UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Luana Cristina Petter

IMPLEMENTAÇÃO DE UMA METODOLOGIA PARA CLASSIFICAÇÃO DE IMAGÉTICA MOTORA EM MÓDULO ORANGE PI ZERO

Santa Maria, RS
2017

Luana Cristina Petter

IMPLEMENTAÇÃO DE UMA METODOLOGIA PARA CLASSIFICAÇÃO DE IMAGÉTICA MOTORA EM MÓDULO ORANGE PI ZERO

Trabalho de Conclusão de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Bacharel em Engenharia de Computação.

ORIENTADOR: Prof. Cesar Ramos Rodrigues

Santa Maria, RS
2017

Luana Cristina Petter

IMPLEMENTAÇÃO DE UMA METODOLOGIA PARA CLASSIFICAÇÃO DE IMAGÉTICA MOTORA EM MÓDULO ORANGE PI ZERO

Trabalho de Conclusão de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Bacharel em Engenharia de Computação.

Aprovado em 15 de dezembro de 2017:

Cesar Ramos Rodrigues, Dr. (UFSM)
(Presidente/Orientador)

Cesar Augusto Prior, Dr. (UFSM)

José Eduardo Baggio, Dr. (UFSM)

Santa Maria, RS
2017

DEDICATÓRIA

Dedico este trabalho aos meus pais, pois confiaram em mim e me deram esta oportunidade de concretizar mais uma caminhada em minha vida.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por ter me dado saúde e força para superar as dificuldades.

Aos meus pais, irmã e amigos, pela convivência e suporte nos momentos necessários.

Ao meu orientador Professor Dr. Cesar Rodrigues, pela orientação e apoio na elaboração deste trabalho. Agradeço também aos Professores Dr. Cesar Prior e Dr. José Baggio por terem aceito fazer parte da banca avaliadora.

Ao "Simba" meu gato de estimação que alegrou meus dias ao longo destes anos de graduação, e principalmente pela sua companhia nas horas de desenvolvimento e escrita deste trabalho.

E a todos que direta ou indiretamente fizeram parte da minha formação.

Você não consegue ligar os pontos olhando pra frente; você só consegue ligá-los olhando pra trás. Então você tem que confiar que os pontos se ligarão algum dia no futuro. Você tem que confiar em algo – seu instinto, destino, vida, carma, o que for. Esta abordagem nunca me desapontou, e fez toda diferença na minha vida.

(Steve Jobs)

RESUMO

IMPLEMENTAÇÃO DE UMA METODOLOGIA PARA CLASSIFICAÇÃO DE IMAGÉTICA MOTORA EM MÓDULO ORANGE PI ZERO

AUTORA: Luana Cristina Petter
ORIENTADOR: Cesar Ramos Rodrigues

Várias pessoas no mundo inteiro vivem com alguma forma de deficiência física. Para que essa parte da população que apresenta alguma deficiência motora, possa se comunicar ou executar tarefas que estão impossibilitadas de realizar é de suma importância o desenvolvimento de tecnologias como Interfaces Cérebro-Computador (BCI). Pois uma aplicação BCI funciona convertendo os sinais cerebrais em comandos para controlar determinados recursos externos, tais como órteses, cadeira de rodas, entre outros dispositivos controláveis. Podendo contribuir significativamente para a melhoria da condição de vida dessas pessoas.

O objetivo principal deste trabalho é implementar em uma plataforma portátil as técnicas já testadas em um computador, referentes a um sistema de detecção de intenção de movimento baseado em sinais de EEG, implementadas por (SILVA, 2017).

Este trabalho apresenta experimentos iniciais para identificação de imagética motora, referente à detecção de intenção de movimento das mãos ou detecção de intenção de movimento referente a mão ou pé. Visando contribuir para o incremento da confiabilidade na detecção de intenções a ponto de permitir sua integração em um sistema BCI com aplicação prática.

Os resultados são apresentados referentes a etapa de treino do algoritmo e simulação em tempo real, onde o algoritmo treinado é executado na plataforma portátil.

Na metodologia proposta por (SILVA, 2017), o sinal é filtrado em ritmos cerebrais relacionados à tarefas de imagética motora. Após a filtragem, atributos estatísticos desses ritmos cerebrais são extraídos e alimentam um classificador (utilizado Florestas Aleatórias e Máquina de Vetores de Suporte) que realiza a predição sobre o conjunto de testes. Neste trabalho, ao invés do classificador ser aplicado sobre o conjunto de testes em uma única vez, a predição de cada tentativa é realizada uma a uma, na etapa de simulação em tempo real.

Os testes foram realizados com classes desbalanceadas e balanceadas. Também foram realizados testes com a detecção de mão direita e mão esquerda ao invés da utilização das classes mão e pé, conforme sugerido por revisores do trabalho apresentado por (SILVA, 2017), para poder verificar a performance da metodologia proposta.

Palavras-chave: EEG, Interface Cérebro-Máquina, Plataforma Portátil, Florestas Aleatórias, Máquina de Vetores de Suporte, Imagética Motora

ABSTRACT

IMPLEMENTATION OF A METHODOLOGY FOR CLASSIFICATION OF MOTOR IMAGERY IN MODULE ORANGE PI ZERO

AUTHOR: Luana Cristina Petter
ADVISOR: Cesar Ramos Rodrigues

Various people around the world live with some form of physical disability. For this part of the population that has some motor deficiency, can communicate or perform tasks that are impossible to perform is of utmost importance the development of technologies such as Brain Computer Interface (BCI). Because a BCI application works by converting the brain signals into commands to control external resources, such as orthotics, wheelchair, among other controllable devices. Being able to contribute significantly to the improvement of the living condition of these people.

The main goal of the current study is to implement in a portable platform the techniques already tested in a computer, referring to a system of detection of movement intention based on EEG signals, implemented by (SILVA, 2017).

This current study presents initial experiments for the identification of motor imagery, referring to the detection of intention of movement of the hands or detection of intention of movement referring to hand or foot. Aiming to contribute to the increase of reliability in the detection of intentions to the point of allowing its integration in a BCI system with practical application.

The results are presented referring to the training stage of the algorithm and simulation in real time, where the trained algorithm is executed in the portable platform.

In the methodology proposed by (SILVA, 2017), the signal is filtered in cerebral rhythms related to the tasks of motor imagery. After filtering, statistical attributes of these brain rhythms are extracted and fed to a classifier (used Random Forest and Support Vector Machine) that performs the prediction on the set of tests. In this current study, instead of the classifier being applied on the test set in a single time, the prediction of each attempt is performed one by one, in the real-time simulation step.

The tests were performed with unbalanced and balanced classes. Were also performed tests with right hand and left hand detection instead of using the of hand and foot classes, as suggested by reviewers of the paper presented by (SILVA, 2017), in order to verify the performance of the proposed methodology.

Keywords: EEG, Brain Computer Interface, Portable Platform, Random Forest, Support Vector Machine, Motor Imagery

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AVC	Acidente Vascular Cerebral
AUC	Área sob a curva
EEG	Eletroencefalograma
PCA	Análise de Componentes Principais
RF	Florestas Aleatórias
SVM	Máquina de Vetores de Suporte
ROC	Característica de operação do receptor
PR	Precisão-revocação
TPR	Taxa de verdadeiros positivos
FPR	Taxa de verdadeiros negativos
IM	Imagética Motora
ICC	Interface Cérebro- Computador
ICM	Interface Cérebro-Máquina
BCI	*Brain Computer Interface*
BMI	*Brain Machine Interface*
$IBGE$	Instituto Brasileiro de Geografia e Estatística
SPI	*Serial Peripheral Interface*
LH	Mão esquerda
RH	Mão direita
F	Pé
H	Mão
R	Relaxamento
SO	Sistema Operacional
CV	Validação cruzada
D	Dimensão
TP	Verdadeiro Positivo
TN	Verdadeiro Negativo

FP	Falso Positivo
FN	Falso Negativo
TPR	*True Positive Rate*
TVP	Taxa de Verdadeiro Positivo
TNR	*True Negative Rate*
TVN	Taxa de Verdadeiro Negativo
κ	Kappa
X	Conjunto de Dados
$\bar{X}$(X)	Média
$V(X)$	Variância
$S(X)$	Obliquidade
$MOSI$	*Master Output Slave Input*
$MISO$	*Master Input Slave Output*
$SCLK$	*Serial Clock*
SS	*Slave Select*

LISTA DE SÍMBOLOS

μ	Banda de frequência mu
β	Banda de frequência beta
M	Matriz
Acc	Acurácia
C	Regularização
σ	*kernel*

SUMÁRIO

1 INTRODUÇÃO

Várias pessoas no mundo inteiro vivem com alguma forma de deficiência física. Essa limitação severa do movimento, pode surgir de uma lesão na coluna cervical, de acidente vascular cerebral (AVC), de alguma doença degenerativa (por exemplo esclerose lateral amiotrófica), ou ainda de qualquer outro mal que tenha como consequência a deficiência motora.

De acordo com o censo demográfico (IBGE, 2010), realizado pelo Instituto Brasileiro de Geografia e Estatística (IBGE) cerca de 45,6 milhões de pessoas declararam ter pelos menos uma das deficiências investigadas (deficiência visual, deficiência auditiva, deficiência motora e deficiência mental/intelectual), correspondendo a 23,9% da população brasileira.

Para que essa parte da população que apresenta alguma deficiência motora, possa se comunicar ou executar tarefas que estão impossibilitadas de realizar é de suma importância o desenvolvimento de tecnologias como Interfaces Cérebro-Computador (BCI). Pois uma aplicação BCI funciona convertendo os sinais cerebrais em comandos para controlar determinados recursos externos, tais como órteses, cadeira de rodas, entre outros dispositivos controláveis. Podendo contribuir significativamente para a melhoria da condição de vida dessas pessoas.

O foco deste trabalho é implementar em uma plataforma portátil (Orange Pi Zero) uma metodologia de reconhecimento de padrões em sinais de EEG para detecção de intenção de movimento referente a mão ou pé. Visando contribuir para o incremento da confiabilidade na detecção de intenções a ponto de permitir sua integração em um sistema BCI com aplicação prática.

Para poder verificar a performance da metodologia proposta, também foram realizados testes com classes de forma balanceadas, com a detecção de intenção de movimento referente a mão direita ou mão esquerda, conforme sugerido por revisores do trabalho apresentado por (SILVA, 2017).

1.1 INTERFACES CÉREBRO-MÁQUINA

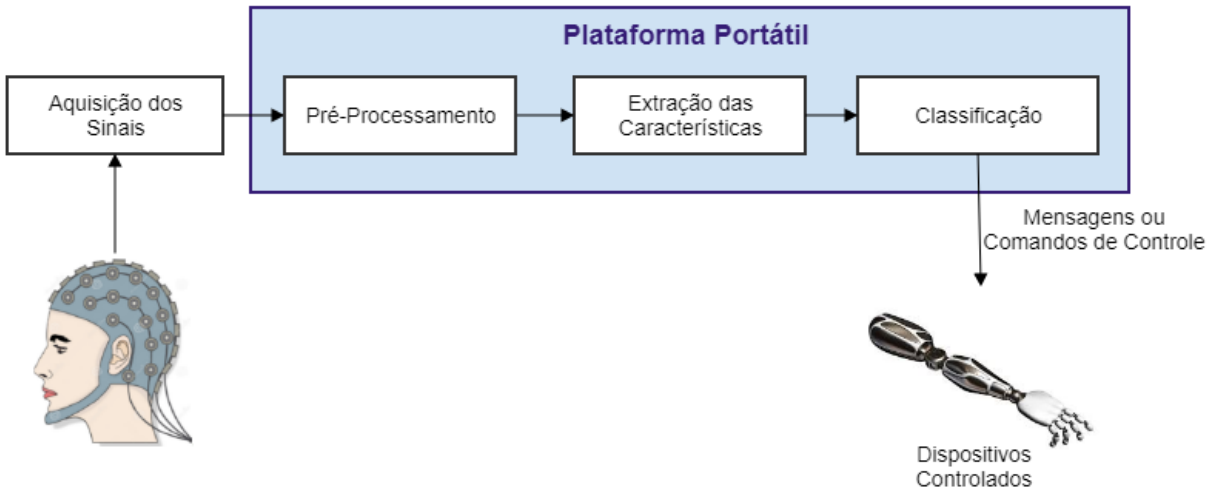
Para que possamos nos comunicar com as máquinas, são necessárias as chamadas Interfaces Cérebro-Máquina (ICM) ou ainda conhecidas como Interfaces Cérebro-Computador (ICC), do inglês *Brain Machine Interface* (BMI) ou *Brain Computer Interface* (BCI).

O principal objetivo de uma ICM é o desenvolvimento de um sistema computacional capaz de interpretar informação codificada na atividade elétrica de grupos neuronais para controlar um dispositivo artificial. Estes sinais devem ser analisados em tempo real e posteriormente traduzidos em comandos para controlar um dispositivo artificial (BARBOSA et al., 2009).

As principais aplicações das ICMs estão relacionadas à mobilidade, comunicação e interação de usuários que perderam o controle muscular voluntário, mas não apresentam danos cognitivos. Desenvolvendo-se como uma forma de tratamento para pacientes com diferentes níveis de paralisia corporal, como a paraplegia (perda das funções sensoriais e motoras nos membros inferiores) e a tetraplegia (perda destas funções também no tronco, incluindo braços) (MUSSATTO; SILVA, 2014).

Um sistema ICM compreende várias etapas: registro dos sinais, pré-processamento, extração características e, por fim, classificação para a geração de saídas artificiais que atuam sobre o ambiente ou sobre o próprio corpo (FARIAS, 2014). A Figura 1.1 que representa a estrutura básica de uma ICM.

Figura 1.1 – Estrutura básica de uma ICM



Fonte: AUTOR

A aquisição dos sinais é a etapa que tem a função de captar os sinais da atividade elétrica cerebral, sejam estes obtidos por meio de métodos invasivos (inserção de eletrodos de forma intracraniana no córtex cerebral) ou não-invasivos (eletrodos colocados externamente no couro cabeludo). A captação não invasiva de sinais de eletroencefalograma (EEG) é utilizada em aplicações ICM, devido ao baixo custo e facilidade de uso, oferecendo um nível de qualidade de sinal aceitável.

A etapa de pré-processamento é destinada a condicionar os sinais em processos de filtragem e remoção dos artefatos (a eletroencefalografia é bastante suscetível a inter-

ferências denominadas artefatos), preparando o sinal para seu posterior processamento.

A extração de características é a etapa em que o sinal EEG é representado por meio de informações (características) extraídas do mesmo a fim de reduzir a dimensão do vetor de dados (sem a perda de informações relevantes), disponibilizando esta representação do sinal para a etapa seguinte, a de classificação. Esta é uma etapa crucial em um sistema de ICM uma vez que interfere no desempenho do algoritmo classificador que decifrará a intenção do usuário (MUSSATTO; SILVA, 2014).

O estágio de classificação, consiste em interpretar as informações do estágio anterior por meio de um algoritmo de classificação, a fim de identificar qual são as intenções do indivíduo. Dentre os algoritmos de classificação estão: classificadores lineares, máquinas de vetores de suporte (Support Vector Machines - SVM), redes neurais, florestas aleatórias (Random Forests - RF), entre outros (NICOLAS-ALONSO; GOMEZ-GIL, 2012).

Por fim, a etapa de interface de controle ou saída dos dados, traduz os sinais classificados em comandos significativos para acionar dispositivos controlados, para assim substituir os movimentos de um membro humano, por exemplo.

1.2 TRABALHOS ANTERIORES

No Grupo de Microeletrônica da UFSM, pesquisas referentes ao processamento de sinais cerebrais usando sinas EEG são desenvolvidos há alguns anos. Sendo este trabalho uma evolução do trabalho de reconhecimento de padrões em sinais de EEG para detecção de intenção de movimento (SILVA, 2017).

1.3 OBJETIVOS

O principal objetivo é implementar em uma plataforma portátil (Orange Pi Zero) as técnicas já testadas em um computador (linguagem Python) referentes a um sistema de detecção de intenção de movimento baseado em sinais de EEG, implementadas por (SILVA, 2017).

O sistema portátil deve ser capaz de simular uma classificação em tempo real, recebendo como entrada um sinal de EEG (amostrado e sem ruído da rede elétrica), gerando uma saída que informe qual era a intenção de movimento do indivíduo em questão. Para isso é utilizada uma base de dados com intenções previamente conhecidas.

1.4 ESTRUTURA DO TEXTO

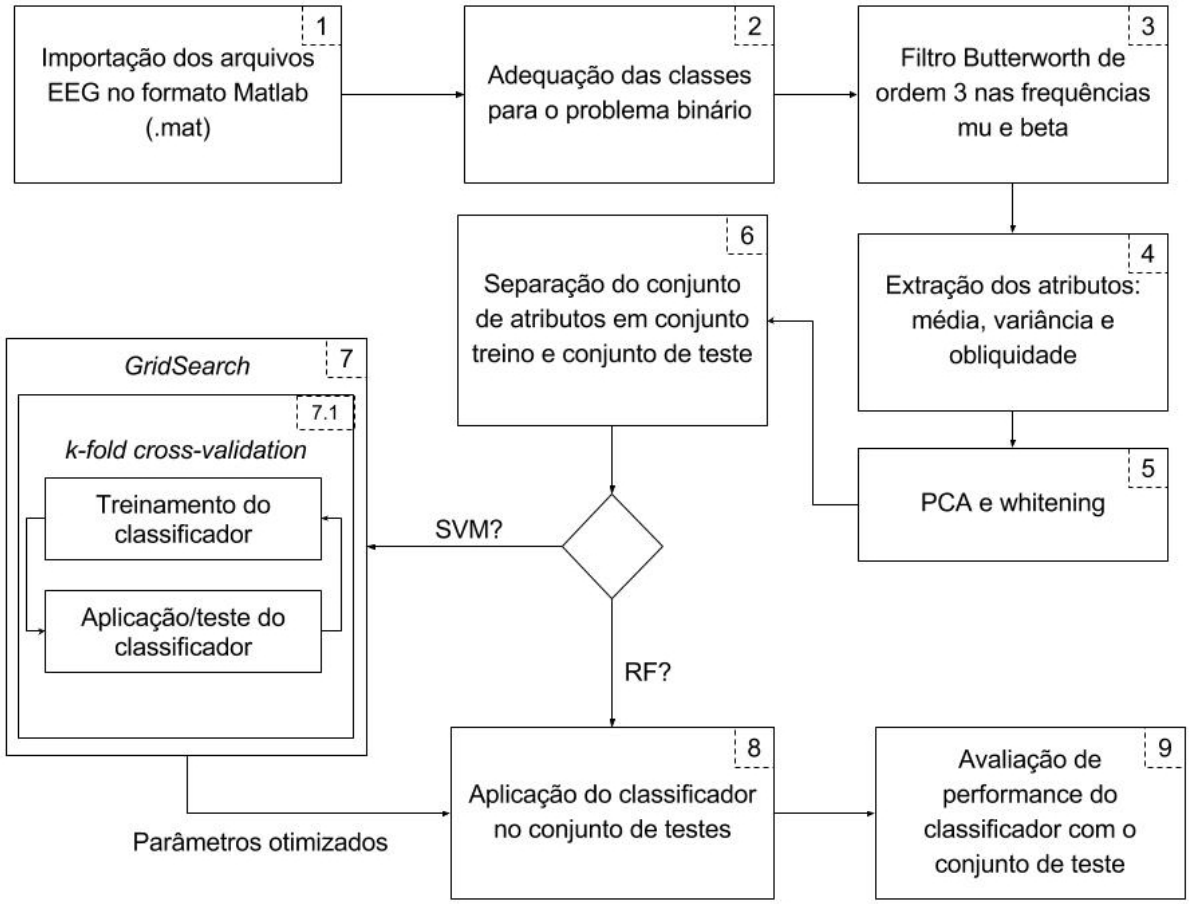
O texto está organizado da seguinte forma: O Capítulo 2 apresenta os materiais e métodos necessários para o desenvolvimento deste trabalho. Na Seção 2.1 é discutido o conceito de imagética motora e o que são os ritmos sensório-motores. Na Seção 2.2 são discutidos conceitos básicos sobre o cérebro, o que são os sinais cerebrais e como são realizadas as medidas do eletroencefalograma (EEG). Na Seção 2.3 é apresentado como foi realizado o pré-processamento do sinal e a forma que foi realizada a filtragem e a extração dos atributos. A seção 2.4 explica o que é o processo de branqueamento (*whitening*). A Seção 2.5 explica como é gerado o modelo de classificador e explora os dois classificadores utilizados neste trabalho. Na seção 2.6 é explicado o conceito de simulação em tempo real. Na Seção 2.7 são explanados os vários tipos de métricas para avaliar a performance de uma metodologia de processamento, assim como as que serão usadas neste trabalho.

