**universidade do vale do Itajaí**

ESCOLA DO MAR, CIÊNCIA E TECNOLOGIA

curso de Engenharia de Computação

**Classificação on-line de emoções usando sinais de eletroencefalografia**

por

Kevin Weiss

Itajaí (SC), julho de 2020

**universidade do vale do Itajaí**

ESCOLA DO MAR, CIÊNCIA E TECNOLOGIA

curso de Engenharia de Computação

**Classificação on-line de emoções usando sinais de eletroencefalografia**

Área de sistemas embarcados

por

Kevin Weiss

Relatório apresentado à Banca Examinadora do Trabalho Técnico-científico de Conclusão de Curso de Engenharia de Computação para análise e aprovação.

Orientador(a): Alejandro Rafael Garcia Ramirez, Doutor

Itajaí (SC), julho de 2020

Resumo

A computação afetiva, uma área de pesquisa em constante desenvolvimento, tem objetivos de integração de emoções à tecnologia. Para que essa integração seja possível, métodos variados para o reconhecimento de emoções foram desenvolvidos. Dentro dos métodos mais utilizados, como voz e imagens, a eletroencefalografia (EEG) se apresenta como um método promissor por ter vantagens únicas em relação aos outros métodos. Por esse motivo, existe bastante interesse no desenvolvimento de sistemas que possam reconhecer emoções por meio de sinais de EEG. Porém, grande parte das pesquisas já realizadas nesse âmbito se limitam a resultados off-line, que consequentemente limitam o escopo de aplicações beneficiadas. Com essa perspectiva, este trabalho faz um estudo sobre a constituição das emoções e como são computacionalmente modeladas. O estudo realizado também aborda as técnicas utilizadas para o processamento de sinais de EEG, e algoritmos utilizados para classificação. Com essa fundamentação teórica, um sistema de reconhecimento de emoções em tempo real é proposto. O sistema tem como objetivo reconhecer as três principais categorias de emoções: positivas, negativas e neutras. O sistema foi desenvolvido e validado com *datasets* padronizados e internacionalmente utilizados pela comunidade científica.

Palavras-chave: Reconhecimento de emoções. EEG. Computação Afetiva. Tempo Real. BCI.

Abstract

Affective computing, a constantly evolving area of ​​research, aims to integrate emotions with technology. To make this integration possible, various methods for emotion recognition have been developed. Among the most used methods, such as voice and images, electroencephalography (EEG) presents itself as a promising method because it has unique advantages over other methods. For this reason, there is a lot of interest in developing systems that can recognize emotions through EEG signals. However, much of the research already done in this area is limited to offline results, which consequently limit the scope of benefited applications. With this perspective, this paper makes a study about the constitution of emotions and how they are computationally modeled. The study also addresses the techniques used for processing EEG signals, and algorithms used for classification. With this theoretical overview, a real-time emotion recognition system is proposed. The system aims to recognize the three main categories of emotions: positive, negative, and neutral. The system was developed and validated with internationally standardized datasets used by the scientific community.

Keywords: Emotion recognition. EEG. Affective Computing. Real time. BCI.

Lista de Figuras

[Figura 1. Teorias clássicas da origem da emoção 18](#_Toc45746587)

[Figura 2. Modelo PAD 20](#_Toc45746588)

[Figura 3. Modelo que mapeia as emoções em duas dimensões, valência (valence) e excitação (arousal). 21](#_Toc45746589)

[Figura 4. Esquemático alternativo do Modelo Circumplexo, que demonstra a existência de ambiguidade na categorização das emoções, exibido pela sobreposição das emoções no plano. 22](#_Toc45746590)

[Figura 5. Roda de emoções 24](#_Toc45746591)

[Figura 6. Díades de emoções 25](#_Toc45746592)

[Figura 7. Músculos utilizados na eletromiografia facial para detectar emoções 30](#_Toc45746593)

[Figura 8. Exemplos de medidas PPG de um usuário enquanto intencionalmente expressava raiva (esquerda) e tristeza (direita). 31](#_Toc45746594)

[Figura 9. fMRI durante tarefas de memória de trabalho 31](#_Toc45746595)

[Figura 10. Cortex Cerebral 32](#_Toc45746596)

[Figura 11. Sistema internacional 10-20 34](#_Toc45746597)

[Figura 12. Duas componentes principais sobre duas dimensões correlacionadas 39](#_Toc45746598)

[Figura 13. Uma função não convexa, com múltiplas mínimas locais, onde a derivada da função é zero. 42](#_Toc45746599)

[Figura 14. Diagrama de blocos do sistema 48](#_Toc45746600)

[Figura 15. Casos de uso 50](#_Toc45746601)

[Figura 16. Diagrama de sequência 52](#_Toc45746602)

[Figura 17. Um exemplo de Jupyter notebook. Note como permite inserir texto e renderização de formulas para fins de documentação, como também a possibilidade plotar gráficos de forma *inline,* sem ter que abrir uma nova janela. 56](#_Toc45746603)

[Figura 18. Estrutura de dados após leitura do *dataset*. 60](#_Toc45746604)

[Figura 19. Mecanismo de geração de características e predição online 60](#_Toc45746605)

[Figura 20. Validação cruzada *K-Fold* 66](#_Toc45746606)

[Figura 21. Validação cruzada aninhada. 67](#_Toc45746607)

[Figura 22. Interface gráfica do OpenBCI 72](#_Toc45746608)

[Figura 23. Interface gráfica do widget criado. 73](#_Toc45746609)

[Figura 24. Tempos de execução em um experimento 76](#_Toc45746610)

Lista de Quadros

[Quadro 1. Diferenciação entre os diferentes fenômenos das áreas de ciência afetiva e neurociência afetiva 18](#_Toc45748624)

[Quadro 2. Emoções básicas codificadas no sistema FACS 27](#_Toc45748625)

[Quadro 3. Definição dos prefixos do sistema 10-20 32](#_Toc45748626)

[Quadro 4. Diferentes bandas de sinais de EEG 33](#_Toc45748627)

[Quadro 5. Níveis de valência 43](#_Toc45748628)

[Quadro 6. Requisitos Funcionais 48](#_Toc45748629)

[Quadro 7. Requisitos não-funcionais 48](#_Toc45748630)

[Quadro 10. Leitura do *dataset* 58](#_Toc45748631)

[Quadro 11. Cálculo da Densidade Espectral 60](#_Toc45748632)

[Quadro 12. Pre-processamento e geração de características 61](#_Toc45748633)

[Quadro 13. Modelos e hiperparâmetros 64](#_Toc45748634)

[Quadro 14. Modelos, hiperparâmetros e validação cruzada 67](#_Toc45748635)

[Quadro 15. Classe *EstimatorSelectionHelper* 68](#_Toc45748636)

[Quadro 16. Tamanho de janela e sobreposição 69](#_Toc45748637)

[Quadro 17. Melhores modelos por usuário 70](#_Toc45748638)

[Quadro 18. Comunicação entre contextos de execução de Java e Python. 73](#_Toc45748639)

[Quadro 19. Servidor TCP no Python 73](#_Toc45748640)

[Quadro 20. Medindo tempo de execução 74](#_Toc45748641)

[Quadro 21. Geração de Características 82](#_Toc45748642)

[Quadro 22. Geração de características de várias janelas e sobreposições 86](#_Toc45748643)

[Quadro 23. Classe RepeatedStratifiedGroupKFold 87](#_Toc45748644)

[Quadro 24. Cálculo de Densidade Espectral 88](#_Toc45748645)

[Quadro 25. Modelos e parâmetros 89](#_Toc45748646)

[Quadro 26. Classe EstimatorSelectionHelper 90](#_Toc45748647)

[Quadro 27. Treino de modelos sujeito dependente 92](#_Toc45748648)

Lista de Equações

[Equação 1 35](#_Toc45748649)

[Equação 2 35](#_Toc45748650)

[Equação 3 36](#_Toc45748651)

[Equação 4 36](#_Toc45748652)

[Equação 5 36](#_Toc45748653)

[Equação 6 36](#_Toc45748654)

[Equação 7 37](#_Toc45748655)

[Equação 8 37](#_Toc45748656)

[Equação 9 37](#_Toc45748657)

[Equação 10 39](#_Toc45748658)

[Equação 11 39](#_Toc45748659)

[Equação 12 39](#_Toc45748660)

[Equação 13 40](#_Toc45748661)

[Equação 14 40](#_Toc45748662)

[Equação 15 40](#_Toc45748663)

[Equação 16 41](#_Toc45748664)

[Equação 17 41](#_Toc45748665)

[Equação 18 42](#_Toc45748666)

Sumário

[1 Introdução 10](#_Toc45748904)

[1.1 Problematização 11](#_Toc45748905)

[1.2 Formulação do Problema 11](#_Toc45748906)

[1.2.1 Solução Proposta 12](#_Toc45748907)

[1.2.2 Justificativa 12](#_Toc45748908)

[1.3 Objetivos 13](#_Toc45748909)

[1.3.1 Objetivo Geral 13](#_Toc45748910)

[1.3.2 Objetivos Específicos 13](#_Toc45748911)

[1.4 Metodologia 14](#_Toc45748912)

[1.5 Estrutura do Trabalho 14](#_Toc45748913)

[2 Fundamentação Teórica 15](#_Toc45748914)

[2.1 Téoria das emoções 15](#_Toc45748915)

[2.2 Origem das emoções e definições 17](#_Toc45748916)

[2.3 Modelos de emoções 18](#_Toc45748917)

[2.3.1 Modelo PAD 18](#_Toc45748918)

[2.3.2 Modelo Circumplexo 19](#_Toc45748919)

[2.3.3 PAD versus Circumplexo 21](#_Toc45748920)

[2.3.4 Roda de Emoções 22](#_Toc45748921)

[2.4 Reconhecimento de emoções 24](#_Toc45748922)

[2.4.1 Processos visuais 25](#_Toc45748923)

[2.4.2 Processos auditivos 25](#_Toc45748924)

[2.4.3 Processos olfativos 25](#_Toc45748925)

[2.5 Reconhecimento automatizado de emoções 26](#_Toc45748926)

[2.5.1 Reconhecimento de emoções por Voz 26](#_Toc45748927)

[2.5.2 Reconhecimento de emoções por Imagens 26](#_Toc45748928)

[2.5.3 Reconhecimento de emoções por texto 27](#_Toc45748929)

[2.5.4 Reconhecimento de emoções por GSR 28](#_Toc45748930)

[2.5.5 Reconhecimento de emoções por EMG 28](#_Toc45748931)

[2.5.6 Reconhecimento de emoções por PPG 29](#_Toc45748932)

[2.5.7 Reconhecimento de emoções por fMRI 30](#_Toc45748933)

[2.5.8 Reconhecimento de emoções por EEG 31](#_Toc45748934)

[2.5.9 Comparação de EEG com outros métodos 34](#_Toc45748935)

[2.6 Processamento de sinais 35](#_Toc45748936)

[2.6.1 Análise de Fourier 35](#_Toc45748937)

[2.6.2 Transformada Discreta de Fourier 36](#_Toc45748938)

[2.6.3 Transformada rápida de Fourier 37](#_Toc45748939)

[2.7 Redução de dimensionalidade 38](#_Toc45748940)

[2.7.1 Análise de componentes principais 38](#_Toc45748941)

[2.8 Aprendizado de máquina 39](#_Toc45748942)

[2.8.1 Regressão Linear 39](#_Toc45748943)

[2.8.2 Regressão Logística 40](#_Toc45748944)

[2.8.3 Redes Neurais Artificiais 42](#_Toc45748945)

[2.9 Trabalhos relacionados 43](#_Toc45748946)

[2.9.1 Real-time EEG-based user's valence monitoring 43](#_Toc45748947)

[2.9.2 CogniMeter: EEG-based Emotion, Mental Workload and Stress Visual Monitoring 44](#_Toc45748948)

[2.9.3 EEG based patient emotion monitoring using relative wavelet energy feature and Back Propagation Neural Network 44](#_Toc45748949)

[2.9.4 Real-Time Movie-Induced Discrete Emotion Recognition from EEG Signals 45](#_Toc45748950)

[2.9.5 Analise comparativa 46](#_Toc45748951)

[3 Projeto 47](#_Toc45748952)

[3.1 Diagrama de blocos 47](#_Toc45748953)

[3.2 Requisitos funcionais 48](#_Toc45748954)

[3.3 Requisitos não-funcionais 48](#_Toc45748955)

[3.4 Casos de uso 48](#_Toc45748956)

[3.5 Diagrama de sequência 50](#_Toc45748957)

[3.6 Plano de testes 52](#_Toc45748958)

[4 Desenvolvimento 53](#_Toc45748959)

[4.1 Análise de datasets 53](#_Toc45748960)

[4.1.1 DEAP: A Database for Emotion Analysis using Physiological Signals 53](#_Toc45748961)

[4.1.2 MAHNOB HCI 53](#_Toc45748962)

[4.1.3 SEED IV 53](#_Toc45748963)

[4.1.4 Comparações e escolha de dataset 54](#_Toc45748964)

[4.2 ferramentas e ambiente de prototipagem 54](#_Toc45748965)

[4.2.1 Jupyter Notebook 54](#_Toc45748966)

[4.2.2 Bibliotecas em Python para computação científica 55](#_Toc45748967)

[4.3 Tratamento do dataset 57](#_Toc45748968)

[4.4 Geração de características 59](#_Toc45748969)

[4.5 Treinamento de algoritmos 62](#_Toc45748970)

[4.5.1 Considerações de organização do dataset 62](#_Toc45748971)

[4.5.2 Modelos e hiperparâmetros 63](#_Toc45748972)

[4.5.3 Validação cruzada 64](#_Toc45748973)

[4.5.4 Resultados e metricas de desempenho 69](#_Toc45748974)

[4.6 Desenvolvimento de software 70](#_Toc45748975)

[4.6.1 Software OpenBCI 71](#_Toc45748976)

[4.6.2 Desenvolvimento do widget no OpenBCI 72](#_Toc45748977)

[4.6.3 Comunicação entre Java e Python 73](#_Toc45748978)

[4.7 Medidas de Execução 74](#_Toc45748979)

[5 Conclusões 76](#_Toc45748980)

[Apêndice A. Códigos-fonte 82](#_Toc45748981)

1. Introdução

Uma interface cérebro-computador (BCI - *Brain-Computer Interface*), resumidamente, é um sistema onde é realizado a captação de sinais elétricos provenientes da atividade neural do cérebro. Os sinais captados são processados e classificados para o controle de uma aplicação (RAO, 2011).

