maneira que métodos mais elaborados para a extração da informação a partir dos sinais neuronais têm sido investigados para aumentar a capacidade do sistema.

Atualmente, a utilização de EEG para captura dos sinais envolve o uso uma touca contendo eletrodos que serão posicionados em locais específicos do escalpo, em geral seguindo um mapa padronizado, como o apresentado na Figura 4. Os eletrodos podem ser bipolares ou unipolares, sendo que o primeiro método mede o potencial entre pares de eletrodos enquanto o segundo método cada eletrodo é comparado com um eletrodo neutro ou com a média de todos os eletrodos.

2.3 Pré-Processamento

Cada um dos eletrodos está ligado a amplificadores e filtros analógicos, com o intuito de condicionar o sinal para sua posterior conversão para o formato digital. O

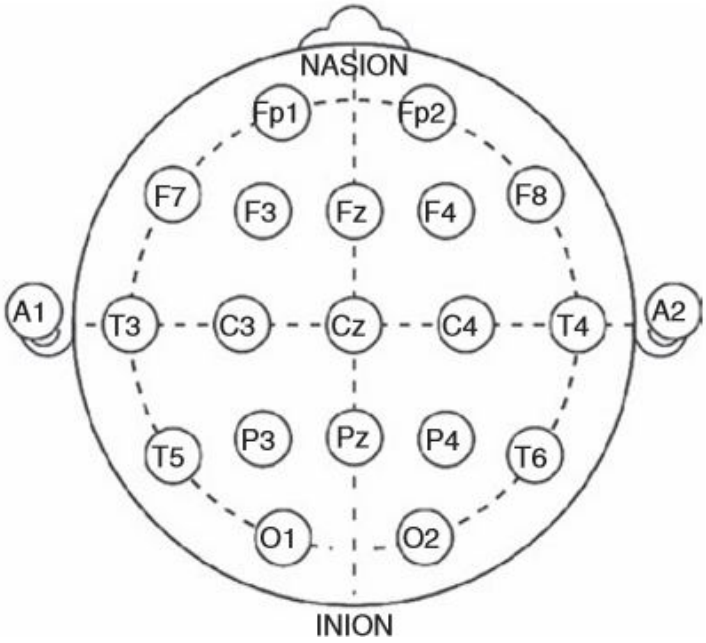

Figura 4: Exemplo de como os sensores são posicionados na cabeça [Rao 2013].

amplificador se faz necessário pois os sinais capturados pelo EEG possuem apenas alguns micro-volts de amplitude, e o filtro tem como objetivo remover sinais interferentes de frequências indesejadas no sistema [Rao 2013], como o ruído proveniente da alimentação da rede elétrica (60 Hz).

Após o devido condicionamento dos sinais, os sinais são convertidos para um sinal de tempo-discreto por meio de conversores analógico-digital, com taxa de amostragem adequada para garantir que nã há perda de informação no processo de conversão. Um resultado importante na teoria de processamento de sinais é expresso pelo teorema da amostragem [Smith 1999], estabelecendo que se o sinal de tempo-contínuo $x(t)$ não contém frequências maiores do que BHz, é possível reconstruir perfeitamente o sinal de tempo-contínuo a partir de suas amostras desde que a taxa de amostragem utilizada tenha sido maior do que $2BHz$ [Smith 1999]. Usualmente, a taxa de amostragem dos equipamentos comerciais se encontra entre $200-500Hz$, o que garante uma grande qualidade na representação dos sinais, tendo em mente que a faixa de interesse para análise dos sinais de EEG dificilmente se estende acima de 100 Hz, como vimos na seção 2.2[Rao 2013].

Uma grande variedade de técnicas de processamento digital de sinais podem ser empregadas para tentar melhorar a qualidade dos sinais de interesse, mas é possível agrupá-las, de uma maneira geral, em dois grandes grupos: técnicas de filtragem espectral e filtragem espacial [Alamdari et al. 2016].

No primeiro grupo se encontram os filtros digitais, projetados de maneira a atender uma especificação fixa de resposta em frequência, em geral definidas de maneira a eliminar sinais interferentes (a exemplo do filtro analógico para remoção da componente em 60

Hz) ou mesmo para limitar a banda de análise dos sinais de interesse. Existe uma vasta literatura abordando o projeto e implementação de filtros digitais (vide, e.g., [Oppenheim, Willsky e Nawab 1996], [Sanei e Chambers 2007]). No presente trabalho, foram utilizados filtros lineares, caracterizados por uma resposta ao impulso de duração finita FIR (do inglês, *Finite Impulse Response*)[Treichler 2012]. Há diversas características interessantes relacionadas a filtros FIR, dentre as quais pode-se destacar a garantia de estabilidade, sua facilidade de implementação e a possibilidade de serem projetados para ter uma fase linear, não introduzindo assim distorções de fase nos sinais a serem analisados.

No segundo grupo, os sinais de diferentes canais são combinados de maneira a produzir sinais com características mais favoráveis à análise e/ou classificação, no caso de sistemas BCI. A combinação dos sinas pode ser feita de diferentes maneiras, visando a otimização de algum critério específico baseado em medidas dos sinais processados. No presente trabalho, foi utilizada a técnica denominada como Padrões Espaciais Comuns - CSP (do inglês, *Common Spatial Pattern*), descrita em maiores detalhes a seguir.

2.3.1 CSP - *Common Spatial Pattern*

O método CSP (Common Spatial Pattern) foi primeiramente introduzido no contexto de análise de sinais de EEG por [Koles, Lazar e Zhou], e tem sido utilizado frequentemente em sistemas BCI baseados em imagética motora. A ideia explorada no CSP consiste em utilizar uma transformação linear para projetar os dados de entrada em diferentes subespaços, de maneira a facilitar a tarefa de classificação dos padrões associados a cada estado mental [Liyanage et al. 2010].

Matematicamente, considere que dois conjuntos de sinais de EEG, C_1 e C_2, com dimensões $N \times T$, onde N é o número de canais e T é o número de amostras por canal, associados a duas classes distintas, Classe 1 e Classe 2, respectivamente. Dessa forma, cada um dos canais do EEG está representado em uma linha da matriz C_i, e uma estimativa para a matriz de covariância espacial das classes podem ser representadas como:

$$R_{C1} = \frac{C_1 C_1^T}{\mathrm{tr}(C_1 C_1^T)} \tag{2.1}$$

$$R_{C2} = \frac{C_2 C_2^T}{\mathrm{tr}(C_2 C_2^T)} \tag{2.2}$$

onde A^T representa a matriz transposta de A e $\mathrm{tr}(A)$ representa o traço da matriz A, i.e., a soma dos elementos diagonais [Grosse-Wentrup* e Buss 2008].

O algoritmo CSP busca por projeções dos dados que maximizem a distinção das duas classes resolvendo o seguinte problema de otimização:

$$w_{opt} = \underset{w \in \Re^N}{argmax} \frac{w^T R_{C1} w}{w^T R_{C2} w} \tag{2.3}$$

onde N representa a dimensão dos dados de entrada (e.g., o número de canais do EEG). Intuitivamente, o algoritmo procura obter direções nas quais a variância da projeção dos dados associados à classe 1 é maximizada, tentando minimizar a projeção dos dados associados à classe 2.

A equação (2.3) corresponde a um quociente de Rayleigh [Golub e Loan 1996], para o qual a solução pode ser obtida por meio do problema de autovalor generalizado, i.e., encontrado w tal que:

$$R_{C1}w^T = \lambda R_{C2}w^T \tag{2.4}$$

onde λ representa o autovalor. Os autovetores associados ao problema são descritos em (2.5) correspondem aos filtros espaciais desejados. É interessante ressaltar que, para um dado filtro espacial w^*, o valor do autovalor associado é dado pela função custo (2.3), ou seja,

$$\lambda^* = \frac{w^{*T}R_{C1}w^*}{w^{*T}R_{C2}w^*} \tag{2.5}$$

indicando que o autovalor serve como um indicador da "qualidade" do filtro espacial obtido. Dessa forma, os autovetores associados aos maiores autovalores correspondem a filtros mais seletivos a sinais pertencentes à classe 1, e os associados aos menores autovalores correspondem a filtros mais seletivos a sinais pertencentes à classe 2.

2.3.2 CSP Multi-Classe

Embora a formulação obtida para o CSP considerando apenas duas classes possua resultados teóricos que garantam soluções ótimas, no caso em que há mais de duas classes a serem separadas não existe uma forma canônica de estender os resultados obtidos anteriormente.

Uma possível abordagem seria aplicar o CSP a cada par de classes existentes, o que poderia levar a um aumento considerável na complexidade da solução no caso de um número grande de classes a serem separadas.

Outra alternativa, explorada no presente trabalho, apoia-se em uma peculiaridade observada na solução do CSP para duas classes. Como as soluções para o problema de otimização descrito em (2.3) são dadas pelos autovetores de um problema de autovalor generalizado (2.5), é possível mostrar que o conjunto de autovetores diagonaliza tanto a matriz R_{C1} como R_{C2} [Grosse-Wentrup* e Buss 2008].

Dessa maneira, o problema de otimização em (2.3) pode ser encarado como um problema de diagonalização conjunta das matrizes R_{C1} e R_{C2}, i.e., os filtros espaciais correspondem às colunas de uma matriz W tal que

$$WR_{Ci}W^T = D_i \tag{2.6}$$

onde D_i denota uma matriz diagonal, para $i = 1, 2$. Com isso, uma possível extensão do conceito do CSP para mais de duas classes consiste em formular o problema de otimização em termos de um problema de diagonalização conjunta de várias matrizes, R_{Ci}, onde a matriz R_{Ci} corresponde à matriz de covariância dos dados associados à classe $i = 1, 2, \ldots, n$.

No caso específico de duas classes apenas, em geral, é possível obter soluções exatas para o problema de otimização. Entretanto, no caso de mais de duas classes, não é possível garantir que as soluções encontradas diagonalizam perfeitamente as matrizes de covariância. Por esse motivo, utiliza-se uma técnica de otimização denominada Diagonalização Conjunta Aproximada - JAD (do inglês, Joint Approximate Diagonalization, JAD) [Grosse-Wentrup* e Buss 2008] para se obter os filtros espaciais.

2.4 Extração e Seleção de Características

A classificação dos sinais neurais em geral é realizada com base em algumas características específicas dos sinais coletados, que permitem identificar mais facilmente a qual classe o conjunto de sinais pertence. Por exemplo, em um sistema automático de reconhecimento de cédulas a partir de imagens, ao invés de alimentar o sistema com as imagens brutas, pode-se realizar um pré-processamento para a extração de informações tais como a cor predominante da cédula, detectar as bordas da cédula e determinar a sua razão de aspecto, entre outras informações relevantes, e a partir de tais características determinar qual o valor da cédula presente na imagem.

Como pode-se imaginar, a escolha por um tipo particular de características depende fortemente do problema em mãos. No caso específico de sistemas BCI baseados em imagética motora, há diferentes abordagens na literatura explorando características espectrais dos sinais, os coeficientes da transformada Wavelet Discreta [Mohammadpour, Ghorbanianm e Mozaffari 2016], características de recorrência dos sinais [Soriano et al. 2014], entre outras [Norani e Md 2010].

No presente trabalho, faz-se uso da densidade espectral de potência como ferramenta para a extração de características dos sinais, que provê informação relevante sobre de que maneira a potência se encontra distribuída na faixa de frequências dos sinais. As figuras 5 e 6 apresentam um exemplo de representação de um sinal no tempo e a sua densidade espectral de potência, respectivamente. A informação no domínio da frequência revela que o sinal analisado possui a energia concentrada em uma única frequência (ou uma faixa bastante estreita), indicando que se trata, de fato, de um sinal senoidal (ou próximo a um sinal senoidal).

Determinar quais características são mais relevantes para o processo de classifica-

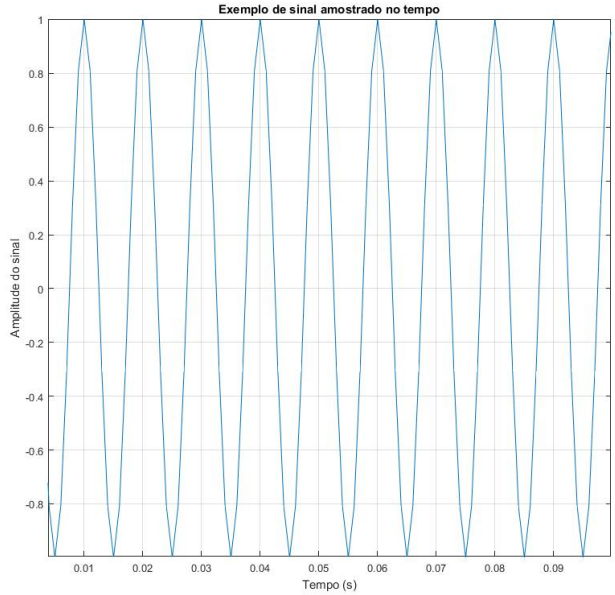

Figura 5: Exemplo de sinal amostrado por um conversor analógico digital.

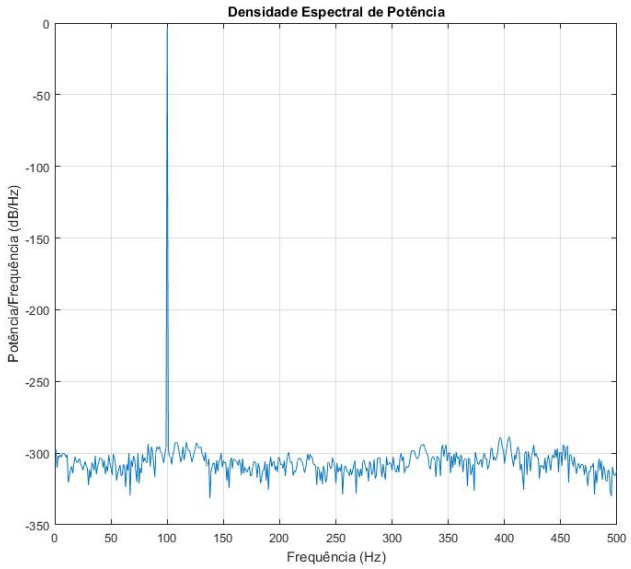

Figura 6: Exemplo da densidade espectral de potência do sinal da figura 5. Pode-se notar que a no eixo das abscissas encontram-se as frequências do sinal. Neste caso, o sinal possui uma frequência de 100 Hz.

ção é um ponto crucial no projeto de sistemas de reconhecimento de padrões, e possui um grande impacto na acurácia da classificação. Além disso, em geral, a extração de características traz consigo o benefício de redução de dimensionalidade dos dados de entrada do classificador, tornando a solução mais simples de ser implementada e, não raramente, mais robusta[1].

[1] Cabe comentar, no entanto, que o efeito de redução da dimensão dos dados utilizados para a classificação também pode ser obtida por meio de métodos de análise de dados multivariados, que buscam

O método de filtragem espacial descrito anteriormente, nesse sentido, pode ser considerado um método de extração de características [Shin et al. 2013] [Ameri, Pouyan e Abolghasemi 2016], uma vez que os dados de entrada são transformados pelos filtros espaciais obtidos no processo de otimização, gerando assim diferentes projeções (características) que são então utilizadas para realizar a classificação. Entretanto, mesmo que seja possível ordenar os filtros espaciais obtidos pelos autovalores associados, não é possível se determinar, *a priori*, quantos filtros devem utilizados para gerar as características - é necessário, portanto, realizar algum procedimento para selecionar quais das características disponíveis seriam as características mais relevantes para a classificação dos sinais.

Na literatura é possível encontrar diferentes metodologias para a seleção de características, com a finalidade de maximizar a acurácia da classificação dos dados e a generalização dos resultados quando operando com dados não utilizados no treinamento. Há três principais métodos para a seleção de características:

- Método de Filtragem: nessa abordagem, utiliza-se uma medida estatística para avaliar cada uma das características. Por exemplo, pode-se avaliar a correlação de uma determinada característica com a saída desejada para o classificador, de maneira que as variáveis que apresentam maior correlação devem, em teoria, contém informação mais relevante para o processo de classificação. Isso, entretanto, pode levar a seleção de características redundantes, uma vez que a abordagem não considera possíveis relações entre as variáveis;

- Método *Wrapper*[Kohavi e John 1997]: nesse método, utiliza-se um modelo para a seleção dos atributos. Por exemplo, no caso de classificação de padrões, o método utiliza um modelo para o classificador e, iterativamente, efetuar um teste com subconjuntos das características, verificando o desempenho obtido com a classificação. Há diferentes estratégias para a escolha das características: por exemplo, na seleção progressiva (do inglês, *forward selection*), iterativamente, o algoritmo acrescenta ao conjunto de características selecionadas o atributo que leva à maior melhoria no desempenho do classificador, até que acrescentar novos atributos não leve a melhorias significativas; de maneira similar, na abordagem de eliminação retrógrada (do inglês, *backward elimination*), inicia-se com todos os atributos e, iterativamente, elimina-se o atributo menos significativo (que possui o menor impacto no desempenho ao ser eliminado do conjunto de características).