O Capítulo 3 reporta e discute os resultados deste estudo. Na Seção 3.1 é descrito como o banco de sinais utilizado esta estruturado e como foi efetuada a aquisição destes dados pelo Laboratório Cichocki de Processamento de Sinais Cerebrais Avançado do Instituto de Ciências do Cérebro no Japão, após esta descrição é apresentado como os dados foram utilizados neste trabalho. Na Seção 3.2 os passos referentes ao treino do algoritmo, compostos pelo procedimento de pré-processamento dos dados, extração dos atributos e por fim o treino do PCA e classificadores são descritos em detalhe. A Seção 3.3 apresenta como é efetuada a simulação em tempo real na placa Orange Pi Zero, mostrando como os dois algoritmos de classificação foram utilizados e seus respectivos resultados para as classes desbalanceadas e balanceadas. A seção 3.4 apresenta uma análise dos resultados obtidos. Por fim, as conclusões e perspectivas futuras desse trabalho são expostas, respectivamente, nos Capítulos 4 e 5.

2 MATERIAIS E MÉTODOS

Para entender a metodologia proposta, inicia-se analisando o fluxograma representado na Figura 2.1, o qual demonstra as etapas do trabalho apresentado por (SILVA, 2017) para classificação de um sinal de imagética motora.

Figura 2.1 – Fluxograma da metodologia proposta por (SILVA, 2017).



Fonte: (SILVA, 2017)

A Figura 2.1 representa resumidamente os principais passos (pré-processamento, extração de atributos e classificação) referentes a classificação de um sinal EEG de imagética motora.

2.1 IMAGÉTICA MOTORA (IM) E OS RITMOS SENSÓRIO-MOTORES

A imagética motora tem sido considerada como um estado cognitivo que corresponde à simulação mental de uma ação motora (STECKLOW, 2006), sem a necessidade

da execução física da tarefa.

As contribuições relativas de cada modalidade sensorial nos processos de simulação mental podem variar. Por exemplo, quando solicitado a simular mentalmente um movimento, o voluntário pode se "sentir" ou se "ver" realizando o movimento (RODRIGUES et al., 2003). No primeiro caso, a simulação ocorrerá a partir de informações somato-motoras (estratégia de imaginação interna ou em perspectiva de primeira pessoa). No segundo, será baseada na percepção visual do movimento imaginado (estratégia de imaginação externa ou em perspectiva de terceira pessoa) (DECETY, 1996). Conforme a ocorrência da simulação mental de um movimento, ativará determinadas áreas do cérebro.

Uma vez monitorada a atividade cerebral, uma BCI baseada na estratégia de imagética motora deve extrair os ritmos sensório-motores, através da filtragem dos sinais cerebrais nas faixas de frequência μ e β (VAZ, 2016).

A onda μ é uma variante da onda alfa (9 a 13 Hz), podendo ser encontrada sobre o córtex motor, responsável pelos grupos musculares e funções motoras específicas do corpo, como o movimento ou a intenção de se mover. São reduzidas com os olhos abertos, sonolência e sono (Simone El Hage, 2017).

E a onda beta (oscila em torno de 20 Hz, podendo chegar até 30 Hz) tem um aumento de presença decorrente de uma atividade cortical relacionada com alta concentração, processos mentais complexos, atividade motora e qualquer tipo de tarefa cognitiva que exija uma maior concentração (NIEDERMEYER; SILVA, 2005).

2.2 ELETROENCEFALOGRAMA (EEG) E SINAIS CEREBRAIS

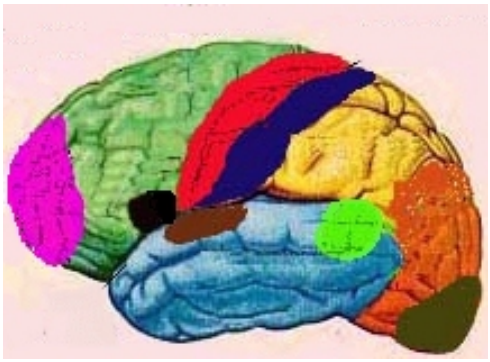
A utilização do EEG tem se mostrado uma importante ferramenta de análise do funcionamento cerebral durante IM (STECKLOW, 2006).

O cérebro gera uma grande quantidade de atividade neural. A captura dessa atividade gera uma infinidade de sinais elétricos, que podem ser usado para sistemas BCI (SILVEIRA, 2013).

As ondas cerebrais estão em atividade, simultaneamente, em todo córtex cerebral, mas, dependendo do momento e da atividade em questão ocorre prevalência sobre alguma área, pois determinadas áreas cerebrais estão mais diretamente ligadas a certas funções.

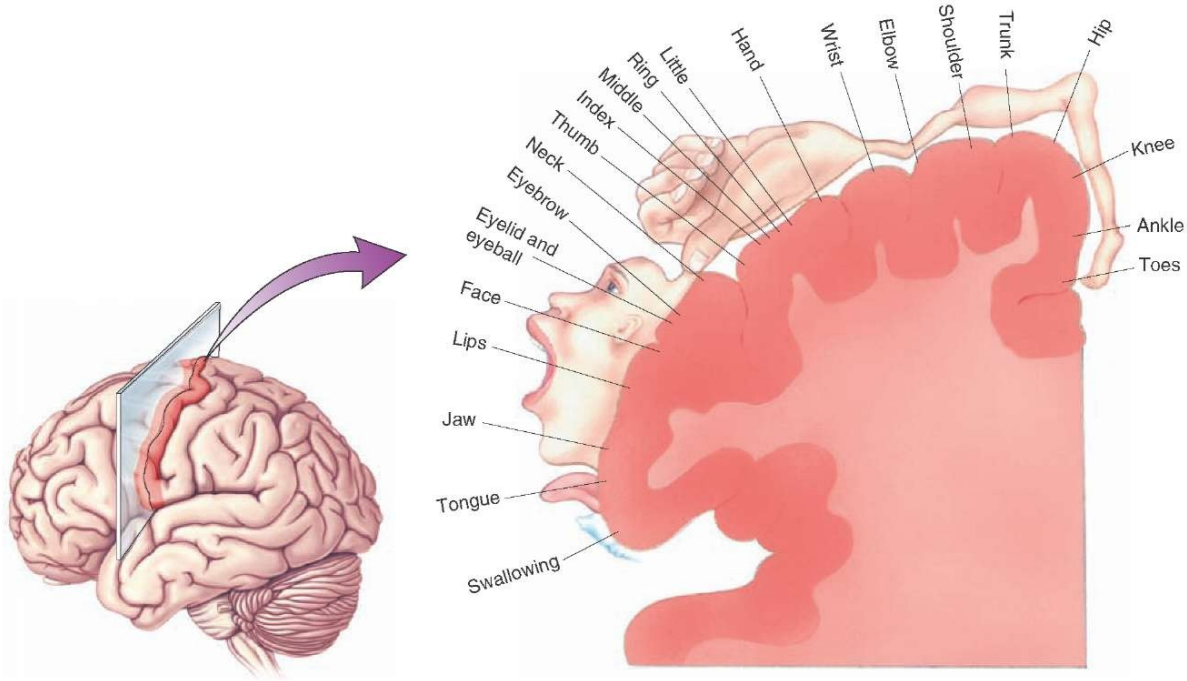
Na Figura 2.2 em vermelho, está representado córtex motor primário, o qual tem a função de iniciação do comportamento motor, controlando diversos movimentos do corpo. Ele está localizado entre o córtex somatossensorial e o córtex pré-motor, podendo ser verificado na Figura 2.3 que há uma assimetria, onde existem regiões muito grandes representando pequenas porções do corpo, como os dedos das mãos onde o ser humano tem muita sensibilidade e regiões menores controlam partes com menor sensibilidade, como pernas e braços (TORTORA, 2000).

Figura 2.2 – Áreas corticais do cérebro



Fonte: (Silvia Helena Cardoso, 2017)

Figura 2.3 – Representação do córtex motor.



Fonte: (http://what-when-how.com, 2017)

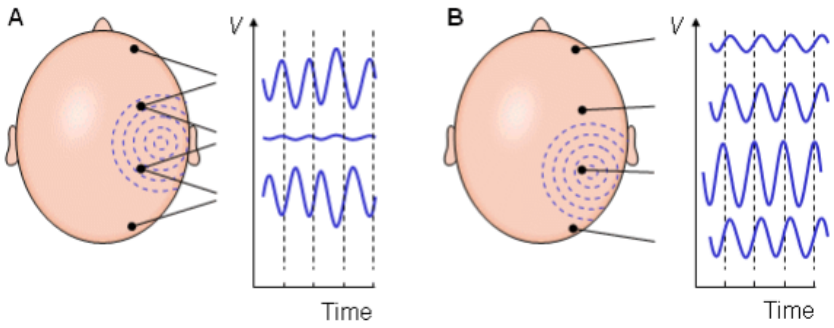
Desta forma os sinais EEG capturados do escalpo (couro cabeludo), captam os potenciais elétricos denominados ondas cerebrais e indicam a atividade do córtex cerebral.

Há duas formas de configuração dos eletrodos para a realização de medidas EEG: bipolares ou unipolares, representados pela Figura 2.4.

No método bipolar a derivação é entre a diferença de potencial entre os dois eletrodos, já no método unipolar o potencial de cada eletrodo é comparado à referência (ponto CZ na Figura 2.5) (MALMIVUO; PLONSEY, 1995).

Muitos sistemas BCI baseados em EEG utilizam uma montagem tipicamente se-

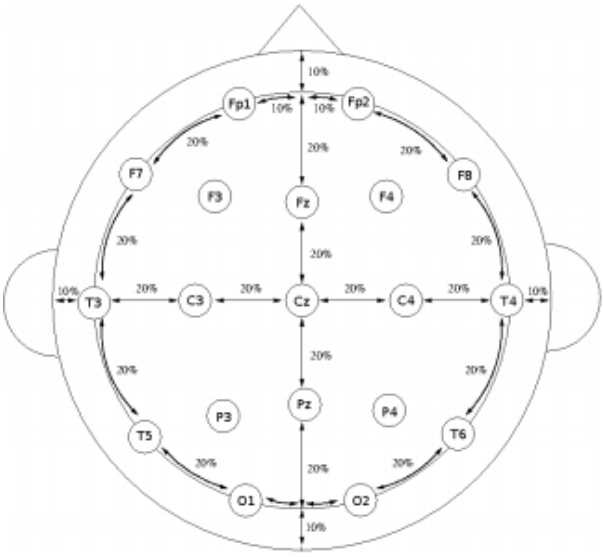
Figura 2.4 – (A) Bipolar (B) Unipolar medidas. A forma de onda do EEG depende do local em que é medido.



Fonte: Adaptado de (MALMIVUO; PLONSEY, 1995)

gundo padrões tais como 10-20, 10-10 e 10-5 (JURCAK; TSUZUKI; DAN, 2007). A Figura 2.5 demonstra o posicionamento de eletrodos segundo o padrão 10-20 para o monitoramento eletroencefalográfico.

Figura 2.5 – Montagem de eletrodos segundo padrão 10-20 para o monitoramento eletroencefalográfico. Cada círculo rotulado corresponde a um eletrodo. Os rótulos indicam: a letra F corresponde à região frontal do escalpo humano; a letra C está associada à região cortical; a letra T indica a região temporal; a letra P corresponde à região parietal; e, finalmente, a letra O indica a região occipital



Fonte: (VAZ, 2016)

O EEG tem sido subdividido em diferentes bandas de frequência, onde seus limites podem variar para cada autor. Tais bandas refletiriam um funcionamento específico do córtex de acordo com estados de comportamento, como os níveis de atenção, estados de sono ou vigília e doenças como epilepsia ou profundidade de coma (BEAR; CONNORS; PARADISO, 2002). Conforme já mencionado anteriormente, as faixas de frequência refe-

rentes a IM são a μ e β.

2.3 PRÉ-PROCESSAMENTO E EXTRAÇÃO DE ATRIBUTOS

Os sinais de EEG foram preparados para seu posterior processamento, sendo organizados em uma matriz da forma [amostras, canais]. Os canais utilizados foram C3, Cp3, C4, Cp4 e Cz, relacionados ao córtex sensório-motor, de onde originam-se os sinais de imagética motora.

Um filtro digital é um sistema temporal discreto que tem a função de remover partes não desejadas do sinal, como ruído, ou extrair partes úteis do sinal, como determinadas componentes de frequência que estão dentro da gama de frequência desejada. Quanta à resposta em frequência, os filtros são classificados em quatro tipos: passa-baixas, passa-altas, passa-faixa e rejeita-faixa. Neste trabalho o sinal EEG original foi filtrado usando um filtro passa-faixa com as frequências dentro dos ritmos sensório-motor (banda μ (9 a 12Hz) e banda β(16 a 30 Hz)). Para isso foi utilizado o filtro *Butterworth* de ordem 3, da biblioteca *Scipy* do Python com os parâmetros fixados, exceto a ordem do filtro. A ordem do filtro é que define a capacidade de atenuar sinais fora da banda de passagem.

Os dados foram organizados com classes de forma desbalanceada e balanceada para verificar a performance dos classificadores, sendo detalhada esta organização de dados na Seção 3.1.1.

A extração de atributos tem a finalidade de reduzir a dimensão do vetor de dados (sem a perda de informações relevantes). Sendo utilizados neste trabalho os seguintes atributos: média ($\bar{X}$), variância (V) e obliquidade (S). A média é a soma dos resultados dividida pelo total de resultados, ou seja, expressa o valor esperado do sinal. A variância é uma das medidas de dispersão que indicam a regularidade de um conjunto de dados em função da média aritmética. E a obliquidade é a medida da assimetria de uma distribuição em relação à normalidade.

Seja n o número total de valores e x_i cada valor do conjunto de dados (X), em que $i = 1, 2, 3, ..., n$.

$$\bar{X}(X) = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.1}$$

$$V(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{X})^2 \tag{2.2}$$

$$S(X) = \frac{1}{n}\frac{\sum_{i=1}^{n}(x_i - \bar{X})^2}{V(X)^{\frac{3}{2}}} \tag{2.3}$$

2.4 PCA E BRANQUEAMENTO

A análise de Componentes Principais (*Principal Component Analysis* - PCA) é uma técnica matemática que descreve um conjunto de dados usando "componentes principais", escrita como combinações lineares dos dados originais em um novo conjunto de dimensões menores (SILVA, 2006).

Se o PCA for configurado com um número de componentes igual à dimensão do vetor de dados, é realizado o branqueamento (*whitening*), onde o conjunto de dados é transformado em um conjunto de dados não-correlacionados, não sendo realizada a compressão destes dados.

Com a aplicação do branqueamento, espera-se ter tratado possíveis ruídos que ainda estivessem presentes no dados, resultando em um sinal que contenha apenas as informações relevantes para a classificação de movimentos.

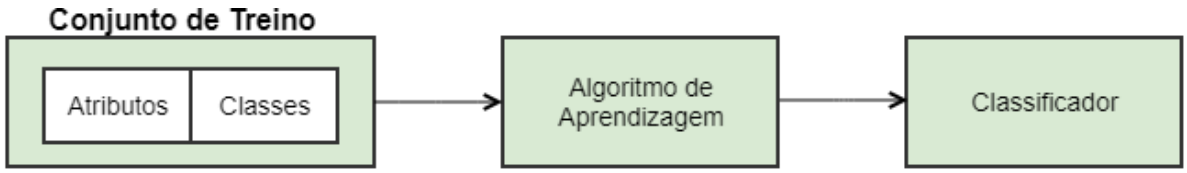
Para mais informações, o procedimento do PCA apresentado por (SILVA, 2017) pode ser consultado no Anexo A.

2.5 CLASSIFICAÇÃO

A informação obtida na extração de atributos e tratada com o método de branqueamento serve como entrada para o algoritmo classificador encarregado de modelar os dados e identificar os padrões.

Geralmente, o conjunto de dados é dividido em dois subconjuntos distintos: conjunto de treinamento e conjunto de testes (SHALEV-SHWARTZ; BEN-DAVID, 2014). Esse procedimento é necessário para assegurar que as medidas obtidas utilizando o conjunto de testes sejam de um conjunto diferente do usado para realizar o aprendizado.

Figura 2.6 – Aprendizado Supervisionado



Fonte: AUTOR

A Figura 2.6 ilustra as etapas da aprendizagem supervisionada. Onde é dado um conjunto de exemplos rotulados (x_i, y_i), onde x_i representa um atributo e y_i a sua classe ou rótulo, devendo produzir um classificador, também denominado modelo ou preditor, capaz de predizer precisamente o rótulo de novos dados. Esse modelo gerado após o treinamento com um conjunto de dados cuja classificação é conhecida é utilizado para predizer

a classe de um conjunto de dados (conjunto de testes) que ainda não foi classificado, de forma que o classificador pode ser visto como uma função f, a qual recebe um dado x e fornece uma predição y, podendo assim ser verificado o grau de efetividade da aprendizagem.

Para realizar a classificação dos atributos foram utilizados dois algoritmos: Máquina de Vetores de Suporte (SVM) e Florestas Aleatórias (RF). Devido a utilização de conjunto de dados desbalanceado, foi necessário considerar se os classificadores realmente seriam eficientes nesse tipo de problema. (SILVA, 2017).

2.5.1 Máquina de Vetores de Suporte

As Máquinas de Vetores de Suporte (*Support Vector Machine* - SVM) têm uma abordagem desenvolvida baseada na teoria do aprendizado estatístico (VAPNIK, 1995). Sendo assim, possuem uma base teórica bem estabelecida dentro da Matemática e Estatística.