Para a captação dos sinais cerebrais, as principais técnicas utilizadas são a invasiva e a não invasiva. O método invasivo envolve a inserção de eletrodos diretamente sobre o cérebro, esta implantação implica na realização de operações cirúrgicas que podem introduzir uma série de vantagens e desvantagens. O dispositivo inserido diretamente no cérebro consegue captar melhor qualidade de sinal, o qual permite melhor processamento e controle da aplicação externa. Porém, além dos riscos envolvidos na cirurgia, a implantação de dispositivos alheios ao cérebro humano pode causar o desenvolvimento de cicatrizes na área onde foi realizada a implantação. No caso de sistemas BCI, estas cicatrizes no cérebro podem enfraquecer ou ocultar por completo o sinal desejado (WILSON, 2013).

Em contraste, o método não invasivo é considerado seguro, pois os eletrodos são alocados na parte externa ao crânio, acima do couro cabeludo, sem a necessidade de uma operação cirúrgica. Assim, uma das grandes vantagens do método não invasivo é o fácil manuseio, já que a preparação e implantação é facilitada em comparação com o método invasivo. Porém, o método não invasivo tem uma desvantagem na qualidade do sinal em comparação com o método invasivo. Essa limitação se deve ao fato de que o sinal captado pode atravessar várias camadas do cérebro, incluindo o crânio, que consequentemente pode causar a distorção ou até a perda do sinal.

Devido as características descritas dos métodos invasivos e não invasivos, a grande maioria dos trabalhos e pesquisas realizadas de sistemas BCI utilizam o método não invasivo que também será o método utilizado neste trabalho.

Há diversas áreas de aplicação para sistemas BCI que estão sendo incluídos cada vez mais ao nosso cotidiano. Uma das principais motivações do uso desses sistemas é na área médica, especificamente, na restauração sensorial e da movimentação motora. Mais recentemente, outras áreas de aplicação estão sendo exploradas, abrangendo entretenimento, reconhecimento biométrico e educação (RAO, 2011). Uma das áreas de pesquisa de sistemas BCI tem como foco o reconhecimento da emoção humana, que é o tema de pesquisa deste trabalho. O reconhecimento de emoções faz parte do ramo de pesquisa da Computação Afetiva, o qual tem como objetivo melhorar a interação entre humanos e dispositivos, com intuito de aprimorar a usabilidade e experiência do usuário. Dentro deste contexto, este trabalho procura fazer uma contribuição na área de reconhecimento de emoções, através da análise e desenvolvimento de um sistema para classificação de sinais de eletroencefalografia (EEG).

* 1. Problematização

Nesta seção é apresentado os desafios envolvidos em sistemas BCI, com uma perspectiva comparativa em relação as formas atuais de reconhecimento de emoções. Em seguida, é apresentada a solução a ser desenvolvida para resolver os problemas levantados.

* 1. Formulação do Problema

O reconhecimento da emoção humana já é alvo de pesquisa com abordagens variadas. Um dos métodos realiza o reconhecimento facial através de processamento de imagens cinematográficas. Nas pesquisas realizadas a expressão facial é utilizada para identificar certas emoções de um indivíduo (LOZANO-MONASOR et al., 2014). Outra abordagem se baseia na classificação de padrões usando entonação da voz, onde a emoção de um indivíduo é identificada pela sua fala (KOOLAGUDI; RAO, 2012). Mesmo tendo sucesso ao reconhecer emoções, estes métodos apresentam uma limitação inerente em comum que podem comprometer um sistema. Nos métodos mencionados, a autenticidade das emoções é facilmente comprometida por uma simulação. Esta realidade afeta negativamente aplicações que dependem de um sentimento legitimo por parte do usuário. Por exemplo, testes de usabilidade e experiência do usuário são afetados por esse problema (RACHED; PERKUSICH, 2012).

Dentro das pesquisas realizadas no âmbito de sistemas BCI para detecção de emoções, grande parte são processados de forma off-line, isto é, resultados são obtidos após o processamento, sem uma perspectiva de processar em tempo real. Consequentemente, a abordagem off-line apresenta um problema de uso prático da tecnologia, o qual restringe seu uso para um número limitado de aplicações.

* + 1. Solução Proposta

Implementar um sistema de classificação de sinais de EEG para o reconhecimento de emoções em tempo real. O sistema deverá identificar três categorias de emoções: emoções positivas, negativas e neutras. Não será objeto desta pesquisa a interação com um hardware BCI para validação do sistema, esta validação será realizada via *datasets*. Será desenvolvido um software com interface de usuário que irá permitir a configuração de uma sessão de classificação, como também visualizar resultados de classificação em tempo real.

* + 1. Justificativa

Com o crescimento do número de pesquisas realizadas com sinais de EEG, e com a disponibilidade de dispositivos de captação de menor custo, novos programas e projetos de pesquisa vêm sendo cogitadas.

Em parceria com o Prof. Dr. Alejandro Ramírez, o doutorando Tiago Catecati da Universidade do Estado de Santa Catarina (UDESC) realizou um trabalho de pesquisa para avaliar a satisfação do usuário por meio de sinais de EEG. Nesse trabalho, é discutido a utilização de ferramentas verbais e questionários para mensurar a emoção do usuário durante a utilização de um software. Apesar de serem os mais utilizados, o trabalho apresenta as limitações inerentes dessas abordagens, e demonstra como o uso de sinais psicofisiológicos, como a EEG, podem ajudar a preencher lacunas deixadas pelos métodos manuais de *feedback* (CATECATI, 2018).

No trabalho de dissertação de mestrado do Marcelus Oldoni da Universidade do Vale de Itajaí (UNIVALI), foi desenvolvido um sistema de interface cérebro-computador para a mensuração da memória de trabalho para auxiliar profissionais no tratamento de pacientes afásicos. Apesar de encontrar resultados promissores para o estudo, os resultados de experimentos levavam alguns dias até serem recebidos pelos profissionais que realizavam o tratamento, devido a motivos de processamento off-line e disponibilidade dos profissionais envolvidos. Sendo assim, a quantidade de experiências realizadas e resultados retornados pelos profissionais foi realizada com pouca frequência (OLDONI et al., 2019).

Desta forma, para que uma pesquisa possa viabilizar um tratamento mais eficiente, o processamento e interpretação dos sinais do experimento precisa ser apresentado de forma on-line. Isso permite um retorno mais ágil do profissional envolvido no tratamento, que consequentemente pode tomar decisões para realizar o melhoramento contínuo do mesmo.

Logo, ambos os trabalhos apresentados contribuem para a justificativa desta pesquisa. O trabalho de CATECATI (2018) demonstra a importância do estudo de emoções a partir de sinais de EEG, com uma aplicação para o estudo da satisfação do usuário. Já o trabalho de OLDONI et al. (2019), demonstra indiretamente a importância de um sistema on-line, por meio das experiências realizadas e resultados adquiridos em parceria com profissionais que tratavam de pacientes afásicos.

Vale ressaltar que os temas abordados neste trabalho trazem conteúdos diversos, combinando disciplinas de engenharia, ciência da computação, neurociência, psicologia e outros. Portanto, há uma série de desafios envolvidos para a realização deste trabalho, tendo em vista a necessidade multidisciplinar do mesmo.

* 1. Objetivos

Nesta seção são apresentados os objetivos específicos e geral do Trabalho de Conclusão de Curso. O objetivo geral apresenta o objetivo central deste projeto. Os objetivos específicos detalham os passos que serão seguidos para alcançar o objetivo geral.

* + 1. Objetivo Geral

Desenvolver um sistema para a classificação de emoções em tempo real, a partir do processamento de sinais de Eletroencefalografia (EEG).

* + 1. Objetivos Específicos

1. Selecionar os algoritmos de classificação e extração de características direcionados para a classificação de sinais de EEG.
2. Desenvolver o software com interface de usuário para configuração e controle de uma sessão de classificação;
3. Implementar um sistema de classificação em tempo real para identificar as emoções positivas, negativas e neutras, a partir de sinais de EEG disponibilizadas em *datasets*;
4. Avaliar os resultados frente aos trabalhos relacionados.
   1. Metodologia

A metodologia a ser utilizada é a hipotética dedutiva, já que o tema sendo pesquisado irá seguir uma linha de várias iterações de análise, levantamento de hipóteses, experimentação e validação das hipóteses. Primeiramente, foi realizado uma pesquisa bibliográfica para estabelecer os fundamentos necessários para a pesquisa. No âmbito de pesquisa científica sobre emoções, é comum a utilização de bibliografias com datas mais antigas, já que este ramo cresce mais lentamente em comparação com o setor tecnológico. Em contrapartida, para a pesquisa bibliográfica sobre algoritmos e sinais de EEG, publicações mais recentes foram revisadas.

Uma revisão dos trabalhos mais recentes e relevantes ao tema foi realizado, a fim de familiarizar-se com as técnicas e algoritmos já existentes. Esta revisão também irá servir para a comparação dos resultados adquiridos entre este trabalho e os trabalhos correlatos. Após a revisão bibliográfica serão implementados os algoritmos e verificadas as métricas de desempenho.

* 1. Estrutura do Trabalho

Este trabalho está estruturado em quatro capítulos. O Capítulo 1 faz uma introdução ao tema de sistemas BCI, diferenciando dos diferentes métodos utilizados em pesquisas. Também é realizado uma introdução ao tema da computação afetiva, apresentando seus métodos e objetivos. O Capítulo 2 realiza a fundamentação teórica do trabalho, fazendo um estudo aprofundado dos modelos das emoções, motivações de uso e origens. O Capítulo 2 também aborda vários conceitos de processamento de sinais e algoritmos de classificação, como também uma análise comparativa dos trabalhos relacionados. O Capítulo 3 exibe o projeto de software e suas especificações com quadros de requisitos e diagramas. O Capítulo 4 apresenta as considerações finais do trabalho.

1. Fundamentação Teórica
   1. Téoria das emoções

Emoções, uma característica intrínseca do ser humano, representam a habilidade de experienciar de forma subjetiva, certos estados do sistema nervoso (PANKSEPP, 2004). A emoção acaba sendo um termo que abrange uma variedade complexa de fenômenos relacionados. No entanto, devido à natureza subjetiva e pessoal dos estados emocionais, muitas vezes é difícil para que um indivíduo descreva com precisão a intensidade e a natureza do efeito.

Em sua forma básica, emoções são respostas específicas e consistentes a um evento interno ou externo significativo com duração curta e consistem em um padrão particular de respostas fisiológicas. Apesar que não existe atualmente uma definição formal aceita pela comunidade cientifica, a emoção pode ser definida como uma experiência positivo ou negativo associada a um padrão particular de atividade fisiológica (SCHACTER et al., 2016).

Na procura de maior compreensão na formação de emoções, várias teorias foram desenvolvidas sobre sua origem. Uma das primeiras teorias, chamada de James-Lange, diz que as emoções são um efeito que ocorre após de uma atividade no sistema nervoso autônomo, que são por sua vez ativados por algum estímulo. Ou seja, um estímulo que desencadeia uma atividade no sistema nervoso autônomo produz certas emoções no cérebro humano.

Para exemplificação de uma teoria de emoção, uma cena comumente utilizada na literatura é o caso no encontro inesperado com um predador, por exemplo, um urso: a princípio, um pode pensar que o primeiro a acontecer é o sentimento de medo, que se seguiria com o batimento cardíaco acelerado com uma tensão muscular nas pernas para se posicionar para uma corrida. Porém, segundo a teoria de James-Lange, a ordem dos acontecimentos fisiológicos e emocionais estão na ordem inversa, sendo assim, primeiro haveria a aceleração do batimento cardíaco, que se seguiria de uma contração muscular das pernas para escapar do perigo, que finalmente se seguiria do sentimento de medo. Portanto, segundo a teoria de James-Lange, a experiência emocional é uma consequência, e não uma causa, das reações fisiológicas de eventos externos (SCHACTER; GILBERT; WEGNER, 2013).

Em contrapartida, a teoria de Cannon-Bard sugere que a atividade do sistema nervoso autônomo e a emoção experienciada ocorrem de forma simultânea, devido a existência de uma série de emoções que não seguem a teoria de James-Lange. Por exemplo, algumas emoções ocorrem momentos antes de uma reação autonômica, que vai contra a teoria James-Lange. Outro fator que vai contra a teoria de James-Lange, é o estímulo não emocional, que pode provocar a mesma atividade fisiológica que um estímulo emocional causaria, como o calor por exemplo. Para complementar sua teoria, Cannon demonstrou que não havia variação suficiente de padrões de atividade fisiológica autonômica para todas as diferentes emoções experienciadas pelo ser humano (SCHACTER; GILBERT; WEGNER, 2013).

Outra teoria, desenvolvido por Stanley Schachter e Jerome Singer, chamada de teoria de Dois Fatores, trabalha em cima das teorias de James-Lange e Cannon-Bard. Sem discordar com elas, a teoria de Dois Fatores apresenta a ideia de que a emoção se trata de um processo de inferência lógica realizada pelo cérebro, após da compreensão das causas da atividade fisiológica.