- Método *Embedded*[Blum e Langley 1997]: nesse caso, o processo de criação do modelo (classificador) e seleção de características é feito de maneira integrada, buscando

representações alternativas para o conjunto de dados coletado explorando critérios estatísticos, como a Análise de Componentes Principais - PCA (do inglês, *Principal Component Analysis*) e a Análise de Componentes Independentes - ICA (do inglês, *Independent Component Analysis* [Romano et al. 2009].

aproveitar as vantagens dos dois métodos descritos anteriormente.

2.5 Classificação

A etapa final no processamento dos sinais em uma interface cérebro-computador se refere ao processo de classificação, e diversas abordagens podem ser encontrados na literatura [Alamdari et al. 2016] [Mohammadpour, Ghorbanianm e Mozaffari 2016] [Norani e Md 2010]. Essencialmente, o classificador deve ser uma estrutura flexível, capaz de gerar um mapeamento que associe os vetores de características obtidos e a sua classe correspondente.

Métodos usualmente empregados em sistemas de BCI incluem [Alamdari et al. 2016]:

- k-Nearest Neighbors (K-NN) [Saugat et al. 2010];
- Support Vector Machine (SVM) [Qin, Li e Sun 2007];
- Classificador Naive Bayes (NBC) [Bassani e J.C. 2010];
- Análise Discriminante Linear (LDA) [Saugat et al. 2010];
- Perceptron de multi-camada (MLP) [Nurse et al. 2015];
- Modelo oculto de Markov (HMM) [Obermaier, Guger e Pfurtscheller 1999];
- Ensemble de classificadores [Silva et al. 2017].

A escolha por um tipo particular de classificador depende de vários fatores, mas em geral busca-se um compromisso entre o custo computacional e seu o desempenho em termos do erro de classificação. Entretanto, é necessário ressaltar que as etapas previamente descritas envolvendo o pré-processamento dos sinais e a extração de características podem influenciar significativamente os resultados finais de classificação. Além disso, outros requisitos desejados para a aplicação, como a operação em tempo real, podem ter um peso decisivo na escolha por um ou outro método de classificação.

No presente trabalho, considerando a motivação inicial de estudar um método para interfaces cérebro-máquina baseados em imagética motora, optou-se pela utilização do algoritmo k-NN, que apresenta um bom compromisso entre complexidade computacional e desempenho de classificação. Dessa forma, na sequência, é discutido em maiores detalhes o algoritmo implementado.

2.5.1 Algoritmo *k-nearest neighbors*

O algoritmo k-NN (do inglês, k-nearest neighbor, ou em português, k vizinhos mais próximos) é um dos 10 algoritmos mais utilizados em mineração de dados e aprendizado de máquina, e com isso, tem sido amplamente utilizado em reconhecimento de padrões [Wu et al. 2008]. De construção simples, a ideia explorada pelo algoritmo consiste em "memorizar"os dados de treinamento e utilizá-los para determinar a qual classe um novo dado de entrada pertence.

A estratégia para se determinar a qual classe pertence o novo exemplar se baseia na noção de distância do novo vetor de dados e os vetores de treinamento, incorporados ao modelo do classificador. O algoritmo, então, determina quais são os k vetores de treino mais próximos do vetor de entrada, e classifica vetor de entrada de acordo com a classe da maioria dos k vizinhos mais próximos (de onde se origina o nome do método). Em outras palavras, o algoritmo k-NN encontra um grupo de k objetos no conjunto de treinamento que estão mais próximos do vetor de teste e baseia a sua classificação na predominância da classe que estiver nesta vizinhança.

A figura 7 ilustra o funcionamento do algoritmo. No exemplo apresentado, o novo padrão (em preto) é comparado a todos os padrões armazenados, e os $k = 4$ padrões mais próximos são selecionados. Dentre os padrões mais próximos, dois deles pertence à classe “vermelho”, e dessa forma o novo padrão de entrada também recebe o mesmo rótulo. O pseudo-código do k-NN é apresentado no algoritmo 1.

Algorithm 1 Algoritmo k vizinhos mais próximos.

D é um conjunto de todos objetos de treinamento.
z é o objeto de teste ou validação.
procedure K-NN(D,z)
D é uma matriz contendo todos os objetos e suas posições.
O objeto z, é o objeto que se deseja classificar.
Computar a distância d, de z e todos os objetos $\in$ D.
Selecionar os k objetos mais próximos de z, que foram computados previamente.
fim procedure
devolve Classificação do objeto será a maioria dos objetos em D_z.

O método se mostra muito eficiente em relação a outros métodos como o classificador bayesiano [Cover e Hart 1967], quando não se possui dados suficientes, no entanto possui desvantagens como alta sensibilidade pelos k vizinhos do vetor a ser previsto e a complexidade computacional para um número muito grande de dados para treinamento. O bom desempenho do algoritmo, cujo pseudo-código é apresentado no algoritmo 1, depende de três elementos chaves: a definição para a métrica de distância entre os vizinhos mais próximos, o conjunto de treinamento armazenado e a quantidade de vizinhos a ser analisado.

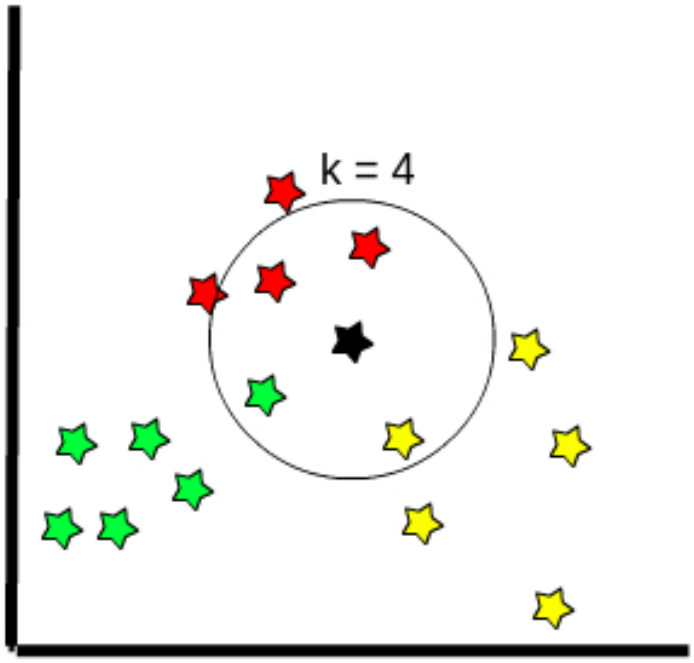

Figura 7: Exemplo prático algoritmo K-NN, onde k=4 a estrela preta é a principal e classificação escolhida, por maioria, será vermelho.

3 Representação Esparsa dos Sinais

Um sinal esparso é fundamentalmente aquele em que predominam, no domínio temporal ou em algum outro domínio relevante, valores nulos ou próximos de zero, sendo que poucos componentes possuem a maior parte da energia do sinal [Peleg e Elad 2012]. Em outras palavras, o conceito de esparsidade nos remete a sinais em que toda a informação está concentrada em uma quantidade pequena de valores que representam aquilo que se analisa [Wright et al. 2009].

Muitos sinais, embora não apresentem elevado grau de esparsidade no tempo, são esparsos em algum domínio especifico - como no domínio frequencial ou tempo-frequencial [Nadalin, Suyama e Attux 2010]. Entretanto, em algumas aplicações específicas, não é possível determinar, *a priori*, qual seria a transformação mais adequada para se gerar uma representação esparsa. Nesses casos, quando se dispõe de dados para treinamento, é possível construir uma base específica - um dicionário - com a qual todos os dados podem ser representados de maneira esparsa.

Nesse capítulo serão discutidos os conceitos fundamentais relacionados à representação esparsa dos sinais, bem como os algoritmos para obtenção dos dicionários utilizados na representação.

3.1 Sinais Esparsos

Uma maneira de se quantificar o grau de esparsidade é determinar quantas amostras ou quantos coeficientes do sinal possuem valores não-nulos, noção que está relacionada à norma ℓ_0[1] do vetor. Como em sinais reais dificilmente observam-se coeficientes exatamente nulos, é possível definir um limiar a partir do qual o coeficiente é considerado nulo, levando à definição de outras normas, como a norma $\ell_{0\varepsilon}$[Nadalin, Suyama e Attux 2010].

Matematicamente, as normas são definidas como funções $\|\cdot\|$ que associam um número estritamente positivo para cada vetor, i.e., um mapeamento de $\mathbb{R}^n \rightarrow \mathbb{R}$, e que possuem as seguintes propriedades [Laub 2005]:

- $\|x\| \geq 0$;
- $\|x\| = 0$ se e apenas se $x = 0$;
- $\|\alpha x\| = |\alpha| \, \|x\|$ para todo $\mathbb{R}^n$ e todo $\alpha \, \epsilon \, \mathbb{R}$;

[1] Cabe comentar que a norma ℓ_0 é considerada uma quase norma, ou seja, é uma função cardinal onde sua definição corresponde ao número de números não nulos no vetor.

- $\|x + y\| \leq \|x\| + \|y\|$ para qualquer vetor onde $x, y \, \epsilon \, \mathbb{R}^n$.

As normas de vetores mais utilizadas pertencem a família das normas p ou normas l_p que são definidas por:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \tag{3.1}$$

Pode ser demonstrado que, para qualquer $0 < p < 1$, $\|\cdot\|_p$ é definido como uma norma de vetor. Conforme comentado anteriormente, a norma ℓ_1 utilizada no contexto de representação esparsa, é obtida considerando $p = 1$, i.e.:

$$\|x\|_1 = |x_1| + |x_2| + ... + |x_n| \tag{3.2}$$

enquanto que a norma euclideana, que fornece a noção habitual de comprimento de um vetor, é obtida considerando $p = 2$, ou seja:

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + ... + x_n^2} \tag{3.3}$$

Em alguns casos, é interessante definir também a norma para $p \to \infty$, gerando a norma ℓ_∞:

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i| \tag{3.4}$$

De maneira similar, é possível definir normas para matrizes, utilizadas para se obter uma noção do tamanho ou da proximidade entre matrizes [Laub 2005]. Assim, de maneira análoga à definição de normas para vetores, a norma de uma matriz A, sendo $A \, \epsilon \, \mathbb{R}^{\{m,n\}}$, deve obedecer a um conjunto de propriedades, a saber:

- $\|A\| \geq 0$;
- $\|A\| = 0$ se e apenas se $A = 0$;
- $\|\alpha A\| = |\alpha| \, \|A\|$ para todo $\mathbb{R}^{\{m,n\}}$ e todo $\alpha \, \epsilon \, \mathbb{R}$;
- $\|A \; + \; B\| \leq \|A\| + \|B\|$ para todo $A, B \, \epsilon \, \mathbb{R}^{\{m,n\}}$.

As normas de matrizes mais utilizadas pertencem à família das normas p (ou normas ℓ_p) que são definidas por:

$$\|A\|_p = \max_{\|x\|_p \neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \max_{\|x\|_p = 1} \|Ax\|_p \tag{3.5}$$

para todo $A \epsilon \mathbb{R}^{m,n}$. Alguns casos particulares são [Laub 2005] a *norma* ℓ_1:

$$\|A\|_1 = \underset{j \ \epsilon \ n}{max} \left(\sum_{i=1}^{m} |a_{ij}| \right) \tag{3.6}$$

a *norma* ℓ_2, a norma euclidiana matricial ou norma de Frobenius (que é um limitante da norma ℓ_2, também representada por ℓ_F):

$$\|A\|_2 = (Tr(AA^T))^{\frac{1}{2}} \tag{3.7}$$

e a *norma* ℓ_∞, definida por:

$$\|A\|_\infty = \underset{j \ \epsilon \ m}{max} \left(\sum_{i=1}^{n} |a_{ij}| \right) \tag{3.8}$$

No contexto do presente trabalho, as normas de matrizes surgem na apresentação dos algoritmos utilizados na criação de dicionários para gerar representações esparsas, apresentados na Seção 3.3.

3.2 Algoritmos para Obtenção da Representação Esparsa

Obter uma representação esparsa para um dado sinal requer que seja definido uma base, i.e., um conjunto de vetores que uma vez combinados adequadamente representam o sinal desejado. Por exemplo, o conjunto de sinais senoidais harmonicamente relacionados forma uma base para representação de qualquer sinal periódico - resultado esse associado à expansão série de Fourier [Oppenheim, Willsky e Nawab 1996]. Assim, o conjunto de vetores base é utilizado como um dicionário para a representação do sinal.

Deseja-se adicionalmente que a representação do sinal utilizando o dicionário escolhido seja esparsa (possua poucos coeficientes significativamente diferentes de zero), problema que pode ser formulado matematicamente como um problema de otimização dado por:

$$\underset{x}{min} ||x||_0 \quad \text{sujeito a} \quad y = Dx \tag{3.9}$$

onde, x representa o vetor contendo a representação esparsa, y o sinal original e D a matriz contendo o dicionário (cada uma de suas colunas representa um vetor base). O problema de otimização proposto impõe uma restrição severa, no sentido de que qualquer solução encontrada para x deve representar perfeitamente o sinal observado. Na prática, essa condição é raramente possível de ser atendida, de maneira que uma forma de se contornar tal problema é relaxar a restrição imposta, alterando o problema de otimização para:

$$\underset{x}{min} ||x||_0 \quad \text{sujeito a} \quad ||y = Dx||_2 \leq \epsilon \tag{3.10}$$

Estes algoritmos foram criados para que, dado um dicionário esparso D e um vetor y consiga-se extrair o vetor esparso x:

$$x = D^{-1} y \tag{3.11}$$

Mesmo que o problema de otimização tenha tido sua restrição amenizada, ainda configura-se como um problema de difícil solução, e diferentes algoritmos foram propostos na literatura para tal fim. Os algoritmos com tal finalidade também são denominados de algoritmos de seleção de vetor esparso. No presente trabalho, foram selecionados alguns dos principais métodos encontrados na literatura, e são descritos na sequência.

3.2.1 Orthogonal Matching Pursuit

O algoritmo OMP (do inglês, *orthogonal matching pursuit*) é um algoritmo do tipo *greedy* (ambicioso) de regressão [Aharon, Elad e Bruckstein 2006]. O algoritmo inicia a operação selecionando o vetor base que mais se assemelha ao vetor original (considerando as colunas da matriz D foram normalizadas, calcula-se o produto interno de cada coluna da matriz D com o vetor x para isso). O valor do coeficiente correspondente em y é calculado de maneira a minimizar o erro residual quadrático da representação, dado por:

$$\| x\, D - y \|^2 \tag{3.12}$$

Na etapa seguinte, o algoritmo seleciona a coluna que apresenta a maior similaridade com o erro residual, e recalcula o valor dos coeficientes em x, visando minimizar o novo erro residual (note que nessa etapa a aproximação é obtida pela combinação linear de dois vetores do dicionário). Dessa forma, o algoritmo continua a realizar, a cada estágio, a seleção de um elemento do dicionário e a atualização dos coeficientes, realizada por mínimos quadrados. O critério de parada do algoritmo pode ser definido pelo número máximo de iterações, i.e., definindo o número máximo de átomos a ser utilizado na representação, ou ainda por um critério que avalie o erro na representação, utilizando a norma do resíduo [Hameed 2012][Aharon, Elad e Bruckstein 2006]. O algoritmo completo, utilizado no presente trabalho, é descrito no Algoritmo 2, onde nele se nota todo o processo de escolha de coluna com maior similaridade, passo a passo, assim como a inversão da respectiva matriz, com colunas similares, para a formação de um novo vetor esparso, obedecendo a equação 3.11.