Consiste em um método de aprendizado que tenta encontrar a maior margem para separar diferentes classes de dados, através de um processo de otimização contínua em que busca-se encontrar o hiperplano (uma reta no caso 2D e um plano no caso 3D) que separe as duas classes da melhor forma possível.

Quando há mais de duas dimensões, é necessária a utilização da função *kernel*. O *kernel* é uma transformação que leva cada ponto do espaço de entrada (espaço de atributos) para um espaço com mais dimensões, de modo que os pontos possam ficar o mais separados possível e que o hiperplano de separação das amostras maximize a separação entre as classes (SILVA, 2017).

A linha do hiperplano busca maximizar a distância entre os pontos mais próximos em relação a cada uma das classes. Sendo $f(x)$ = $(wx) + b$ um hiperplano, podemos definir a margem como a menor distância entre os exemplos do conjunto de treinamento e o hiperplano utilizado para separação destas classes (LORENA; CARVALHO, 2003). A margem determina quão bem duas classes podem ser separadas (SMOLA, 1999).

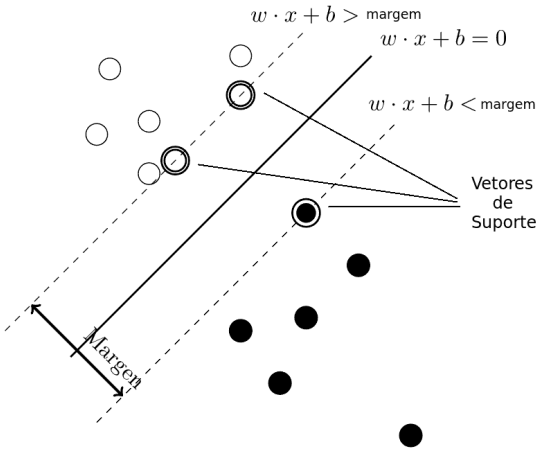
É através dos vetores de suporte (exemplos de treinamento que estiverem mais próximos da função de separação) que será definido o hiperplano. A Figura 2.7 representa o hiperplano de separação entre duas classes, demonstrando as margens.

Neste trabalho o *kernel* utilizado foi Função de Base Radial (*Radial Basis Function* - RBF) ou Gaussiano, dado pela equação 2.4, onde $(x_i - x_j^2)$ é a distância Euclidiana entre dois vetores de atributos e σ um parâmetro livre.

$$\phi(x_i, x_j) = exp(-\frac{x_i - x_j^2}{2\sigma^2}) = exp(-\gamma(x_i - x_j^2)) \tag{2.4}$$

A função *kernel* da SVM é difícil de ser escolhida para aplicações práticas devido

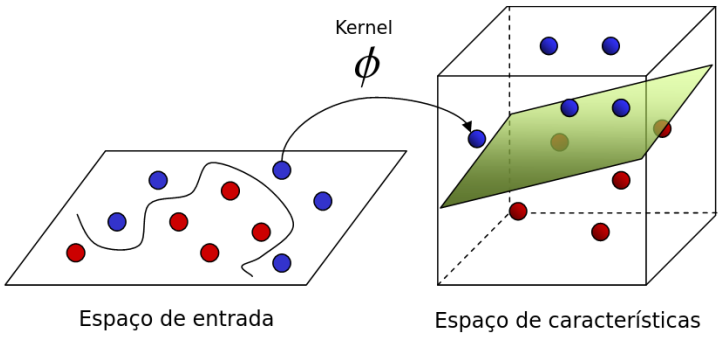
Figura 2.7 – Hiperplano (reta) de separação entre duas classes em um plano bidimensional.



Fonte: (SILVA, 2017)

à aleatoriedade e não-estacionariedade da natureza dos sinais EEG e à falta de conhecimento prévio a respeito da distribuição característica do sinal cerebral (SILVA, 2017).

Figura 2.8 – Uso do kernel como uma transformada de um espaço menor para um espaço maior no algoritmo SVM.



Fonte: (SILVA, 2017)

A Figura 2.8 mostra como seria impossível traçar uma reta que separasse eficientemente os valores do espaço de entrada.

2.5.1.1 Otimização dos parâmetros

É crucial a seleção do conjunto ótimo de parâmetros para a utilização efetiva da SVM. Os parâmetros de constante de regularização C e o parâmetro do *kernel* σ são selecionados através de um *GridSearch*. O *GridSearch* foi implementado em Python, sendo uma combinação, a partir de um conjunto de valores pré-estabelecidos, de todos os valores

com todos os valores a fim de maximizar uma medida. A medida foi a acurácia, a mais amplamente utilizada para apresentar a performance do algoritmo (KEVRIC; SUBASI, 2017). Porém, essa não é a melhor medida para o caso de dados desbalanceados, apesar de obter-se um bom resultado, pois quando as classes são desbalanceadas, a acurácia geralmente confere maior peso à classe com maior ocorrência (FATOURECHI et al., 2008). No entanto, seria possível utilizar outras medidas a serem maximizadas no algoritmo de *GridSearch*, a saber a área sob a curva ROC (Receiver Operating Characteristic - ROC) insensível a dados desbalanceados (SILVA, 2017). Porém, não foi possível a utilização destas métricas devido a conflitos internos da implementação dos algoritmos do SVM em Python e do cálculo dessa área, pois o algoritmo SVM exige que os dados tenham marcadores nominais e o cálculo da área exige que os dados tenham marcadores binarizados, de forma que na implementação atual disponível no *scikit-learn* não é possível realizar essa combinação (SILVA, 2017).

Ao avaliar diferentes configurações de hiperparâmetros, há o risco de sobreajuste (*overfitting*) no conjunto de teste (superajuste dos parâmetros aos dados de teste, não criando um padrão generalizado para o classificador), mostrando-se ineficaz para prever novos resultados, ou seja, *overfitting* ocorre quando o método de aprendizado não consegue generalizar os resultados para dados que não foram utilizados no processo de treino. Para resolver o problema o conjunto de dados pode ser dividido em 3 conjuntos (conjunto de treinamento, conjunto de validação e conjunto de teste), utilizando o conjunto de treinamento para fazer a aprendizagem do algoritmo e posteriormente se utiliza o conjunto de validação para verificar a generalização do algoritmo (ajustar os parâmetros), e somente quando o experimento resultar em uma boa performance, uma avalização final é efetuada com o conjunto de testes. Porém, conforme exposto por (SILVA, 2017) ao realizar a partição dos dados desta forma, o número de amostras que pode ser usado para o modelo aprender é drasticamente reduzido e os resultados podem depender de uma escolha aleatória específica dos conjuntos de pares (treino, validação).

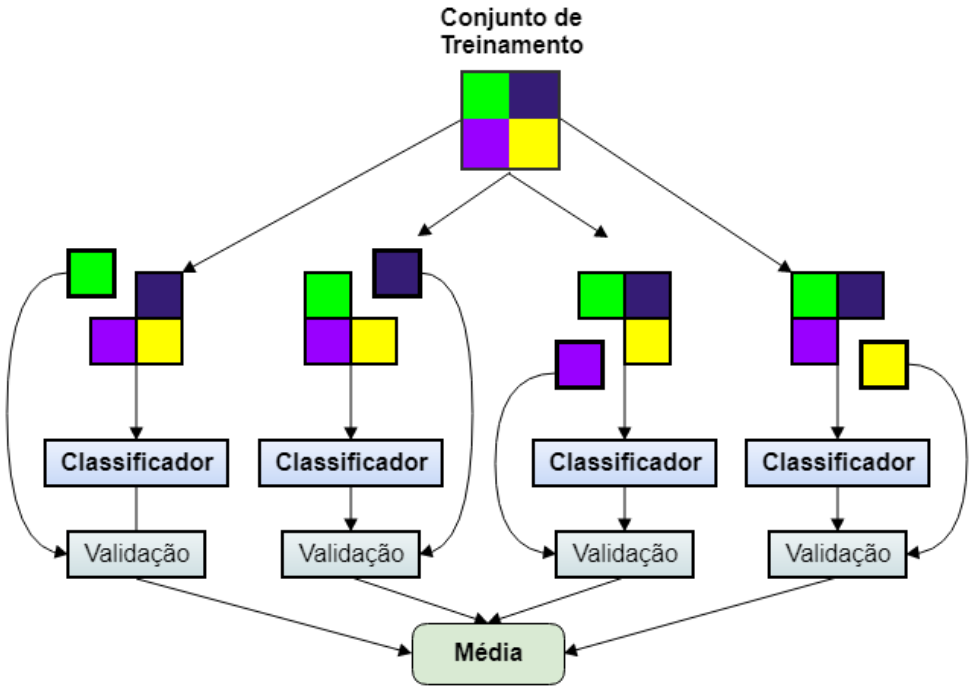
Uma solução para este problema é o procedimento chamado validação cruzada (*cross-validation* - CV) (PEDREGOSA et al., 2011). Não sendo mais necessário o conjunto de validação, apenas o conjunto de treinamento e teste.

A Figura 2.9 representa o método *k-fold cross-validation* para 4 subconjuntos. No método *k-fold cross-validation*, o conjunto de treino de tamanho n (n exemplos) é dividido em k conjuntos disjuntos de tamanho $\frac{n}{k}$. O procedimento descrito a seguir é repetido k vezes, de modo que cada um dos k subconjuntos sejam utilizados exatamente uma vez como dado de teste para validação do modelo:

- Um modelo é treinado usando $k-1$ *folds* do conjunto de treino;
- O modelo resultante é validado com o seu respectivo subconjunto k.

No final deste processo tem-se o desempenho médio do classificador nos k testes.

Figura 2.9 – Método *k-fold cross-validation* para 4 subconjuntos



Fonte: AUTOR

O objetivo de repetir os testes múltiplas vezes é aumentar a confiabilidade da estimativa da precisão do classificador.

Conforme exposto por (SILVA, 2017), essa abordagem pode ser computacionalmente custosa, mas não “desperdiça” tantos dados, sendo essa uma grande vantagem para problemas nos quais o número de amostras é pequeno.

2.5.2 Florestas Aleatórias

Florestas Aleatórias (*Random Forest* - RF) são um tipo de *ensemble learning*, método que gera muitos classificadores e combina os seus resultados. Os classificadores baseados em árvores de decisão foram propostos em 1995 por Tin Kam Ho (HO, 1995).

As florestas aleatórias são computacionalmente muito efetivas, além de evitarem sobreajuste (*overfitting*) e serem pouco sensíveis a ruídos (BREIMAN, 2001).

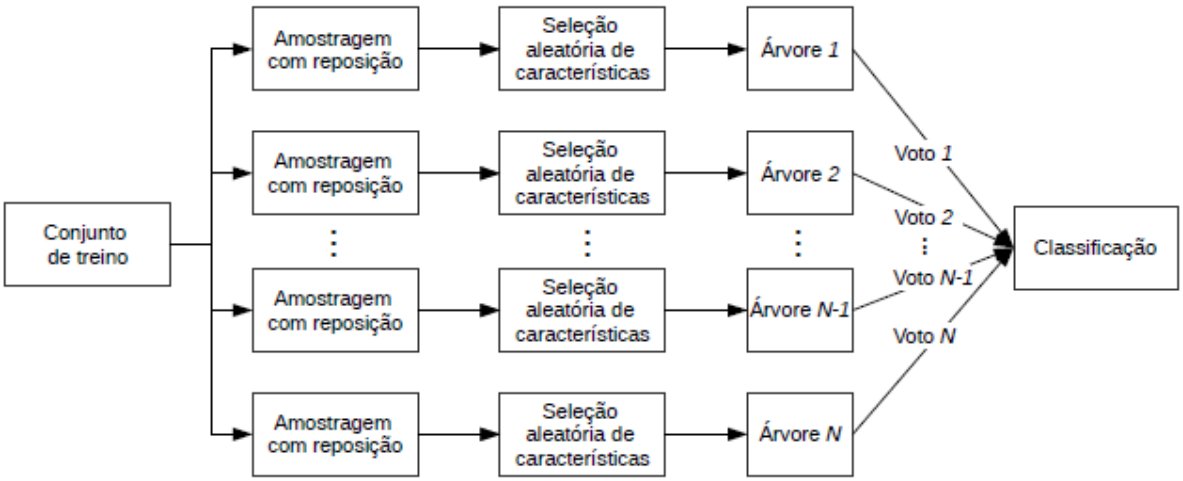
O objetivo é combinar as classificações individuais de N árvores de decisão, cada uma construída com F atributos, em uma só rotulagem. O número de atributos F obedece $F = [\log_2 P + 1]$, sendo P a quantidade total de atributos do conjunto de dados (SILVEIRA, 2016).

Segundo (SILVEIRA, 2016) a construção de uma floresta aleatória se dá da seguinte forma: A técnica de amostragem com reposição é aplicada sobre todo o conjunto

de treino, gerando N conjuntos de mesmo tamanho, chamados de *in-bag*. Cada uma das N árvores de decisão considera seu conjunto *in-bag* como conjunto de treino, selecionando aleatoriamente F atributos para seu crescimento.

Floresta Aleatória utiliza árvores aleatórias como algoritmo para os N preditores. Cada árvore cresce ao máximo e nenhuma técnica de poda é aplicada. A classificação de uma floresta aleatória é dada pela moda dos votos das N árvores de decisão que a compõe (SILVEIRA, 2016). Na Figura 2.10 esta representado o fluxograma do procedimento de construção de uma floresta aleatória.

Figura 2.10 – Arquitetura de um classificador do tipo floresta aleatória



Fonte: (SILVEIRA, 2016)

O número das árvores de classificação a serem construídas e o número de dimensões aleatoriamente escolhidas por nodo (número de atributos), impactam fortemente na acurácia e esforço computacional de um classificador RF (SILVA, 2017).

2.6 SIMULAÇÃO EM TEMPO REAL

Um sistema de tempo real é um sistema de processamento de informações que responde a eventos/estímulos externos dentro de um período específico e finito. Algumas aplicações de tempo real apresentam restrições de tempo mais rigorosas do que outras, por exemplo, sistemas responsáveis pelo monitoramento de pacientes em hospitais, sistemas embarcados em robôs e veículos e sistemas de supervisão e controle em plantas industriais (FARINES; FRAGA; OLIVEIRA, 2000).

Um sistema é dito de tempo real se o sinal de controle é gerado suficientemente rápido para suprir as necessidades do processo. Pois computação de tempo real não quer

dizer execução rápida, mas sim o cumprimento de prazos (AROCA, 2008).

Na simulação em tempo real, tem-se a modelagem de um processo/sistema, de forma que o modelo imite as respostas do sistema real numa sucessão de eventos que ocorrem ao longo de um período, ou seja, os eventos ocorrem e são tratados na mesma escala de tempo correspondente ao sistema real.

No intuito de integrar a plataforma portátil contendo a metodologia para classificação de imagética motora, em um sistema BCI que controle uma cadeira de rodas, por exemplo, o sinal de controle gerado entre 1 s a 2 s após a imaginação da atividade motora supre a necessidade do processo.

2.7 AVALIAÇÃO DA PERFORMANCE DA CLASSIFICAÇÃO

A estratégia de avaliação e métricas usadas para análise devem ser escolhidas cuidadosamente para representar de forma precisa a performance da BCI (SILVA, 2017).

A taxa de acerto e o erro são as medidas de avaliação mais comuns para modelos de classificação (WHITTEN; FRANK; HALL, 2011). Essas medidas podem ser calculadas a partir da Matriz de Confusão que mede o desempenho do modelo através dos acertos e erros. No problema de decisão binária (Tabela 2.1), denomina-se uma classe como positiva e a outra como negativa, desta forma, a matriz de confusão indica quatro possibilidades de acertos e de erros do classificador.

- Verdadeiros positivos (TP): quando a instância “Positivo” for predita pelo classificador como “Positivo”.
- Falsos negativos (FN): quando a instância “Positivo” for predita pelo classificador como “Negativo”.
- Verdadeiros negativos (TN): quando a instância “Negativo” for predita pelo classificador como “Negativo”.
- Falsos positivos (FP): quando a instância “Negativo” for predita pelo classificador como “Positivo”.

Tabela 2.1 – Matriz de confusão para problemas com duas classes

	Classificação	
Classe verdadeira	Positivo	Negativo
Positivo	TP	FN
Negativo	FP	TN

Fonte: (SILVA, 2017)

Com base na matriz de confusão, diversas métricas podem ser obtidas para avaliar o desempenho do modelo: sensitividade, especificidade, confiabilidade positiva, e confiabilidade negativa.

A sensitividade (*Recall* ou *True Positive Rate* - TPR) ou Taxa de Verdadeiro Positivo (TVP) é a proporção de exemplos positivos que foram classificados corretamente, já a especificidade (*True Negative Rate* - TNR) ou Taxa de Verdadeiro Negativo (TVN) é a proporção de exemplos negativos que foram classificados corretamente.

$$Sensitividade = \frac{TP}{TP + FN} \tag{2.5}$$

$$Especificidade = \frac{TN}{TN + FP} \tag{2.6}$$

O gráfico da proporção de verdadeiros positivos *versus* a proporção de falsos positivos quando um parâmetro qualquer de um classificador é variado é conhecido como a curva ROC (Curva de Característica de Operação do Receptor) (RAO, 2013).

Para comparar a performance de dois ou mais classificadores através da curva ROC, é interessante sumarizar esta performance com um valor escalar único, ou seja, calculando a área sob a curva (*Area Under the Curve* - AUC) (SILVA, 2017). A AUC é uma porção da área de um quadrado unitário e seu valor estará sempre entre 0 e 1.0 (SILVA, 2017). Sendo o gráfico cortado por uma linha diagonal, representando a probabilidade de uma classificação aleatória, desta forma, modelos que são classificados acima da linha são considerados bons, e abaixo da linha são descartados.

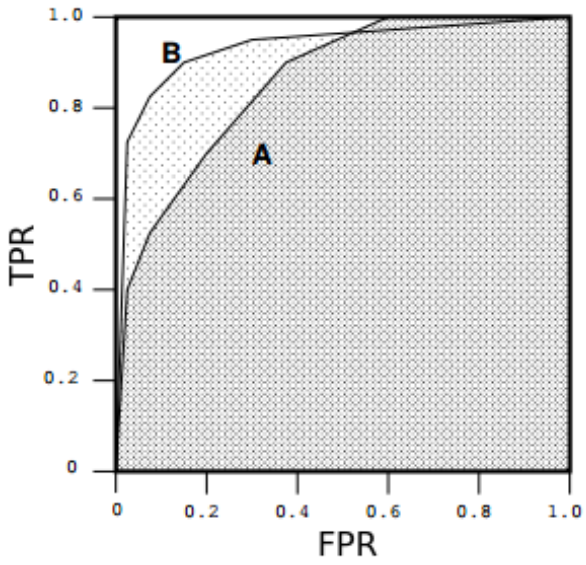
A Figura 2.11 mostra a área sob duas curvas ROC, A e B. Onde é possível verificar que o classificador B tem uma performance melhor, devido a sua área ser maior que a área do classificador A. O objetivo do espaço ROC é a curva estar o mais próximo possível do canto superior esquerdo, onde a Taxa de Verdadeiros Positivos (TPR) é máxima e Taxa de Falsos Positivos (FPR) é mínima.