Utilizando o mesmo exemplo de um encontro inesperado com um predador, a teoria de Dois Fatores relata que após avistar o predador, haverá uma leve aceleração no batimento cardíaco, que fará com que o cérebro realize uma interpretação da situação. Após de compreender que há um predador a cerca e que há uma aceleração cardíaca, o cérebro fará uma inferência lógica da situação, que consequentemente irá produzir uma experiência emocional de temor. De forma semelhante, caso haja a mesma reação fisiológica, porém em um contexto alegre, o cérebro irá produzir uma emoção positiva. De forma resumida, A Figura 1 demonstra a sequência das reações das três teorias discutidas.

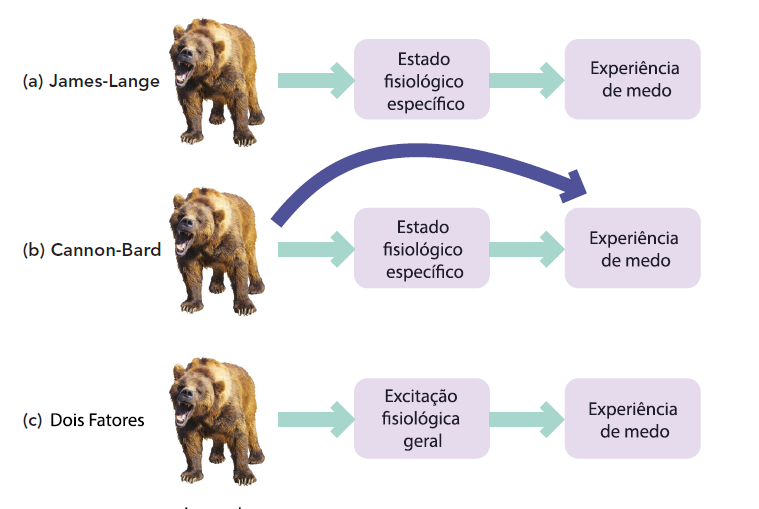


Figura 1. Teorias clássicas da origem da emoção

Fonte: Adaptado de Schacter et al. (2016)

* 1. Origem das emoções e definições

Nos últimos anos, houve um crescimento significativo em pesquisas da área de neurociência e psicologia direcionadas à experiência das emoções, sua origem, e como são desenvolvidas no cérebro humano. Porém, devida a incerteza da natureza das emoções, ainda há pouco consenso no campo da Computação Afetiva em relação aos fundamentos das emoções. Atualmente, há duas proposições que são mais amplamente reconhecidas.

A primeira delas assume que as emoções são biologicamente integradas e governadas por regiões de circuitos neurais inerentes ao cérebro. Essa perspectiva sugere que a natureza concedeu uma série de emoções básicas, que são reconhecidas e experienciadas de forma universal. As emoções básicas universalmente reconhecidas são felicidade, medo, desgosto, raiva e medo. Estas emoções são facilmente reconhecidas, e são consideradas as emoções que existem na natureza de forma “natural”, independente de percepção delas.

Já a segunda hipótese propõe, que, emoções representam uma coordenação entre vários componentes fisiológicos, psicológicos e comportamentais que trabalham em conjunto com objetivo principal a sobrevivência do organismo. Esta hipótese, trabalha no conceito que estes componentes são sincronizados momentaneamente em resposta a algum evento que seja de alta relevância ao indivíduo (FOX, 2018).

Devido a subjetividade da disciplina, a pesquisa cientifica das emoções sofre do fato de que vários fenômenos e expressões são estudados de forma sinônima. Termos como emoção, ânimo e sentimento são utilizados de forma indiferente, apesar que são pesquisados em subcampos diferentes. Para obter uma distinção clara dos termos, e uma melhor ramificação da representação das pesquisas na área devido à natureza multidisciplinar da ciência afetiva, uma definição para cada um dos termos é apresentada no Quadro 1. Mesmo sem o consenso geral das definições apresentadas, estas ajudam a delimitar o escopo da definição do estudo das emoções.

Quadro . Diferenciação entre os diferentes fenômenos das áreas de ciência afetiva e neurociência afetiva

|  |  |  |  |
| --- | --- | --- | --- |
| **Termo** | **Duração típica** | **Ocasionador comum** | **Função** |
| Emoções | Segundos a minutos | Objeto ou situação específica | Influenciar ações |
| Ânimo | Horas, dias ou semanas | Sistemas fisiológicos internos e fatores do meio ambientes externo | Influenciar cognição |
| Sentimentos | Variável, provavelmente de curto prazo | Ativação de sistema fisiológico interno | Trazer consciência |

Fonte: Adaptado de FOX (2018)

* 1. Modelos de emoções
     1. Modelo PAD

Devido a necessidade de quantificar as emoções, pesquisadores da área criaram modelos visuais para poder conceituar e mensurar as emoções. Um modelo desenvolvido por Albert Mehrabian e James A. Russell, chamado do modelo PAD (Pleasure, Arousal and Dominance), demonstra o mapeamento das emoções em um espaço tridimensional, traçado por três eixos independentes, denominados de Prazer, Excitação e Dominância. O modelo PAD foi originalmente desenvolvido para o estudo da psicologia ambiental, que tem como objetivo compreender como os ambientes podem influenciar o comportamento e emoções de indivíduos (MEHRABIAN, 1980).

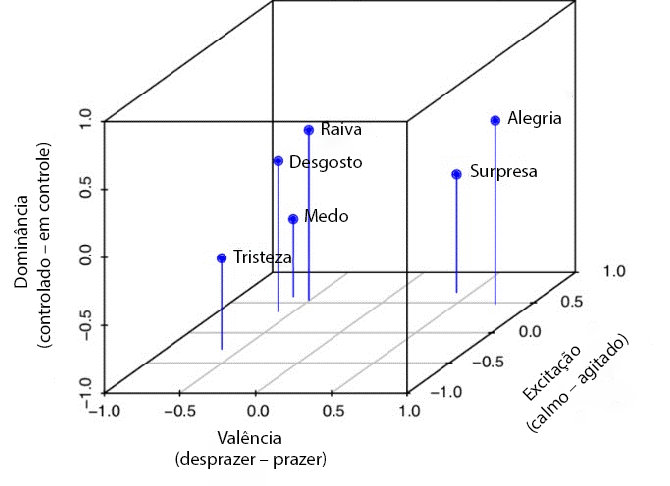


Figura 2. Modelo PAD

Fonte: Adaptado de Buechel e Hahn (2016)

* + 1. Modelo Circumplexo

Outro modelo dimensional desenvolvido por James A. Russel que foi amplamente adotado, chamado do Modelo Circumplexo, utiliza o escalonamento multidimensional para mapear as emoções em um plano projetado por dois eixos. O eixo horizontal do plano, chamado de Valencia, representa quão positivo ou agradável a experiência emocional é. Já o eixo vertical, chamado de Excitação, representa a intensidade da emoção, ou seja, quão passiva ou ativa é a emoção. O modelo desenvolvido agrupa as emoções segundo sua similaridade com as outras emoções, a partir das dimensões de Valencia e Excitação (SCHACTER; GILBERT; WEGNER, 2013).

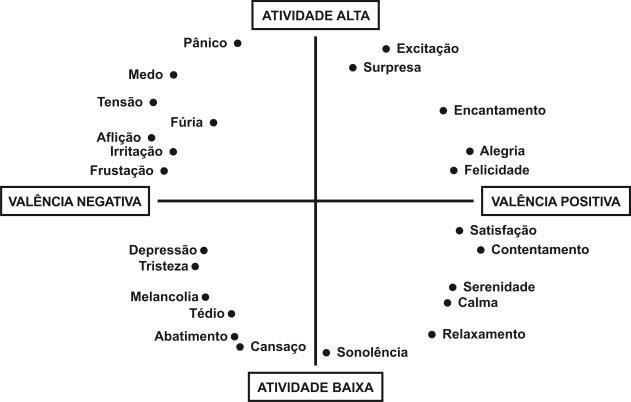


Figura 3. Modelo que mapeia as emoções em duas dimensões, valência (valence) e excitação (arousal).

Fonte: Nogueira (2018)

O modelo Circumplexo foi altamente influenciado pelo modelo PAD. Embora os dois compartilham o mesmo subconjunto de dimensões (Valencia e Excitamento), o modelo Circumplexo acrescenta outras informações entre as dimensões utilizadas. Devido a caracterização similar entre as emoções, o modelo as põe em uma circunferência, demonstrando a similaridade das emoções através das suas distâncias no plano. Por exemplo, as emoções de alegre e animado são semelhantes, e por isso são representados de forma quase adjacente no modelo. Em contrapartida, a emoção de tristeza está certa de 180º graus da emoção de alegria, que demonstra o sentido oposto das emoções no modelo (SCARANTINO, 2016).

Outra propriedade do modelo Circumplexo, é que este possibilita demonstrar a ambiguidade de certas emoções, o que exibe claramente a natureza difusa ou *fuzzy* das mesmas (POSNER et al., 2009; RUSSELL, 1980).



Figura 4. Esquemático alternativo do Modelo Circumplexo, que demonstra a existência de ambiguidade na categorização das emoções, exibido pela sobreposição das emoções no plano.

Fonte: Adaptado de POSNER et al. (2009)

* + 1. PAD versus Circumplexo

Em comparação com o modelo PAD, o modelo Circumplexo não exibe todas as dimensões associadas a emoções e deixa de explicitar as diferenças entre as emoções adjacentes. Por exemplo, as emoções de temor e raiva estão próximas no modelo Circumplexo, porém isso falha de exibir corretamente suas diferenças. Sendo assim, o modelo Circumplexo é visto como um modelo para visualizar as similaridades entre emoções, e não comunicar características únicas das mesmas. No que diz respeito da psicologia ambiental, o modelo Circumplexo implica a representação interna das emoções de um indivíduo, que apesar de serem influenciados externamente (estímulos provocados pelo ambiente), o estado interno pode variar de forma independente. Já o modelo PAD, com uma dimensão a mais do modelo Circumplexo, chamado de Dominância, consegue apresentar corretamente a diferença entre emoções adjacentes, pois este conta com um fator de avaliação do ambiente externo. No caso do exemplo das emoções de temor ou raiva, estes diferem por fatores externos, como causa e reação comportamental (SCARANTINO, 2016).

O modelo PAD, apesar de poder apresentar melhor as emoções e suas diferenças, tem sido relativamente difícil de incorporar em pesquisas devido ao seu fator de ser dependente do ambiente externo, representado pela dimensão de Dominância. Segundo MEHRABIAN e RUSSELL (1974) a dificuldade de encontrar estímulos externos que podem compor amplamente o espectro emocional de forma balanceada pode explicar o motivo pelo qual vários pesquisadores se basearem exclusivamente nas dimensões de prazer e excitação, que compõe o modelo Circumplexo. O modelo PAD tem sido utilizado extensivamente em pesquisas direcionadas ao consumidor. Em algumas dessa s pesquisas, resultados indicam que a fator de Dominância é relativamente fraca em relação aos outros fatores, e outros resultados sugerem que o fator de Dominância pode ser ignorado por completo (BAMFORD; NOBBS, 2007). Semelhantemente, RUSSELL e PRATT (1980) sugerem que o fator de dominância não se aplica à ambientes em que está sendo estudado uma reação afetiva. Sendo assim, estudos propõem que o uso prático do fator de dominância pode depender de certos aspectos do ambiente utilizado, que pode explicar resultados menos confiáveis do fator de dominância em comparação aos fatores de prazer e excitação (FOXALL, 1997).

* + 1. Roda de Emoções

O modelo da Roda de Emoções, desenvolvido por Robert Plutchik, surgiu de sua teoria psicoevolucionária que define que há oito emoções primárias que são biologicamente evoluídas através do instinto comportamental de sobrevivência. O modelo é visualizado como uma roda de cores, onde emoções similares são posicionadas de forma adjacente, e emoções opostas estão a 180º à parte. Outras emoções são formadas através da combinação de emoções elementares que são adjacentes entre si, que intuitivamente segue a mesma analogia da combinação de cores. O modelo de Plutchik, que compartilha certa semelhança aos modelos Circumplexo e PAD, também introduz uma terceira dimensão que representa a intensidade das emoções. Na versão bidimensional do modelo, a intensidade é representada segundo a distância ao centro da roda, ou seja, quanto mais próximo do centro da roda mais alto é a intensidade da emoção e vice-versa. Já na versão tridimensional do modelo, o modelo tem um formato curvo, semelhante à de um cone (PLUTCHIK, 2001).

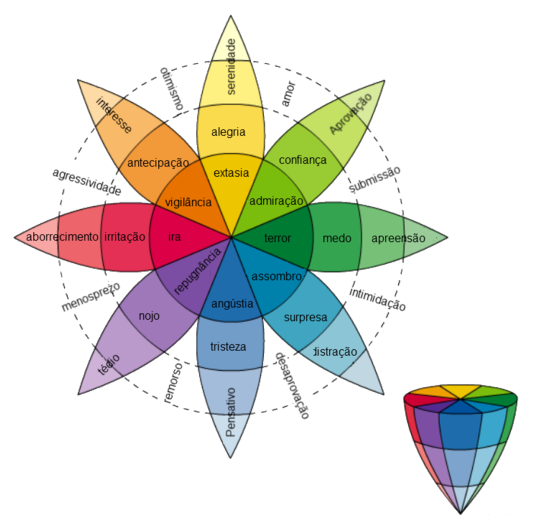


Figura 5. Roda de emoções

Fonte: Bragança (2010)

No modelo de Plutchik, a combinação de duas emoções primárias revela novas emoções, como também, revela quão frequente a emoção é experienciada, que é função da proximidade das emoções no modelo. Por exemplo, a combinação de duas emoções primárias adjacentes é chamada de díade primária e tem uma ocorrência mais frequente. Já a combinação de duas emoções primárias não adjacentes, chamado de díade secundária, é menos frequente.



Figura 6. Díades de emoções

Fonte: Adaptado de KOŁAKOWSKA et al. (2015)

O modelo de Plutchik apresenta várias visualizações para tentar correlacionar e idealizar as complexidades e diferentes relações que emoções tem entre si mesmas. Porém, o modelo de Plutchik é raramente utilizado em aplicações práticas (KOŁAKOWSKA et al., 2015).