Algorithm 2 OMP - Orthogonal Matching Pursuit

função OMP(D,y)
  $D \in \Re\{n, m\}$
  Define-se máxima esparsidade, que terá no máximo metade do vetor y:
  $\underline{m} \leftarrow n/2$
  Inicializa vetor residual como y:
  $r \leftarrow y$
  Inicializa vetor esparso:
  $x \leftarrow 0$
  Inicializa contador:
  $\underline{j} \leftarrow 0$
  enquanto não obedecer critérios de parada ou $||x||_0 < \underline{\mathbf{m}}$ **faça**
    Escolha de projeção
    $z \leftarrow D^T r$
    Escolha de coluna com maior similaridade:
    $p_{\underline{j}} \leftarrow$ posição de $arg\_max(z)$
    Inicializar $M \in \Re\{n, m\}$:
    $M \leftarrow 0$
    $M_{:,p} \leftarrow D_{:,p}$
    A partir da inversa de M, atualiza-se x:
    $x \leftarrow M^+ x$
    $r \leftarrow x - (Mx)$
    $\underline{j} \leftarrow \underline{j} + 1$
  fim enquanto
fim função
devolve Vetor esparso: **x**

3.2.2 Least-Angle Regression

O algoritmo de LARS (do inglês, Least Angle Regression) é outro algoritmo que pode ser utilizado para obtenção da representação esparsa, e foi originalmente proposto como método para ajuste de modelos lineares em cenários envolvendo dados de alta dimensão [Hameed 2012]. Assim como o OMP, o algoritmo LARS inicia com um vetor com todos os elementos nulos, vetor de erro residual r igual ao vetor dado y, e iterativamente adiciona uma das colunas da matriz de dicionário D com vetor base utilizado para a representação esparsa de y.

A ideia básica explorada nas etapas executadas pelo algoritmo é ilustrada na figura 8, onde é ilustrado o vetor dado y e duas possíveis colunas da matriz D, d_1 e d_2.

Em sua primeira iteração, o algoritmo seleciona a coluna de D que apresenta maior correlação (forma o menor ângulo) com o vetor de erro residual r, de maneira análoga ao OMP (No caso ilustrado, o vetor selecionado corresponde a d_1). Ajustando-se o coeficiente associado ao vetor base d_1, verifica-se que, na medida em que o seu valor γ aumenta, o ângulo entre o vetor de erro residual e o vetor dado y aumenta. O valor de γ é ajustado

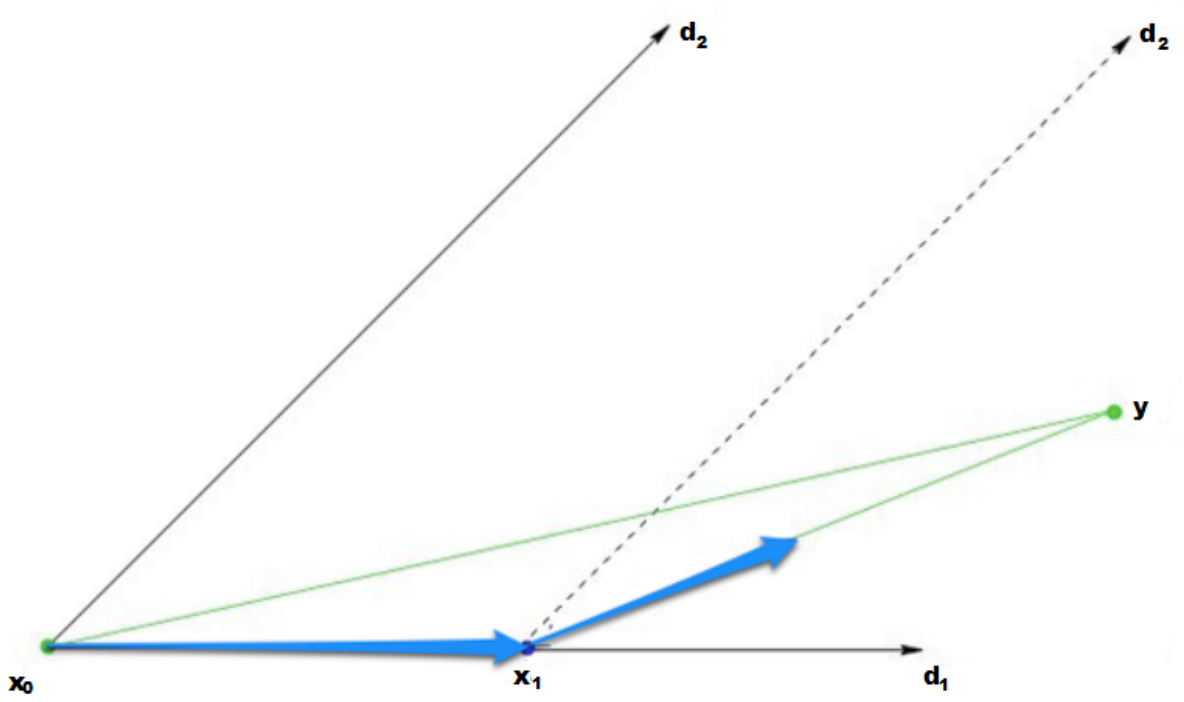

Figura 8: O modelo começa com a resposta de vetor esparso inicial x_0, a linha residual verde nos mostra no exemplo que a coluna d_1 possui um ângulo menor que a coluna d_2, portanto o algoritmo utiliza da coluna d_1 do dicionário D. Deste modo, algoritmo muda de direção de acordo com ângulos residuais diferentes, onde x se aproxima cada vez mais de y dado.

até que alguma outra coluna de D apresente maior correlação (forma o menor ângulo) com o erro residual, que no exemplo corresponde a d_2.

Na iteração seguinte, o algoritmo prossegue o ajuste do segundo coeficiente da representação esparsa, mas seguindo uma direção equiangular em relação aos dois vetores base já selecionados, até que uma terceira coluna de D apresente maior correlação com o vetor de erro residual, e assim sucessivamente com as demais colunas do dicionário [Bradley et al. 2004]. O critério de parada do algoritmo é geralmente definido em termos da magnitude do vetor de erro residual ou pela quantidade de vetores base utilizados na representação. A descrição do método é apresentação no Algoritmo 3. Percebe-se através do algoritmo 3 e o algoritmo 2 que ambos são similares, porém, o método LARS possui mecanismos mais sensíveis para a localização de novas colunas que possam criar um melhor vetor esparso.

Algorithm 3 Algoritmo LARS.

função LARS(y,D)
  $D \in \Re\{n, m\}$
  $x \leftarrow 0$ - Vetor esparso inicializado com zeros
  $\underline{\lambda} \leftarrow arg\_max(D^T\, y)$ - É o maior valor no vetor
  enquanto não obedecer critérios de parada faça
    Escolha de melhor direção:
    $z \leftarrow \frac{D^T(y - D*x)}{\underline{\gamma}}$
    $\underline{j} \leftarrow$ posição de $arg\_max(z)$ - Posição do argumento máximo
    Atualização do vetor esparso:
    $x_j \leftarrow \frac{D^T_{:,j} y - \underline{\gamma} * sign(x_j)}{D^T_{:,j} D_{:,j}}$
    se $||y - D\,x||_2^2 <$ critério de parada) **então**
      Fim de algoritmo de obtenção de vetor
    fim se
    Escolher: $0 < \underline{\alpha} < 1$ - Fator que melhor se encaixe
    $\underline{\gamma} \leftarrow \underline{\gamma} - \underline{\alpha}\,\underline{\gamma}$ - Fator de decrescimento
  fim enquanto
fim função

devolve Vetor esparso: **x**

3.3 Criação de Dicionário

Tanto o algoritmo OMP quanto LARS baseiam-se na existência de um dicionário D com o qual é possível representar o vetor esparso, dado y. A qualidade da representação, bem como o nível de esparsidade da solução encontrada, dependem não apenas da capacidade dos algoritmos de encontrar os coeficientes da representação, mas principalmente dos vetores presentes no dicionário. Em algumas aplicações, por exemplo, é usual utilizar vetores base pré-definidos - como as funções base da transformada de Fourier ou de uma transformada Wavelet. Entretanto, em várias outras situações, é interessante que o próprio dicionário seja construído especificamente para a aplicação em mãos. Dessa maneira, é possível encontrar na literatura diferentes propostas para realizar o Aprendizado de Dicionário, i.e., métodos para criação da matriz D de modo que o dicionário se adapte para responder ao seu propósito, por exemplo, um dicionário capaz de representar um subespaço específico de sinais [Skretting 2011]. A maioria das técnicas de criação de dicionário consiste em utilizar dois estágios para seu aprendizado: obtenção dos coeficientes para a representação esparsa e atualização de dicionário [Dai, Xu e Wang 2012]. No primeiro estágio, o objetivo é encontrar uma matriz com vetores de representação esparsa dos vetores originais, minimizando $\|Y - DX\|_F^2$, como apresentado anteriormente. No segundo estágio, dada a representação obtida com o dicionário inicial, é realizada a atualização de D, buscando promover uma nova representação para o vetor y com maior esparsidade. A ideia descrita anteriormente é sintetizada no Algoritmo 4.

Algorithm 4 Típico algoritmo criação de dicionário

1: **função** CRIAÇÃO DE DICIONÁRIO
2: Tarefa: encontrar o melhor dicionário que represente os dados da matriz Y
3: Inicialização: Começar com um dicionário randômico.
4: **enquanto** não convergir **faça**
5: Encontrar uma matriz esparsa X com o dicionário atual D
6: Atualizar dicionário D utilizando matriz esparsa X para que se encontre o mínimo local.
7: **fim enquanto**
8: **fim função**

Para exemplificar, de forma parcimoniosa, o funcionamento dos métodos para criação de dicionários, suponha que um vetor de tamanho $n = 6$ a ser representado utilizando um dicionário D, de tamanho $n = 6 \times m = 10$. Se com este dicionário for possível representar, de maneira parcimoniosa, o vetor de entrada y, utilizando qualquer algoritmo para obtenção dos coeficientes da representação, obteremos um vetor x, esparso, com o tamanho de $n = 10$.

No vetor esparso, x, cada elemento representará a proporção que deve ser utilizada de cada coluna de D. Suponha que o sinal a ser representado é dado por

$$y = [0, 0, 0, 1, 1, -2]^T \tag{3.13}$$

Considerando que a matriz D é dada por

$$D = \begin{bmatrix} 0 & 1 & 1 & -1 & -1 & 0 & -1 & 0 & -1 & 0 \\ -1 & -1 & 0 & 1 & -1 & 0 & 0 & -1 & 0 & 1 \\ 1 & -1 & 1 & -1 & 0 & -1 & 1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 & -1 & -1 & 1 & 1 \\ -1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & -1 \\ 0 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & 0 \end{bmatrix} \tag{3.14}$$

Um algoritmo de seleção de vetor seria encarregado de encontrar a representação esparsa do sinal para o dicionário D, que neste caso é dada por:

$$x = [0, 0, 1, 0, 0, 1, 0, 0, 1, 0]^T \tag{3.15}$$

O vetor x, representa exatamente quais as colunas de D estão sendo utilizadas para a reconstrução de y. Nesse caso, o vetor x pode ser reconstruído perfeitamente utilizando o dicionário D e o vetor y. Normalmente soluções criadas para criação de dicionários são baseadas em soluções de mínimos quadrados respeitando a equação 3.10. Entretanto, dada a complexidade do dicionário e vetores, nem sempre é possível garantir a representação perfeita dos vetores, especialmente quanto se impõem restrições em relação à esparsidade, como por exemplo restringir o número máximo de coeficientes não nulos

na representação x. Dessa forma, os algoritmos utilizados e apresentados nesse trabalho, não diferente do descrito acima, são responsáveis por criar um dicionário que consiga representar e/ou reconstruir a maior parte dos sinais treinados, isto porque, normalmente os algoritmos conseguem chegar apenas em uma solução local dos resultados, e não em uma solução global. Na sequência são apresentados os métodos que foram utilizados no presente trabalho.

3.3.1 Método de Direções Ótimas

MOD (do inglês, *method of optimal directions*), ou método de direções ótimas, é inspirado no algoritmo de generalizado de Lloyd [Engan, Aase e Husoy 1999][Dai, Xu e Wang 2012], ou k-means, que é capaz de encontrar, no espaço euclidiano, pontos equidistantes de subconjuntos no espaço [Chen 2004]. O algoritmo, muito utilizado em problemas de *clusterização* de dados, busca pela centroide de cada agrupamento de dados, iteração por iteração [Engan, Aase e Husøy 2000].

O algoritmo MOD utiliza uma estratégia em duas etapas distintas para resolver o problema de encontrar o dicionário "ótimo". A ideia é utilizar um algoritmo para a seleção de vetor para que, a partir de vetores de treinamento, se obtenha a representação esparsa dos vetores de treinamento, e a partir dessa representação atualizar o dicionário inicial para que, ao final do processo iterativo, se obtenha um dicionário que consiga representar vetores esparsos e reconstruí-los com o menor erro possível.

Em geral, a qualidade do dicionário é mensurada por meio do erro residual da reconstrução, i.e.,

$$r = y_{original} - y_{reconstruído} \tag{3.16}$$

onde $y_{original}$ é vetor de treinamento e $y_{reconstruído}$ é o vetor reconstruído a partir da sua representação esparsa e o dicionário utilizado [Engan, Aase e Husøy 2000]. O MOD busca ajustar o dicionário de tal maneira que o erro quadrático médio (MSE, do inglês mean square error) definido por

$$MSE_r = \sum_i ||r_i||_2^2 = ||R||_2^2 \tag{3.17}$$

seja o menor possível. Esse é, em essência, o mesmo procedimento adotado para a obtenção da representação esparsa utilizada no algoritmo OMP para a seleção de vetores. A diferença, no entanto, é que naquele caso, a minimização era realizada ajustando-se os coeficientes da representação; no caso do MOD, o ajuste é feito alterando-se os vetores base presentes no dicionário D.

A cada iteração, o algoritmo propõe alterações em um conjunto de colunas do dicionário, de maneira que cada coluna modificada é representada por:

$$\widetilde{D}_{:,j} = d_{:,j} + \delta_j \tag{3.18}$$

onde δ representa a modificação adicionada à coluna da matriz D, de maneira que o erro residual ao final de cada iteração pode ser dado por:

$$\widetilde{r} = r + \sum_{j=1}^{K} y_j \delta_j \tag{3.19}$$

onde $\underline{x}_j$ representa o coeficiente do vetor esparso associado à coluna j. Com isso, o objetivo do MOD é, a cada iteração, reduzir o erro quadrático médio (MSE), de modo que:

$$\sum ||\widetilde{r}||_2^2 \leq \sum ||r||_2^2 \tag{3.20}$$

O problema de minimização indicado em (3.20), pode ser solucionado por meio da iteração[Engan, Aase e Husøy 2000]:

$$\widetilde{D} = D + R_{rx} R_{xx}^{-1} \tag{3.21}$$

onde,

$$R_{xx} = XX^T \tag{3.22}$$

$$R_{rx} = RX^T \tag{3.23}$$

sendo que X é a matriz contendo a representação esparsa de todos os vetores de treinamento, R é a matriz contendo os erros residuais obtidos para cada um dos vetores de treinamento. Dessa forma, R_{xx} e R_{rx} representam as matrizes de autocorrelação de x e de correlação cruzada entre x e r, respectivamente. Alternativamente, mostra-se que (3.21) é equivalente a[Engan, Aase e Husøy 2000]:

$$\widetilde{D} = R_{yx} R_{xx}^{-1} \tag{3.24}$$

onde R_{yx} é a matriz de correlação cruzada entre os vetores de treinamento e a sua representação esparsa. Após a obtenção do novo dicionário, busca-se obter uma nova representação dos dados e novamente atualiza-se o dicionário. O procedimento completo é descrito no algoritmo 5.

3.3.2 K-SVD

O algoritmo K-SVD, assim com o algoritmo MOD, busca atualizar o dicionário de maneira iterativa, também inspirado no algoritmo k-means, de onde se originou seu nome[Aharon, Elad e Bruckstein 2006].

A metodologia adotada no K-SVD é similar ao algoritmo MOD, inicialmente, a partir de um dicionário D, selecionam-se os vetores esparsos para todas as amostras

Algorithm 5 MOD - Method of Optimal Directions

função MOD(Y,D)
  Criar dicionário D com $D \in \Re\{n, m\}$.
  $j \leftarrow 0$;
  para $j \leftarrow 1$ **até** m **faça**
    Obter matriz X extraindo x com qualquer algoritmo de seleção de vetor.
    Atualizar matrizes de correlação R_{yx} e R_{xx}.
    Atualizar dicionário com: $\widetilde{D} = R_{yx}R_{xx}^{-1}$.
    Normalizar nova matriz D.
    $j \leftarrow j + 1$;
  fim para
fim função
devolve Dicionário: **D**

disponíveis. Depois, busca-se aprimorar o dicionário em busca de um mínimo local. Porém uma distinção importante em relação ao MOD, que é atualizar apenas uma coluna por vez, do dicionário, a ponto de se reduzir o erro quadrático médio dos vetores esparsos reconstruídos utilizando o dicionário provido pela iteração anterior e os sinais originais. Este processo de atualização, que utiliza uma coluna por vez, possui uma solução direta baseada no SVD (do inglês, singular value decomposition) [Aharon, Elad e Bruckstein 2006], permitindo que a convergência do dicionário para um mínimo seja mais rápida.