Conforme apresentado por (SILVA, 2017), quando utiliza-se conjuntos de dados altamente desbalanceados, as curvas Precisão-Revocação (*Precision-Recall* - PR) produzem um panorama mais informativo da performance do algoritmo, e apesar da curva ROC não ser afetada pelo desbalanceamento das classes, ela pode acabar mascarando performances ruins. (SILVA, 2017) recomenda usar mais de uma métrica para avaliar a classificação ou normalizar as classes usando reamostragem aleatória.

Outra métrica utilizada é a acurácia de classificação (Acc), definida como o grau de proximidade de uma estimativa com seu parâmetro (ou valor verdadeiro). Conforme demonstrado por (SILVA, 2017), pode ser derivada da matriz de confusão M, como segue:

$$Acc = \frac{\sum_{i=1}^{Q} M_{ii}}{\sum_{i=1}^{Q} \sum_{j=1}^{Q} M_{ij}}, \tag{2.7}$$

Figura 2.11 – Duas curvas ROC. Pode-se notar que a curva ROC B está mais próxima do canto superior esquerdo do que a curva ROC B, indicando que ela é melhor



Fonte: (SILVA, 2017)

sendo M_{ij} o elemento da i-ésima linha e j-ésima coluna da matriz de confusão M e Q a ordem desta matriz e também o número de classes do problema.

O coeficiente *kappa* mede a acurácia de um método, subtraindo a fração relacionada à probabilidade de acerto ao acaso que, hipoteticamente, qualquer classificador poderia ter (SILVEIRA, 2016). O coeficiente *kappa* é calculado através de:

$$\kappa = \frac{Acc(M) - EA(M)}{1 - EA(M)} \tag{2.8}$$

sendo

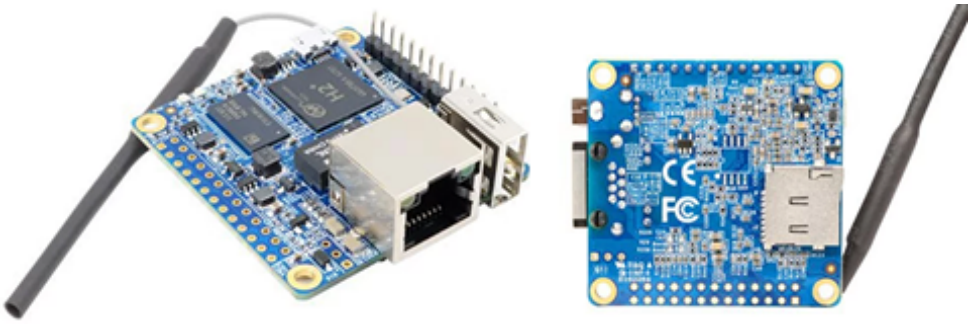
$$EA(M) = \frac{\sum_{i=1}(\sum_{j=1}^{Q} M_{ij} \sum_{j=1}^{Q} M_{ji})}{(\sum_{i=1}^{Q} \sum_{j=1}^{Q} M_{ij})^2} \tag{2.9}$$

2.8 ORANGE PI ZERO

Orange Pi Zero (Figura 2.12) é uma placa que possibilita embarcar Sistemas Operacionais (SO) como Debian, Ubuntu e Android, desenvolvida com vistas na aplicação em projetos de desenvolvimento de software, robótica e automação.

Esta placa é considerada um computador em placa com baixo custo e com boa performance em relação a demais placas deste tipo, porém sua performance para uso como um computador convencional pode não ser tão boa.

Figura 2.12 – Orange Pi Zero - Parte superior e inferior da placa



Fonte: AUTOR

Possui um processador quad-core, conta com 256 MB de memória RAM, porta USB 2.0, porta Ethernet, Wi-Fi integrado e entrada para cartão de memória. Pesando apenas 26 g.

Para a utilização da placa foi utilizado um cartão SD de 16 GB classe 10, no qual foi gravada a imagem da distribuição Debian Jessie, feita pela Armbian (Armbian é uma espécie de “fabricante” de distribuições Linux extremamente leves para CPUs ARM).

O código da metodologia de detecção de movimento baseado em sinais de EEG, foi programado na linguagem Python. Portanto para poder rodar os códigos na placa, foi instalado o miniconda, que além do interpretador Python, instala também o conda, que ajuda a gerenciar a instalação e manutenção dos pacotes do Python.

2.8.1 Método de Comunicação Serial

A Orange Pi Zero possibilita comunicação serial com outros dispositivos, através do método de comunicação serial síncrona *Serial Peripheral Interface* (SPI), o qual possui um fluxo de dados do tipo *Full Duplex*, ou seja, a troca de dados acontece sempre em ambas as direções.

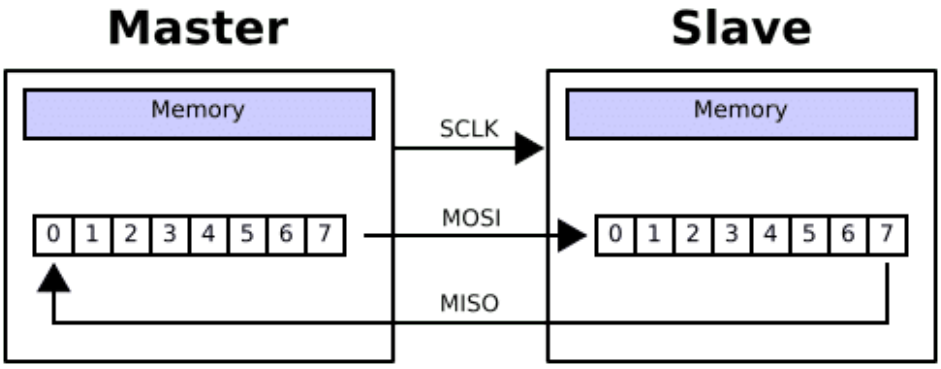
Na comunicação serial síncrona há o conceito de Mestre-Escravo, onde normalmente o gerador do sinal de sincronismo é definido como o Mestre (*Master*) da comunicação e os dispositivos que utilizam do sinal de sincronismo gerado, denominam-se Escravo (*Slave*).

Na comunicação entre dispositivos SPI, a ligação do *Master* para o *Slave* recebe o nome padrão de MOSI (*Master Output Slave Input*), enquanto do *Slave* para o *Master* o nome padrão é MISO (*Master Input Slave Output*). Sendo que o pino de *clock* recebe o nome padrão SCLK (*Serial Clock*) e para a seleção de *Slave* o nome padrão é SS (*Slave Select*).

O fundamento do SPI é um circuito *shift-register*, ou seja, cada bit de dado envi-

ado do *Master* para o *Slave*, transfere também um bit de dado do *Slave* para o *Master*. Conforme a representação da Figura 2.13.

Figura 2.13 – Transmissão de um byte



Fonte: (Francesco Sacco, 2014)

Através da comunicação SPI é possível realizar a comunicação da placa Orange Pi Zero com um dispositivo que comporte o circuito ADS1299-4, o qual tem a função de realizar a aquisição de sinais de EEG. Quando for realizada esta comunicação entre os dois dispositivos, não será mais necessário o uso de um banco de sinais de EEG armazenado no cartão SD. Sendo que o fluxograma representado na Figura 3.11 seguirá a partir do passo 5, realizando a leitura do sinal EEG referente a uma tentativa, no passo 6, conforme explicado na seção 3.3.

2.8.2 Linguagem Python

Python é a linguagem ideal para aplicações científicas, pois é uma linguagem expressiva, em que é fácil traduzir o raciocínio em um algoritmo. Sendo que em aplicações científicas, o raciocínio é essencialmente complicado e o cientista deve se concentrar exclusivamente no problema que está estudando ao invés de perder tempo com alocação de memória, gerenciamento de recursos, etc.

A versão do Python escolhida para utilização foi a 2.7, pois tem um grande apoio da comunidade online, sendo que a versão 2 foi lançada no final de 2000 e tem sido usada até hoje. Além de ter uma infinidade de bibliotecas de terceiros. Havendo um grande número de módulos que funcionam apenas na versões 2.x, mesmo já havendo a versão 3.x.

A seguir uma lista das bibliotecas utilizadas para os cálculos e análise de dados:

- **Numpy** para representação eficiente de *arrays* e matrizes. Esta biblioteca também contém funções básicas de álgebra linear, transformações de *Fourier* e capacidades avançadas de números aleatórios (KUNAL JAIN, 2016).

- **SciPy** é construída sobre NumPy e usada para realizar computação sobre matrizes, *arrays*, além de cálculos numéricos, resolução sistemas de equações lineares, etc (Felipe Martins dos Santos, 2017).

- **Matplotlib** é usada para verificar os dados em gráficos, desde gráficos de histogramas até gráficos de calor.

- **Pandas** para operação e manipulação de dados estruturados. É amplamente utilizado para preparação de dados (KUNAL JAIN, 2016).

- **Scikit Learn** para a aprendizagem de máquina. Construído sobre NumPy, SciPy e matplotlib, esta biblioteca contém uma grande quantidade de ferramentas eficientes para aprendizado de máquina e modelagem estatística, incluindo classificação, regressão, *clustering* e redução de dimensionalidade (KUNAL JAIN, 2016).

De forma resumida, as bibliotecas mencionadas anteriormente foram utilizadas da seguinte maneira no código da metodologia.

- A biblioteca **SciPy** foi utilizada para carregar os dados dos sinais de EEG armazenados no cartão SD utilizado na Orange Pi Zero.

- Esses dados foram reorganizados, conforme descrito na seção 3.2.1, sendo posteriormente utilizada a biblioteca **SciPy** para uso do filtro *Butterworth*.

- Para a extração dos atributos foi necessário o uso da biblioteca **Numpy** para a média e variância e a biblioteca **SciPy** para obliquidade.

- Após a extração dos atributos, esses dados foram guardados em um *dataframe* fazendo-se o uso da biblioteca **Pandas**. Um *dataframe* é semelhante a uma planilha do Excel, a diferença é que no *dataframe* os nomes das colunas e números de linha são conhecidos como colunas e índices de linha.

- Nesta fase, os dados estão de forma útil para a modelagem. Sendo utilizada a biblioteca **Scikit Learn** (sklearn) para construir o modelo preditivo, realizar a classificação e fazer avaliação da performance da classificação.

- Os gráficos foram obtidos com o uso da biblioteca **Matplotlib**.

Também foram utilizados os módulos *time* para a verificação dos tempos de execução e *pickle* que é explicado na seção 2.8.2.1.

2.8.2.1 arquivos pickle (.pkl)

"Para não obrigar os usuários a escrever e depurar constantemente código para salvar estruturas de dados, Python oferece o módulo padrão *pickle*" (Python Software Foundation, 2012).

Os arquivos *pickle* servem para compartilhar informações entre diferentes programas Python ou entre programas que possuem diferentes sessões de execução. No caso deste trabalho é necessário o uso dos arquivos *pickle*, devido as duas etapas: treino do algoritmo e a simulação em tempo real (predição das tentativas).

O módulo *pickle* permite converter praticamente qualquer objeto Python para uma *string* de bytes, sendo este processo denominado *pickling*. O processo reverso que consiste em reconstruir o objeto a partir de sua representação como *string de bytes*, denomina-se *unpickling*.

O processo *pickling* foi utilizado na etapa de treino do algoritmo, conforme consta nos passos 6, 8 e 11 do fluxograma apresentado na Figura 3.3.

Já o processo de *unpickling* foi utilizado na etapa de simulação em tempo real, nos passos 3 e 5 do fluxograma da Figura 3.11, para que não houvesse a necessidade de executar novamente os processos executados na fase de treino do algoritmo.

2.9 FLUXOGRAMA DO MÉTODO

A Figura 3.3 representa resumidamente os passos explicados nas seções anteriores, sendo que o modelo de classificador gerado nesta etapa de treinamento será utilizado na etapa de classificação (onde os dados serão obtidos em tempo real, quando a plataforma portátil integrar um sistema BCI), a qual está representada pelo fluxograma da Figura 3.11.

3 RESULTADOS E DISCUSSÕES

3.1 CONJUNTO DE DADOS

O banco de sinais de EEG de imagética motora foi obtido do repositório do Laboratório Cichocki de Processamento de Sinais Cerebrais Avançado do Instituto de Ciências do Cérebro no Japão (CICHOCKI; ZHAO, 2011).

Esse banco de dados contém arquivos no formato Matlab (.mat), os quais estão organizados da seguinte forma [canais, amostras, tentativas], onde canais indica o número de eletrodos, amostras é a duração de cada tarefa de imagética motora e tentativas é a quantidade de tarefas realizadas pelo sujeito. Essas tarefas são separadas em paradigmas binário (com duas classes) representando a tarefa de imagética motora de mão esquerda e mão direita ou em multi-classes (três classes) representando a tarefa de mão esquerda, mão direita e pé ou representando a tarefa de mão esquerda, mão direita e relaxamento.

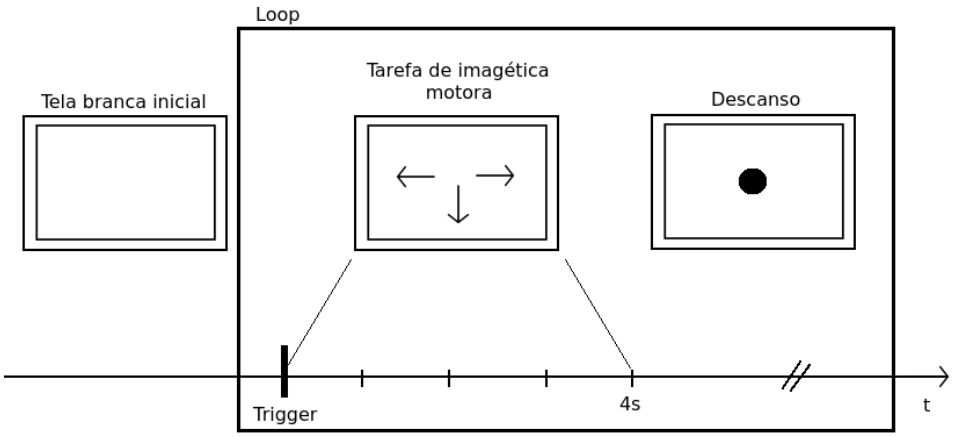
Este conjunto de dados de EEG foi obtido de vários indivíduos saudáveis. Sendo utilizado diferentes paradigmas de BCI, os quais consistiram na imaginação de duas ou três tarefas de imagética motora. Sendo realizada várias sessões em dias diferentes para alguns sujeitos.

Os sujeitos estavam sentados em uma poltrona confortável em frente a uma tela de computador. O teste iniciava com a tela em branco. Depois de dois segundos (t=2s), era apresentada uma sugestão em forma de seta apontado para a esquerda, para direita ou para baixo (correspondendo a mão esquerda, mão direita e pé, respectivamente) permanecendo na tela por um período determinado de 4 segundos. Desta forma o sujeito entendia que deveria começar a realizar a tarefa de imagética motora. Foi requisitado aos sujeitos para continuar com a imaginação da tarefa até que a sugestão desaparecesse da tela e que evitassem piscar os olhos e movimentá-los durante a imaginação. Ao desaparecer a sugestão da tela ocorria uma pausa de 2 segundos. A sequência de sugestões era aleatória. Esse paradigma é ilustrado na Figura 3.1. Para cada sujeito, a primeira execução era chamada de procedimento de inicialização, sendo apenas apresentada a sugestão sem nenhum *feedback*.

Neste conjunto de dados, utilizaram-se dois dispositivos g.tec e Neuroscan para gravar os sinais EEG. Informações referentes a estes dispositivos, podem ser obtidas em (g.tec, 2017) e (Compumedics NeuroScan, 2017).

Os sinais de EEG foram filtrados usando um filto passa-faixa. Para os sujeitos que se utilizou o dispositivo g.tec, os sinais foram filtrados entre 2 e 30 Hz com uma taxa de amostragem de 256 Hz e um filtro rejeita-faixa em 50 Hz. Enquanto para o dispositivo Neuroscan foram filtrados entre 0.1 e 100 Hz com uma taxa de amostragem de 250 Hz. Os

Figura 3.1 – Paradigma de imagética motora dos sinais de EEG utilizados.

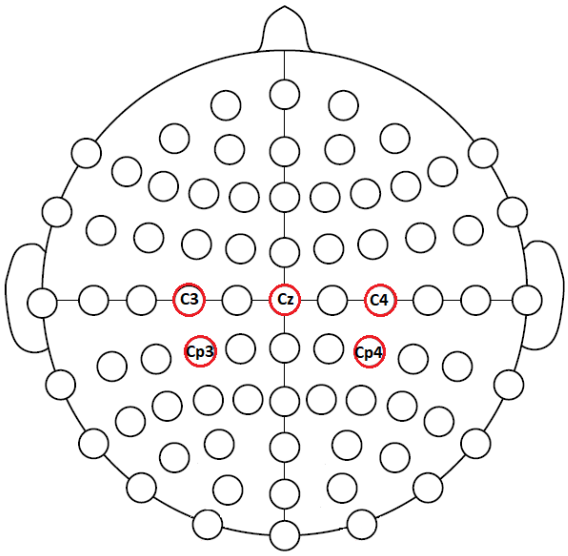


Fonte: (SILVA, 2017)

sinais foram medidos em μV e V para o Neuroscan e o g.tec, respectivamente.

O número de eletrodos é diferente nos conjuntos de dados disponíveis, tendo configuração de 5, 6 e 14 canais. Neste trabalho foram utilizados os sinais EEG que contém informações de 5 eletrodos, correspondentes ao C3, Cp3, C4, Cp4 e Cz, conforme na Figura 3.2.

Figura 3.2 – Sistema internacional 10-20 extendido, indicando os eletrodos utilizados utilizados neste trabalho.



Fonte: AUTOR

3.1.1 Organização dos dados

O conjunto de dados foi organizado com classes de forma balanceada e desbalanceada, sendo utilizada classes de mão (H) e pé (F) ou mão esquerda (LH) e mão direita (RH), com o objetivo de aumentar a performance na detecção de intenções. Em um trabalho anterior de (SILVA, 2017) revisores sugeriram a utilização de classes LH e RH no lugar de H e F, hipótese testada neste trabalho. Além da troca de classes, analisou-se o efeito do balanceamento sobre a performance da metodologia utilizada.

Classes desbalanceadas ocorrem quando há uma grande desproporção entre o número de exemplos de cada classe. Ou seja, em aplicações reais, teremos classes desbalanceadas. Sendo assim, está é a analise de maior interesse neste trabalho. Já as classes de forma balanceada são testadas a fim de verificar a performance dos classificadores utilizados.

3.1.1.1 Classes desbalanceadas

As classes foram reestruturadas da seguinte forma: os rótulos de *left hand* (LH) e *right hand* (LR) virassem apenas a classe referente a mão (H - *hand*) e os rótulos de *foot* (F) foram apenas nomeados, compondo a classe pé (F - *foot*).

Tabela 3.1 – Propriedades dos sinais EEG de cada sujeito utilizado com classes desbalanceadas

	Sujeito A	**Sujeito B**	**Sujeito C**
Dispositivo	g.tec	Neuroscan	g.tec
Total de tentativas	270	174	180
Classes	H/F	H/F	H/F
Tentativas por classe	180/90	116/58	120/60

Fonte: AUTOR

Arquivos do banco de sinais de EEG de imagética motora, utilizados, referentes a Tabela 3.1:

- Para o Sujeito A: SubA_5chan_3LRF.mat;
- Para o Sujeito B: SubB_5chan_3LRF.mat;
- Para o Sujeito C: SubC_5chan_3LRF.mat.