* 1. Reconhecimento de emoções

O reconhecimento de emoções trata-se de um processo natural realizado por nós humanos, geralmente através da interpretação de expressões faciais e verbais. Humanos demonstram consistência no reconhecimento de emoções, porém, isso também têm se demonstrado certa variabilidade que depende dos indivíduos e suas habilidades. Para o reconhecimento biológico de emoções, há diferentes processos que captam sinais externos que realizam a interpretação de emoções, estes processos podem ser categorizados como processos visuais, auditivos, olfativos e fisiologicamente sensoriais.

* + 1. Processos visuais

O reconhecimento de emoções através de processos visuais tem bastante foco na compreensão das emoções emitidas a partir de expressões faciais. Vários estudos independentes entre si indicam que há uma percepção universal de expressões faciais das emoções de surpresa, desgosto, tristeza, felicidade, raiva e medo. Os principais sinais para identificação de expressões faciais estão centralizados sobre os olhos e a boca, por exemplo, raiva, medo e tristeza são identificados com mais facilidade pelos olhos. Já o sentimento de felicidade e desgosto são melhor identificados por movimentos da boca, por fim, o sentimento de surpresa é igualmente identificado por sinais visuais dos olhos e a boca (MOST; AVINER, 2009).

* + 1. Processos auditivos

Compreensão linguística contém vários fatores que contribuem para o entendimento do ouvinte. A percepção de emoções através de processos auditivos é apenas um dos fatores entre as existentes, como semântica e informações não verbais. Geralmente, a percepção de emoção é baseada em processos auditivos e visuais, porém, alguns estudos demonstram que emoções podem ser identificados apenas pela audição. Por exemplo, a raiva é identificada por ter uma frequência fundamental alta, uma grande variação de frequências fundamentais e intensidade médias elevadas. Em contrapartida, a tristeza é caracterizada por ter uma média baixa de intensidade e frequências fundamentais, e um ritmo mais lento na fala. Desse modo, foi demonstrado que a percepção emocional na audição tem alguns sinais essenciais, como características de frequências fundamentais, intensidade média, distribuição de energia no espectro e duração (MOST; AVINER, 2009).

* + 1. Processos olfativos

A percepção do olfato, conhecido como odor, também tem efeitos no ânimo. HERZ et al. (2004) demonstrou por meio de análises de ressonância magnética, havia uma ativação significativa nas regiões da amígdala e hipocampo do cérebro durante a recordação de memórias provocadas pelo odor sendo testado. A intensidade das ativações do cérebro gravadas, eram maiores de qualquer outro processo como auditivo ou visual, também, a reação comportamental dos participantes sinalizou uma maior reação emocional provocado pelo odor. Sendo assim, há evidências que o odor pode provocar reação emocional por meio da recuperação de memórias significativas.

* 1. Reconhecimento automatizado de emoções
     1. Reconhecimento de emoções por Voz

O reconhecimento de emoções através de voz pode tomar vantagem de alterações na voz gerada pelo sistema nervoso. Por exemplo, quando é experienciado o sentimento de raiva, medo ou alegria, há uma reação de excitação no sistema nervoso. Consequentemente, a emoção irá induzir uma série de reações fisiológicas como, uma aceleração no batimento cardíaco, maior pressão sanguínea, mudanças no movimento respiratório, seca da boca, e possível tremor de músculo. Essas reações afetam a voz, deixando-a mais rápida, mais alta, e mais definida com uma voz mais aguda. Por outro lado, quando é experienciado o sentimento tristeza, cansaço ou tédio, o resultado é uma desaceleração no batimento cardíaco, aumento de salivação, e baixa pressão sanguínea. A voz resultante geralmente é lenta, arrastada e com pouca energia. Portanto, os efeitos da emoção tendem a afetar a voz significantemente. Os principais fatores afetados são o tom, temporização, qualidade da voz e articulação (BREAZEAL; ARYANANDA, 2002).

O reconhecimento de emoções apenas por sinais vocais ocorre por meio de classificação de características extraídas do sinal da fala. Entonação de voz, temporização e outros parâmetros vocais que descrevem características prosódicas podem ser processadas por algoritmos com técnicas de reconhecimento de padrões.

* + 1. Reconhecimento de emoções por Imagens

Para formalmente categorizar diferentes emoções manifestadas de expressões faciais, EKMAN e FRIESEN (1971) definiram os fundamentos do sistema FACS (*Facial Action Coding System*). O sistema FACS, na sua forma mais simples, define a contração e relaxamento de um ou mais músculos faciais a partir de códigos para unidade de ação. Por exemplo, o código 1 representa a movimentação do musculo Frontalis, que movimenta a sobrancelha interna, e o código 6 representa a movimentação do músculo orbicular do olho. Desta forma uma movimentação coletiva dos músculos é representada por uma soma descritiva dos códigos.

Utilizando este sistema, é possível rotular manualmente qualquer expressão facial, expressando á em unidades de ação (AU – *Action Units*). O sistema de AU resolve problemas de subjetividade ao interpretar emoções de expressões faciais, o qual permitiu uma padronização internacional na descrição de emoções faciais. No Quadro 2, estão listadas as seis emoções básicas, sabidos por serem reconhecidos universalmente.

Quadro 2. Emoções básicas codificadas no sistema FACS

|  |  |
| --- | --- |
| **Emoção** | **Unidades de ação** |
| Felicidade | 6+12 |
| Tristeza | 1+4+15 |
| Surpresa | 1+2+5B+26 |
| Medo | 1+2+4+5+20+26 |
| Raiva | 4+5+7+23 |
| Desgosto | 9+15+16 |

Fonte: adaptado de Hager ([s.d.])

Com o sistema FACS, foi possível definir as AU especificas de expressões emocionais faciais, o qual permitiu o desenvolvimento de sistemas computacionais que possam identificação automaticamente as emoções a partir de imagens.

* + 1. Reconhecimento de emoções por texto

A detecção de emoções manifestados a partir de texto compartilha semelhanças na área de análise sentimental de textos. A maior diferença entre as duas áreas de pesquisa, é a profundidade de informações que está sendo extraído no texto. No caso da análise sentimental, os algoritmos procuram a opinião ou sentimento de forma mais genérica, atribuindo apenas níveis de sentimento positivo, negativo ou neutro em relação a uma entidade ou suas características (HU; LIU, 2004). Em contraste, a detecção de emoções em texto procura encontrar mais detalhes a partir da classificação de emoções mais objetivas no texto que podem agregar mais valor e compreensão da análise textual.

O problema de detecção de emoções em texto faz parte de uma área de pesquisa maior denominada de processamento de linguagem natural (NLP – *Natural Language Processing*), que estuda soluções para a geração e compreensão automática de linguagens humanas naturais.

* + 1. Reconhecimento de emoções por GSR

Outras formas de reconhecimento de emoções também estão sendo utilizadas em pesquisas, especialmente sinais biológicos extraídos de sensores, diferente das formas mais comuns utilizadas, como vídeo, voz e texto. Um dos sensores utilizados com grande sucesso, é o sensor que mede a resistência galvânica. Este sensor mede a impedância da pele que depende de quão úmido está a pele e é formalmente conhecido por medir resposta galvânica da pele (GSR). A lógica do seu funcionamento está baseada no sistema nervoso, de tal forma que as glândulas que produzem o suor são controladas pelo sistema nervoso. Sendo assim, foi possível traçar evidência que há uma correlação com o GSR e a dimensão de excitação, por exemplo, quanto mais excitado o estado do indivíduo se apresenta, mais informações que o sensor GSR lê, ou seja, a pele tem mais condutância.

* + 1. Reconhecimento de emoções por EMG

A eletromiografia facial (fEMG) refere-se a uma técnica de eletromiografia (EMG) que mede a atividade muscular, o qual detecta e amplifica pequenos impulsos elétricos que são gerados pelas fibras musculares quando se contraem. Apesar do rosto poder expressar uma grande variedade de emoções, apenas dois músculos, corrugador do supercílio e zigomático maior são utilizados com este sensor ao identificar emoções. O músculo corrugador do supercílio geralmente é utilizado para testar emoções negativas, e o músculo zigomático maior para as emoções positivas (LARSEN; NORRIS; CACIOPPO, 2003).

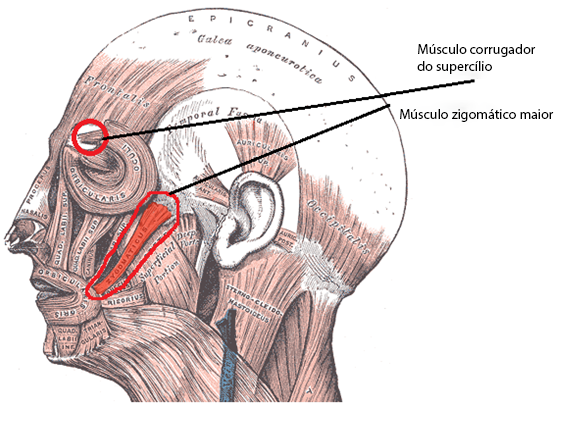


Figura 7. Músculos utilizados na eletromiografia facial para detectar emoções

Fonte: Henry Gray (2011)

* + 1. Reconhecimento de emoções por PPG

Uma forma alternativa de detectar emoções se baseia no processo de fotopletismografia (PPG), que possibilita a detecção de alterações no volume sanguíneo. A PPG geralmente é adquirida por meio de um oxímetro de pulso ou infravermelho junto com LED vermelho, onde este mede as mudanças de absorção de luz (PICARD, 1998; SHELLEY; SHELLEY, 2001). Variações no volume sanguíneo geralmente são resultado de uma reação de uma série de eventos externos diferentes, estresse emocional sendo um deles.

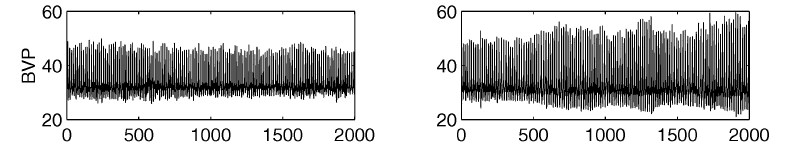


Figura 8. Exemplos de medidas PPG de um usuário enquanto intencionalmente expressava raiva (esquerda) e tristeza (direita).

Fonte: PICARD (2001)

* + 1. Reconhecimento de emoções por fMRI

O uso de imagem por ressonância magnética (fMRI - *Functional Magnetic Resonance Imaging*) possibilita a visualização das regiões do cérebro que são ativadas a partir de um estimulo. Estas ativações cerebrais são definidas por variações de fluxo sanguíneo causadas por uma atividade neural. Desta forma, quando uma região do cérebro está em uso, o fluxo sanguíneo desta região aumenta (LOGOTHETIS et al., 2001).

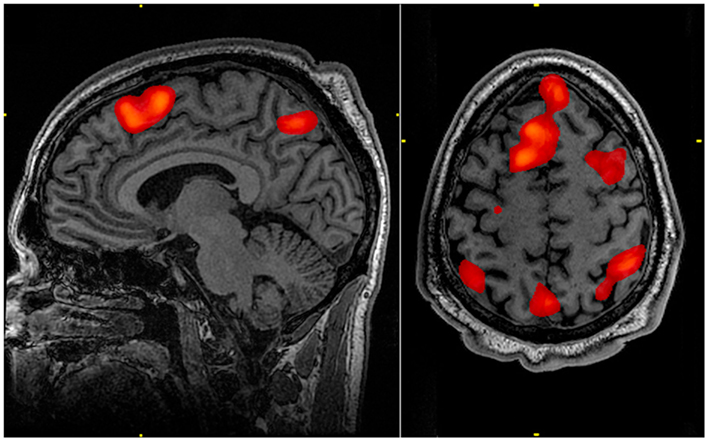


Figura 9. fMRI durante tarefas de memória de trabalho

Fonte: Graner (2010)

POSNER et al. (2009), através de resultado obtidos de fMRI, demonstrou que existe evidência de atividade neural que dê suporte a teoria de emoções que diz respeito das dimensões de valência e excitação. A pesquisa demonstra que há uma relação linear entre a atividade neural e variações incrementais nas duas dimensões de afeto. Sendo assim, os resultados sustentam a teoria do modelo Circumplexo, o qual defende a ideia que todas as emoções podem ser representadas por duas dimensões, valência e excitação.

* + 1. Reconhecimento de emoções por EEG

A eletroencefalografia (EEG) é uma técnica não invasiva para a captação de sinais emitidos pelo cérebro através de dispositivos com eletrodos posicionados sobre o couro cabeludo. Sinais de EEG são o resultado da junção de milhares de potenciais pós-sinápticos emitidos por neurônios. Diferente de sinais de EEG, correntes elétricas que fluem de forma paralela ao couro cabeludo não são detectadas por eletrodos. Semelhantemente, correntes que são de origem profunda do cérebro também não são detectadas devido à perda de energia até ser propagado a superfície do crânio. Portanto, a principal fonte de sinais de EEG são do córtex cerebral (RAO, 2011, p. 26).