Para se computar os coeficientes esparsos x, precisa-se primeiramente, de um dicionário D que será utilizado no algoritmo de seleção de vetor e seus coeficientes esparsos[Elad 2013]. Para o K-SVD neste trabalho usa-se o OMP (do inglês, Orthogonal Matching Pursuit) para o cálculo de seus coeficientes e SVD para calcular e atualizar o dicionário de valores.

No passo de aprendizado de dicionário, considere que os vetores contendo as representações esparsas das amostras são agrupadas em uma matriz X, de maneira que a aproximação das amostras é dada por DX. O objetivo na atualização do dicionário é minimizar o resíduo da representação para todos as amostras, i.e.,

$$\min \|Y - DX\|_2^2 \tag{3.25}$$

O método proposto no algoritmo MOD prevê a atualização de todos os vetores conjuntamente em um único passo, o que depende da inversão de matrizes que podem apresentar dimensão elevada. No algoritmo k-SVD, a aproximação DX é escrita em termos da soma de K matrizes de *rank* 1, i.e.,

$$DX = \sum_{j=1}^{K} d_j x_T^j \tag{3.26}$$

onde x_T^j denota a j-ésima linha de X. Dessa forma, (3.25) pode ser reescrita como [Aharon,

Elad e Bruckstein 2006]:

$$\|Y - DX\|_2^2 = \|Y - \sum_{j=1}^{K} d_j x_T^j\|_2^2 = \|(Y - \sum_{j=1, j\neq k}^{K} d_j x_T^j) - d_k x_T^k\|^2 = \|E_k - d_k x_\tau^k\|^2 \quad (3.27)$$

onde E_k corresponde ao vetor de erro quando se exclui arbitrariamente o k-ésimo elemento do dicionário de D.

Com isso, é possível utilizar a SVD para encontrar a aproximação de *rank*-1 que mais se aproxima de E_k, minimizando assim o erro global. Essa abordagem, entretanto, pode levar à obtenção de um vetor x_T^j que não seja esparso (já que a restrição de esparsidade não é imposta em nenhum momento).

Para contornar o possível problema, antes de efetuar a aproximação, deve-se remover as colunas de E_k que não fazem uso, efetivamente, do vetor d_k em suas representações. Com isso, o cálculo da SVD da matriz E_k reduzida produz $U\Delta V^T$ onde a primeira coluna de U fornece a solução para d_k, e a primeira linha de V, multiplicada por $\Delta(1,1)$, a solução para x_T^k. O mesmo procedimento pode ser efetuado para cada uma das colunas de D [Rubinstein, Zibulevsky e Elad 2008]. O método é sintetizado no Algoritmo 6.

Algorithm 6 K-SVD

função K-SVD(Y)
  Criar dicionário D com $D \in \Re\{n, m\}$.
  para $j \leftarrow 1$ **até** m **faça**
    Obter representação esparsa x com Y e criar matriz X.
  fim para
  $R \leftarrow Y - DX$ - Matriz das diferenças
  para $\underline{col} \leftarrow 1$ **até** m **faça**
    $i \leftarrow$ posição de elementos não nulos em $R_{\underline{col},:}$
    $E \leftarrow R_{:,i} + D_{\underline{col}} X_{\underline{col},i}$ - Erro geral da matriz.
    $[U, S, V] \leftarrow svd(E)$ - Calcula-se SVD de E.
    $D_{:,\underline{col}} \leftarrow U_{:,1}$. - Atualização do dicionário.
    $X_{\underline{col},i} \leftarrow S_{1,1}\, V_{:,1}^T$. - Ajustes dos vetores esparsos.
    $R_{:,i} \leftarrow E - D_{:,\underline{col}}\, X_{\underline{col},i}$ - Ajuste de matriz para próximas iterações.
  fim para
fim função
devolve Dicionário: **D**

3.3.3 LS-DLA e RLS-DLA

Conforme apresentado anteriormente, o método MOD consiste, basicamente, no ajuste das colunas da matriz D visando minimizar o erro residual obtido na representação dos vetores de treinamento - cuja representação é obtido com um dos algoritmos de seleção de vetor mencionados na Seção 3.2. O ajuste do dicionário, nesse caso, segue uma abordagem de minimização de mínimos quadrados, e por esse motivo os algoritmos

baseados no MOD também são conhecidos como algoritmos para aprendizado de dicionário de mínimos quadrados iterativos (do inglês, iterative least square dictionary learning algorithm - LS-DLA).

Denotando Y como a matriz contendo as amostras do sinal e y_l como suas colunas e X como a matriz esparsa que representa os seus pesos correspondentes e x_l, a abordagem seguida no algoritmo MOD é, dada uma representação esparsa, obter a atualização do dicionário utilizando a solução clássica do problema de mínimos quadrados, conforme indicado no algoritmo (2). A complexidade computacional, entretanto, aumenta consideravelmente quando considera-se o aprendizado de um dicionário para um conjunto muito grande de dados.

Como alternativa ao MOD, algumas modificações foram propostas na literatura, como dividir o conjunto de dados em subconjuntos e, a cada iteração, realizar o ajuste do dicionário apenas considerando um (ou alguns) dos subconjuntos de dados. Os subconjuntos de dados podem ser escolhidos aleatoriamente, ou ainda pode ser utilizado um fator de esquecimento para atualização dos vetores, dando origem ao algoritmo RLS-DLA [Skretting 2011] (do inglês, Recursive Least Squares Dictionary Learning Algorithm).

Definindo-se as matrizes

$$A^{(i)} = \sum_m x_m x_m^T \tag{3.28}$$

e

$$B^{(i)} = \sum_m y_m x_m^T \tag{3.29}$$

para a $i-$ésima iteração, a solução para o dicionário na $i-$ésima iteração do algoritmo, $D^{(i)}$[2], é dada por $D^{(i)} = B^{(i)} A^{-1\,(i)}$. É importante ressaltar que as matrizes $A^{(i)}$ e $B^{(i)}$ correspondem, respectivamente, às matrizes R_{xx} e R_{yx} do algoritmo MOD, mas considerando apenas um subconjunto dos dados de treinamento (indexados por m).

A atualização das matrizes $A^{(i)}$ e $B^{(i)}$ no algoritmo LS-DLA é efetuada por meio das equações

$$A^{(i)} = \underline{\lambda}^{(i)} A^{(i-1)} + \sum_m x_m x_m^T \tag{3.30}$$

$$B^{(i)} = \underline{\lambda}^{(i)} B^{(i-1)} + \sum_m y_m x_m^T \tag{3.31}$$

onde λ_i é um fator de esquecimento. Dessa maneira, preserva-se uma certa memória no processo de adaptação do dicionário, mesmo que nem todos os vetores estejam sendo utilizados na atualização de D na iteração atual. Os passos do algoritmo são sintetizados no Algoritmo 7.

[2] O uso do termo $'(i)'$ indica a iteração atual do passo do algoritmo, assim como $'(i-1)'$ indica a iteração passada e $'(i+1)'$ a iteração futura.

Algorithm 7 LS-DLA - Interactive Least Squares Dictionary Learning Algorithm

função LS-DLA(Y)
    Criar dicionário D com $D \in \Re\{n, m\}$
    Itn é quantidade de iterações do algoritmo até a convergência
    para $i \leftarrow 1$ **até** m **faça**
        Para o subconjunto de dados de entrada, extrair as representações esparsas com algum algoritmo de seleção de vetor;
        Atualizar matrizes de correlação $A^{(i)}$ e $B^{(i)}$ de acordo com o fator de decrescimento $\underline{\lambda}$ escolhido.
        Atualizar dicionário com: $\widetilde{D} = B\,A^{-1}$.
        Normalizar nova matriz D.
    fim para
fim função
devolve Dicionário: **D**

Naturalmente, uma vez que se emprega uma abordagem de mínimos quadrados, é possível empregar métodos de otimização mais eficientes, como método de mínimos quadrados recursivo, o que dá origem ao algoritmo RLS-DLA (do inglês, Recursive Least-Squares Dictionary Learning Algorithm). O RLS-DLA tenta resolver o problema utilizando um modo que continuamente atualiza o dicionário[Skretting 2011].

Na derivação do RLS-DLA, a cada nova iteração do algoritmo, um novo vetor de treinamento x_i é utilizado, de maneira que são definidas as matrizes (indexadas pela iteração do algoritmo) $Y^{(i)} = [y_1, ..., y_i]$ com o tamanho $N \times i$, $X^{(i)} = [x_1, ..., x_i]$de tamanho $K \times i$ e $C^{(i)} = (X^{(i)}X^{(i)T})^{-1}$, assim como o dicionário $D^{(i)}$, que é resultado da minimização de mínimos quadrados.

A cada novo vetor de treinamento Y_i, obtém-se os pesos correspondentes na representação esparsa x_i, que são encontrados com o dicionário ajustado na iteração anterior, D_{i-1}, e algum algoritmo de seleção de vetor. Utilizando o lema de inversão de matrizes (c) [Skretting 2011] em $C^{(i)}$, é possível obter as seguintes regras de atualização:

$$C^{(i)} = C^{(i-1)} - \alpha u u^T \tag{3.32}$$

$$D^{(i)} = D^{(i-1)} - \alpha u r_i u^T \tag{3.33}$$

onde,

$$u = C^{(i-1)} x_i \tag{3.34}$$

$$\underline{\alpha} = \frac{1}{(1 + x_i^T u)} \tag{3.35}$$

$$r_i = y_i - D^{(i-1)} x_i \tag{3.36}$$

Sendo r_i o erro residual na iteração i.

O fator esquecimento λ_i, introduzido no algoritmo RLS-DLA, faz com que o dicionário seja muito menos dependente do dicionário inicial assim como melhora a convergência do algoritmo[Skretting 2011]. O λ é um parâmetro a ser escolhido, e geralmente está entre $0 < \lambda < 1$. Assim,

$$C^{(i)} = (\lambda^{-1} C^{(i-1)}) - \alpha u u^T \tag{3.37}$$

e,

$$u = (\lambda^{-1} C^{(i-1)}) x_i \tag{3.38}$$

e o pseudo-código sintetizando do método RLS-DLA é mostrado no Algoritmo 8.

Algorithm 8 RLS-DLA

função RLS-DLA(Y)
    Criar dicionário D com $D \in \Re\{n, m\}$;
    Criar Matriz C com $C \in \Re\{m, m\}$;
    $\underline{\lambda} \leftarrow$ valor arbitrário onde $0 < \underline{\lambda} < 1$
    para $y \leftarrow 1$ **até** Y **faça**
        Obter representação esparsa x.
        Utilizando lema de inversão de matrizes para atualização de dicionário D:
        $r \leftarrow y - Dx$
        $C \leftarrow \frac{1}{\lambda} C$
        $u \leftarrow Cw$
        $\underline{\alpha} \leftarrow \frac{1}{1 + x^T u}$
        $D \leftarrow D + \underline{\alpha} r u^T$
        $C \leftarrow C - \underline{\alpha} u u^T$
    fim para
fim função

devolve Dicionário: **D**

3.3.4 Aprendizado de Dicionário Online

Outra proposta encontrada na literatura para lidar com o treinamento de dicionários quando há um grande número de dados para treinamento é o ODL (do inglês, Online Dicitionary Learning)[Mairal et al. 2009]. Este método se torna interessante pois permite que o dicionário seja retreinado em tempo real de maneira mais eficiente.

Como os demais algoritmos para aprendizado de dicionário, o algoritmo ODL é dividido em duas partes [Mairal et al. 2009]: a etapa de seleção de vetor e a atualização da matriz D. Ou seja, pode-se atualizar o dicionário separadamente e não apenas de uma só vez.

A cada iteração, o algoritmo utiliza apenas um vetor y_i (sinal de entrada) e extrai, através de um algoritmo de seleção de vetor esparso, a sua representação esparsa x_i a partir de $D^{(i-1)}$, i.e., o dicionário na iteração anterior.

A atualização do dicionário visa minimizar a função custo dada por[Mairal et al. 2009]:

$$\widehat{f}_i(D) \triangleq \frac{1}{i}\sum_{j=1}^{i} \|y_j - D\ x_j\|_2^2 + \lambda\,\|x_j\|_1 \tag{3.39}$$

ou seja, assim como nos algoritmos descritos anteriormente, a proposta do novo dicionário tem como objetivo minimizar a norma do erro residual, porém diferentemente do método RLS-DLA, o ODL não utiliza fator de esquecimento, visto que o processo de inversão de matrizes é diferente. O método utilizado para atualizar as colunas da matriz D é apresentado sinteticamente no Algoritmo 9. No algoritmo 9, entende-se facilmente que a atualização da coluna do dicionário é dada através de uma atualização entre o erro normalizado e a coluna do dicionário que está sendo atualizada no momento [Mairal et al. 2009].

Algorithm 9 ODL - Online Dictionary Learning

função ODL(Y)
  Criar dicionário D com $D \in \Re\{n, m\}$
  Criar Matriz A e preencher com **0** com $A \in \Re\{m, m\}$
  Criar Matriz B e preencher com **0** com $A \in \Re\{n, m\}$
  para $y \leftarrow 1$ **até** Y **faça**
    Obter representação esparsa x, através de y
    $A \leftarrow A + y_j\ y_j^T$
    $B \leftarrow B + x\ y_j^T$
    para $i \leftarrow 1$ **até** n **faça**
      Atualização de dicionário utilizando lema de mínimos quadrados:
      $u \leftarrow \frac{1}{D_{1,1}}(B_i - D^T A_i) + D_{:,i}^T$
      $\underline{max} \leftarrow arg\_max((u^T u)^{\frac{1}{2}})$
      se $\underline{max} < 1$ **então**
        $\underline{max} \leftarrow 1$
      fim se
      $D_{:,i} \leftarrow \frac{1}{\underline{max}} u^T$
    fim para
  fim para
fim função

devolve Dicionário: **D**

4 BCI baseado em Imagética Motora utilizando Representação Esparsa

Conforme comentado no Capítulo 2, uma das abordagens consideradas para a construção de um BCI explora características dos sinais de EEG buscando relacioná-los com a imaginação de movimentos.

Nessa abordagem, em geral, utilizam-se principalmente os canais de EEG localizados na região próxima ao córtex motor, por estarem, teoricamente, captando sinais provenientes de atividade relacionada à atividade motora. Entretanto, os demais canais também podem fornecer informações relevantes e auxiliar na tarefa de classificação - por exemplo, um canal próximo da região frontal da cabeça pode conter informação relevante de possível atividade muscular dos olhos da pessoa, o que, por sua vez, pode contribuir para minimização da influência desse tipo de artefato nos demais sinais sob análise.

No capítulo, é apresentado estudo proposto para o trabalho, no qual explora-se o esquema básico de um BCI, explicado no capítulo 2, mas incluindo etapas adicionais de processamento buscando explorar os métodos de aprendizado de dicionário e representação esparsa discutidos no capítulo anterior.

4.1 Abordagens para BCI baseado em Imagética Motora

No presente trabalho foram estudadas diferentes abordagens para realizar a classificação dos sinais neurais provenientes de EEG. Em todos os casos, a primeira etapa no pré-processamento dos dados consiste em filtrar os sinais com um filtro passa banda, tipo FIR, projetado com método de janelamento (janela de Kaiser) de ordem 20, com banda passante entre 0,5 Hz e 20 Hz. A razão para essa etapa decorre dos trabalhos já realizados na área de BCI baseado em imaginação do movimento, que indicam uma maior atividade relacionada a atividades motoras concentrado em frequências abaixo de 20 Hz. Dessa forma, ao realizar a filtragem, foca-se a atenção na faixa de frequências de interesse dos sinais.

Ao todo, são consideradas 4 abordagens distintas, envolvendo os métodos de processamento apresentados nos Capítulos 2 e 3, que são descritos de maneira mais detalhada a seguir.

4.1.1 Método 1 - Abordagem Canônica

No primeiro método, denominada de abordagem canônica, os sinais são classificados baseados na densidade espectral de potência, conforme ilustrado na figura 9.