3.1.1.2 Classes balanceadas

Para o sujeito A, as classes foram reestruturadas, desconsiderados os dados (sinais) de EEG referentes a classe *foot*, considerando apenas as classes de mão esquerda (LH - *left hand*) e mão direita (LR - *right hand*).

O arquivo de dados do sujeito B deste caso, continha 6 canais, sendo considerado apenas os 5 canais utilizados nos demais casos. Não foi necessário desconsiderar nenhum sinal, pois os dados continham duas classes, mão esquerda (LH - *left hand*) e mão direita (LR - *right hand*).

Referente ao sujeito C, haviam as classes referentes a mão esquerda (LH - *left hand*), mão direita (LR - *right hand*) e relaxamento (R - *relaxation*). Suas classes foram reestruturadas, desconsiderados os dados (sinais) de EEG referentes a classe *relaxation*, considerando apenas as classes de mão esquerda (LH - *left hand*) e mão direita (LR - *right hand*). Também continha o registro de 14 eletrodos, sendo considerado apenas os 5 canais já utilizados nos demais casos.

Tabela 3.2 – Propriedades dos sinais EEG de cada sujeito utilizado com classes balanceadas

	Sujeito A	**Sujeito B**	**Sujeito C**
Dispositivo	g.tec	Neuroscan	Neuroscan
Total de tentativas	180	162	232
Classes	LH/RH	LH/RH	LH/RH
Tentativas por classe	90/90	81/81	116/166

Fonte: AUTOR

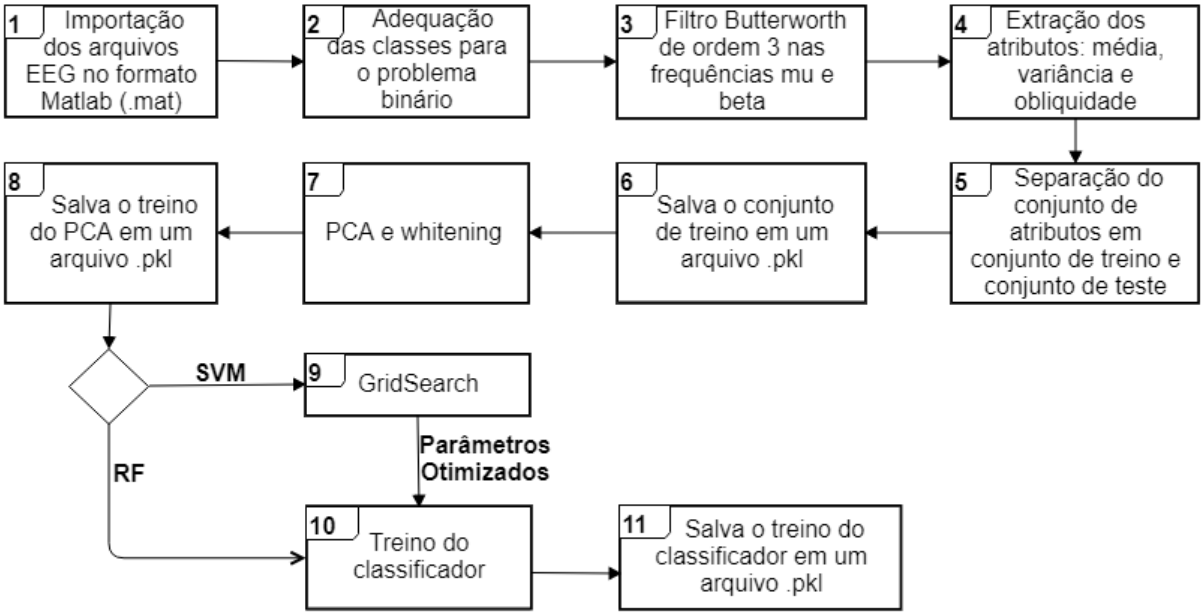
Arquivos do banco de sinais de EEG de imagética motora, utilizados, referentes a Tabela 3.2:

- Para o Sujeito A: SubA_5chan_3LRF.mat;
- Para o Sujeito B: SubB_6chan_2LR.mat;
- Para o Sujeito C: SubC_14chan_3LRR.mat.

3.2 TREINO DO ALGORITMO

A Figura 3.3 representa o fluxograma dos passos executados na etapa de treino do algoritmo. Esta etapa pode ser executada na própria placa (Orange Pi Zero) ou em um computador, desde que posteriormente os arquivos *pickle* (.pkl) gerados referentes ao treino do PCA e classificadores sejam transferidos para a placa.

Figura 3.3 – Fluxograma da metodologia - etapa do treino do algoritmo.



Fonte: AUTOR

O passo 6 é necessário apenas para que na etapa de classificação (simulação de leitura do sinal EEG em tempo real) possa ser avaliada corretamente a performance da classificação.

O fluxograma da Figura 3.3 referente a etapa de treino do algoritmo, representa as etapas de pré-processamento, extração de atributos e treino do PCA e classificadores. Sendo assim, o detalhamento dos passos deste fluxograma será apresentado nas seções a seguir.

A metodologia foi implementada na linguagem Python. Informações referentes às bibliotecas utilizadas, podem ser consultadas na seção 2.8.2.

3.2.1 Pré-processamento

Os dados do sinal EEG importado (formato .mat), continha todas as informações de forma agrupada, então para uma melhor manipulação destes dados no decorrer do código, eles foram reorganizados em variáveis contendo os dados (sinal), as classes, quantidade de tentativas, tamanho da amostra e frequência.

As classes foram reestruturadas conforme consta na seção 3.1.1.1 e seção 3.1.1.2. Não sendo realizada nenhuma análise para remoção de tentativas (*trials*) inválidas pois inferiu-se que o conjunto de sinais disponibilizado por (CICHOCKI; ZHAO, 2011) já estivesse pré-selecionado.

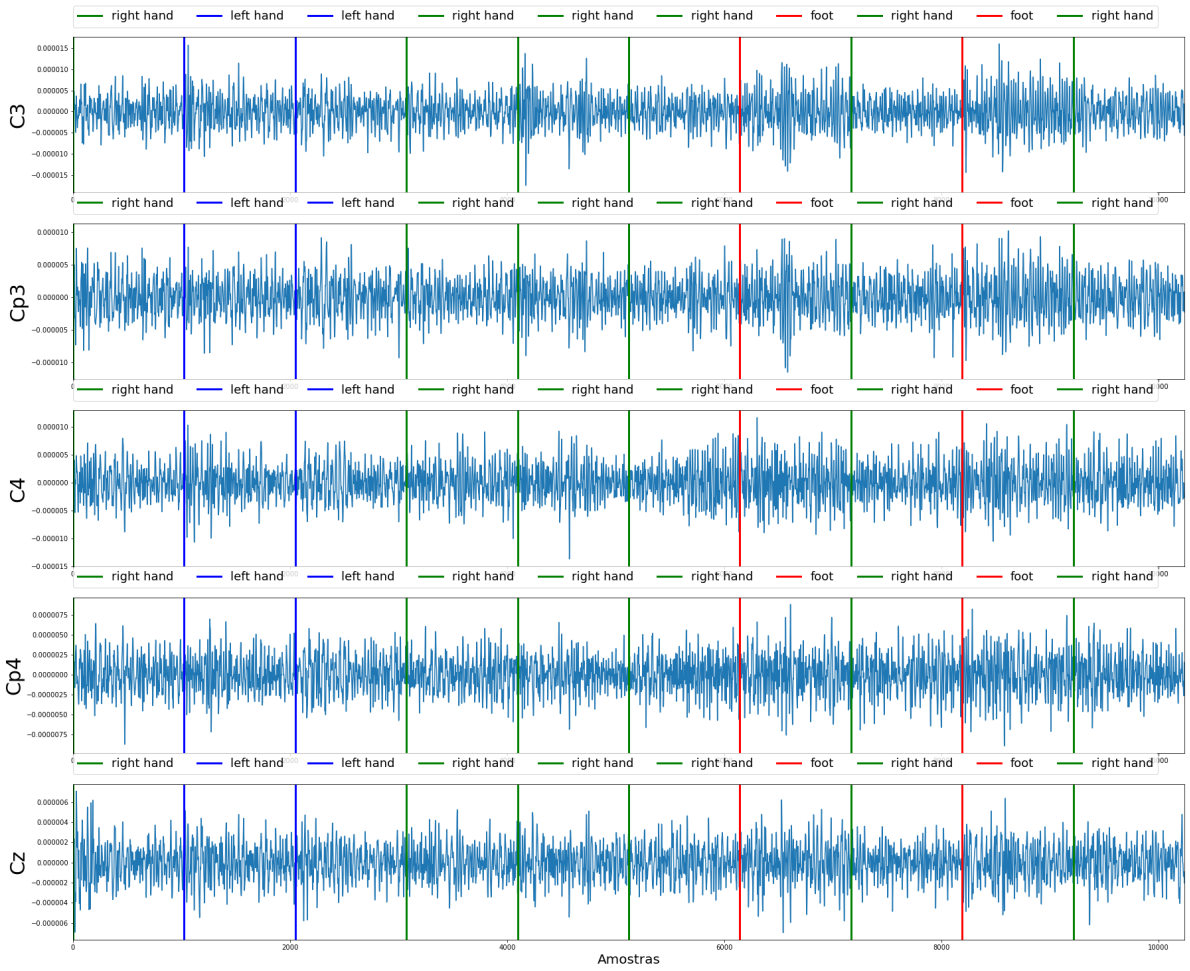
Os dados organizados na forma [canais, amostras, tentativas] foram armazenados em um *dataframe* contendo as amostras de cada tentativa como índices de linhas e em cada coluna um canal dos canais utilizados. Conforme demonstrado na Figura 3.4.

Figura 3.4 – 4 índices de linha de cada canal do *dataframe*.

	C3	Cp3	C4	Cp4	Cz
0	-2.378336e-06	-1.307604e-06	-2.785485e-07	6.563342e-07	9.572987e-07
1	-3.772074e-06	-2.868773e-06	-1.367637e-06	-3.139187e-07	6.121963e-07
2	-3.998892e-06	-3.452935e-06	-2.371728e-06	-1.292090e-06	1.246743e-07
3	-3.120829e-06	-3.025374e-06	-3.026765e-06	-2.079096e-06	-2.688373e-07
4	-1.584262e-06	-1.901159e-06	-3.200633e-06	-2.573274e-06	-3.682343e-07

Fonte: AUTOR

Figura 3.5 – Sinal EEG orginal referente ao sujeito A, com cada tarefa separada por cores diferentes.



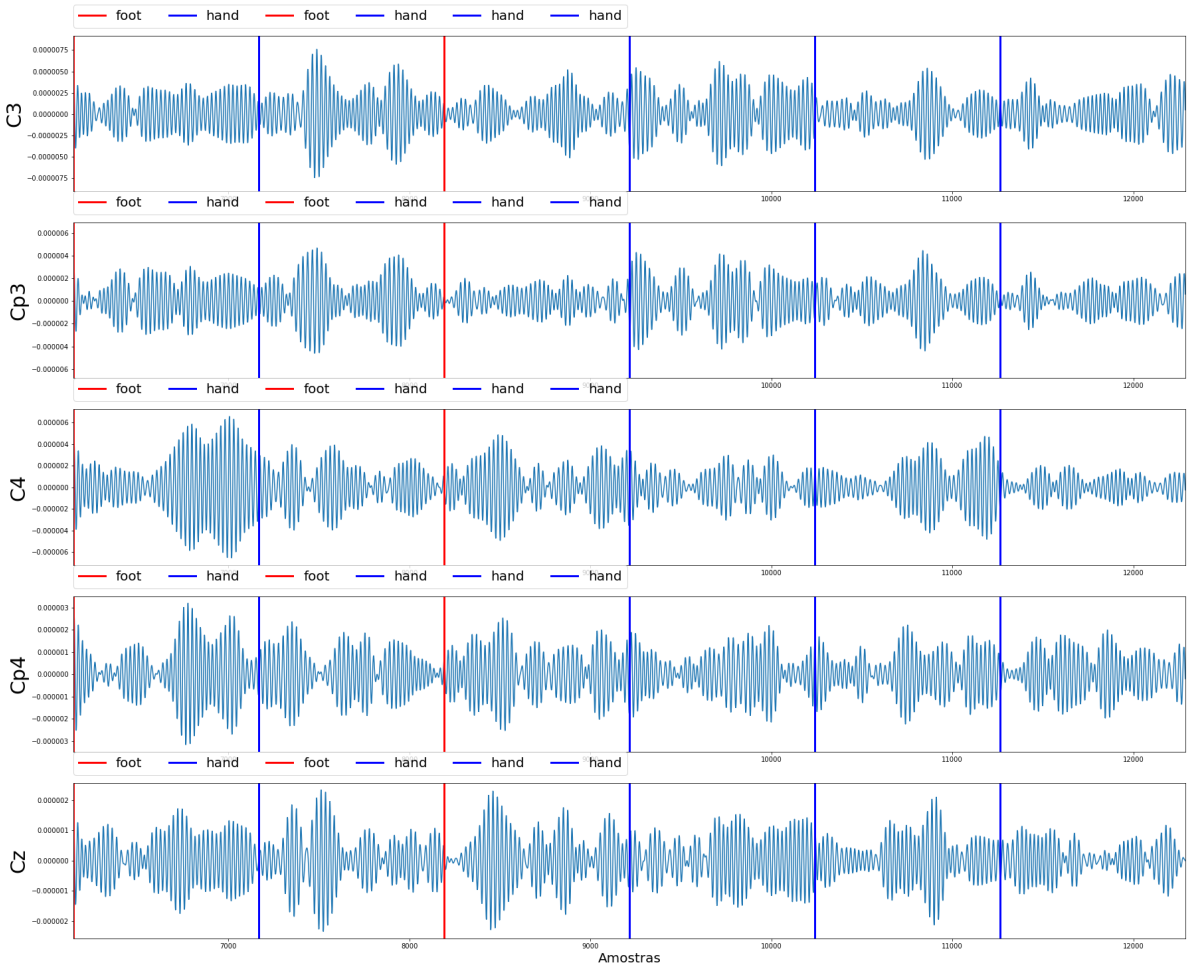
Fonte: AUTOR

Na Figura 3.5 consta o sinal EEG original de cada canal, para as 10 primeiras tentativas, referente ao sujeito A (referenciado na Tabela 3.1), demonstrando as 3 classes separadamente.

Os sinais (de cada canal) foram filtrados em μ e β com um filtro *Butterworth* de ordem 3. De forma que ao final do passo 3 da Figura 3.3 referente ao fluxograma do treino do algoritmo, tem-se 10 sinais filtrados.

A Figura 3.6 demonstra o sinal original (representado na Figura 3.5) filtrado na banda μ, para as classes mão (*hand*) e pé (*foot*) a partir da 7ª tentativa.

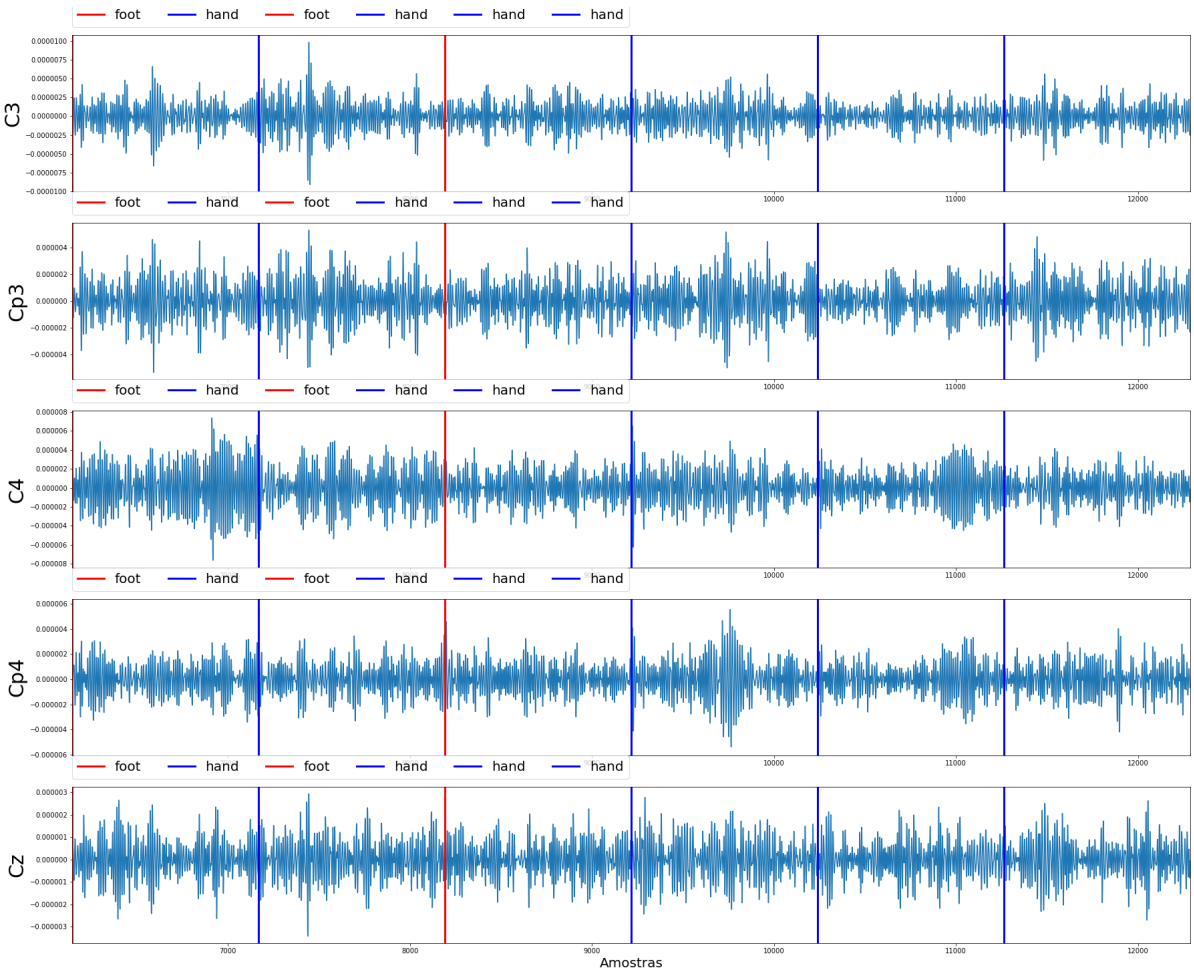
Figura 3.6 – Sinal referente ao sujeito A, considerando as classes de mão (*hand*) e pé (*foot*), filtrado na banda mu.



Fonte: AUTOR

Sendo que a Figura 3.7 demostra o sinal original (representado na Figura 3.5) filtrado na banda β para as classes mão (*hand*) e pé (*foot*) a partir da 7ª tentativa.

Figura 3.7 – Sinal referente ao sujeito A, considerando as classes de mão (*hand*) e pé (*foot*), filtrado na banda beta.