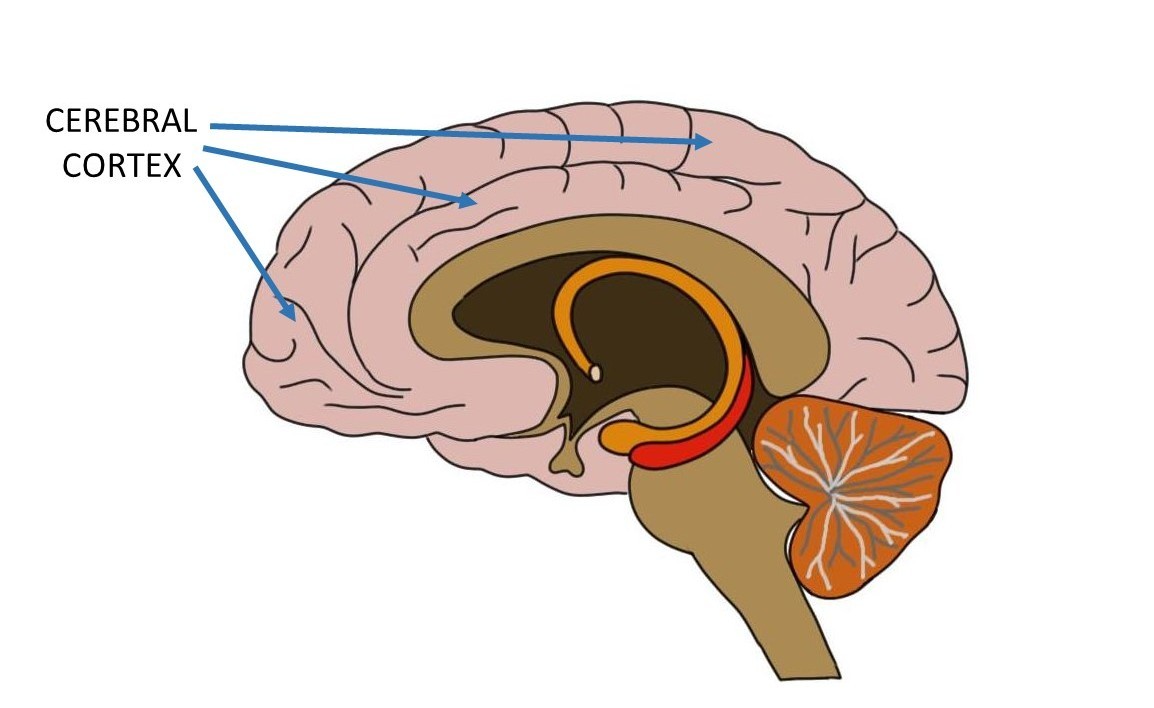
****

Figura 10. Cortex Cerebral

Fonte: Neuroscientifically Challenged (2014)

Sinais de EEG sofrem de pouca resolução espacial, especificamente na ordem de centímetros quadrados. Isso significa que o sinal de EEG captado pelo eletrodo não representa por completo a atividade neural logo abaixo a área de superfície de contato, o sinal captado é o resultado da mescla de vários sinais emitidos por regiões próximos. O motivo da fraca resolução espacial se deve ao fato da existência das várias camadas (meninges, líquido cefalorraquidiano, crânio e couro cabeludo) entre a superfície de contato dos eletrodos e a fonte do sinal. Em contrapartida, a resolução temporal em sinais de EEG é considerada ótima, na ordem de milissegundos. A boa resolução temporal representa a frequência em que o sinal pode ser amostrado, portanto, uma alta resolução temporal neste contexto significa uma melhor detecção de variações no sinal que ocorrem em períodos curtos, que poderiam ser omitidos numa detecção com pouca resolução temporal (RAO, 2011, p. 26).

Sinais captados de EEG estão na ordem de microvolts, isto implica na necessidade do uso de amplificadores e filtros para processamento do sinal e remoção de ruído. Devido sua baixa amplitude, sinais de EEG são facilmente corrompidos pela movimentação muscular e interferência elétrica. Por exemplo, movimentações nos olhos, piscar de olhos, movimentação da cabeça, podem causar ruído e outros artefatos. Para reduzir o impacto dessas interferências, são utilizados algoritmos sofisticados para remover completamente ou parcialmente as secções afetadas do sinal.

Para a captação de sinais de EEG, geralmente envolve a utilização de uma touca na qual os eletrodos são fixados. Para melhorar a qualidade do sinal, as vezes é realizada um leve esfolamento no couro cabeludo para reduzir a impedância causada por células mortas da pele. Também, geralmente é colocado gel condutor antes de encaixar os eletrodos na touca para melhorar a captação do sinal.

O Sistema Internacional 10-20 é uma convenção utilizada para especificar posições de eletrodos sobre o couro cabeludo. O sistema define uma codificação prefixadas por letras que definam as localizações sobre o couro cabeludo.

Quadro . Definição dos prefixos do sistema 10-20

|  |  |
| --- | --- |
| **Prefixo** | **Significado** |
| C | Central |
| P | Parietal |
| T | Temporal |
| F | Frontal |
| Fp | Frontal polar |
| O | Occipital |
| A | Mastoids |

Fonte: adaptado de RAO (2011, p. 26-27)

Por exemplo, para a posição dos eletrodos de referência, as posições A1 e A2 podem ser utilizadas (nos lóbulos da orelha). Outros pontos de referência também podem ser utilizados para referência, como a parte superior do nariz e o ínion.

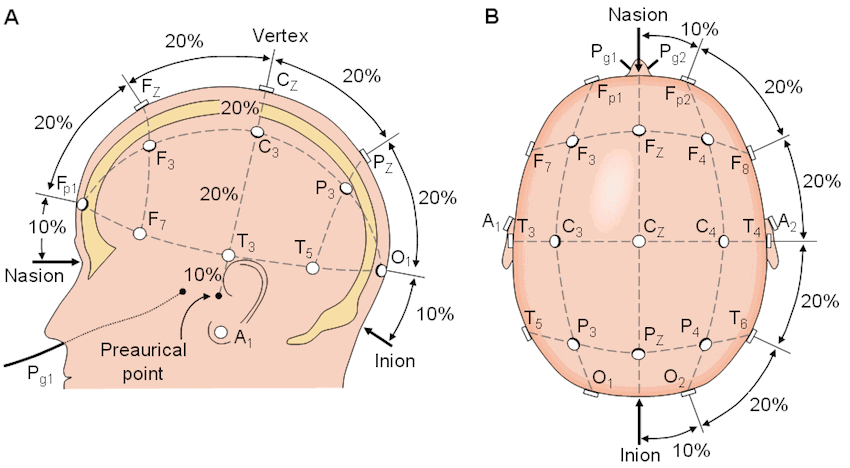


Figura 11. Sistema internacional 10-20

Fonte: Sklar (2017)

Eletrodos bipolares ou unipolares são utilizados na captação do sinal de EEG. O método Bipolar utiliza a diferença de potencial de um par de eletrodos para medir o sinal. Já no método unipolar, os eletrodos são comparados com uma referência em comum, alternativamente, é utilizada a média dos sinais como referência, conhecido como CAR (do inglês Common Average Reference).

Sinais de EEG contém diferentes bandas de frequência, a intensidade dessas bandas é diretamente influenciada pelo estado cerebral. No Quadro 4, são especificados diferentes bandas e o estado mental correspondente.

Quadro 4. Diferentes bandas de sinais de EEG

|  |  |  |
| --- | --- | --- |
| **Nome** | **Frequência (Hz)** | **Estado cerebral** |
| Gamma | > 35 | Concentração |
| Beta | 12-35 | Atenção externa e ativa |
| Alpha | 8-12 | Relaxado, atenção passiva |
| Theta | 4-8 | Sonolência |
| Delta | 0,5-4 | Dormente |

Autor: Adaptado de EHROTRA (2018)

Recentemente, várias pesquisas para o reconhecimento de emoções através da EEG foram realizadas. Devido à natureza complexa de sinais de EEG, existe uma ampla variedade de técnicas e métodos sendo explorados para a decodificação das emoções dos sinais de EEG. Em (ZANGENEH SOROUSH et al., 2017), é realizado uma revisão das pesquisas mais recentes realizadas nessa área, demonstrando a variedade de métodos em 46 trabalhos diferentes. Os principais componentes das pesquisas realizadas são as variações em que as emoções são identificadas, estímulos utilizados, *datasets* empregados e algoritmos de extração de características e classificação.

* + 1. Comparação de EEG com outros métodos

Sinais de EEG tem várias vantagens em relação a outros métodos. As vantagens se baseiam no fato de que EEG tem alta resolução temporal, que é importante para o estudo de processos neurocognitivos. O motivo disso, é que a captação do sinal ocorre no mesmo período do que a própria cognição. A grande maioria dos processos cognitivos ocorrem em intervalos de milissegundos e podem durar de algumas centenas de milissegundos a alguns segundos. Em contrapartida, métodos hemodinâmicos como a fotopletismografia, tem uma resposta de duas a três mais lenta ao processo cognitivo em comparação a uma resposta eletrofisiológica (COHEN, 2014, p. 15).

Outra vantagem da EEG para o estudo de processos neurocognitivos, é que os sinais medidos são diretamente relacionados a atividade neural. A variação esporádica de tensões medidas na EEG provém de fenômenos biofisiológicos, gerados a partir de uma mescla de sinais de uma população de neurônios. Essa vantagem pode ser vista em comparação com outros métodos como a fMRI em conjunto com a BOLD (do inglês, Blood Oxygenation Level Dependent). O método BOLD não mede a atividade neural diretamente, pois este mede as variações no fluxo sanguíneo (COHEN, 2014, p. 15–16).

* 1. Processamento de sinais
     1. Análise de Fourier

A análise de Fourier consiste na decomposição do sinal em uma soma ponderada de senos e cossenos de diferentes frequências. Esta análise permite a expressão de uma função periódica e contínua no tempo a partir de uma soma de senoides com diferentes frequências e amplitudes definido sobre um intervalo em , onde T é um período do sinal periódico :

|  |  |
| --- | --- |
|  | Equação |

Onde é a frequência angular definido como onde é a frequência ordinária. A expansão infinita da série acima é denominada a série de Fourier, que permite a análise de um sinal, a partir dos coeficientes e , que representam as amplitudes das senoides. Os valores dos coeficientes podem ser estimados através das equações abaixo:

|  |  |
| --- | --- |
|  | Equação |

Que fundamentalmente realiza uma correlação cruzada entre a senoide de uma determinada frequência e o sinal periódico , e dessa forma, consegue “codificar” a semelhança entre os dois sinais no coeficiente sendo calculado. Uma forma alternativa mais usada de representar a série de Fourier faz o uso da representação complexa dos parâmetros e , denominado por , que descreve o sinal como:

|  |  |
| --- | --- |
|  | Equação |

Tal que, , é descrito como

|  |  |
| --- | --- |
|  | Equação |

Para funções que tem o valor de zero fora do intervalo , a Equação 4 é equivalente a Transformada de Fourier, e a Equação 3 é Transformada Inversa de Fourier (RAO, 2011; SMITH, 1997, p. 144–145).

* + 1. Transformada Discreta de Fourier

Aplicações que utilizam sinais contínuos precisam discretizar o sinal para possibilitar o armazenamento e processamento. Para processar os sinais, aplicações utilizam a forma discretizada da Transformada de Fourier, denominado de Transformada Discreta de Fourier (DFT, do inglês, Discrete Fourier Transform). A DFT transforma um sinal discretizado com amostras para seus respectivos coeficientes no domínio da frequência, e é definida como:

|  |  |
| --- | --- |
|  | Equação |

Também, a Transformada Inversa de Fourier (IDFT, do inglês, *Inverse Discrete Fourier Transform*) é definida como:

|  |  |
| --- | --- |
|  | Equação |

Os coeficientes no domínio da frequência, , contém informações de amplitude e fase das componentes sinusoidais que podem ser extraídas através das propriedades da forma polar de números complexos:

|  |  |
| --- | --- |
|  | Equação |
|  | Equação |

Onde, e denominam as partes reais e imaginária de . Em aplicações que usam sinais de EEG, geralmente é analisado a magnitude de diferentes frequências durante a execução de uma tarefa, que basicamente é expresso como o quadrado da Transformada de Fourier. De forma alternativa, a densidade espectral pode ser calculada como o quadrado da amplitude definido por :

|  |  |
| --- | --- |
|  | Equação |

A definição acima é conhecida como a densidade espectral do sinal, e é amplamente utilizado em aplicações que realizam o processamento de sinais (RAO, 2011).

* + 1. Transformada rápida de Fourier

Na prática, a DFT de um dado sinal acaba sendo lento devido à grande quantidade de amostras em um pequeno intervalo de tempo e o custo das operações matemáticas, que na notação de complexidade computacional é descrito por . O algoritmo Transformada rápida de Fourier (*Fast Fourier Transform*, do inglês, ou FFT) prova ser eficiente, reduzindo a complexidade computacional para . Umas das implementações mais comuns do algoritmo, chamado de Cooley-Turkey, utiliza técnicas de divisão e conquista, que divide o sinal em pequenas seções de forma recursiva quais são aplicadas as operações matemáticas do algoritmo FFT (COOLEY; LEWIS; WELCH, 1967; RAO, 2011).

* 1. Redução de dimensionalidade
     1. Análise de componentes principais

A análise de componentes principais (PCA – *Principal Component Analysis*), tem como objetivo delinear um conjunto de pesos que representam os componentes principais. Os componentes principais são calculados a partir da covariância de um conjunto de variáveis correlacionadas. Os componentes tentam encapsular a maioria da variação existente em conjunto de dados em um número menor de dimensões, de forma que os componentes não sejam correlacionados entre si.

Por exemplo, o primeiro componente principal, comumente denominado de PC1, é o componente que codifica a variação de uma dimensão que contém a maior variação entre as outras dimensões do conjunto de dados. Do mesmo modo, o segundo componente denominado de PC2, encapsula a maior variação residual para a segunda dimensão mais variável, sendo ortogonal à PC1, e assim por diante.

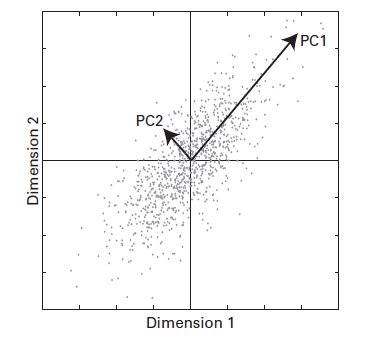
[](https://univali-my.sharepoint.com/personal/kev_asher_edu_univali_br/Documents/tcc/ttc-relatorio.docx?web=1)

Figura 12. Duas componentes principais sobre duas dimensões correlacionadas

Autor: Cohen (2014)

Dessa forma, a análise de componentes principais consegue capturar a essência da variabilidade de um conjunto de dados com várias dimensões, efetivamente reduzindo as dimensões em um conjunto de componentes principais de menor dimensionalidade (COHEN, 2014).