Na fase de treinamento, cada um dos canais presentes nos registros dos dados, foi calculado a densidade espectral de potência, e os dados transformados são utilizados para compor a base de treinamento do sistema. Nessa etapa, foi realizada uma redução na dimensionalidade dos dados para classificação, retendo apenas os primeiros 50 coeficientes da PSD, pois para treinamento foi utilizado apenas as 50 primeiras frequências amostradas (0 à 50Hz, da PSD), e uma vez que na etapa de pré-processamento dos dados os sinais foram filtrados, a maior concentração da energia dos sinais pré-processados estava presente até a amostra 50 da PSD calculada.

Na fase de classificação, o sistema opera de maneira similar, calculando a PSD de cada sinal e realizando a classificação de maneira independente para cada canal. A classe final atribuída ao bloco de dados do teste é obtida por meio de voto majoritário, i.e., a classe obtida para o maior número de canais do bloco de testes.

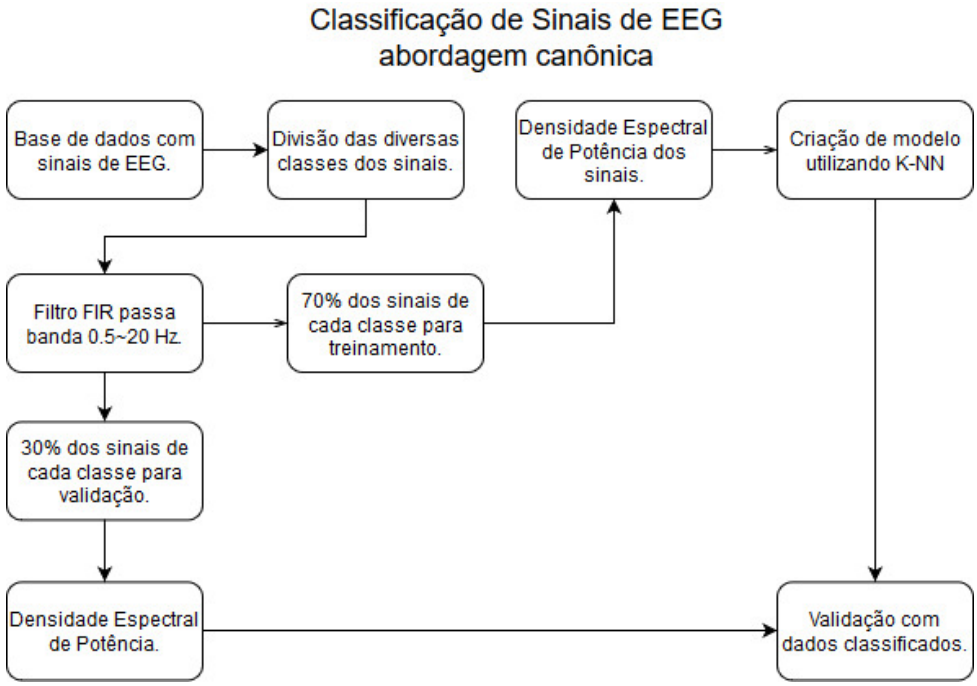

Figura 9: Método utilizado para classificação de sinais de EEG sem a utilização de vetores esparsos e sem utilização de CSP.

4.1.2 Método 2 - CSP

O segundo método avaliado é descrito na figura 10, e consiste, basicamente, em utilizar o CSP para realizar um pré-processamento dos sinais, buscando ressaltar diferenças entres as classes de sinais, de maneira que tais diferenças tornem-se mais pronunciadas

quando calcula-se a densidade espectral de potência dos sinais filtrados. A abordagem foi originalmente descrita em [Wang, Gao e Gao 2005].

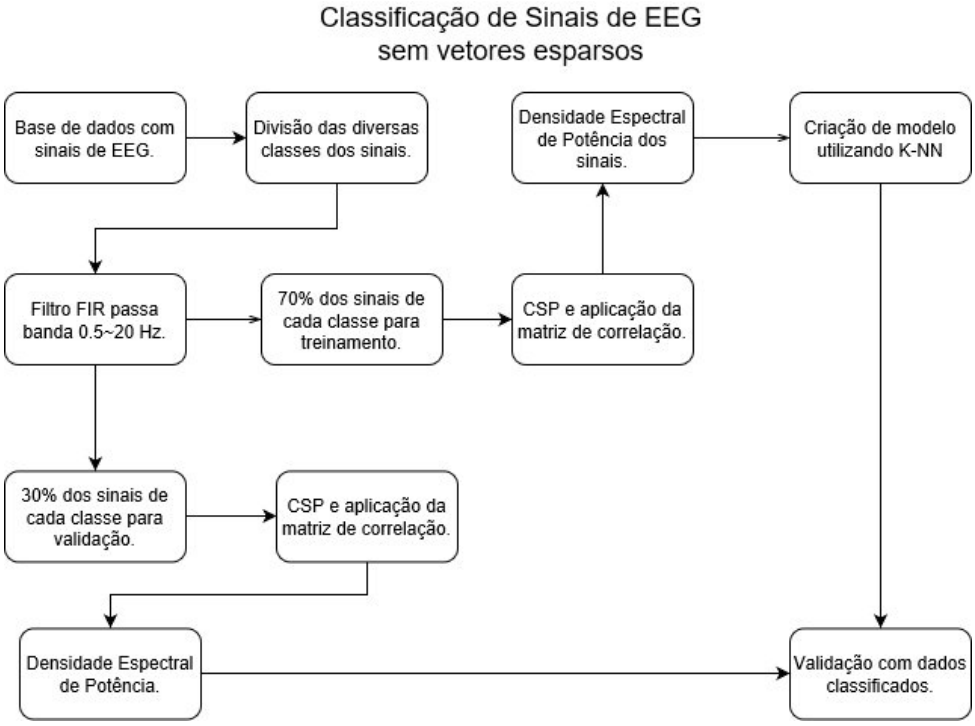

Figura 10: Método utilizado para classificação de sinais de EEG, com a utilização de CSP.

Nos testes realizados, foram consideradas dois cenários distintos: em um primeiro cenário, a classificação é realizada considerando-se duas classes apenas (par-a-par), e no segundo, a classificação multi-classe. No primeiro caso, o pré-processamento dos dados é obtido por meio do algoritmo CSP para duas classes apenas, enquanto que no segundo caso foi utilizado o algoritmo CSP multi-classe (M-CSP).

4.1.3 Método 3 - Abordagem Canônica + Representação Esparsa

A fim de avaliar o impacto da representação esparsa no problema de classificação de sinais na BCI baseada em imagética motora, foi utilizado um algoritmo de criação de dicionário para buscar representar as densidades espectrais de potência de maneira esparsa, conforme indicado na Figura 11.

4.1.4 Método 4 - CSP + Representação Esparsa

Por fim, a mesma ideia para representação esparsa foi utilizada no caso em que os dados foram pré-processados com o algoritmo CSP, conforme indicado na Figura 12;

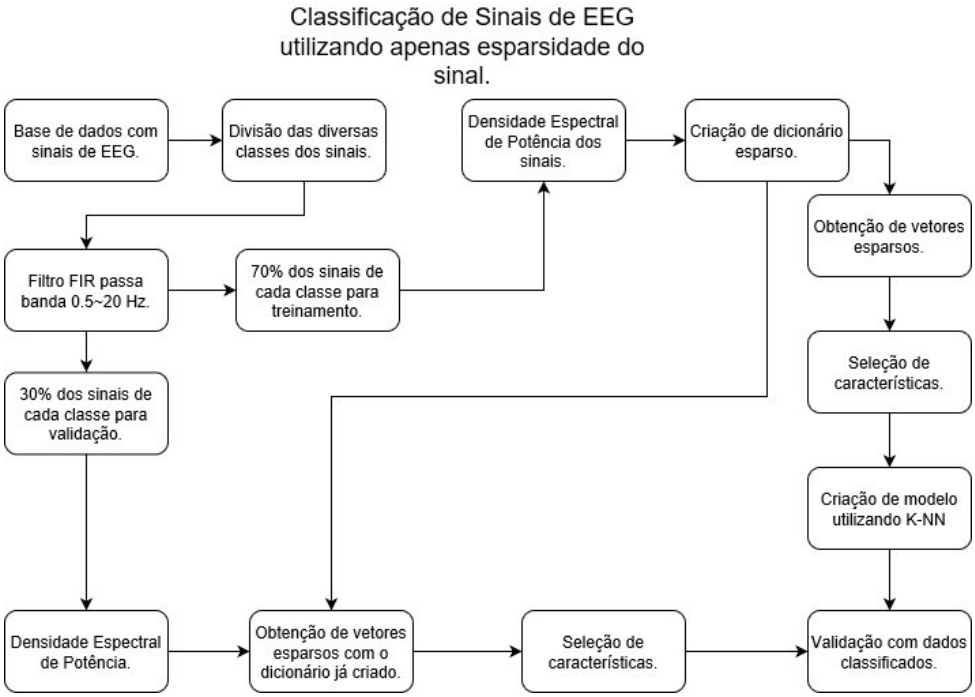

Figura 11: Método utilizado para classificação de sinais de EEG com a utilização de vetores esparsos.

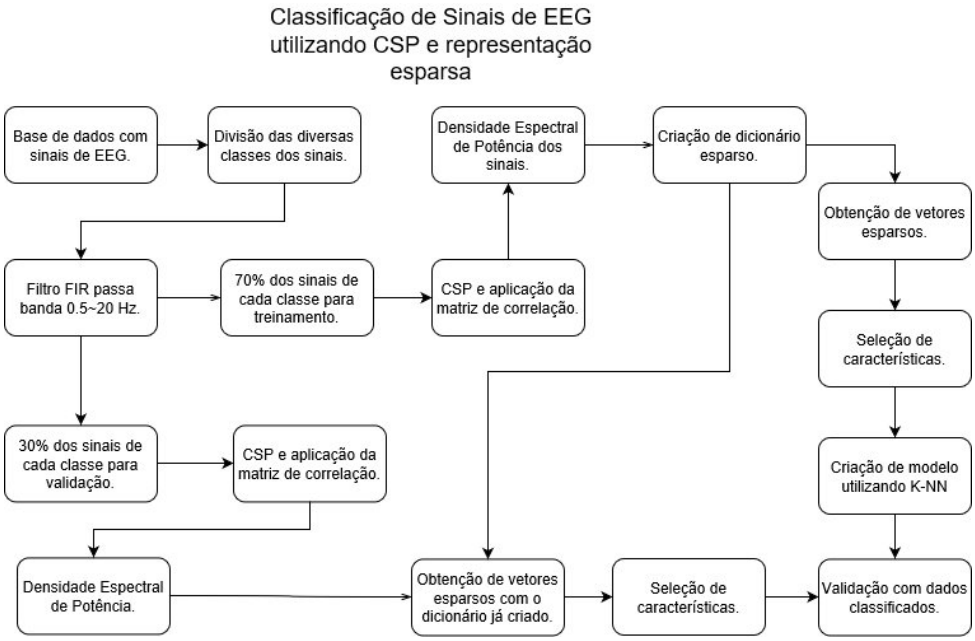

Figura 12: Método utilizado para classificação de sinais de EEG com a utilização de vetores esparsos e CSP.

4.2 Dados de Teste

O presente estudo faz uso de um conjunto de dados disponível livremente na internet, e corresponde à uma das bases de dados utilizada na BCI Competition IV (data set 2A) [Tangermann et al. 2012]. Em particular, foi utilizado o conjunto de dados relacionado à imaginação de movimentos, contendo a gravações de 22 canais de EEG, fornecidas

no formato ". mat" do Matlab.

O conjunto de dados do EEG utilizado nesse trabalho, consiste em gravações de 9 indivíduos diferentes, onde foram geradas 9 diferentes bases para treinamento, associadas a 4 tarefas motoras que foram imaginadas pelos indivíduos:

- Movimentação da mão esquerda (a1);
- Movimentação da mão direita (a2);
- Movimentação de ambos os pés (a3);
- Movimentação da língua (a4).

O procedimento experimental para coleta dos dados consistiu em: cada um dos 9 indivíduos permaneceu sentado em uma cadeira confortável em frente de uma tela de computador. No início do teste ($t = 0s$), uma cruz aparece na tela de computador, e após dois segundos ($t = 2s$), uma flecha apontando para uma direção surge na tela (apontando para a direita, esquerda, baixo ou cima), correspondendo às 4 classes a serem treinadas, i.e., mão direita e esquerda, pés e língua, e permanecendo na tela por $1,25s$. O indivíduo, então, deveria realizar a tarefa de imaginação de movimento selecionada, sem que houvesse qualquer tipo de *feedback*, até que a cruz se apagasse da tela. O período entre $t = 3s$ e $t = 6s$ era devidamente gravado e associado à classe correspondente. Após um breve intervalo, o procedimento era repetido, num total de 48 trials (12 para cada classe).

Todas as gravações foram realizadas com 22 eletrodos, separados de aproximadamente 3,5 cm uns dos outros, conforme ilustrado na Fig. 13. Os sinais gravados foram amostrados a uma frequência de 250 Hz, sendo filtrados com um filtro passa-banda entre as frequências de 0,5 Hz e 100 Hz, e com um filtro notch com frequência central de 50Hz para suprimir interferência da rede de alimentação. A sensibilidade do amplificador de EEG foi configurada para $100\mu V$ [Tangermann et al. 2012].

Apenas para demonstrar o desafio encontrado na classificação dos sinais, a figura 14 ilustra a diferença entre os sinais associados a diferentes classes, apresentando a média dos canais de EEG para cada uma das classes, considerando o indivíduo 9 do dataset.

Embora o comportamento médio das 4 classes não seja exatamente o mesmo, não é uma tarefa trivial implementar um sistema que seja capaz de distingui-las utilizando apenas o sinal no domínio do tempo. Por esse motivo, as abordagens estudadas neste trabalho consideram a representação dos sinais no domínio da frequência.

A representação esparsa, nesse caso, tem um papel importante pois tende a aumenta a diferença entre as classes e permite que a classificação tenha um erro menor. Para maior entendimento, na figura 15 é apresentada uma PSD de um sinal associado à

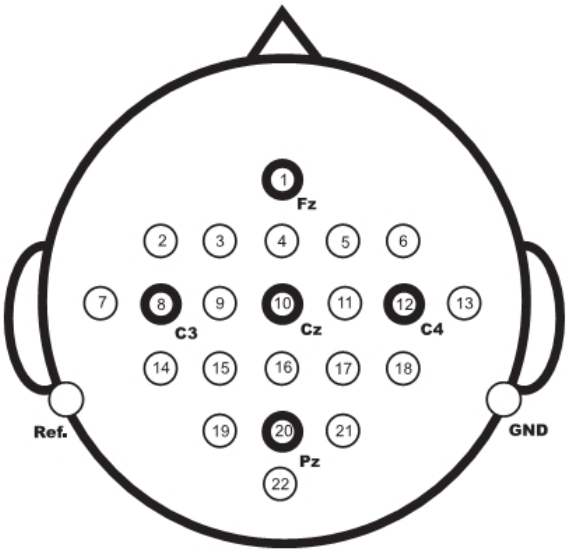

Figura 13: Posicionamento dos eletrodos utilizados na captura dos sinais para o IV BCI Competition (extraído de [Tangermann et al. 2012]).

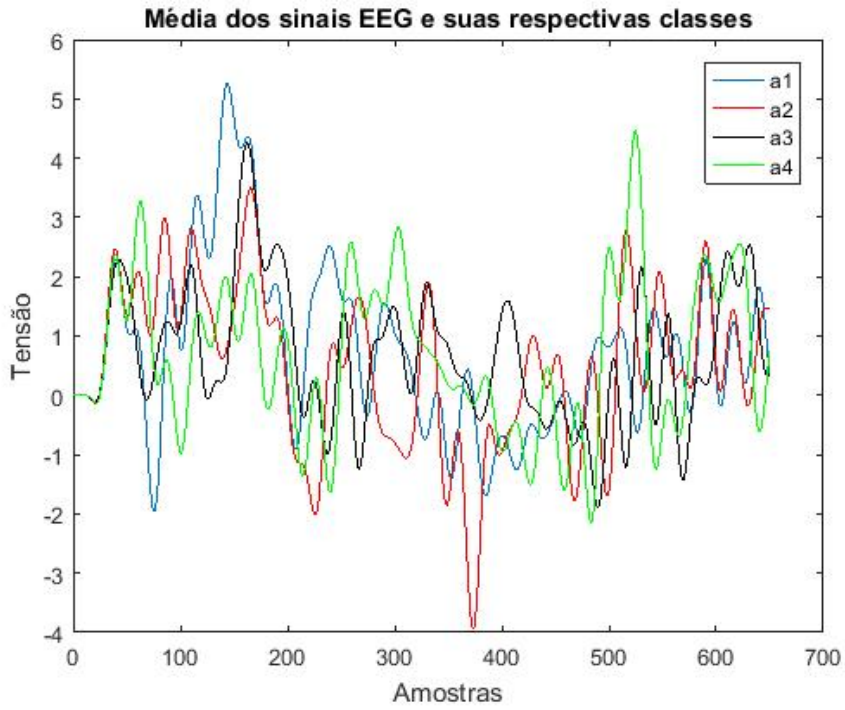

Figura 14: Média dos sinais de EEG e suas respectivas classes para o indivíduo 9.

classe $a2$, e na figura 16 a sua representação com o dicionário obtido a partir da etapa de treinamento. Observa-se que a representação com o dicionário apresenta um número muito inferior de valores não-nulos, sendo portanto mais esparso que o sinal original. Finalmente, na figura 17 é ilustrado a PSD da média de todas as representações dos sinais associados às 4 classes, e é possível observar que, nesse caso, a tarefa de distinguir os vetores de classes distintas fica bem mais simples de ser realizado.