Fonte: AUTOR

3.2.2 Extração de Atributos

A cada 4s (1024 pontos), o qual corresponde ao tempo total de cada tentativa, foram extraídos os atributos referentes a média, variância e obliquidade.

Para cada um dos 10 sinais, obtidos após o passo 3 da Figura 3.3, foram calculados os estimadores estatísticos das Equações 2.1, 2.2 e 2.3, totalizando os 30 atributos. O vetor de atributos é mostrado na Equação 3.1.

$$\begin{aligned} V = [&\bar{X}_{C3\mu}, \bar{X}_{Cp3\mu}, \bar{X}_{C4\mu}, \bar{X}_{Cp4\mu}, \bar{X}_{CZ\mu}, \bar{X}_{C3\beta}, \bar{X}_{Cp3\beta}, \bar{X}_{C4\beta}, \bar{X}_{Cp4\beta}, \bar{X}_{CZ\beta}, \\ &V_{C3\mu}, V_{Cp3\mu}, V_{C4\mu}, V_{Cp4\mu}, V_{CZ\mu}, V_{C3\beta}, V_{Cp3\beta}, V_{C4\beta}, V_{Cp4\beta}, V_{CZ\beta}, \\ &S_{C3\mu}, S_{Cp3\mu}, S_{C4\mu}, S_{Cp4\mu}, S_{CZ\mu}, S_{C3\beta}, S_{Cp3\beta}, S_{C4\beta}, S_{Cp4\beta}, S_{CZ\beta}] \end{aligned} \tag{3.1}$$

Esses dados são armazenados em um *dataframe*, constando as tentativas como

índices de linhas e em cada coluna um dos atributos. A Figura 3.8 demonstra parte deste *dataframe*.

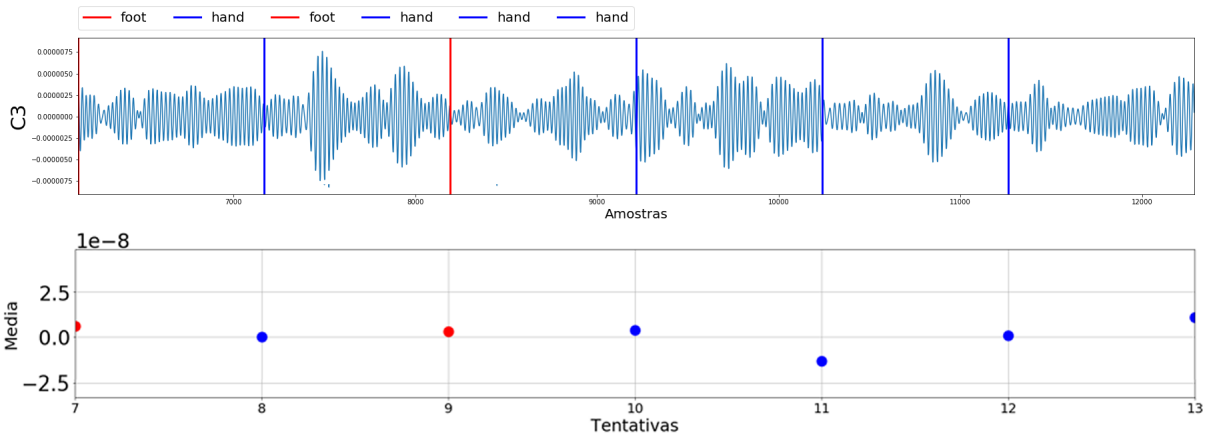
Figura 3.8 – Parte do *dataframe* que contém as tentativas como índices de linha e cada atributo como coluna.

	mean_C3_mu	mean_Cp3_mu	mean_C4_mu	mean_Cp4_mu	mean_Cz_mu	var_C3_mu	...
0	3.210164e-09	5.031355e-09	5.649628e-09	6.688298e-10	8.703913e-09	9.877763e-13	...
1	2.775066e-09	2.144599e-09	2.509615e-09	-1.906938e-09	7.639858e-10	3.027900e-12	...
2	8.811057e-09	1.127483e-08	-2.434438e-10	1.583445e-09	-1.144857e-09	1.177644e-12	...
3	-6.937062e-09	-8.076348e-09	-5.147430e-10	-1.965927e-09	3.306208e-09	1.892673e-12	...
4	-3.740371e-09	-5.514940e-09	-6.323517e-09	-3.559979e-09	-3.097134e-09	4.120941e-12	...
5	-8.490657e-09	-2.311254e-09	-6.860712e-09	-1.637673e-09	-4.010374e-09	2.050332e-12	...
6	5.880435e-09	3.578444e-09	2.929620e-09	4.574497e-09	4.932101e-10	6.519183e-12	...

Fonte: AUTOR

A Figura 3.9 mostra o atributo média, extraído do sinal C3, filtrado na banda mu. Demonstrando que após a extração de atributos o sinal é comprimido, a fim de reduzir a dimensão do vetor de dados (sem a perda de informações relevantes). Ou seja, cada 1024 pontos foi reduzido a um ponto da característica demonstrada.

Figura 3.9 – Atributo Média, extraído do sinal referente ao canal C3, filtrado na banda mu.



Fonte: AUTOR

3.2.3 Treino do PCA e classificadores

Essa etapa é representada a partir do passo 5 do fluxograma da etapa de treino do algoritmo (Figura 3.3).

Após a extração dos atributos este conjunto foi divido em conjunto de treino (60% dos dados) e conjunto de teste (40% dos dados). Geralmente, quanto maior o conjunto de treinamento melhor a performance de classificação (WHITTEN; FRANK; HALL, 2011).

Após essa separação dos conjuntos foi efetuada a transformada de PCA nos dados a fim de descorrelacioná-los e deixá-los “brancos”.

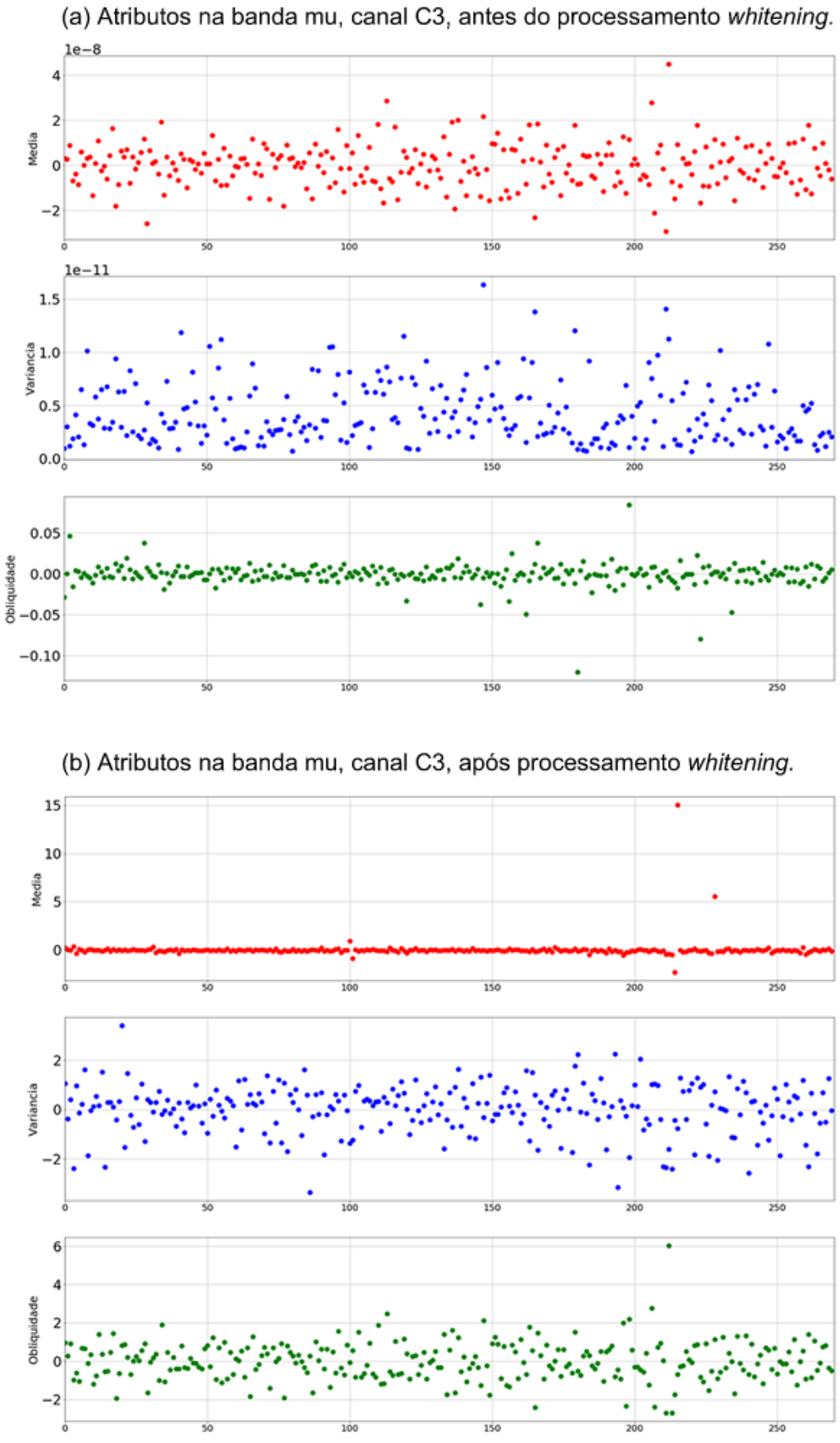
A Figura 3.10a demonstra os 270 pontos (cada ponto representa uma tentativa) de atributos extraídos do sujeito A antes da transformação usando o PCA.

Após a transformada de PCA nos dados, conforme a Figura 3.10b, nota-se que as ordens de grandeza dos atributos mudou.

Conforme o passo 8 do fluxograma da etapa de treino do algoritmo (Figura 3.3), o treino do PCA é salvo em um arquivo no formato *pickle* (.pkl) para ser usado posteriormente na fase de classificação dos dados reais. Para mais informações sobre os arquivos *pickle* (.pkl), consultar seção 2.8.2.1.

Executados os 8 passos do fluxograma da etapa de treino do algoritmo (Figura 3.3), é efetuado o treino de cada um dos classificadores utilizados, sendo que o classificador SVM contém uma etapa a mais, referente a otimização de parâmetros, a qual não é necessária para o classificador RF. Por fim é gerado um arquivo *pickle* (.pkl) referente a cada classificador treinado.

Figura 3.10 – Diferença os atributos antes e depois do processamento *whitening*.



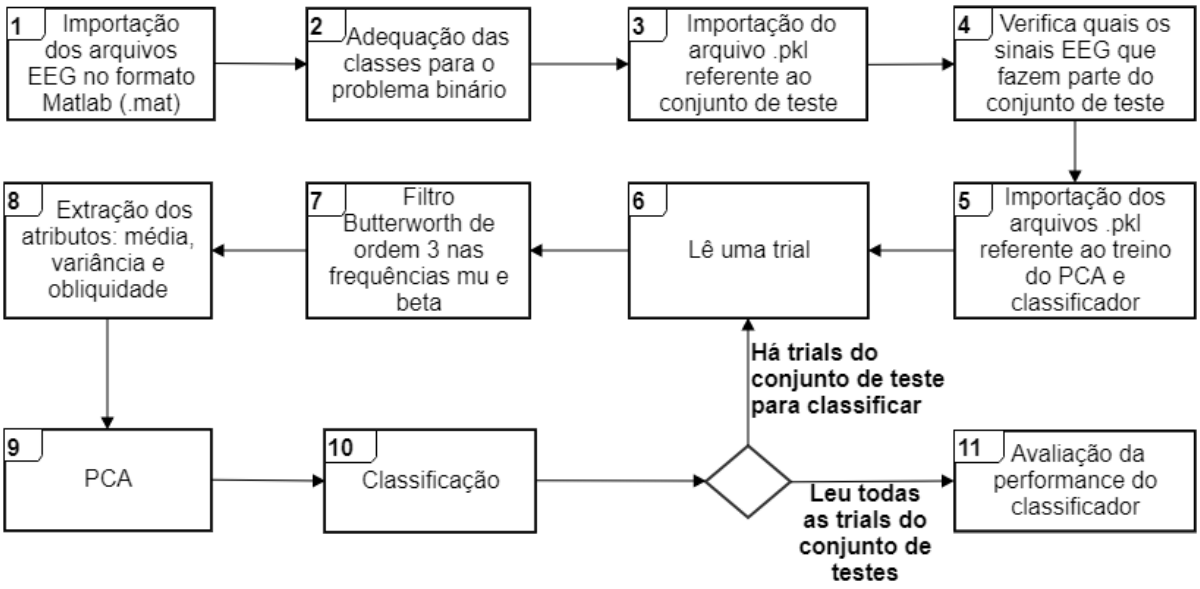
Fonte: AUTOR

3.3 SIMULAÇÃO EM TEMPO REAL

Na metodologia proposta (Figura 2.1), não era realizada nenhuma simulação em tempo real. Após o treino dos classificadores, aplicava-se o classificador no conjunto de testes e posteriormente se realizava a avaliação da performance do classificador com o conjunto de teste.

Esta etapa é executada na placa Orange Pi Zero, utilizando os arquivos *pickle* (.pkl) gerados na etapa de treino do algoritmo. Foi inclusa essa etapa, para que seja possível realizar a aquisição de um sinal EEG em tempo real e realizar a sua predição, quando a plataforma portátil for integrada com um dispositivo que realize a aquisição do sinal EEG. Neste trabalho é realizada a simulação da aquisição de um sinal EEG de um banco de sinais que estará armazenado no cartão SD da Orange Pi Zero.

Figura 3.11 – Fluxograma da metodologia - etapa de classificação de um sinal EEG



Fonte: AUTOR

A Figura 3.11 representa o fluxograma da etapa de classificação de um sinal EEG. Os passos 1, 2, 3 e 4 foram inclusos a fim de simular a leitura de um sinal EEG que não seja conhecido pelo algoritmo (não faça parte dos sinais do conjunto de treino). Quando este trabalho for integrado com outro que faça a aquisição do sinal EEG de uma pessoa em tempo real, o fluxograma segue a partir do passo 5, onde no passo 6 será feita a leitura do sinal EEG recebido em tempo real.

Da mesma forma efetuada na etapa de treino do algoritmo, os dados do sinal EEG importados (formato .mat), são reorganizados em variáveis contendo informações relevantes. E efetuada a reestruturação das classes para o problema binário.

Em seguida é importado o conjunto de teste (40% dos dados totais) gerado na etapa

de treino do algoritmo, para que com o índice da lista deste conjunto, seja possível verificar quais os dados dos sinais EEG faziam parte do conjunto de teste na etapa de treinamento, podendo assim desconsiderar os dados que fizeram parte do conjunto de treino (dados já conhecidos pelo algoritmo).

Após ter definido quais os sinais EEG que o algoritmo ainda não conhece, é importado os demais arquivos *pickle* (.pkl) gerados na etapa de treinamento do algoritmo, referentes ao treino do PCA e treino do classificador (SVM ou RF).

Conforme o fluxograma da Figura 3.11, o passo 6 é repetido até que tenha sido classificada cada tentativa (*trials*) do sujeito em questão.

O tempo médio de classificação de cada tentativa, é apresentado nas tabelas de confusão. Esse tempo inicia no passo 6 do fluxograma da Figura 3.11 e finaliza após a predição da tentativa, apresentada no passo 10. Esse é o tempo que o dispositivo levará para fazer o processamento necessário para a predição de uma tentativa adquirida de outro dispositivo que fará a aquisição do sinal EEG em tempo real.

Para cada tentativa são seguidos os passos 7, 8, 9 e 10. Ao selecionar uma tentativa, cada canal (C3, Cp3, C4, Cp4 e Cz) deste sinal é filtrado na banda μ e β, resultando em 10 sinais (5 canais x 2 bandas de frequência). Após são extraídos os atributos referentes a média, variância e obliquidade para os 10 sinais, resultado em 30 atributos para a tentativa.

No próximo passo, passa-se o PCA nos dados da tentativa e posteriormente é realizada a classificação desta. Para que ao fim de todas as leituras referentes as tentativas (*trials*) que faziam parte do conjunto de testes na fase de treinamento, seja possível realizar a avaliação da performance do classificador utilizado, é guardado em uma lista os dados referente ao sinal da tentativa após passar o PCA, em outra lista é guardada a informação de qual era a predição esperada e em outra lista é guardada a informação de qual foi a predição para cada tentativa em questão.

3.3.1 Máquina de Vetores de Suporte

A SVM implementada neste trabalho utilizou o *kernel* conhecido como Função de Base Radial (*Radial Basis Function* - RBF).

Usando o algoritmo de GridSearch foram escolhidos os parâmetros possíveis variando de 2 a 6 para o parâmetro C e de -5 a 0 para o parâmetro σ (SILVA, 2017).

3.3.1.1 Classes desbalanceadas

Os dados referentes ao dispositivo utilizado na aquisição do sinal de cada sujeito, assim como quais as classes e quantidade total de tentativas (*trials*), constam na Tabela 3.1.

A Tabela 3.3 mostra os resultados para os parâmetros **C** e σ após o uso do algoritmo GridSearch.

Tabela 3.3 – Parâmetros do classificador SVM para classes desbalanceadas

	Sujeito A	**Sujeito B**	**Sujeito C**
C	1000.0	1000.0	1000.0
σ	1.01e-05	1.01e-05	0.0001778

Fonte: AUTOR

A Tabela 3.4 apresenta a matriz de confusão referente aos sujeitos A, B e C, assim como a precisão para cada classe, os respectivos tempos de processamento, quantidade total de tentativas (*trials*) do conjunto de testes e acurácia.

A média de precisão obtida para a classe Pé (Foot) foi de 0,78 e para a classe Mão (Hand) foi de 0,83. O tempo médio de classificação de cada tentativa (*trial*) foi 1.31 segundos. O tempo total de execução mostrado na tabela é referente ao tempo que o algoritmo levou para executar os passos demonstrados no fluxograma da Figura 3.11, levando mais tempo de execução para um conjunto de testes maior, conforme esperado. No caso da execução em tempo real o tempo será menor, pois neste caso de simulação foi necessária a inclusão de verificações para que a simulação fosse a mais próxima de um caso real, por exemplo, foi necessário verificar em código quais tentativas faziam parte do conjunto de teste gerado na etapa de treino.

Tabela 3.4 – Matriz de confusão para os sujeitos A, B e C - classificador SVM com classes desbalanceadas

		Sujeito A		**Sujeito B**		**Sujeito C**	
		Pé	Mão	Pé	Mão	Pé	Mão
Classes reais	Pé	**34**	2	**21**	4	**11**	9
	Mão	8	**64**	8	**37**	11	**41**
Precisão		0.944	0.889	0.840	0.822	0.550	0.788
Tempo de classificação (s)		1.33		1.28		1.32	
Tempo total de execução (s)		144.95		90.11		95.84	
Trials do conjunto de testes		108		70		72	
Acurácia		0.907		0.828		0.722	

Fonte: AUTOR

3.3.1.2 Classes balanceadas

Os dados referentes ao dispositivo utilizado na aquisição do sinal de cada sujeito, assim como quais as classes e quantidade total de tentativas (*trials*), constam na Tabela 3.2.