* 1. Aprendizado de máquina
     1. Regressão Linear

Regressão linear é um algoritmo que tem como objetivo obter uma relação linear que possa prever de forma aproximada uma função dada o conjunto de amostras de uma ou mais variáveis. A Regressão Linear é considerada uma abordagem simples de aprendizagem supervisionada, que apesar da simplicidade de implementação é bastante utilizado na prática. Contudo, mesmo tendo o nome Regressão Linear na literatura, existem variações que apresentam formas não lineares do algoritmo, que as vezes é denominado de Regressão não Linear (STANFORD.EDU, [s.d.]).

A função sendo aproximada, é definido como:

|  |  |
| --- | --- |
|  | Equação |

Onde os coeficientes são atualizados conforme a função é “treinada”. Os coeficientes , também chamado de parâmetros, representam as características das variáveis , com exceção de , que sempre tem o valor 1.

Uma forma alternativa de representação, é na sua forma vetorial, que geralmente é utilizado para simplificação de notação, eficiência de execução e implementação:

|  |  |
| --- | --- |
|  | Equação |

A atualização dos parâmetros ocorre através do método de gradiente, que é um método numérico de otimização. O método gradiente tem como objetivo minimizar uma função que captura a magnitude média de diferença do valor previsto e o valor desejado , denominada da função de custo, que é descrito como:

|  |  |
| --- | --- |
|  | Equação |

Onde, e representam o i-ésimo vetor de amostra. Para minimizar a função de custo , o método de gradiente calcula a derivada parcial da função de custo em relação aos coeficientes , definido como:

|  |  |
| --- | --- |
|  | Equação |

Em seguida atualiza os parâmetros iterativamente com a equação abaixo, até que houver um valor mínimo satisfatório da função de custo:

|  |  |
| --- | --- |
|  | Equação |

Tal que, é definido como o vetor de gradiente:

|  |  |
| --- | --- |
|  | Equação |

O parâmetro na Equação 14 define qual rápido irá convergir ao valor mínimo da função de custo. O parâmetro precisa ser ajustado corretamente para não houver divergência, como também convergir em um tempo aceitável (COURSERA.ORG, [s.d.]).

* + 1. Regressão Logística

Regressão Logística é um algoritmo com implementação semelhante ao do algoritmo de Regressão Linear, mas difere no aspecto em que a predição realizada é categórica, ou seja, é previsto como resposta desse algoritmo um número discreto de classes, por exemplo, duas classes no caso da classificação binária.

Para realizar a classificação, a Regressão Logística redefina a função (originalmente definida na Equação 11) para calcular a probabilidade da função resultar no valor 1, ou seja, é utilizado a função logística, também chamado da função sigmoide.

|  |  |
| --- | --- |
|  | Equação |

Tal que,

|  |  |
| --- | --- |
|  | Equação |

É definido como a função sigmoide, que mapeia qualquer valor real para um intervalo . Sendo assim, a função sigmoide é apropriada para classificação binária, onde há apenas duas classes.

A função de custo para regressão logística não pode ser a mesma utilizada na regressão linear devido ao problema da não convexidade. A não convexidade implica na existência de vários mínimos locais, consequentemente, o algoritmo do método de gradiente não consegue atingir um mínimo global da função de custo e se estabelece em um mínimo local, fazendo com que a função preditiva, , não consiga os valores desejáveis para os parâmetros .

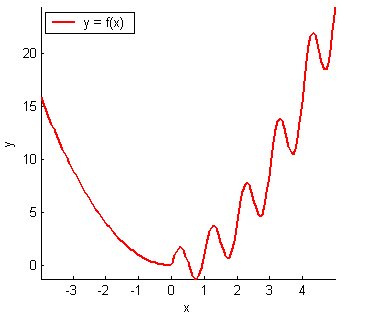


Figura 13. Uma função não convexa, com múltiplas mínimas locais, onde a derivada da função é zero.

Fonte: Sampler (2009)

Para resolver este problema, a função de custo é redefinida. A função de custo de entropia cruzada garante a convexidade dado a nova definição da função . Esta é definida como

|  |  |
| --- | --- |
|  | Equação |

Para a classificação de mais de duas classes, a abordagem tomada ao utilizar regressão logística é conhecida como *one-vs-all*. Esta abordagem calcula separadamente para cada classe a probabilidade da classificação, de tal forma que é realizada a regressão logística binária para cada classe. Em seguida, o resultado que retornou a maior probabilidade é usado como predição (COURSERA.ORG, [s.d.]; DATA SCIENCE ACADEMY, [s.d.]).

* + 1. Redes Neurais Artificiais

O modelo computacional de Redes Neurais Artificiais (RNA) foi inspirado na forma que neurônios de seres vivos funcionam e aprendem. O modelo é semelhante a organização biológico neural nos quesitos de comunicação através de sinapses e seus milhares de interconexões com outros neurônios. Redes neurais artificiais aprendam na forma de exemplos, com o objetivo de generalizar ao máximo sem ter regras explicitas de como deve se comportar dada uma amostra específica dos dados de entrada (CARVALHO, 2009).

Dependendo da arquitetura de uma RNA, esta compartilha características semelhantes ao modelo de Regressão Logística descrita na seção 2.6.2. Na forma mais simples de uma RNA por exemplo, um *Perceptron* de uma camada, há somente uma função de ativação. Caso a função de ativação for a função logística, então, a rede neural é idêntica ao modelo da Regressão Logística.

O algoritmo utilizado por uma RNA para realizar o treinamento é denominado de *Backpropagation*. De forma semelhante ao modelo de Regressão Logística, o algoritmo de *Backpropagation* tenta minimizar a função de custo iterativamente, atualizando os parâmetros das conexões entre os nós da rede. O diferencial do algoritmo *Backpropagation*, é que este permite o cálculo das derivadas parciais para todos os parâmetros da rede, em uma única passagem para frente (*forward pass*) seguido por uma passagem para atrás (*backward pass*) (DATA SCIENCE ACADEMY, [s.d.]; NIELSEN, 2015).

* 1. Trabalhos relacionados

Esta seção descreve os trabalhos relacionados envolvendo o reconhecimento de emoções por meio de sinais de EEG. Um resumo breve de cada trabalho é realizado, destacando os algoritmos e modelos utilizadas para classificação e resultados adquiridos.

* + 1. Real-time EEG-based user's valence monitoring

No trabalho de Lan et al. (2015), é apresentado um novo algoritmo para o reconhecimento de 4 níveis de valência do modelo Circumplexo. O algoritmo proposto é sujeito dependente, isto é, requer algumas sessões que realizam uma calibração inicial no sistema para que o mesmo possa calcular os limiares necessários para funcionamento. O algoritmo toma vantagem de padrões assimétricos que existem no cérebro. A assimetria se refere à ativação dos diferentes hemisférios do cérebro após um estimulo. No caso de um estímulo de afetivo positivo (valência positiva), o hemisfério esquerdo do cérebro é mais ativado, enquanto que, quando há um estimulo de afetivo negativo (valência negativa) o hemisfério direito se mostra mais dominante na atividade cerebral. Quando um hemisfério é mais ativado do que o outro, os sinais captados por eletrodos posicionados nesse hemisfério são mais complexos, no sentido de ter mais variações no sinal. Em base disso, é possível calcular a diferença de atividade neural dos hemisférios a partir do cálculo da dimensão fractal dos sinais.

Segundo o autor, trabalhos relacionados geralmente utilizavam somente de 2 a 3 níveis de valência. Neste trabalho, foi utilizado 4 níveis de valência. Os níveis delimitados representam intervalos de valores de valência, segmentados de 1 a 4 descritos pela tabela abaixo:

Quadro 5. Níveis de valência

|  |  |  |
| --- | --- | --- |
| **Nível de valência** | **Intervalo de valores** | **Interpretação** |
| Nível 1 | 1, 2 | Muito desagradável |
| Nível 2 | 3, 4 | Desagradável |
| Nível 3 | 5, 6 | Agradável |
| Nível 4 | 7, 8, 9 | Muito agradável |

Para o desenvolvimento e validação do sistema foi utilizado o DEAP *dataset*. Foi realizado uma série de transformações e considerações em relação ao *dataset* para que o mesmo tornasse apropriado para a validação do sistema. O sistema final conseguiu uma acurácia máxima de 79,31% e uma acurácia média de 49,40%.

* + 1. CogniMeter: EEG-based Emotion, Mental Workload and Stress Visual Monitoring

Hou et al. (2016), implementou um sistema em tempo real para o reconhecimento de emoções, carga de trabalho e monitoramento de estresse. Para o reconhecimento de emoções foi utilizado um algoritmo de aprendizado de máquina em conjunto com características estatísticas e dimensão fractal. As características estatísticas utilizadas incluem o valor médio, desvio padrão, média dos valores absolutos das primeiras diferenças, média dos valores absolutos das primeiras diferenças dos sinais normalizados, média dos valores absolutos das segundas diferenças e a média das segundas diferenças dos sinais normalizados. As características estatísticas e os valores de dimensão fractal são normalizados e entregue à um classificador SVM. A classificação é realizada de forma que é sujeito dependente. Portanto, para usar esse sistema é necessário realizar uma etapa de treinamento.

Para medir a carga de trabalho é calculado a densidade espectral da banda theta (4-8Hz), já que tem sido demonstrado que há uma forte correlação entre a carga de trabalho e as ondas theta. O monitoramento do estresse foi realizado a partir dos dados adquiridos de classificação de emoção e carga de trabalho, onde foi encontrado uma correlação positiva entre carga de trabalho e estresse, e uma correlação negativa entre estresse e valência. Portanto, foi proposto a combinação das medidas de carga de trabalho e emoção para reconhecer o estresse.

Em suma, o sistema desenvolvido consegue classificar 8 emoções com 52,67% de acurácia, mas caso somente houvesse emoções positivas ou negativas, a melhor acurácia poderá alcançar 91,07%.

* + 1. EEG based patient emotion monitoring using relative wavelet energy feature and Back Propagation Neural Network

Neste trabalho, Purnamasari et al. (2015) apresentam um método para o reconhecimento de emoções em tempo real a partir de características extraídas por meio da transformada de *wavelet.* As características extraídas são de origem daenergia relativa da onduleta (RWE – Relative Wavelet Energy). Para a classificação, foi utilizado uma rede neural clássica com 3 camadas, sendo uma camada oculta. Foram utilizadas 4 classes para classificação das emoções em base dos valores de valência e excitamento do modelo Circumplexo. As classes foram dividas conforme os 4 quadrantes do modelo Circumplexo, que são divididos como alta excitação/alta valência (HAHV), baixa excitação/alta valência (LAHV), baixa excitação/baixa valência (LALV) e alta excitação/baixa valência (HALV).

Para obter o melhor resultado, foi realizado testes com janelas de tempo diferentes. A melhor acurácia alcançada foi na janela de 30 segundos, com uma acurácia média de 92,03%.

* + 1. Real-Time Movie-Induced Discrete Emotion Recognition from EEG Signals

Para o desenvolvimento desse sistema, Liu et al. (2018) criaram seu próprio *dataset* para realizar a classificação das emoções. O *dataset* é formado por 16 trechos de filmes com alto índice emocional. Em base das diferentes categorias de emoções que são induzidas pelos trechos escolhidos, é proposto um sistema que de reconhecimento de emoções em tempo real por meio de sinais de EEG. Para a realização do experimento, 30 participantes assistiram o conteúdo audiovisual previamente compilado, com objetivo de reconhecer 8 emoções discretas, incluindo a emoção neutra. Para segmentar os sinais de EEG, foi utilizado uma janela de dois segundos, com 50% de sobreposição entre segmentos consecutivos. Para cada janela de tempo, é realizado a classificação dos sinais em relação as dimensões de valência e excitação, como também uma classificação discreta das emoções. Para a extração de características dos sinais de EEG, foi utilizado o STFT (*Short-Time Fourier Transform*) com uma janela deslizante. A transformada permite a extração e normalização de características em base da análise de tempo-frequência. Mais de 105 características dos sinais de EEG foram obtidas, sendo assim, foi necessário reduzir a dimensionalidade dos dados. Para a seleção automatizada das melhores características, foi utilizado o LDA. As características selecionadas foram entregues como entrada para um classificador SVM.

O reconhecimento das emoções discretas foi desenvolvido em três estágios. No primeiro estágio, é realizado a classificação de emoções neutras e não neutras, nessa etapa foi alcançado uma acurácia média de 92,26%. No segundo estágio, é realizado a classificação entre emoções positivas e negativas, a partir dos dados das emoções não neutras adquiridas no primeiro estágio. A acurácia média obtida por meio desse método foi de 86,63%. No terceiro estágio, emoções discretas foram classificadas. Por meio do conjunto de dados obtidos no segundo estágio, emoções de alegria, diversão e ternura foram classificadas do conjunto de emoções positivas com uma acurácia média de 86,43%. Emoções de raiva, tristeza, medo e desgosto foram classificadas do conjunto de emoções negativas com uma acurácia média 65,09%. Sendo assim, foi possível estabelecer uma classificação de 3 níveis que pode classificar em até 8 emoções discretas com acurácia geral de 60,55%. O método de classificação em 3 níveis é comparado com o método que tenta classificar 8 classes diferentes diretamente. Este obteve um desempenho bem inferior em contraste com o método de 3 níveis, com uma acurácia máxima de 32,31%.