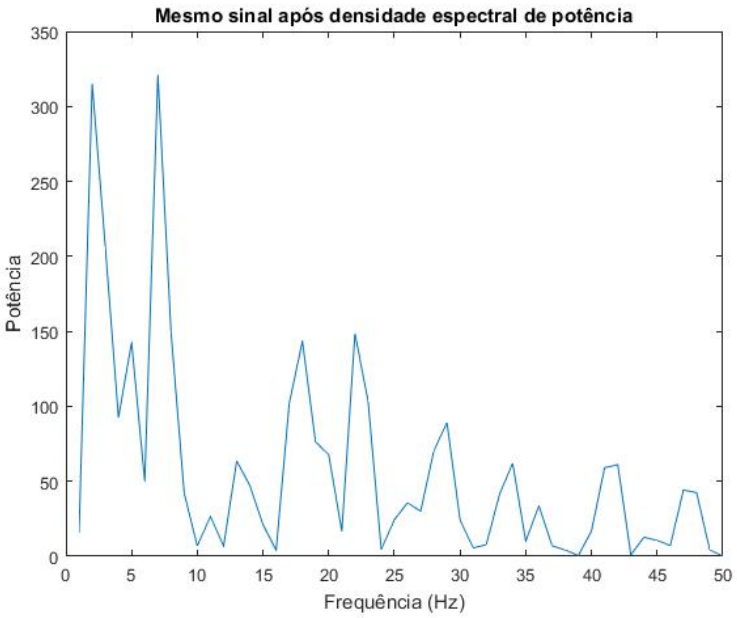

Figura 15: Densidade espectral de potência de um sinal associado à classe a2.

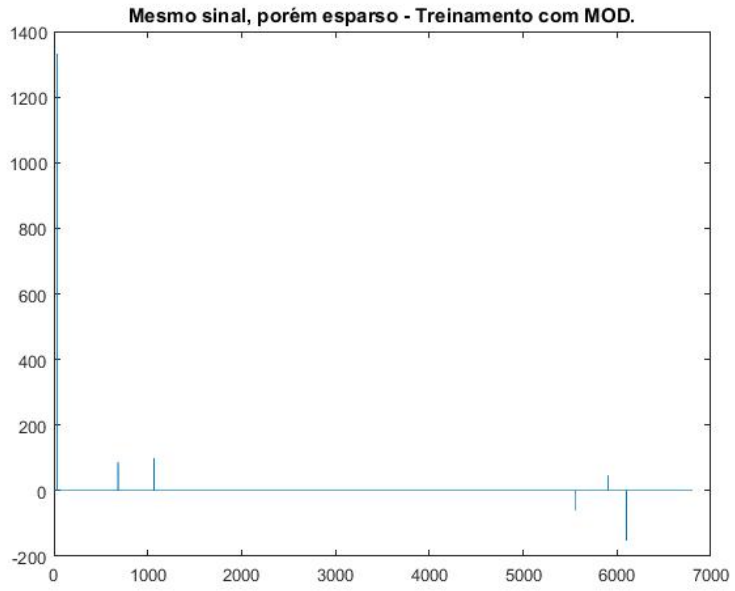

Figura 16: Representação esparsa do vetor obtido na densidade espectral de potência do sinal associado à classe a2.

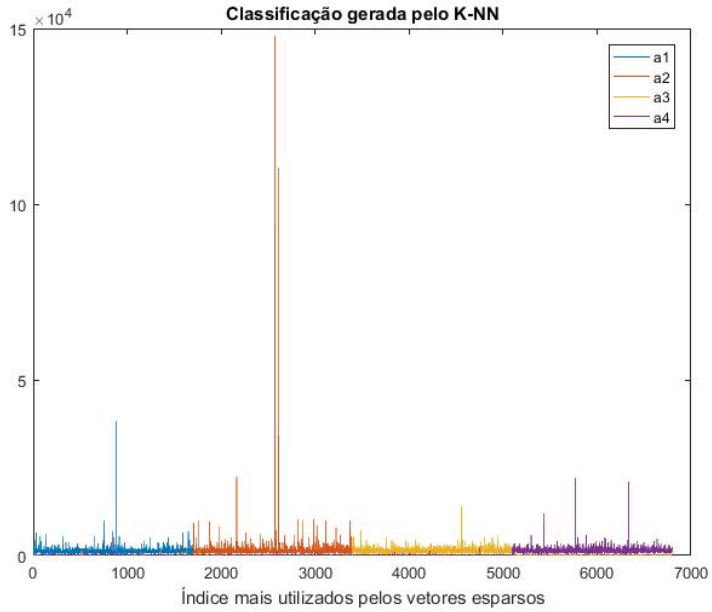

Figura 17: Visualização final de treinamento k-NN com suas distintas classes (a1, a2, a3 e a4). Podemos ver que o dicionário esparso, inicializado corretamente consegue separar as diferentes classes de maneira intuitiva.

5 Resultados

Os métodos apresentados no Capítulo 4 foram testados considerando-se o conjunto de dados disponibilizados na BCI Competition IV (data set 2A), relativos à tarefa de imaginação de movimentos (4 classes distintas).

Foram realizados dois cenários de testes: no primeiro deles, considerou-se que a classificação é realizada par-a-par, i.e., avaliou-se apenas a acurácia na classificação entre duas classes distintas; enquanto que no segundo cenário, foi avaliada a acurária na classificação das quatro classes em conjunto.

5.1 Métrica de Avaliação - Coeficiente Kappa

A acurácia da classificação ou taxa de erro (1-acurácia) é o critério mais utilizado em sistemas de BCI, pois pode ser facilmente interpretado e calculado. Em sistemas de BCI, além de se utilizar a acurácia como critério de desempenho, usa-se o coeficiente Cohen Kappa, nada mais é que uma medição quantitativa dos resultados obtidos utilizando o grau de concordância entre as taxas, visto que existem vários canais para analisar em um BCI [Schlögl et al. 2007].

O coeficiente Cohen Kappa, κ, para ser calculado utiliza a acurácia da classificação, que é dado como:

$$ACC = p_0 = \frac{\sum_{i=1}^{m} n_i}{t} \tag{5.1}$$

Onde, m é o número de classes utilizado na classificação, t é o total de tentativas (inclui tentativas com acertos e sem acertos) e n é dado pelo número de acertos corretos, dado uma entrada na classificação. E utiliza a chance de concordância geral, dado como p_e:

$$p_e = \frac{\sum_{i=1}^{m} N_{:,i} N_{i,:}}{t^2} \tag{5.2}$$

Onde $N_{:,i}$ e $N_{i,:}$ são a soma da i-ésima coluna e da i-ésima linha da matriz de classificação, respectivamente. A matriz de classificação ou matriz de confusão, é uma matriz onde se coloca todas os resultados obtidos pela classificação em uma validação. Cada linha, da matriz de classificação, é considerada como objeto esperado e cada coluna,

como objeto classificado pelo sistema desenvolvido. Portanto, ao final, tem-se uma matriz com todos os resultados esperados e obtidos do sistema testado pela classificação [Schlögl et al. 2007].

Portanto, a estimação do coeficiente κ é dada como:

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{5.3}$$

E o desvio padrão é obtido por:

$$\sigma_e(\kappa) = \frac{\sqrt{(p_0 + p_e^2 - \sum_{i=1}^{m}[N_{:,i}N_{i,:}(N_{:,i} + N_{i,:})]/t^3)}}{(1 - p_e)\sqrt{t}} \tag{5.4}$$

Para maior compreensão, o coeficiente Kappa é zero se as classes previstas não mostram nenhuma correlação. Se o número de amostras são igualmente distribuídas entre as classes, a chance de concordância global, p_e, é de $1/m$ e o coeficiente Kappa e acurácia são dados como [Schlögl et al. 2007]:

$$\kappa = \frac{p_0 - p_e}{1 - p_e} = \frac{mp_0 - 1}{m - 1} \tag{5.5}$$

$$ACC = p_0 = \frac{M\kappa - \kappa + 1}{m} \tag{5.6}$$

5.2 Resultados - Classificação Par a Par

Considerando um primeiro conjunto de testes, foram realizadas simulações para avaliar a acurácia na classificação dos sinais considerando apenas classificações par-a-par, ou seja, o sistema deve decidir se o sinal deve ser classificado como pertencente à classe a_1 ou a_2, e assim sucessivamente. Embora não seja propriamente o objetivo final da BCI, configura-se em um teste importante para realizar uma avaliação preliminar das abordagens propostas.

Para todos os métodos considerados, escolhe-se aleatoriamente 70% dos dados para realizar o treinamento do classificador e os 30% restante para teste. Nos métodos empregando o CSP, utiliza-se apenas 1 filtro para realizar o pre-processamento, uma vez que há apenas comparações entre duas classes de cada vez.

Após esse processo, os métodos de criação de dicionário utilizaram uma matriz D inicializada com as próprias amostras dos dados de treinamento, criando uma matriz inicial de dimensão 50×2520, alocando uma coluna para cada amostra. O OMP foi utilizado como algoritmo de seleção de vetor para todos os métodos de criação de dicionário menos ODL, onde se utilizou LARS. Representação esparsa (um vetor com 2520 posições) é obtida, e partir dái se classifica os vetores utilizando o classificador k-NN.

Na Tabela 1 são apresentados os valores do coeficiente kappa, que varia entre -1 (pior resultado) e 1 (classificação perfeita), para o Método 1. Os resultados são bastante expressivos, conseguindo valores para o coeficiente kappa superiores a 0.9. Os valores expressivos de acurácia são obtidos às custas de um maior custo computacional - visto que a classificação é realizada de maneira independente para os 22 canais do EEG. Como todos os resultados passaram por validação cruzada, ao lado do coeficiente Cohen's Kappa, é dado seu respectivo desvio padrão, calculado através da equação 5.4.

Tabela 1: Classificação utilizando o Método 1 (Canônico).

Resultados provenientes do método 1						
Indivíduo	**Kappa** (± 0.04)					
–	**a1 x a2**	**a1 x a3**	**a1 x a4**	**a2 x a3**	**a2 x a4**	**a3 x a4**
1	0.9073	0.9562	0.9600	0.9619	0.9593	0.9543
2	0.9244	0.9150	0.9094	0.9156	0.9162	0.9025
3	0.9119	0.9056	0.9210	0.9187	0.9238	0.9169
4	0.9606	0.9463	0.9456	0.9543	0.9563	0.9419
5	0.9281	0.9413	0.9318	0.9337	0.9244	0.9280
6	0.9125	0.9613	0.9620	0.9630	0.9610	0.9612
7	0.9381	0.9288	0.9125	0.9350	0.9262	0.9262
8	0.9413	0.9388	0.9317	0.9375	0.9381	0.9280
9	0.9382	0.9150	0.9315	0.9219	0.9406	0.9362
Média	**0.9278**	**0.9342**	**0.9342**	**0.9381**	**0.9378**	**0.9382**

Em uma tentativa de se obter uma redução no custo computacional da tarefa de classificação, optou-se pelo uso do CSP no método 2, para o qual, infelizmente, observa-se uma grande diferença negativa nos resultados na Tabela 2. Atribui-se a grande diferença nos resultados a escolha de apenas um filtro no pré-processamento do CSP, que, embora seja capaz de preservar grande parte da informação relevante para a classificação dos sinais, aparentemente despreza informação que poderia ajudar a melhorar no desempenho do sistema.

De qualquer forma, a baixa acuidade na classificação, nesse caso, evidencia alguns pontos anteriormente não observados. A partir dos dados, constata-se inicialmente que há uma variação de desempenho entre os diferentes indivíduo, algo que era esperado visto que há uma grande variabilidade nos padrões de diferentes indivíduos - até mesmo para um mesmo indivíduos, se forem consideradas respostas em diferentes dias, por exemplo.

Além disso, constata-se que existem classes que representam um maior desafio para o sistema BCI, como é o caso de se distinguir entre as classes $a1$ e $a2$, que levou ao pior desempenho para todos os algoritmos testados. Os valores obtidos para o coeficiente kappa mostram que o sistema opera significativamente melhor do que simplesmente realizar uma classificação aleatória.

Tabela 2: Classificação utilizando o Método 2 - CSP.

Resultados provenientes do método 2						
Indivíduo	**Kappa** (± 0.04)					
–	**a1 x a2**	**a1 x a3**	**a1 x a4**	**a2 x a3**	**a2 x a4**	**a3 x a4**
1	0.5600	0.8200	0.8300	0.8275	0.8125	0.8500
2	0.6550	0.6975	0.5925	0.6775	0.6525	0.5850
3	0.5455	0.5825	0.5625	0.6000	0.5975	0.6800
4	0.7550	0.7850	0.7925	0.7525	0.7600	0.7825
5	0.7300	0.6450	0.6150	0.6250	0.6750	0.5625
6	0.5625	0.7475	0.8500	0.7775	0.8750	0.8650
7	0.6375	0.6400	0.6001	0.5800	0.5650	0.6225
8	0.6500	0.5875	0.6660	0.6375	0.7475	0.6450
9	0.6350	0.6200	0.5800	0.6725	0.5950	0.6076
Média	**0.6367**	**0.6806**	**0.6758**	**0.6833**	**0.6978**	**0.6889**

Pode-se observar uma grande diferença nos resultados ao considerar o segundo método, utilizando apenas as densidades espectrais de potência para realizar a classificação, apresentados na tabela 1. Atribui-se a grande diferença nos resultados a escolha de apenas um filtro no pré-processamento do CSP, que, embora seja capaz de preservar grande parte da informação relevante para a classificação dos sinais, aparentemente despreza informação que poderia ajudar a melhorar no desempenho do sistema.

O terceiro método avaliado, baseado na representação esparsa das densidades espectrais de potência, que é menos complexo que o método 4, apresenta os resultados descritos nas tabelas 3, 4, 5, 6 e 7 .

Comparando-se os resultados obtidos com os diferentes algoritmos de criação de dicionário, observa-se que não há grande disparidade nas performances obtidas. O desempenho dos algoritmos, conforme comentado no capítulo 3, depende bastante da inicialização do dicionário D - indicando que a metodologia adotada para a inicialização do dicionário utilizando os vetores de treinamento é bastante adequada para o problema. Além disso, os resultados obtidos com o terceiro método foram melhores do que o método canônico, aumentando a acuidade média do sistema para todos os métodos de criação de dicionário avaliados.

Finalmente, utilizando o Método 4, que explora a representação esparsa das densidades espectrais de potência, foram obtidos os resultados apresentados nas tabelas 8, 9, 10, 11, 12, para diferentes algoritmos de criação de dicionário.

Tabela 3: Classificação utilizando o método 3 - Canônico + Esparso, com algoritmo MOD.

Resultados provenientes do método 3 - MOD						
Indivíduo	**Kappa** (±0.04)					
–	**a1 x a2**	**a1 x a3**	**a1 x a4**	**a2 x a3**	**a2 x a4**	**a3 x a4**
1	0.9375	0.9337	0.9331	0.9319	0.9425	0.9325
2	0.9337	0.9256	0.9312	0.9294	0.9344	0.9375
3	0.9387	0.9331	0.9319	0.9381	0.9412	0.9400
4	0.9713	0.9731	0.9681	0.9688	0.9688	0.9681
5	0.9450	0.9381	0.9425	0.9481	0.9444	0.9331
6	0.9406	0.9425	0.9425	0.9369	0.9450	0.9462
7	0.9312	0.9237	0.9419	0.9306	0.9456	0.9394
8	0.9419	0.9450	0.9362	0.9419	0.9319	0.9375
9	0.9281	0.9369	0.9425	0.9400	0.9450	0.9500
Média	**0.9409**	**0.9391**	**0.9411**	**0.9406**	**0.9443**	**0.9427**

Tabela 4: Classificação utilizando o método 3 - Canônico + Esparso, com algoritmo RLS-DLA.