A Tabela 3.5 mostra os resultados para os parâmetros **C** e σ após o uso do algoritmo GridSearch.

Tabela 3.5 – Parâmetros do classificador SVM para classes balanceadas

	Sujeito A	**Sujeito B**	**Sujeito C**
C	100.0	100.0	10000.0
σ	0.0031622	0.0031622	0.0001778

Fonte: AUTOR

A Tabela 3.6 apresenta a matriz de confusão referente aos sujeitos A, B e C, assim como a precisão para cada classe, os respectivos tempos de processamento, quantidade total de tentativas (*trials*) do conjunto de testes e acurácia.

A média de precisão obtida para a classe Mão Esquerda (LH) foi de 0,81 e para a classe Mão Direita (RH) foi de 0,74. O tempo médio de classificação de cada tentativa (*trial*) foi 1,28 segundos. O tempo total de execução mostrado na tabela é referente ao tempo que o algoritmo levou para executar os passos demonstrados no fluxograma da Figura 3.11, conforme já mencionado anteriormente.

Tabela 3.6 – Matriz de confusão para os sujeitos A, B e C - classificador SVM com classes balanceadas

		Sujeito A		**Sujeito B**		**Sujeito C**	
		LH	RH	LH	RH	LH	RH
Classes reais	LH	**32**	4	**27**	8	**40**	13
	RH	7	**29**	6	**24**	15	**25**
Precisão		0.889	0.806	0.771	0.800	0.755	0.625
Tempo de classificação (s)		1.30		1.25		1.28	
Tempo total de execução (s)		95.58		83.52		119.83	
Trials do conjunto de testes		72		65		93	
Acurácia		0.847		0.785		0.699	

Fonte: AUTOR

3.3.2 *Florestas Aleatórias*

Para a classificação usando RF não é necessária a otimização dos parâmetros (SILVA, 2017).

Tabela 3.7 – Parâmetros do classificador RF

Parâmetro	Valor
#árvores	10
#dimensões por nodo	6

Fonte: (SILVA, 2017)

Onde o número de dimensões por nodo foi definido como $\sqrt{\#atributos} = \sqrt{30}$ = 5,477 sendo arredondado para 6.

3.3.2.1 Classes desbalanceadas

Os dados referentes ao dispositivo utilizado na aquisição do sinal de cada sujeito, assim como quais as classes e quantidade total de tentativas (*trials*), constam na Tabela 3.1.

A Tabela 3.8 apresenta a matriz de confusão referente aos sujeitos A, B e C, assim como a precisão para cada classe, os respectivos tempos de processamento, quantidade total de tentativas (*trials*) do conjunto de testes e acurácia.

A média de precisão para a classe Pé (Foot) foi de 0,58 e para a classe Mão (Hand) foi de 0,87. O tempo médio de classificação de cada tentativa (*trial*) foi 1,32 segundos. O tempo total de execução mostrado na tabela é referente ao tempo que o algoritmo levou para executar os passos demonstrados no fluxograma da Figura 3.11, conforme já mencionado anteriormente.

3.3.2.2 Classes balanceadas

Os dados referentes ao dispositivo utilizado na aquisição do sinal de cada sujeito, assim como quais as classes e quantidade total de trials, constam na Tabela 3.2.

A Tabela 3.9 apresenta a matriz de confusão referente aos sujeitos A, B e C, assim como a precisão para cada classe, os respectivos tempos de processamento, quantidade total de tentativas (*trials*) do conjunto de testes e acurácia.

A média de precisão para a classe Mão Esquerda (LH) foi de 0,55 e para a classe Mão Direita (RH) foi de 0,70. O tempo médio de classificação de cada tentativa (*trial*) foi 1,32 segundos. O tempo total de execução mostrado na tabela é referente ao tempo que

Tabela 3.8 – Matriz de confusão para os sujeitos A, B e C - classificador RF com classes desbalanceadas

		Sujeito A		**Sujeito B**		**Sujeito C**	
		Pé	Mão	Pé	Mão	Pé	Mão
Classes reais	Pé	**31**	5	**11**	14	**9**	11
	Mão	15	**57**	3	**42**	6	**46**
Precisão		0.861	0.791	0.440	0.933	0.450	0.885
Tempo de classificação (s)		1.33		1.31		1.32	
Tempo total de execução (s)		144.72		92.18		95.96	
Trials do conjunto de testes		108		70		72	
Acurácia		0.815		0.757		0.764	

Fonte: AUTOR

o algoritmo levou para executar os passos demonstrados no fluxograma da Figura 3.11, conforme já comentado anteriormente.

Tabela 3.9 – Matriz de confusão para os sujeitos A, B e C - classificador RF com classes balanceadas

		Sujeito A		**Sujeito B**		**Sujeito C**	
		LH	RH	LH	RH	LH	RH
Classes reais	LH	**22**	14	**18**	17	**25**	28
	RH	8	**28**	6	**24**	16	**24**
Precisão		0.667	0.694	0.514	0.800	0.471	0.600
Tempo de classificação (s)		1.34		1.30		1.31	
Tempo total de execução (s)		97.09		84.88		122.31	
Trials do conjunto de testes		72		65		93	
Acurácia		0.694		0.646		0.527	

Fonte: AUTOR

3.3.3 Curva ROC

A Curva ROC é utilizada para demonstrar as relações entre sinal-ruído, onde o sinal é interpretado como os verdadeiros positivos (sensibilidade) e o ruído como os verdadeiros negativos (especificidade). Sendo assim, o objetivo é de que as curvas que estão sendo avaliadas devam estar o mais próximo possível do canto superior esquerdo.

3.3.3.1 Classes desbalanceadas

A Figura 3.12 demonstra a comparação entre os dois classificadores utilizados, referente a cada sujeito, para as classes desbalanceadas.

Para o classificador SVM, a média da AUC da curva ROC foi de 0.863. Já para o classificador RF obteve um média da AUC da curva ROC de 0.837.

O classificador SVM obteve, em média, um desempenho de 2,6% melhor, referente ao classificador RF.

3.3.3.2 Classes balanceadas

A Figura 3.13 demonstra a comparação entre os dois classificadores utilizados, referente a cada sujeito, para as classes balanceadas.

Para o classificador SVM, a média da AUC da curva ROC foi de 0.520. Já para o classificador RF obteve um média da AUC da curva ROC de 0.653.

O classificador RF obteve, em média, um desempenho de 13.3% melhor, referente ao classificador SVM.

3.3.4 Curva PR

O objetivo é de as curvas que estão sendo avaliadas estarem o mais próximo possível do canto superior esquerdo.

3.3.4.1 Classes desbalanceadas

A Figura 3.14 demonstra a comparação entre os dois classificadores utilizados, referente a cada sujeito, para as classes desbalanceadas.

Para o classificador SVM, a média da AUC da curva PR foi de 0.757. Já para o classificador RF obteve um média da AUC da curva PR de 0.707.

O classificador SVM obteve, em média, um desempenho de 5% melhor, referente ao classificador RF.

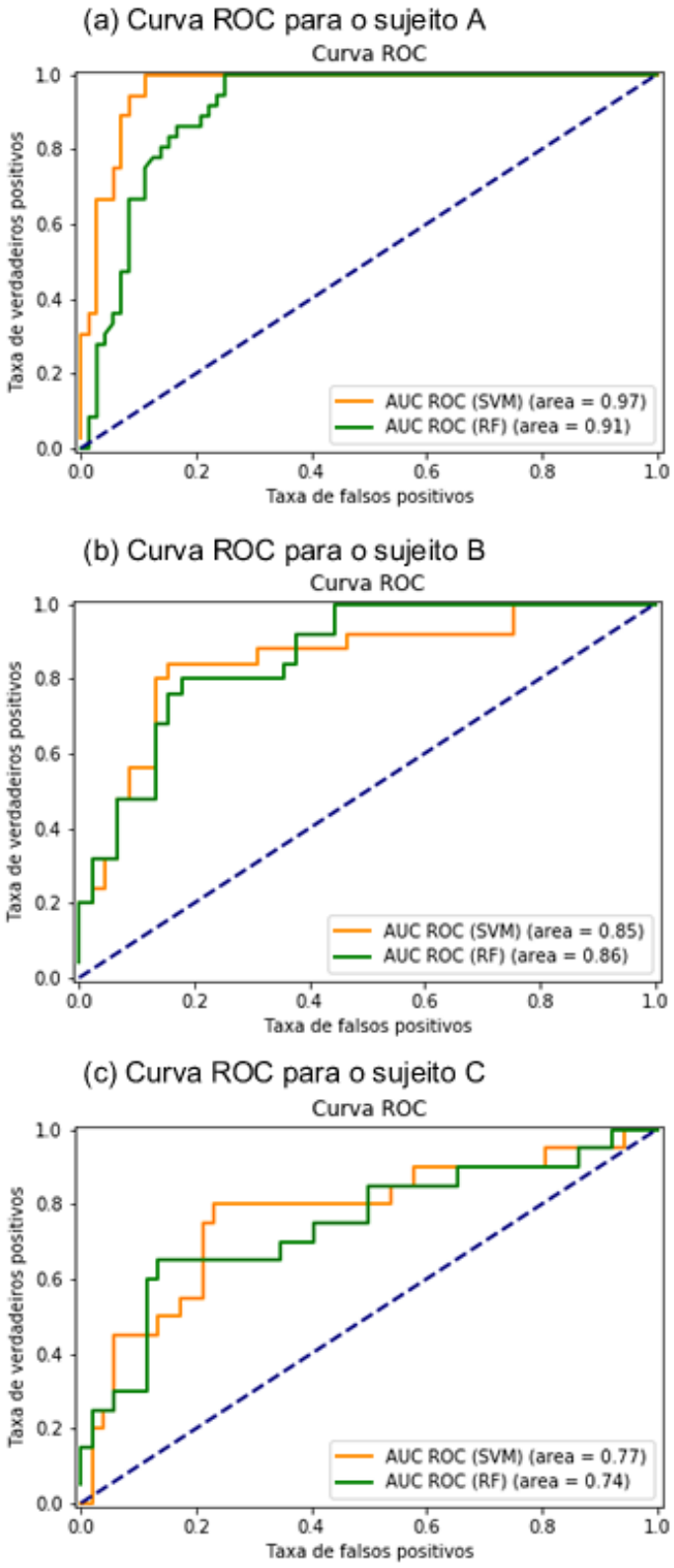
3.3.4.2 Classes balanceadas

A Figura 3.15 demonstra a comparação entre os dois classificadores utilizados, referente a cada sujeito, para as classes balanceadas.

Para o classificador SVM, a média da AUC da curva PR foi de 0.567. Já para o classificador RF obteve um média da AUC da curva PR de 0.687.

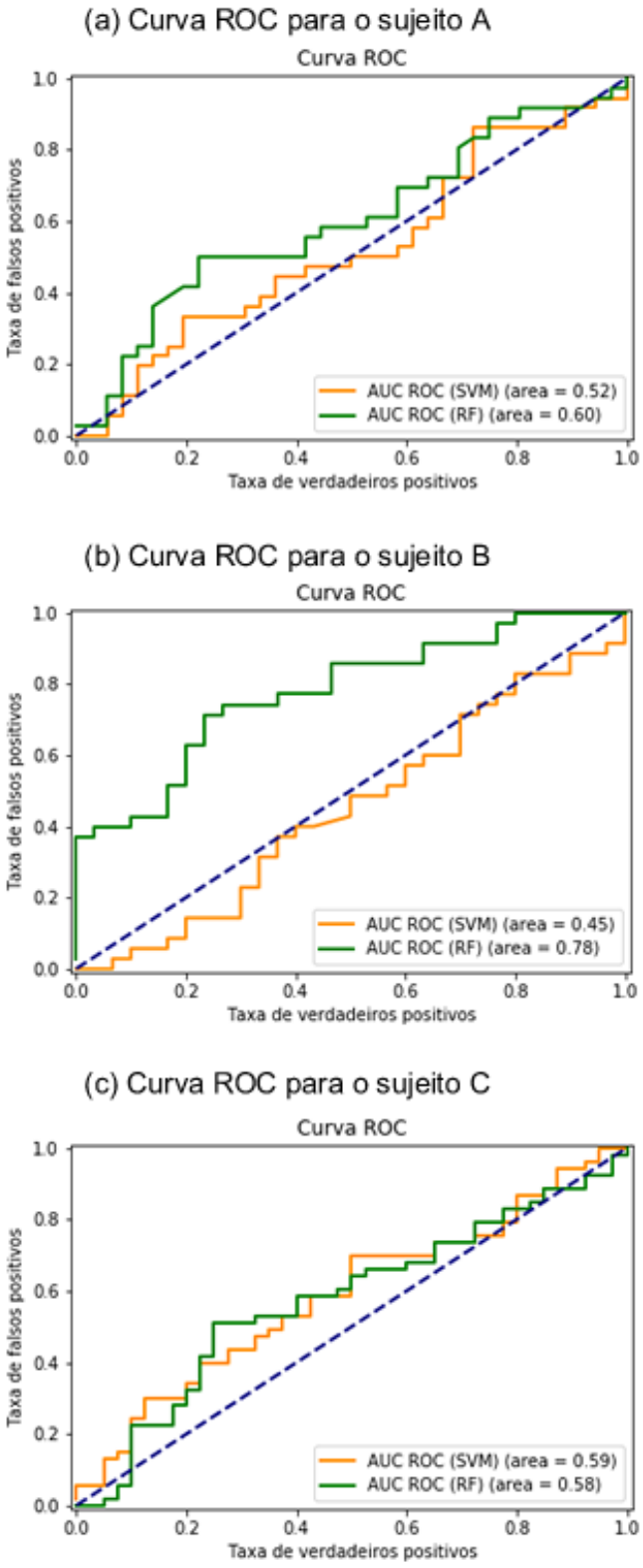
O classificador RF obteve, em média, um desempenho de 12% melhor, referente ao classificador SVM.

Figura 3.12 – Curva ROC para os Sujeitos A, B e C. Classes desbalanceadas (H/F)



Fonte: AUTOR

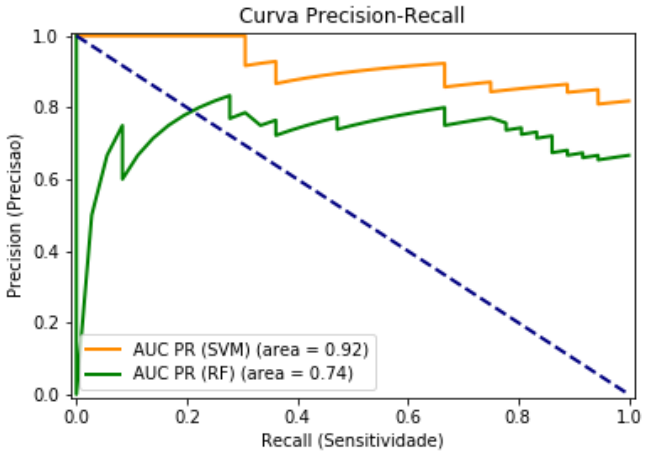
Figura 3.13 – Curva ROC para os Sujeitos A, B e C. Classes balanceadas (LH/RH)



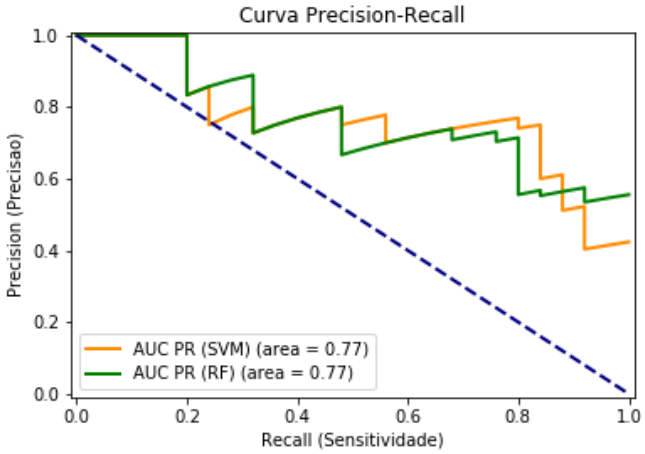
Fonte: AUTOR

Figura 3.14 – Curva PR para os Sujeitos A, B e C. Classes desbalanceadas (H/F)

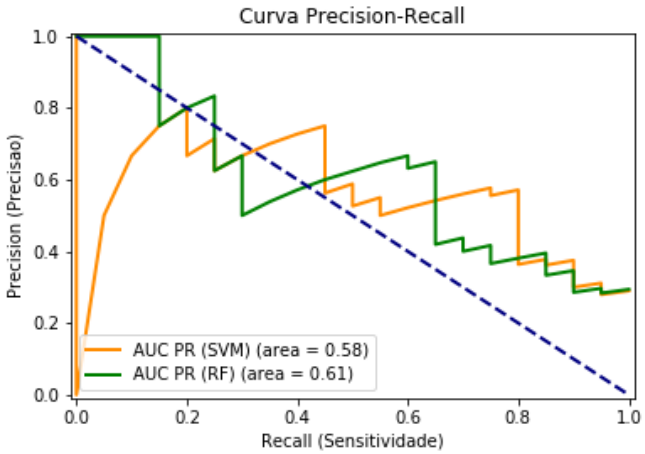
(a) Curva PR para o sujeito A



(b) Curva PR para o sujeito B



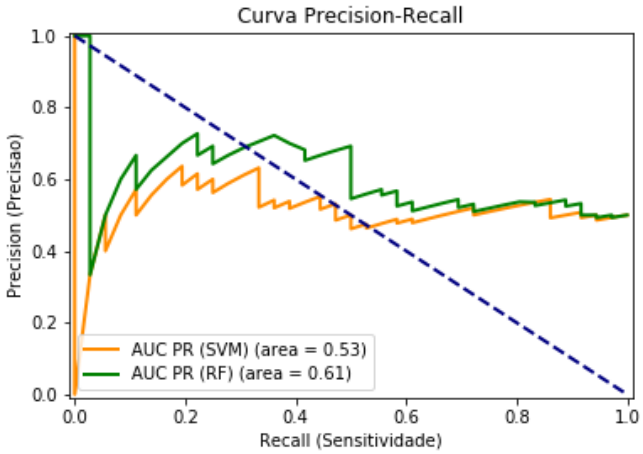
(c) Curva PR para o sujeito C



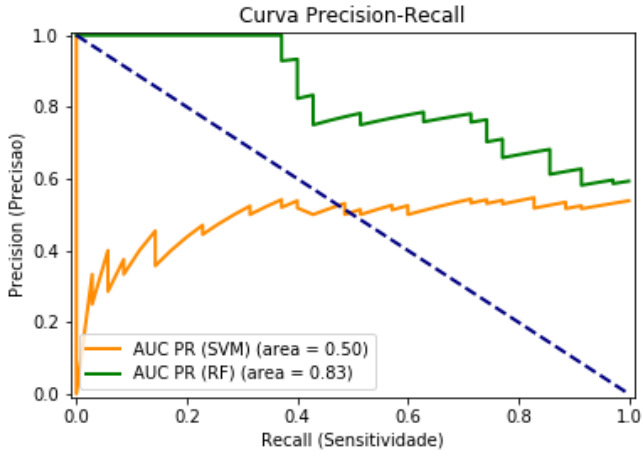
Fonte: AUTOR

Figura 3.15 – Curva PR para os Sujeitos A, B e C. Classes balanceadas (LH/RH)

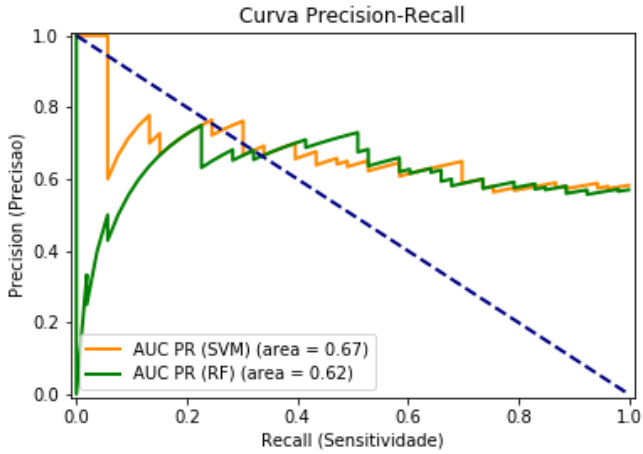
(a) Curva PR para o sujeito A



(b) Curva PR para o sujeito B



(c) Curva PR para o sujeito C



Fonte: AUTOR

3.4 ANÁLISE DOS RESULTADOS

Foram apresentados resultados usando métodos de classificação para classes balanceadas e desbalanceadas a fim de verificar a performance dos classificadores utilizados (SVM e RF), sendo utilizado 5 canais de EEG. A metodologia utilizada é composta pelas etapas de treino do algoritmo e simulação em tempo real. Uma parte do pré-processamento já é feita pelo distribuidor do conjunto de sinais, que o separou em tentativas de 4 segundos cada. Não foi analisada a existência de tentativas inválidas.