* + 1. Analise comparativa

Diferentes métodos foram empregados nos trabalhos apresentados. Diversas definições para a distinção das emoções, extração de características e algoritmos de classificação foram utilizadas. No quadro abaixo, é destacado as principais características dos trabalhos e seus resultados.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Autor** | **Classe de emoção** | **Extração de características** | **Classificador** | **Acurácia** |
| Lan et al. (2015) | Muito desagradável, desagradável, agradável e muito agradável | Fractal | *Threshold* com *max-voting* | 49,40% |
| Hou et al. (2016) | Positivo e negativo | Estatístico e Fractal | SVM | 91,07% |
| Purnamasari et al. (2015) | HAHV, LAHV, LALV e HALV | Energia relativa de wavelet | Rede Neural | 92,03% |
| Liu et al. (2018) | Positivo e negativo | Tempo-frequência com LDA | SVM | 86,63% |
| Este trabalho | Positivo, negativo e neutro | \* | \* |  |

\* Diversos algoritmos diferentes serão experimentados. O que apresentar melhor desempenho será selecionado para comparação.

1. Projeto

Este capítulo tem o objetivo de apresentar os requisitos previstos para este projeto, assim como, apresentar a estrutura do projeto.

A seguintes seções irão apresentar os requisitos funcionais e não-funcionais, os casos de uso, o diagrama de sequência, o planejamento de execução, planos de teste e o cronograma previsto para a próxima fase do TCC.

* 1. Diagrama de blocos

Uma visão geral do sistema é exibida na Figura 14 com um diagrama de blocos.

Uma imagem contendo captura de tela

Descrição gerada automaticamente

Figura 14. Diagrama de blocos do sistema

No diagrama da Figura 14, é evidenciado a sequência, fluxo de informações e processamento realizados pelo sistema. Os sinais eletroencefalográficos no diagrama já estarão digitalizados e prontos para serem processados. Para que o sistema possa extrair corretamente as características necessárias para processamento do sinal, é necessário realizar o pré-processamento do sinal para fins de remoção de artefatos indesejados que geralmente são fruto de movimentos musculares faciais e outras interferências externas. Após ser filtrado, o sinal é entregue ao sistema de reconhecimento de características, que irá redimensionar o sinal em uma série de características que o sistema de classificação é projetado para funcionar.

* 1. Requisitos funcionais

Quadro 6. Requisitos Funcionais

|  |  |
| --- | --- |
| **Requisito** | **Descrição** |
| RF1 | O sistema deve realizar a classificação das emoções, com no mínimo 3 classes: emoções positivas, negativas e neutras. |
| RF2 | O sistema deve exibir na interface do usuário a emoção classificada. |
| RF3 | O sistema deve exibir na interface do usuário a porcentagem de confiança na classificação realizada. |
| RF4 | O sistema deve permitir a configuração da fonte de sinal. |
| RF5 | O sistema deve permitir iniciar, pausar e retomar uma sessão. |

* 1. Requisitos não-funcionais

Quadro 7. Requisitos não-funcionais

|  |  |
| --- | --- |
| **Requisito não funcional** | **Descrição** |
| RNF1 | O sistema deve realizar a classificação das emoções em tempo real, dentro dos intervalos estabelecidos nos trabalhos relacionados. |
| RNF2 | O sistema deve tratar de artefatos indesejados no sinal EEG. |
| RNF3 | A interface do usuário deve ser desenvolvida com linguagens de interface Web: HTML5, CSS e Javascript. O Electron deverá ser utilizado para permitir execução em desktops de forma multiplataforma. |
| RNF4 | O sistema de classificação deve ser implementado com a linguagem Python e/ou Javascript. |
| RNF5 | Para a modelagem do sistema deve ser utilizado Matlab, Octave e/ou Python. |

* 1. Casos de uso

A Figura 15 demonstra os casos de uso da aplicação. Este diagrama deixa explícito as formas com que o usuário irá interagir com o sistema.

Uma imagem contendo texto, mapa

Descrição gerada automaticamente

Figura 15. Casos de uso

|  |  |
| --- | --- |
| **UC01** | **Configurar fonte de entrada do sinal** |
| Descrição | Configurar fonte de entrada do sinal |
| Pré-condições | Caso for utilizar um dispositivo para os sinais, então o dispositivo precisa estar conectado ao computador. Caso for um arquivo, então o arquivo precisar estar acessível pelo software. |
| Pós-condições | Software exibe os botões para controlar a sessão, isto indica que a fonte do sinal foi aceita pelo software. |
| Objetivos | Selecionar a fonte do sinal que será utilizado em uma sessão. O usuário poderá selecionar uma fonte pré-gravada de sinais armazenado em um arquivo, ou um dispositivo conectado no computador. |
| Atores | Usuário |
| Fluxo Principal | 1. Acessar o software 2. Acessar a opção de selecionar uma fonte 3. Clicar em uma das opções: selecionar arquivo ou dispositivo. 4. Selecionar fonte. |
| Fluxos de exceção | Caso houver um erro na fonte de sinal inserida no software, avisar o usuário com mais detalhes do problema. |

|  |  |
| --- | --- |
| **UC02** | **Visualizar emoção classificada** |
| Descrição | Visualizar resultado da classificação realizada pelo sistema |
| Pré-condições | A fonte de sinal precisar estar configurada |
| Pós-condições | Resultados de classificação contínua serão exibidos. |
| Objetivos | Visualizar as emoções sendo classificadas pelo sistema e outros dados relacionados. |
| Atores | Usuário |
| Fluxo Principal | 1. Clicar no botão para iniciar a sessão 2. Monitorar resultado exibidos na tela |
| Fluxos de exceção | Caso houver um erro na execução do sistema, avisar o usuário com mais detalhes do problema e providenciar opções para reinicialização da sessão. |

|  |  |
| --- | --- |
| **UC03** | **Iniciar, pausar ou retomar uma sessão** |
| Descrição | Controles básicos para a execução de uma sessão |
| Pré-condições | A fonte de sinal precisar estar configurada. Para as opções de pausar ou resumir, a sessão já estará iniciada. |
| Pós-condições | Processamento e classificação dos sinais irão obedecer às opções de controle ativadas. |
| Objetivos | Controlar uma sessão. |
| Atores | Usuário |
| Fluxo Principal | * Clicar para iniciar a sessão * Clicar para pausar uma sessão * Clicar para resumir uma sessão |
| Fluxos de exceção | Caso houver um erro na execução do sistema, avisar o usuário com mais detalhes do problema e providenciar opções para reinicialização da sessão. |

* 1. Diagrama de sequência

Na Figura 16, é representado a sequência lógica da interação do usuário com o sistema, conforme descrito nos casos de uso.

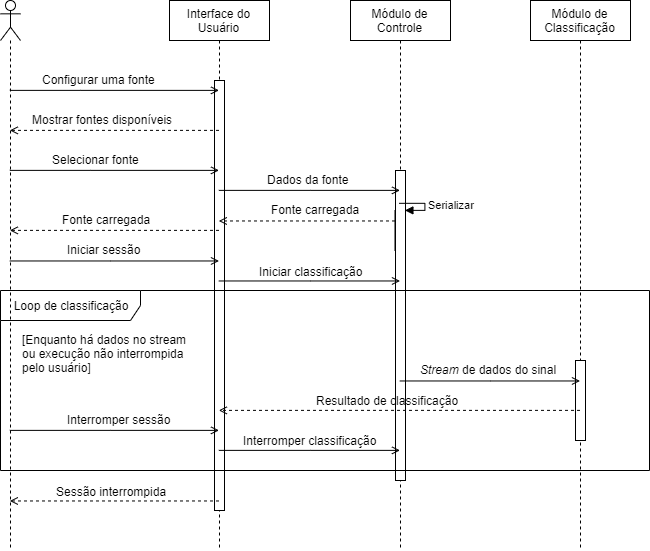


Figura 16. Diagrama de sequência

No diagrama da Figura 16, é evidenciado a necessidade de configurar a fonte do sinal antes de iniciar uma sessão. Após selecionar uma fonte de sinal, os dados são enviados para o Módulo de Controle e são serializados em um formato que possa ser processado pelo Módulo de Classificação.

Com o sistema configurado com um sinal de entrada, o usuário poderá iniciar uma sessão. Assim que for iniciado a sessão, o Módulo de Controle irá encaminhar ao Módulo de Classificação um *Stream* de dados. O fluxo de dados será continuamente processado e classificado até que haja a interrupção do usuário ou da fonte de sinal.

* 1. Plano de testes

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso** | **Passo** | **Descrição** | **Resultado esperado** |
| UC01 | 01 | Acessar configurações de fonte do sinal | Exibir opções de arquivo ou dispositivo |
| 02 | Selecionar uma das opções, de arquivo, ou dispositivo. | Caso for selecionado a opção de arquivo, deverá abrir o menu para selecionar um arquivo do computador. Caso for selecionado a opção de um dispositivo, deverá listar os dispositivos disponíveis para conectar à aplicação. |
| 03 | Carregar a fonte do sinal, da opção selecionada (arquivo ou dispositivo). | O sistema irá mostrar pela interface, indícios que o mesmo foi carregado corretamente no sistema. |
| UC02 | 01 | Com uma fonte já carregada, clicar na opção para iniciar a sessão | O sistema irá mostrar pela na interface o início da sessão. |
| 02 | Visualizar resultados de classificação | O sistema irá mostrar na interface o resultado de classificação. |
| UC03 | 01 | Iniciar uma sessão | O sistema irá mostrar pela na interface o início da sessão. |
| 02 | Pausar uma sessão | O sistema irá mostrar pela na interface a pausa da sessão. Os resultados de classificação também serão pausados. |
| 03 | Retomar uma sessão | Retomar onde havia pausado a sessão. No caso de uma fonte do tipo arquivo, a classificação irá continuar onde parou. |

1. Desenvolvimento
   1. Análise de *datasets*

Para fins de comparação entre diferentes pesquisas realizadas, algumas pesquisas compartilharam publicamente os *datasets* utilizados para serem utilizadas na comunidade acadêmica. No contexto de reconhecimento de emoções a partir de sinais EEG, os *datasets* mais relevantes encontradas são a [DEAP](https://www.eecs.qmul.ac.uk/mmv/datasets/deap/doc/tac_special_issue_2011.pdf), MAHNOB-HCI e SEED IV.

* + 1. DEAP: A Database for Emotion Analysis using Physiological Signals

O DEAP *dataset* foi disponibilizado com a participação de 32 indivíduos, onde foram registrados sinais de EEG e outros sinais fisiológicos. Cada participante realizou uma sessão que incluiu 40 trechos de videoclipe com 1 minuto de duração. Após as sessões os participantes reportavam sua experiência em termos valência e excitação (Modelo PAD), como também a experiência em termos de gosto pessoal e familiaridade com os trechos de vídeo assistidos.

* + 1. MAHNOB HCI

O MAHNOB HCI *dataset* foi um dos primeiros *datasets* publicamente compartilhados com comunidade cientifica com o objetivo de identificar emoções a partir de diferentes sinais fisiológicos, incluindo a EEG. No total de 30 indivíduos participaram em diferentes sessões. Nas sessões gravadas foram visualizados fragmentos de filmes e fotos, enquanto os monitoravam com 6 câmeras de vídeo, incluindo gravação de voz, rastreamento ocular, além de sensores fisiológicos que mediam ECG, EEG (32 canais), respiração e temperatura da pele.

* + 1. SEED IV

O *dataset* SEED IV, uma versão melhorada do *dataset* SEED, apresenta sinais fisiológicos de 15 participantes, incluindo sinais de EEG com 62 canais e rastreamento ocular. Os participantes assistiram 72 trechos de filmes com uma média de 2 minutos cada. Os trechos foram selecionados de forma rigorosa, incluindo somente os que apresentavam alto índice de indução emocional, o qual foi estabelecido por um estudo preliminar. Os trechos selecionados têm como objetivo provocar as emoções de alegria, tristeza, temor e emoção neutra. As sessões foram realizadas em períodos diferentes (diferença de alguns dias) para cada participante, assim para evitar qualquer possível fadiga e viés no sinal adquirido.

* + 1. Comparações e escolha de *dataset*

Visto as principais características apresentadas dos *datasets*, o SEED IV é o mais apropriado para este trabalho, já que o mesmo apresenta as características de classificação que procuramos. Apesar o SEED IV apresenta um número menor de participantes do que os outros *datasets*, este apresenta mais tempo de amostragem por participante, como também organização de sessões e análise mais crítica dos dados que podem possibilitar melhores resultados.

* 1. ferramentas e ambiente de prototipagem

Nesta seção é discutido as linguagens e ferramentas utilizadas para a prototipação, treinamento, e testes do modelo de aprendizado de máquina.

* + 1. Jupyter Notebook

O Jupyter notebook um ambiente de desenvolvimento que permite a rápida prototipação e execução de trechos de código Python. A ferramenta é bastante popular entre os profissionais de ciência de dados, devido a sua facilidade de uso, que permite ser usado em tarefas de organização de dados, modelagem estatística, execução de modelos de aprendizado de máquina, como também a plotagem de grafos e outros casos de uso para a computação científica.

O Jupyter notebook foi desenvolvido a partir do IPython, que é uma linha de comando interativa para execução de Python. O IPython é o motor de execução do Jupyter notebook, esta proporciona ferramentas de alta desempenho para a computação paralela e distribuída.