Resultados provenientes do método 3 - RLS-DLA						
Indivíduo	**Kappa** (±0.04)					
–	**a1 x a2**	**a1 x a3**	**a1 x a4**	**a2 x a3**	**a2 x a4**	**a3 x a4**
1	0.9300	0.9312	0.9250	0.9312	0.9400	0.9269
2	0.9312	0.9312	0.9350	0.9394	0.9369	0.9337
3	0.9237	0.9275	0.9281	0.9287	0.9312	0.9337
4	0.9688	0.9694	0.9700	0.9650	0.9637	0.9719
5	0.9369	0.9294	0.9375	0.9419	0.9462	0.9344
6	0.9437	0.9556	0.9444	0.9469	0.9444	0.9469
7	0.9281	0.9206	0.9319	0.9319	0.9394	0.9319
8	0.9306	0.9375	0.9369	0.9337	0.9331	0.9400
9	0.9400	0.9444	0.9450	0.9412	0.9537	0.9437
Média	**0.9370**	**0.9385**	**0.9393**	**0.9400**	**0.9432**	**0.9403**

Tabela 5: Classificação utilizando o método 3 - Canônico + Esparso, com algoritmo ODL.

Resultados provenientes do método 3 - ODL						
Indivíduo	**Kappa** (±0.04)					
–	**a1 x a2**	**a1 x a3**	**a1 x a4**	**a2 x a3**	**a2 x a4**	**a3 x a4**
1	0.9362	0.9275	0.9344	0.9294	0.9350	0.9206
2	0.9350	0.9225	0.9262	0.9287	0.9287	0.9281
3	0.9450	0.9269	0.9181	0.9287	0.9350	0.9300
4	0.9644	0.9700	0.9644	0.9694	0.9656	0.9706
5	0.9425	0.9344	0.9425	0.9456	0.9462	0.9319
6	0.9344	0.9456	0.9387	0.9469	0.9362	0.9444
7	0.9275	0.9237	0.9387	0.9250	0.9306	0.9325
8	0.9437	0.9375	0.9400	0.9512	0.9369	0.9444
9	0.9450	0.9544	0.9444	0.9556	0.9512	0.9550
Média	**0.9415**	**0.9381**	**0.9386**	**0.9423**	**0.9406**	**0.9397**

Tabela 6: Classificação utilizando o método 3 - Canônico + Esparso, com algoritmo K-SVD.

Resultados provenientes do método 3 - K-SVD						
Indivíduo	**Kappa** (±0.04)					
–	**a1 x a2**	**a1 x a3**	**a1 x a4**	**a2 x a3**	**a2 x a4**	**a3 x a4**
1	0.9300	0.9319	0.9331	0.9250	0.9362	0.9350
2	0.9294	0.9337	0.9331	0.9331	0.9406	0.9337
3	0.9425	0.9262	0.9344	0.9244	0.9375	0.9262
4	0.9594	0.9719	0.9637	0.9600	0.9656	0.9675
5	0.9312	0.9269	0.9387	0.9369	0.9287	0.9325
6	0.9225	0.9394	0.9369	0.9456	0.9362	0.9387
7	0.9312	0.9337	0.9412	0.9400	0.9444	0.9500
8	0.9456	0.9444	0.9487	0.9387	0.9412	0.9419
9	0.9412	0.9406	0.9431	0.9419	0.9531	0.9444
Média	**0.9370**	**0.9387**	**0.9415**	**0.9384**	**0.9426**	**0.9411**

Tabela 7: Classificação utilizando o método 3 - Canônico + Esparso, com algoritmo LS-DLA.

Resultados provenientes do método 3 - LS-DLA						
Indivíduo	**Kappa** (±0.04)					
–	**a1 x a2**	**a1 x a3**	**a1 x a4**	**a2 x a3**	**a2 x a4**	**a3 x a4**
1	0.9387	0.9306	0.9287	0.9269	0.9319	0.9300
2	0.9319	0.9325	0.9344	0.9306	0.9356	0.9319
3	0.9381	0.9231	0.9256	0.9250	0.9381	0.9319
4	0.9612	0.9650	0.9650	0.9650	0.9600	0.9662
5	0.9400	0.9387	0.9406	0.9500	0.9450	0.9387
6	0.9212	0.9400	0.9319	0.9256	0.9194	0.9356
7	0.9319	0.9369	0.9475	0.9425	0.9431	0.9500
8	0.9312	0.9381	0.9306	0.9406	0.9362	0.9394
9	0.9425	0.9525	0.9369	0.9387	0.9487	0.9362
Média	**0.9374**	**0.9397**	**0.9379**	**0.9383**	**0.9398**	**0.9400**

Tabela 8: Classificação utilizando o método 4 - CSP + Esparso, com o algoritmo MOD.

Resultados provenientes do método 4 - MOD						
Indivíduo	**Kappa** (±0.04)					
–	**a1 x a2**	**a1 x a3**	**a1 x a4**	**a2 x a3**	**a2 x a4**	**a3 x a4**
1	0.6875	0.8675	0.7500	0.8750	0.8375	0.8250
2	0.6850	0.6850	0.6755	0.6175	0.5650	0.5925
3	0.6500	0.5625	0.6425	0.5900	0.5900	0.6475
4	0.8575	0.7850	0.8025	0.8075	0.7775	0.7975
5	0.7400	0.7550	0.6475	0.6400	0.6900	0.6550
6	0.6450	0.8650	0.8650	0.8625	0.8475	0.8350
7	0.6345	0.6425	0.6125	0.6150	0.6500	0.6350
8	0.6600	0.7550	0.7225	0.7525	0.7225	0.6875
9	0.6552	0.6375	0.6005	0.6775	0.7075	0.6475
Média	**0.6917**	**0.7283**	**0.7019**	**0.7152**	**0.7097**	**0.7025**

Tabela 9: Classificação utilizando o método - CSP + Esparso, com algoritmo ODL.

Resultados provenientes do método 4 - ODL						
Indivíduo	**Kappa** (±0.04)					
–	**a1 x a2**	**a1 x a3**	**a1 x a4**	**a2 x a3**	**a2 x a4**	**a3 x a4**
1	0.6875	0.8675	0.7500	0.8750	0.8375	0.8250
2	0.6850	0.6850	0.6750	0.6175	0.5650	0.5925
3	0.6500	0.5625	0.6425	0.5900	0.5900	0.6475
4	0.8575	0.7850	0.8025	0.8075	0.7775	0.7975
5	0.7400	0.7555	0.6475	0.6400	0.6900	0.6550
6	0.6456	0.8950	0.8650	0.8625	0.8475	0.8350
7	0.6450	0.6425	0.6125	0.6150	0.6500	0.6350
8	0.6600	0.7550	0.7225	0.75825	0.7225	0.6875
9	0.6550	0.6375	0.6000	0.6775	0.7075	0.6475
Média	**0.6917**	**0.7283**	**0.7019**	**0.7153**	**0.7097**	**0.7025**

Tabela 10: Classificação utilizando o método 4 - CSP + Esparso, com o algoritmo K-SVD.

Resultados provenientes do método 4 - K-SVD						
Indivíduo	**Kappa** (±0.04)					
–	**a1 x a2**	**a1 x a3**	**a1 x a4**	**a2 x a3**	**a2 x a4**	**a3 x a4**
1	0.5250	0.7950	0.7975	0.7875	0.7975	0.7825
2	0.5725	0.5700	0.5925	0.6075	0.5850	0.5675
3	0.5200	0.5850	0.5775	0.5255	0.5500	0.5900
4	0.8275	0.7621	0.7625	0.7975	0.7550	0.7825
5	0.6050	0.5775	0.5855	0.5600	0.5825	0.5950
6	0.5750	0.7975	0.7725	0.8125	0.7975	0.8025
7	0.5275	0.5525	0.5500	0.5200	0.5575	0.5925
8	0.5410	0.5725	0.5700	0.6125	0.5925	0.6175
9	0.5600	0.5850	0.5655	0.5625	0.5955	0.5825
Média	**0.5836**	**0.6444**	**0.6413**	**0.6425**	**0.6458**	**0.6558**

Tabela 11: Classificação utilizando o método 4 - CSP + Esparso, com o algoritmo RLS-DLA.

Resultados provenientes do método 4 - RLS-DLA						
Indivíduo	**Kappa** (±0.04)					
–	**a1 x a2**	**a1 x a3**	**a1 x a4**	**a2 x a3**	**a2 x a4**	**a3 x a4**
1	0.6500	0.7925	0.7675	0.7800	0.7925	0.7675
2	0.6275	0.6425	0.5770	0.6000	0.5975	0.5650
3	0.5450	0.6275	0.5775	0.6675	0.6075	0.6625
4	0.7850	0.7750	0.7800	0.8075	0.7800	0.7850
5	0.6850	0.6050	0.6650	0.6350	0.6275	0.6600
6	0.5800	0.8575	0.8025	0.8025	0.8325	0.8300
7	0.5900	0.7000	0.5325	0.5325	0.6175	0.5900
8	0.6100	0.6775	0.6550	0.6550	0.6675	0.7025
9	0.5925	0.5625	0.5450	0.5450	0.6250	0.5750
Média	**0.6294**	**0.6933**	**0.6555**	**0.6775**	**0.6831**	**0.6819**

Tabela 12: Classificação utilizando o método 4 - CSP + Esparso, com o algoritmo LS-DLA.

Resultados provenientes do método 4 - LS-DLA						
Indivíduo	**Kappa** (± 0.04)					
–	**a1 x a2**	**a1 x a3**	**a1 x a4**	**a2 x a3**	**a2 x a4**	**a3 x a4**
1	0.6950	0.8425	0.8200	0.8475	0.8525	0.8350
2	0.6400	0.6725	0.6050	0.6450	0.6425	0.6775
3	0.6100	0.6550	0.6155	0.7000	0.6350	0.5825
4	0.8000	0.8050	0.8350	0.8050	0.7800	0.7875
5	0.7300	0.6900	0.6900	0.7225	0.7075	0.6625
6	0.5775	0.8550	0.8125	0.7900	0.8300	0.8200
7	0.6450	0.5825	0.6125	0.5925	0.6225	0.6125
8	0.5850	0.6925	0.6950	0.6850	0.7150	0.6625
9	0.6525	0.6475	0.6625	0.6525	0.6550	0.6750
Média	**0.6594**	**0.7125**	**0.7053**	**0.7156**	**0.7156**	**0.7017**

Embora o quarto método tenha apresentado resultados melhores do que o método 2 - que utiliza apenas o CSP, sem a representação esparsa - não foi possível igualar o mesmo patamar de desempenho apresentado pelo método 3 , reforçando a ideia de que o uso do CSP, nesse caso, não traz benefícios ao problema de classificação dos sinais. Conjectura-se que o pré-processamento realizado pelo CSP pode descartar informações importantes na criação do dicionário para as densidades espectrais de potência.

5.3 Resultados - Classificação Multi-Classe

Considerando o problema global de classificação das 4 classes distintas, foram adotadas os mesmos métodos descritos, com a diferença de que é utilizado o método M-CSP (multiclass CSP) para o pré-processamento dos sinais, considerando os 4 principais filtros (cada um associado a uma classe).

O método 1, que obteve resultados bastante expressivos no caso de classificação binária, os valores de kappa no caso de 4-classes, apresentou um desempenho bastante inferior, conforme pode-se observar na Tabela 13. Aparentemente, embora fosse possível distinguir facilmente as densidades espectrais entre duas classes distintas, ao considerar um número maior de classes, as diferenças entre as densidades espectrais de potências tornam-se menos visíveis, levando a uma queda bastante significativa no desempenho do método.

Na tabela 14 são apresentados os resultados obtidos com o método 2, que considera o uso de pré-processamento com o M-CSP e a classificação diretamente sobre a densidade espectral obtida.

Observa-se que os valores de kappa para esse caso são ligeiramente inferiores ao

Tabela 13: Classificação utilizando o método 1 - Canônico.

Resultados provenientes do método 1	
Indivíduo	**Kappa**
1	0.4095± 0.03
2	0.3500± 0.03
3	0.3213± 0.03
4	0.3687± 0.03
5	0.3788± 0.03
6	0.3575± 0.03
7	0.3404± 0.03
8	0.3408± 0.03
9	0.3341± 0.03
Média	**0.3557 ± 0.03**

caso de classificação binária - um resultado já esperado dada a complexidade do problema de classificação. Cabe ressaltar que, nesse caso, utilizam-se quatro filtros no CSP - um associado a cada classe - o que possivelmente contribui para um melhor desempenho do método em relação aos resultados com apenas duas classes.

Tabela 14: Classificação utilizando o método 2 - CSP.

Resultados provenientes do método 2	
Indivíduo	**Kappa**
1	0.5891± 0.03
2	0.5958± 0.03
3	0.5883± 0.03
4	0.5904± 0.03
5	0.5775± 0.03
6	0.6829± 0.03
7	0.7341± 0.03
8	0.6254± 0.03
9	0.5562± 0.03
Média	**0.6155 ± 0.03**

Para o método 3, foram obtidos os resultados apresentados na tabela 15. Claramente, corresponde à melhor abordagem dentre as analisadas, uma vez que os valores de kappa, para todos os indivíduos, é superior a 0,8. Novamente, a exemplo do que já foi observado anteriormente, há uma ligeira diferença no desempenho para os diferentes métodos de criação de dicionário, mas em todos os casos foi possível atingir um patamar de desempenho bastante elevado.

Finalmente, com o método 4, o desempenho também não se destacou, e os resultados podem ser vistos na Tabela 16. Comparando os resultados com os obtidos no método 2, na tabela 14, nesse caso, a utilização da representação esparsa levou a uma melhoria no desempenho da interface.

Tabela 15: Classificação utilizando o método 3 - Canônico + Esparso.

Resultados provenientes do método 3					
Indivíduo	**Kappa**				
–	**MOD**	**RLS-DLA**	**LS-DLA**	**ODL**	**K-SVD**
1	0.9150± 0.03	0.8588± 0.03	0.8950± 0.03	0.9062± 0.03	0.8737± 0.03
2	0.8713± 0.03	0.9038± 0.03	0.8963± 0.03	0.8713± 0.03	0.8875± 0.03
3	0.8838± 0.03	0.8388± 0.03	0.9125± 0.03	0.8838± 0.03	0.8975± 0.03
4	0.9463± 0.03	0.9338± 0.03	0.9388± 0.03	0.9225± 0.03	0.9625± 0.03
5	0.9000± 0.03	0.8925± 0.03	0.8913± 0.03	0.9037± 0.03	0.9413± 0.03
6	0.9125± 0.03	0.9063± 0.03	0.9238± 0.03	0.8950± 0.03	0.9062± 0.03
7	0.9013± 0.03	0.9113± 0.03	0.8962± 0.03	0.8712± 0.03	0.9038± 0.03
8	0.9250± 0.03	0.9288± 0.03	0.8735± 0.03	0.9037± 0.03	0.9138± 0.03
9	0.9200± 0.03	0.9138± 0.03	0.9037± 0.03	0.9100± 0.03	0.9837± 0.03
Média	**0.9083 ± 0.03**	**0.8986 ± 0.03**	**0.9035 ± 0.03**	**0.8964 ± 0.03**	**0.9124 ± 0.03**

Embora no caso de classificação binária (2 classes) o uso da representação esparsa em conjunto com o CSP não tenha trazido benefícios, no caso de várias classes, é possível que a utilização de mais de um filtro no pré-processamento, lembrando que nesse caso são utilizados 4 filtros, tenha preservado a informação relevante para a criação do dicionário e posterior classificação dos sinais.

Tabela 16: Classificação utilizando o método 4 - CSP + Esparso.