Ao realizar a simulação de tempo real, onde a predição de cada tentativa é realizada uma a uma, obteve-se uma melhor performance para ambas as metodologias, em comparação a aplicação do classificador sobre todo o conjunto de testes em uma única vez para realizar a predição das tentativas.

Tabela 3.10 – Comparação dos resultados apresentados na seção 3.3

	Classes Desbalanceadas	**Classes Balanceadas**
	SVM	**SVM**
precisão média	0,78	0,81
especificidade média	0,90	0,74
média da acurácia	0,819	0,777
média AUC ROC	0,863	0,520
média AUC PR	0,757	0,567
tempo médio de classificação	1,31 s	1,28 s
	RF	**RF**
precisão média	0,58	0,55
especificidade média	0,83	0,73
média da acurácia	0,779	0,622
média AUC ROC	0,837	0,653
média AUC PR	0,707	0,687
tempo médio de classificação	1,32 s	1,32 s

Fonte: AUTOR

Nos testes realizados para classes de forma balanceada a média de acurácia obtida foi de 70%, sendo inferior a obtida para as classes desbalanceadas.

Analisando os resultados apresentados na Tabela 3.10, referente a média da acurácia, média AUC ROC e média AUC PR para as classes de forma balanceada, o classificador RF obteve um melhor resultado comparado ao classificador SVM, sendo esse resultado também visível na demonstração da Figura 3.13 e Figura 3.15. O resultado precário referente ao classificador SVM pode ter resultado do não ajuste dos parâmetros (utilizado os parâmetros proposto por (SILVA, 2017) para dados desbalanceados).

Já para as classes de forma desbalanceadas, o melhor resultado foi obtido pelo classificador SVM.

O tempo médio de classificação de uma tentativa, é aproximadamente igual independente dos classificadores ou organização das classes.

4 CONCLUSÃO

Este trabalho teve por objetivo mostrar que a metodologia proposta por (SILVA, 2017) poderia ser implementada em uma plataforma portátil (Orange Pi Zero), com a finalidade de utilização em interfaces cérebro-máquina baseadas em imagética motora.

A metodologia proposta aplicava o classificador sobre o conjunto de testes. Neste trabalho foi inclusa a etapa de simulação em tempo real, onde cada tentativa é predita uma a uma. Visando contribuir para o incremento da confiabilidade na detecção de intenções a ponto de permitir sua integração em um sistema BCI com aplicação prática.

Ambos classificadores obtiveram um melhor desempenho na performance ao fazer a predição de cada tentativa ao invés da predição de todo o conjunto de testes em uma única vez. Esses resultados são exibidos na Tabela 3.10.

O tempo médio de classificação obtido, para ambas as organizações de classes e ambos classificadores, foi aproximadamente igual. Desta forma, o tempo de execução não impacta na escolha de qual seria o melhor classificador a ser escolhido ou qual a melhor forma de organização das classes.

Em aplicações reais, as classes serão de forma desbalanceadas, sendo o classificador SVM quem obtém a melhor performance para esta organização dos dados.

Visando a integração da plataforma portátil em um sistema BCI para controlar uma cadeira de rodas, utilizando a metodologia de classes desbalanceadas com o classificador SVM, a aquisição do sinal EEG será realizada a cada 4s e o sinal de controle será gerado mais o menos 1,30 s após a imaginação da atividade motora, o que é considerado um tempo bom para suprir as necessidades de um cadeirante.

5 TRABALHOS FUTUROS

- Integração deste trabalho em um sistema BCI;
- Ajuste dos parâmetros do classificador SVM para classes balanceadas;
- Remoção de outliers (trials cujos valores dos atributos sejam muito maiores que os demais;
- Melhorar o tempo de processamos do código na plataforma portátil, o qual tem um resultado inferior ao de um computador;
- Propor modificações na metodologia visando a utilização de atributos no domínio *wavelet*, que melhor reflitam os comportamentos cerebrais para cada tarefa mental, conforme realizado por SILVEIRA (2016).

REFERÊNCIAS BIBLIOGRÁFICAS

AROCA, R. V. **Análise de Sistemas Operacionais de Tempo Real para Aplicações de Robótica e Automação**. 2008. Dissertação (Mestrado) — Universidade de São Paulo, São Carlos - SP, 2008.

BARBOSA, A. F. et al. Implementação de classificador de tarefas mentais baseado em eeg. In: ANAIS DO IX CONGRESSO BRASILEIRO DE REDES NEURAIS /INTELIGÊNCIA COMPUTACIONAL, IX., 2009, Ouro Preto. Minas Gerais, 2009.

BEAR, M. F.; CONNORS, B. W.; PARADISO, M. A. **Neurociências: Desvendando o Sistema Nervoso**. 2. ed. Porto Alegre: Artmed, 2002.

BREIMAN, L. Random forest. **Machine Learning Journal**, Hingham, v. 45, p. 5–32, 2001.

CICHOCKI, A.; ZHAO, Q. **EEG motor imagery dataset**. [S.l.: s.n.], 2011.

Compumedics NeuroScan. **Compumedics Neursocan**. 2017. Acessado em 02 dezembro 2017. Disponível em: <https://compumedicsneuroscan.com/products-overview/>.

DECETY, J. The neurophysiological basis of motor imagery. **Behavioural Brain Research**, n. 77, p. 45–52, 1996.

FARIAS, F. C. **Interface Cérebro-Máquina: Reconhecimento de Sinais Cerebrais Através de Reservoir Computing**. 2014. 51 f. Monografia (Trabalho de Conclusão de Curso) — Curso de Graduação em Engenharia de Computação, Escola Politécnica de Pernambuco, Recife - PE, 2014.

FARINES, J.-M.; FRAGA, J. da S.; OLIVEIRA, R. S. de. **Sistemas de Tempo Real**. Florianópolis: Departamento de Automação e Sistemas - Universidade Federal de Santa Catarina, 2000.

FATOURECHI, M. et al. Comparison of evaluation metrics in classification applications with imbalanced datasets. **Seventh International Conference on Machine Learning and Applications**, p. 777–782, 2008.

Felipe Martins dos Santos. **Aprendizado de Máquina Supervisionado com Python**. 2017. Acessado em 01 dezembro 2017. Disponível em: <https://iascblog.wordpress.com/2017/03/17/aprendizado-de-maquina-supervisionado-com-python/>.

Francesco Sacco. **Comunicação SPI - Parte 2**. Embarcados, 2014. Acessado em 01 de dezembro 2017. Disponível em: <https://www.embarcados.com.br/comunicacao-spi-parte-2/>.

g.tec. **g.tec medical engineering**. 2017. Acessado em 02 dezembro 2017. Disponível em: <http://www.gtec.at/Products>.

HO, T. K. Random decision forests. **IEEE. Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on**, p. 278–282, 1995.

http://what-when-how.com. 2017. Acessado em 30 novembro 2017. Disponível em: <http://what-when-how.com/wp-content/uploads/2012/04/tmp15F87.jpg>.

IBGE. **Censo demográfico: 2010: características gerais da população, religião e pessoas com deficiência**. 2010. Acessado em 25 novembro 2017. Disponível em: <https://biblioteca.ibge.gov.br/pt/biblioteca-catalogo?view=detalhes&id=794>.

JURCAK, V.; TSUZUKI, D.; DAN, I. 10/20, 10/10, and 10/5 systems revisited: Their validity as relative head-surface-based positioning systems. **NeuroImage**, p. 1600 –1611, 2007.

KEVRIC, J.; SUBASI, A. Comparison of signal decomposition methods in classification of eeg signals for motor-imagery bci system. **Biomedical Signal Processing and Control**, Elsevier, v. 31, p. 398–406, 2017.

KUNAL JAIN. **Um tutorial completo para aprender Data Science com Python do zero**. 2016. Acessado em 01 dezembro 2017. Disponível em: <https://www.vooo.pro/insights/um-tutorial-completo-para-aprender-data-science-com-python-do-zero/>.

LORENA, A. C.; CARVALHO, A. C. P. L. F. de. Uma introdução às support vector machines. **São Carlos - SP**, 2003.

MALMIVUO, J.; PLONSEY, R. **Biolectromagnetism - Principles and Applications of Bioelectric and Biomagnetic Filds**. New York: OXFORD UNIVERSITY PRESS, 1995.

MUSSATTO, G. G.; SILVA, S. de Avila e. Perspectivas e potencialidades da interface cérebro-máquina. **Revista de Sistemas de Informação da FSMA**, n. 13, p. 51–56, 2014.

NICOLAS-ALONSO, L. F.; GOMEZ-GIL, J. Brain computer interfaces, a review. **Sensors**, v. 12, n. 2, p. 1211 –1279, 2012.

NIEDERMEYER, E.; SILVA, F. L. da. **Electroencephalography: Basic Principles, Clinical Applications and Related Fields**. Baltimore MD: Lippincott Williams & Wilkins, 2005.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. **Journal of Machine Learning Research**, v. 12, p. 777–782, 2011.

Python Software Foundation. **O módulo pickle**. Modelos de Support Vector Machine, 2012. Acessado em 18 novembro 2017. Disponível em: <http://turing.com.br/pydoc/2.7/tutorial/inputoutput.html\#o-modulo-pickle>.

RAO, R. P. N. **Brain-computer interfacing: an introduction**. [S.l.]: Cambridge University Press, 2013.

RODRIGUES, E. C. et al. Efeito da estratégia de simulação mental sobre o controle postural. **Rev Bras Psiquiatr**, n. 25(Supl II), p. 33–5, 2003.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding machine learning: From theory to algorithms**. [S.l.]: Cambridge University Press, 2014.

SILVA, A. F. da. **Reconhecimento de Faces via PCA: Análise de Desempenho**. 2006. 141 p. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Uberlândia, Uberlândia, 2006.

SILVA, L. da. **Reconhecimento de padrões em sinais de EEG para detecção de intenção de movimento para aplicação em interface cérebro-máquina**. 2017. 58 f. Monografia (Trabalho de Conclusão de Curso) — Curso de Graduação em Engenharia de Controle e Automação, Universidade Federal de Santa Maria, Santa Maria, 2017.

SILVEIRA, J. de Ávila. **Análise de Sinais Cerebrais Utilizando Árvores de Decisão**. 2013. 103 p. Dissertação (Mestrado em Modelagem Computacional) — Universidade Federal do Rio Grande, Rio Grande, 2013.

SILVEIRA, T. **Classificação de estágios de sono através da aplicação de transformada wavelet discreta sobre um único canal de eletroencefalograma**. 2016. Dissertação (Mestrado - Tese (Doutorado)) — Universidade Federal de Santa Maria, Santa Maria, 2016.

Silvia Helena Cardoso. **Funções Especializadas do Córtex Cerebral**. Cérebro Mente, 2017. Acessado em 19 novembro 2017. Disponível em: <http://www.cerebromente.org.br/n01/arquitet/cortex.htm>.

Simone El Hage. **Frequências Mentais**. 2017. Acessado em 30 novembro 2017. Disponível em: <http://simoneelhage.com.br/frequencias-mentais/>.

SMOLA, A. J. Introduction to large margin classifiers. In: ____. [S.l.]: MorganKauffman, 1999. cap. 1, p. 1–28.

STECKLOW, M. V. **Imagética motora em tarefa complexa: Análise na banda alfa fo eletroencefalograma**. 2006. 96 f. Dissertação (Mestrado em Engenharia Biomédica) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2006.

TORTORA, G. J. **Corpo Humano**: Fundamentos de anatomia e fisiologia. Porto Alegre: Artmed, 2000.

VAPNIK, V. N. **The Nature of Statistical Learning Theory**. New York, NY, USA: Springer-Verlag New York, 1995.

VAZ, Y. **Extração de características para a classificação de imagética motora em interfaces cérebro-computador**. 2016. 111 f. Dissertação (Mestrado em Ciência de Computação e Matemática Computacional) — Universidade de São Paulo, São Paulo, 2016.

WHITTEN, I. H.; FRANK, E.; HALL, M. A. **Data mining: practical machine learning tools and techniques**. San Francisco: Morgan Kaufmann, 2011.

ANEXO A – *PRINCIPAL COMPONENT ANALYSIS*

Então, dada uma matriz MxN na qual os atributos em questão estão colocadas nas N colunas, e cada uma das M linhas corresponde a uma amostra do atributo representado pela coluna na qual se encontra, cuja representação pode ser dada da seguinte forma:

$$\vec{X} = [\vec{x_1}\vec{x_2}\dots\vec{x_N}] \tag{A.1}$$

$$\vec{x_i} = \begin{bmatrix} x_i(1) \\ x_i(2) \\ \vdots \\ x_i(M) \end{bmatrix} \tag{A.2}$$

Em geral, cada amostra dos atributos é um vetor no espaço vetorial N-dimensional, onde N é o número de atributos que estamos analisando. Esse espaço vetorial é gerado por uma base ortonormal de vetores. Todas as amostras são combinações lineares dessa base de vetores de comprimento unitário (SHLENS, 2014). Um exemplo simples dessa base $\vec{B}$ poderia ser a matriz identidade $\vec{I}$.

$$\vec{B} = \begin{bmatrix} \vec{b_1} \\ \vec{b_2} \\ \vdots \\ \vec{b_N} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} = \vec{I} \tag{A.3}$$

onde cada linha é um vetor base $\vec{b_1}$, e tem N componentes. Para fins de explicação a representação é mostrada abaixo:

$$\vec{X'} = \vec{B}\vec{X} = \begin{bmatrix} \vec{b_1} \\ \vec{b_2} \\ \vdots \\ \vec{b_N} \end{bmatrix} [\vec{x_1}\vec{x_2}\dots\vec{x_N}] = \begin{bmatrix} \vec{b_1}\vec{x_1} & \vec{b_1}\vec{x_2} & \cdots & \vec{b_1}\vec{x_N} \\ \vec{b_2}\vec{x_1} & \vec{b_2}\vec{x_2} & \cdots & \vec{b_2}\vec{x_N} \\ \vdots & \vdots & & \vdots \\ \vec{b_N}\vec{x_1} & \vec{b_N}\vec{x_2} & \cdots & \vec{b_N}\vec{x_N} \end{bmatrix} \tag{A.4}$$

Como a base ortonormal é a matriz identidade:$\vec{X'} = \vec{X}$.

A PCA faz uma rigorosa porém poderosa suposição: a linearidade. A partir dessa suposição é possível simplificar o problema ao restringir o conjunto de potenciais bases. Define-se $\vec{Y}$ como a re-representação dos dados e $\vec{P}$ como a transformação linear que irá transformar $\vec{X}$ em $\vec{Y}$.

$$\vec{Y} = \vec{P}\vec{X}^T = \vec{P}_{[NxN]}\vec{X^T}_{[NxM]} \tag{A.5}$$

$$\vec{Y} = \begin{bmatrix} \vec{p_1} \\ \vec{p_2} \\ \vdots \\ \vec{p_N} \end{bmatrix} [\vec{x_1}\vec{x_2}\ldots\vec{x_N}] = \begin{bmatrix} \vec{p_1}\vec{x_1} & \vec{p_1}\vec{x_2} & \cdots & \vec{p_1}\vec{x_N} \\ \vec{p_2}\vec{x_1} & \vec{p_2}\vec{x_2} & \cdots & \vec{p_2}\vec{x_N} \\ \vdots & \vdots & & \vdots \\ \vec{p_N}\vec{x_1} & \vec{p_N}\vec{x_2} & \cdots & \vec{p_N}\vec{x_N} \end{bmatrix} \tag{A.6}$$

Pode-se notar a forma de cada coluna de $\vec{Y}$

$$y_i = \begin{bmatrix} \vec{p_1}\vec{x_i} \\ \vdots \\ \vec{p_N}\vec{x_i} \end{bmatrix} \tag{A.7}$$

É possível reconhecer que cada coeficiente y_i é o produto escalar de x_i com cada linha correspondente de $\vec{P}$. Em outras palavras, o j-ésimo coeficiente de y_i é uma projeção de x_i na j-ésima linha de $\vec{P}$. As linhas de $\vec{P}$ são, de fato, o novo conjunto de vetores ortonormais que formarão a base para representar as colunas de $\vec{X}$ (SHLENS, 2014). Como assumimos linearidade no problema, encontrar as componentes principais se resume, na verdade, a uma mudança de base apropriada.

A fim de mostrar as redundâncias que podem ocorrer nos nossos dados é calculado o quanto os atributos em análise variam um em relação aos outros. É nessa análise de variância que é baseada a escolha para a melhor base de vetores.

Como a matriz de covariância é simétrica, por resultados de álgebra linear (BOLDRINI et al., 1980) sabe-se que existe uma base ortonormal de autovetores desta matriz que pode ser ortogonalizada da seguinte forma: $C = PDP^T$, sendo D a matriz diagonal que contém apenas os autovalores de C, e P a matriz dos autovetores associados. A matriz de transformação P é utilizada para expressar os dados na base de autovetores. Essa matriz P é a matriz com os autovetores da matriz de covariância nas colunas. Mais informações sobre a PCA podem ser encontradas em (CASTELLS et al., 2007; SHLENS, 2014; SMITH, 2002).

A operação de “branqueamento” (*whitening*) toma os dados na base de autovetores e divide cada dimensão pelo autovalor a fim de normalizar a escala. A interpretação geométrica dessa transformação é que os dados de entrada são gaussianos multivariáveis, e os dados *whitened* serão gaussianos com média zero e matriz de covariância igual à identidade.