Resultados provenientes do método 4					
Indivíduo	**Kappa**				
–	**MOD**	**RLS-DLA**	**LS-DLA**	**ODL**	**K-SVD**
1	0.7212± 0.03	0.6712± 0.03	0.7474± 0.03	0.7662± 0.03	0.6612± 0.03
2	0.7062± 0.03	0.6787± 0.03	0.7337± 0.03	0.7374± 0.03	0.6912± 0.03
3	0.6940± 0.03	0.6374± 0.03	0.6124± 0.03	0.5912± 0.03	0.6712± 0.03
4	0.6612± 0.03	0.5487± 0.03	0.5887± 0.03	0.5375± 0.03	0.6037± 0.03
5	0.6474± 0.03	0.6287± 0.03	0.6750± 0.03	0.6150± 0.03	0.6562± 0.03
6	0.6862± 0.03	0.7424± 0.03	0.7662± 0.03	0.7475± 0.03	0.7637± 0.03
7	0.8474± 0.03	0.8174± 0.03	0.8200± 0.03	0.8237± 0.03	0.8299± 0.03
8	0.8474± 0.03	0.7962± 0.03	0.7825± 0.03	0.8337± 0.03	0.7637± 0.03
9	0.7112± 0.03	0.6924± 0.03	0.6212± 0.03	0.7187± 0.03	0.6787± 0.03
Média	**0.7248 ± 0.03**	**0.6904 ± 0.03**	**0.7053 ± 0.03**	**0.7079 ± 0.03**	**0.7022 ± 0.03**

A título de comparação, na tabela 17 são mostrados os valores de Kappa obtidos na competição de BCI - BCI Competition IV[Tangermann et al. 2012], de onde foram obtidos os dados para os testes. Os resultados obtidos são bastante animadores, mostrando-se significativamente superiores ao primeiro colocado na competição, à época.

Tabela 17: Resultados publicados pelo BCI Competition IV [Tangermann et al. 2012].

Resultados publicados					
Indivíduo	**Kappa - Colocados**				
–	**1º**	**2º**	**3º**	**4º**	**5º**
1	0.68	0.69	0.38	0.46	0.41
2	0.42	0.34	0.18	0.25	0.17
3	0.75	0.71	0.48	0.65	0.39
4	0.48	0.44	0.33	0.31	0.25
5	0.40	0.16	0.07	0.12	0.06
6	0.27	0.21	0.14	0.07	0.16
7	0.77	0.66	0.29	0.00	0.34
8	0.75	0.73	0.49	0.46	0.45
9	0.61	0.69	0.44	0.42	0.37
Média	**0.57**	**0.52**	**0.31**	**0.30**	**0.29**

Em resumo, o uso dos algoritmos de criação de dicionário para a representação esparsa das densidades, em geral, traz benefícios ao problema de classificação. Apenas no problema de classificação binária, com o uso de apenas um filtro no pré-processamento pelo CSP, a abordagem explorando a representação esparsa não foi capaz de melhorar o desempenho do sistema.

O desempenho obtido para os diferentes algoritmos de criação de dicionário foram similares, indicando que a metodologia de inicialização é efetiva nesse caso. A escolha por um método em detrimento do outro, portanto, poderia ser feita baseada em outro requisito, como o de complexidade computacional, facilidade de implementação, ou flexibilidade para lidar com um conjunto maior de dados para treinamento.

6 Conclusão e Considerações Finais

Neste trabalho foram investigadas algumas maneiras de se utilizar técnicas de representação esparsa no problema de classificação de sinais neuronais, especificamente na construção de uma interface cérebro-computador.

Foram estudados diferentes métodos para estimação da representação esparsa e para a criação de dicionários. Os métodos foram utilizados para a criação de dicionários para representação do espectro dos sinais neuronais, visando facilitar a tarefa de classificação dos mesmos.

Os diferentes algoritmos para criação de dicionário não apresentaram diferenças significativas, atingindo desempenho satisfatório nos casos avaliados. A estratégia para a criação do dicionário inicial, utilizando as próprias amostras dos sinais utilizados no treinamento, aparentemente mostrou-se adequada para o problema de classificação.

A partir de dicionários criados com as próprias densidades espectrais dos sinais para treinamento pode-se verificar, por meio do coeficiente kappa, que o uso da representação esparsa é bastante efetiva no problema de classificação dos sinais.

Em particular, no caso de utilização para representação da densidade espectral de potência, sem qualquer uso de pre-processamento com o CSP, os resultados obtidos foram superiores ao melhor colocado da competição de BCI, conseguindo acurácia de **0.8986** (RLS-DLA), **0.9035** (LS-DLA), **0.9124** (K-SVD), **0.8964** (ODL) e **0.9083** (MOD). Esta abordagem estudada é diferente das proposta encontradas na literatura, e pode ser considerada uma contribuição original do presente trabalho.

Como perspectivas para trabalhos futuros, vislumbra-se investigar formas de redução do custo computacional (como utilizar outras técnicas para seleção de características), visando a implementação dos métodos em hardware específico para o teste em tempo real de interfaces BCI; realização de um estudo mais amplo da efetividade das soluções estudadas em situação real de utilização; investigação sobre a extensão das técnicas para a classificação de uma quantidade maior de classes, bem como a aplicação em outras abordagens de BCI (como no paradigma de SSVEP).

Referências

[Aharon, Elad e Bruckstein 2006]AHARON, M.; ELAD, M.; BRUCKSTEIN, A. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, v. 54, n. 11, p. 4311–4322, 2006.

[Alamdari et al. 2016]ALAMDARI, N. et al. A review of methods and applications of brain computer interface systems. In: *2016 IEEE International Conference on Electro Information Technology (EIT)*. [S.l.: s.n.], 2016. p. 0345–0350.

[Allison, Wolpaw e Wolpaw 2007]ALLISON, B. Z.; WOLPAW, E. W.; WOLPAW, J. R. Brain–computer interface systems: progress and prospects. *Expert Review of Medical Devices*, Taylor Francis, v. 4, n. 4, p. 463–474, 2007. Disponível em: <https://doi.org/10.1586/17434440.4.4.463>.

[Ameri, Pouyan e Abolghasemi 2016]AMERI, R.; POUYAN, A.; ABOLGHASEMI, V. Projective dictionary pair learning for eeg signal classification in brain computer interface applications. *Neurocomputing*, v. 218, p. 382 – 389, 2016. ISSN 0925-2312. Disponível em: <http://www.sciencedirect.com/science/article/pii/S092523121630978X>.

[Bassani e J.C. 2010]BASSANI, T.; J.C., N. Brain-computer interface using wavelet transformation and naïve bayes classifier. *PubMed, U.S. National Library of Medicine*, 2010.

[Blankertz et al. 2008]BLANKERTZ, B. et al. The berlin brain - computer interface: Accurate performance from first-session in bci-naïve subjects. *IEEE Transactions on Biomedical Engineering*, 2008. ISSN 00189294.

[Blum e Langley 1997]BLUM, A. L.; LANGLEY, P. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, v. 97, n. 1, p. 245 – 271, 1997. ISSN 0004-3702. Relevance. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0004370297000635>.

[Bradley et al. 2004]BRADLEY, E. et al. Least angle regression. *The Annals of Statistics*, v. 2, 2004.

[Brunner et al. 2011]BRUNNER, P. et al. Rapid communication with a "p300" matrix speller using electrocorticographic signals (ecog). *Frontiers in Neuroscience*, v. 5, p. 5, 2011. ISSN 1662-453X. Disponível em: <https://www.frontiersin.org/article/10.3389/fnins.2011.00005>.

[Chen 2004]CHEN, C. Q. An enhanced generalized lloyd algorithm. *IEEE Signal Processing Letters*, 2004. ISSN 10709908.

[Cover e Hart 1967]COVER, T. M.; HART, P. E. Nearest neighbor pattern classification. *IEEE Transactions on information theory*, p. 21–27, 1967.

[Dai, Xu e Wang 2012]DAI, W.; XU, T.; WANG, W. Simultaneous codeword optimization (simco) for dictionary update and learning. *IEEE Transactions on Signal Processing*, 2012. ISSN 1053587X.

[Elad 2013]ELAD, M. *Sparse and Redunant Representations*. [S.l.: s.n.], 2013. 1689–1699 p.

[Engan, Aase e Husoy 1999]ENGAN, K.; AASE, S. O.; HUSOY, J. H. Method of optimal directions for frame design. In: *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*. [S.l.: s.n.], 1999. v. 5, p. 2443–2446 vol.5. ISSN 1520-6149.

[Engan, Aase e Husøy 2000]ENGAN, K.; AASE, S. O.; HUSØY, J. H. Multi-frame compression: Theory and design. *Signal Processing*, 2000. ISSN 01651684.

[Golub e Loan 1996]GOLUB, G. H.; LOAN, C. F. V. *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996. ISBN 0-8018-5414-8.

[Graimann, Allison e Pfurtscheller 2010]GRAIMANN, B.; ALLISON, B.; PFURTSCHELLER, G. *Brain-Computer Interfaces, Revolutionizing Human-Computer Interaction*. [S.l.]: Springer, 2010. 1-397 p.

[Grosse-Wentrup* e Buss 2008]GROSSE-WENTRUP*, M.; BUSS, M. Multiclass common spatial patterns and information theoretic feature extraction. *IEEE Transactions on Biomedical Engineering*, v. 55, n. 8, p. 1991–2000, Aug 2008. ISSN 0018-9294.

[Hameed 2012]HAMEED, M. A. *Comparative analysis of orthogonal matching pursuit and least angle regression*. 81 p. Dissertação (Mestrado), 2012.

[Hwang, Lee e Lee 2017]HWANG, J.-Y.; LEE, M.-H.; LEE, S.-W. A brain-computer interface speller using peripheral stimulus-based ssvep and p300. *2017 5th International Winter Conference on Brain-Computer Interface (BCI)*, 2017.

[Jeannerod 1995]JEANNEROD, M. Mental imagery in the motor context. *Neuropsychologia*, 1995. ISSN 00283932.

[Kohavi e John 1997]KOHAVI, R.; JOHN, G. H. Wrappers for feature subset selection. *Artificial Intelligence*, v. 97, n. 1, p. 273 – 324, 1997. ISSN 0004-3702. Relevance. Disponível em: <http://www.sciencedirect.com/science/article/pii/S000437029700043X>.

[Koles, Lazar e Zhou]KOLES, Z. J.; LAZAR, M. S.; ZHOU, S. Z. *Brain Topography*, Human Sciences Press, v. 2.

[Laub 2005]LAUB, A. J. *Matrix Analysis for Scientists & Engineers*. 4. ed. [S.l.]: SIAM, 2005.

[Liyanage et al. 2010]LIYANAGE, S. R. et al. Eeg signal separation for multi-class motor imagery using common spatial patterns based on joint approximate diagonalization. In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2010. p. 1–6. ISSN 2161-4393.

[Lotte, Bougrain e Clerc 2015]LOTTE, F.; BOUGRAIN, L.; CLERC, M. Electroencephalography (eeg)-based braincomputer interfaces. *Wiley Encyclopedia of Electrical and Electronics Engineering*, Wiley, 2015.

[Mairal et al. 2009]MAIRAL, J. et al. Online dictionary learning for sparse coding. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, 2009. (ICML '09), p. 689–696. ISBN 978-1-60558-516-1. Disponível em: <http://doi.acm.org/10.1145/1553374.1553463>.

[Mohammadpour, Ghorbanianm e Mozaffari 2016]MOHAMMADPOUR, M.; GHORBANIANM, M.; MOZAFFARI, S. Comparison of eeg signal features and ensemble learning methods for motor imagery classification. In: *2016 Eighth International Conference on Information and Knowledge Technology (IKT)*. [S.l.: s.n.], 2016. p. 288–292.

[Nadalin, Suyama e Attux 2010]NADALIN, E. Z.; SUYAMA, R.; ATTUX, R. Um Breve Estudo sobre Análise de Componentes Esparsos. In: *Anais do III Encontro dos Alunos do DCA - EADCA (FEEC/UNICAMP)*. [S.l.: s.n.], 2010.

[Nicolas-Alonso e Gomez-Gil 2012]NICOLAS-ALONSO, L. F.; GOMEZ-GIL, J. Brain computer interfaces, a review. *Sensors*, 2012. ISSN 14248220.

[Norani e Md 2010]NORANI; MD, N. A Review of Signal Processing in Brain Computer Interface System. *IEEE EMBS Conference on Biomedical Engineering & Sciences*, IEEE, 2010.

[Nurse et al. 2015]NURSE, E. S. et al. A generalizable brain-computer interface (bci) using machine learning for feature discovery. *PLOS ONE*, 2015.

[Obermaier, Guger e Pfurtscheller 1999]OBERMAIER, B.; GUGER, C.; PFURTSCHELLER, G. Hidden markov models used for the offline classification of eeg data. *PubMed, U.S. National Library of Medicine*, 1999.

[Oppenheim, Willsky e Nawab 1996]OPPENHEIM, A. V.; WILLSKY, A. S.; NAWAB, S. H. *Signals systems (2nd ed.)*. 2nd. ed. [S.l.]: Prentice Hall, 1996. 957 p. ISBN 0-13-814757-4.

[Peleg e Elad 2012]PELEG, T.; ELAD, M. Exploiting statistical dependencies in sparse representations for signal recovery. *IEEE Transactions on Signal Processing*, v. 60, n. 5, p. 2286–2303, 2012.

[Pfurtscheller e Neuper 2001]PFURTSCHELLER, G.; NEUPER, C. Motor imagery and direct brain–computer communication. *Proceeding of the IEEE*, 2001.

[Pfurtscheller et al. 2003]PFURTSCHELLER, G. et al. Graz-BCI: State of the art and clinical applications. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2003. ISSN 15344320.

[Qin, Li e Sun 2007]QIN, J.; LI, Y.; SUN, W. A semisupervised support vector machines algorithm for bci systems. *PubMed, U.S. National Library of Medicine*, 2007.

[Rao 2013]RAO, R. P. N. *Brain-Computer Interfacing An introduction*. 1. ed. [S.l.]: Cambridge, 2013. ISBN 9780521769419.

[Romano et al. 2009]ROMANO, J. M. T. et al. *Independent Component Analysis and Signal Separation*. [S.l.]: Springer, 2009. 313 p. ISBN 978-3-642-00599-2.

[Rubinstein, Zibulevsky e Elad 2008]RUBINSTEIN, R.; ZIBULEVSKY, M.; ELAD, M. Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. *CS Technion*, p. 1–15, 2008.

[Sanei e Chambers 2007]SANEI, S.; CHAMBERS, J. *EEG Signal Processing*. 1st. ed. [S.l.]: John Wiley Sons, 2007. 313 p. ISBN 9780470511923.

[Saugat et al. 2010]SAUGAT, B. et al. Performance analysis of lda, qda and knn algorithms in left-right limb movement classification from eeg data. *International Conference on Systems in Medicine and Biology (ICSMB)*, 2010.

[Schlögl et al. 2007]SCHLöGL, A. et al. Evaluation Criteria for BCI Research. In: _____. *Toward Brain-computer Interfacing*. [S.l.]: MIT Press, 2007. cap. 19, p. 327–342.

[Shin et al. 2013]SHIN, Y. et al. Performance increase by using a eeg sparse representation based classification method. In: *2013 IEEE International Conference on Consumer Electronics (ICCE)*. [S.l.: s.n.], 2013. p. 201–203. ISSN 2158-3994.

[Silva et al. 2017]SILVA, V. F. et al. Ensemble learning based classification for bci applications. *IEEE 5th Portuguese Meeting on Bioengineering (ENBENG).*, 2017.

[Skretting 2011]SKRETTING, K. E. K. *Learned dictionaries for sparse image representation: properties and results*. 2011. 8138 - 8138 - 14 p. Disponível em: <http://dx.doi.org/10.1117/12.892684>.

[Smith 1999]SMITH, S. W. *The Scientist and Engineer's Guide to Digital Signal Processing*. 2. ed. [S.l.]: California technical Publishing, 1999. ISBN 0-9660176-4-1,0-9660176-6-8,0-9660176-7-6,9780966017649.

[Soriano et al. 2014]SORIANO, D. C. et al. A recurrence-based approach for feature extraction in brain-computer interface systems. *Springer International Publishing*, 2014.

[Tangermann et al. 2012]TANGERMANN, M. et al. Review of the bci competition iv. *Frontiers in Neuroscience*, v. 6, p. 55, 2012. ISSN 1662-453X. Disponível em: <https://www.frontiersin.org/article/10.3389/fnins.2012.00055>.

[Treichler 2012]TREICHLER, J. *Notes on Design of Optimal FIR Filters*. [S.l.: s.n.], 2012. 1–47 p.

[Wang, Gao e Gao 2005]WANG, Y.; GAO, S.; GAO, X. Common spatial pattern method for channel selelction in motor imagery based brain-computer interface. In: *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*. [S.l.: s.n.], 2005. p. 5392–5395. ISSN 1094-687X.

[Wright et al. 2009]WRIGHT, J. et al. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 31, n. 2, p. 210–227, 2009.

[Wu et al. 2008]WU, X. et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, v. 14, n. 1, p. 1–37, Jan 2008. ISSN 0219-3116. Disponível em: <https://doi.org/10.1007/s10115-007-0114-2>.