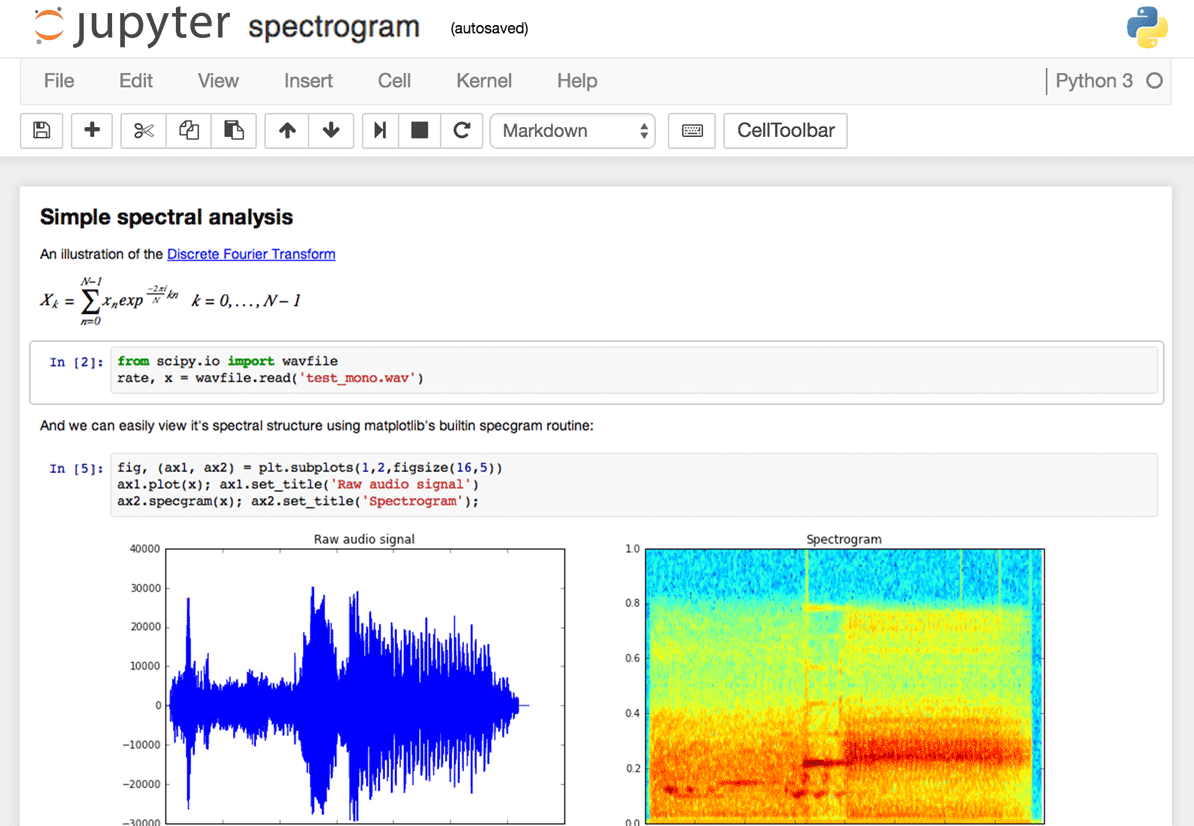


Figura 17. Um exemplo de Jupyter notebook que demonstra a inserção de texto e renderização de formulas para fins de documentação, como também a possibilidade plotar gráficos de forma *inline,* sem ter que abrir uma nova janela.

Fonte: Devlin ([s.d.])

O diferencial introduzida pela Jupyter Notebook em comparação de apenas executar códigos em Python, é a possiblidade de executar Python em modo de REPL (Read Eval Print Loop) com ferramentas de edição de texto e visualização dentro do próprio editor, o qual permite a execução de trechos de código que mantém o estado das variáveis (de forma similar ao MATLAB). Vale salientar que, o Jupyter Notebook é um ambiente de exploração de dados, e não uma ferramenta para desenvolvimento de programas que contém estruturas complexas de lógica de programação, que são mais apropriados para ambientes de desenvolvimento integrado (IDE – *Integrated Development Environment*).

* + 1. Bibliotecas em Python para computação científica

No ecossistema do Python, existem dezenas de bibliotecas de código aberto que são utilizadas na exploração científica. As mais notáveis são a Numpy, Scipy, Matplotlib, Pandas e Scikit-Learn. As bibliotecas mencionadas resolvem problemas distintos, portanto não competem entre si, e geralmente são utilizados em conjunto. Aqui segue uma breve descrição dos objetivos das bibliotecas, e o seus casos de uso:

* A biblioteca Numpy é considerada a fundação da computação cientifica no Python. A Numpy supri as operações matemáticas básicas para a manipulação de vetores multidimensionais, que permitem realizar operações com matrizes e outras operações utilizadas na álgebra linear. Porém, como o Python é uma linguagem interpretada, operações em grandes conjuntos de dados podem sofrer problemas de desempenho. Por isso, a Numpy implementa estrutura de dados e operações matemáticas diretamente na linguagem C. A comunicação entre a linguagem C e Python é realizada por meio do uso da linguagem Cython, que é considerada uma extensão da linguagem Python que permite a importação módulos implementados na linguagem C. Além da computação em baixo nível, os módulos em C realizam operações vetorizadas, isto é, elas aproveitam de instruções *SIMD* (Single Instruction, Multiple Data) que usam registradores dedicados da CPU que são capazes de realizar operações matemáticas em paralelo. Todas essas otimizações permitem a realização de operações em grandes conjuntos de dados com alto desempenho.
* A biblioteca Scipy utiliza o Numpy como base para suas operações. A Scipy fornece várias API’s comumente utilizadas na computação cientifica, como cálculo integral, cálculo estatístico, processamento de sinais e outros. Para esta pesquisa, vamos utilizar as API’s de FFT e cálculo espectral.
* A biblioteca Matplotlib tem como objetivo a criação de gráficos interativos que podem ser 2D ou 3D. Ela tem uma API similar a do MATLAB, e suporta as estruturas de dados fornecidos pelo Numpy. Ela também fornece uma API orientado a objetos, que fornece maior flexibilidade para aplicações de propósito geral.
* A biblioteca Pandas apresenta funcionalidades de manipulação, indexação, agrupamento e filtro de dados. Esta usa o Numpy internamente, então consegue realizar operações de forma eficiente. A biblioteca Pandas é comparado com a linguagem SQL, no quesito de comandos DML (Data Manipulation Language), e é comumente utilizada em projetos de ciência de dados.
* A Scikit-Learn é uma biblioteca de código aberta de aprendizado de máquina. Ela fornece várias API’s para execução de vários algoritmos como a regressão linear, SVM, Redes Neurais e outros. Além de fornecer os algoritmos de aprendizado de máquina, esta incluí ferramentas para tratamento de dados como escalonamento e normalização.
  1. Tratamento do *dataset*

O SEED IV *dataset* consiste em 3 sessões realizados em dias diferentes, onde cada sessão consiste em 24 ensaios. Como houve a participação de 15 indivíduos, no total de 1080 ensaios foram registrados. Para cada ensaio é apresentado um vídeo clipe de aproximadamente 2 minutos de duração com objetivo de estimular uma das 4 emoções: alegria, tristeza, temor ou neutro. As 4 emoções foram igualmente distribuídas nos 24 ensaios, dessa forma, há 6 ensaios por emoção em uma dada sessão.

No *dataset*, as 3 sessões estão divididas por pastas, e dentro de cada pasta estão 15 arquivos de dados do MATLAB que representam os ensaios realizados pelos 15 indivíduos. Em cada arquivo, está registrado 24 ensaios realizados por um indivíduo, e para cada ensaio, está registrado os sinais EEG de 62 canais. O *dataset* está bem organizado, porém está em uma estrutura relativamente complexa. Para podermos realizar qualquer tipo de processamento nos sinais, uma estrutura mais simples e direta é necessária.

As primeiras considerações averiguadas antes de realizar a leitura do *dataset*, são a ordem dos eletrodos no arquivo de dados do MATLAB e a ordem dos ensaios a respeito da emoção sendo estimulada. Por exemplo, os 62 canais de dados estão organizados por índice de vetor e não com os nomes das posições dos eletrodos do sistema internacional 10-20. Portanto, para realizar qualquer tipo de busca e processamento nos dados uma reestruturação precisa ser realizada.

O Quadro 10 demonstra como foi realizado a leitura do *dataset* para permitir a nomeação dos eletrodos, informações de ensaio e sessão. Nesta função, as pastas do *dataset* são percorridas e os dados dos arquivos são carregados na memória. As variáveis armazenadas nos arquivos do MATLAB são extraídas e são armazenadas em uma nova estrutura de dados que permite a indexação dos dados com mais facilidade. A Figura 18 mostra a estrutura de dados após a leitura parcial do *dataset*.

Quadro 10. Leitura do *dataset*

|  |
| --- |
| […]  def read\_dataset(dataset\_dir, session\_list=None, user\_list=None, trial\_list=None):      column\_names = ['session', 'user', 'trial'] + electrode\_names      df\_eeg\_session\_trial\_list = []      with os.scandir(dataset\_dir) as folders:          for folder in list(folders):              session = folder.name              if session\_list != None and int(session) not in session\_list:                  continue                with os.scandir(folder) as files:                  for file in list(files):                        user = file.name.split('\_')[0]                      if user\_list != None and int(user) not in user\_list:                          continue                        mat = spio.loadmat(file.path)                      for trial in trials:                          if session\_labels[session][trial] not in allowed\_session\_labels:                              continue                          if trial\_list != None and int(trial) not in trial\_list:                              continue                            trial\_key = find\_regex\_index(r'^.\*eeg{}$'.format(trial+1), mat)                          mat\_subset = [val for i, val in enumerate(mat[trial\_key]) if i in electrode\_names\_indices]                            df\_eeg\_session\_trial = pd.DataFrame(np.array(mat\_subset).T)                          df\_eeg\_session\_trial.columns = electrode\_names                          df\_eeg\_session\_trial.insert(0, column='session', value=int(session))                          df\_eeg\_session\_trial.insert(1, column='user', value=int(user))                          df\_eeg\_session\_trial.insert(2, column='trial', value=int(trial))                            df\_eeg\_session\_trial\_list.append(df\_eeg\_session\_trial)      return pd.concat(df\_eeg\_session\_trial\_list).reset\_index(drop=True) |

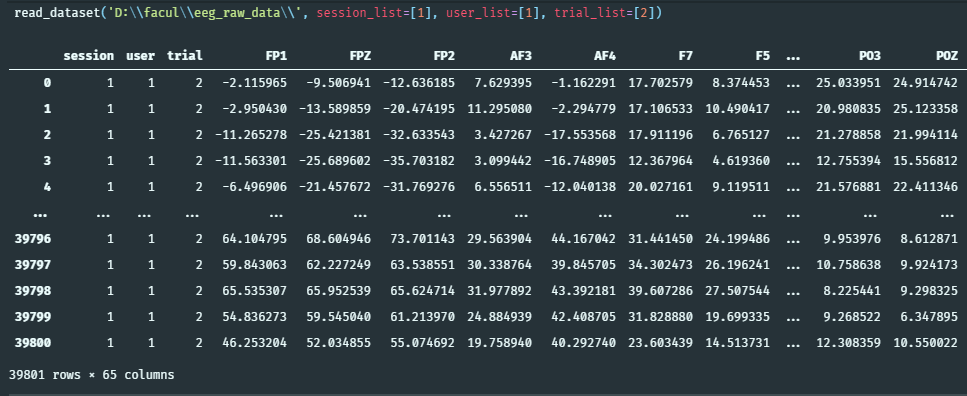


Figura 18. Estrutura de dados após leitura do *dataset*.

* 1. Geração de características

Como o sistema proposto precisa funcionar em um ambiente online, a geração de características e a predição precisam ser realizadas de forma incremental, isto é, conforme a chegada de sinais ao sistema. Para isso, é necessário definir parâmetros de largura e sobreposição de uma janela deslizante, que irá percorrer os sinais do *dataset*.

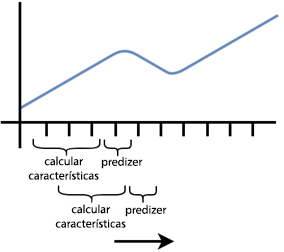


Figura 19. Mecanismo de geração de características e predição online

Nos trabalhos relacionados, os parâmetros de largura de janela e sobreposição não estão padronizados e diferem entre as pesquisas. Por exemplo, em ZHENG et al. (2019) foi utilizado uma janela de 4 segundos sem sobreposição. Já em HOU et al. (2016) foi utilizado uma janela de 4 segundos com 3 segundos de sobreposição. Em LIU et al. (2018) foi utilizado uma janela de 2 segundos com sobreposição de 1 segundo. Portanto, não há um tamanho padronizado referente a estes parâmetros, por esse motivo iremos experimentar com vários tamanhos de janela diferentes, e utilizar o que reportar os melhores resultados de classificação.

A cada instância de uma janela percorrida no sinal, características que descrevem a porção do sinal são calculadas. Uma das características comumente utilizadas em sistemas BCI é a densidade espectral, onde é calculada a energia do sinal no domínio da frequência, esta característica é definida pela Equação 9. Neste trabalho, é utilizado a densidade espectral para calcular as energias das diferentes bandas definidas no Quadro 4. A escolha das características e das bandas coincidem com o trabalho dos autores do *dataset* (ZHENG et al., 2019), assim para ter uma linha de comparação direta aos resultados obtidos.

Para o cálculo da densidade espectral, foi utilizado a implementação no Quadro 11, onde uma função recebe como parâmetro um vetor multidimensional *fft\_coefs*, que contém os coeficientes de FFT calculada previamente em um vetor de sinais. Os outros parâmetros são referente ao tamanho da FFT, frequência de amostragem, banda e finalmente fator de normalização. A vantagem dessa implementação é que permite o cálculo vetorizado da densidade espectral de vários sinais de forma simultânea, evitando a necessidade de utilizar laços de repetição que iriam degradar o desempenho de execução.

Quadro 11. Cálculo da Densidade Espectral

|  |
| --- |
| […]  def multi\_psd\_band(fft\_coefs, N, sf, band, norm\_factor):  # get band boundaries  low, high = band  # get frequency axis  freqs = fftfreq(N, 1/sf)  # get indexes for the given frequnecy band  idx\_band = np.logical\_and(freqs >= low, freqs <= high)  # gotta rescale fft since half of the amplitude is in negative frequencies  scaling\_factor = 2    # calculate the psd  psd = np.abs(fft\_coefs[:, idx\_band])\*\*2  # return scaled and normalized psd, since we are summing across  # a frequency band, we need to average it by the N fft bins  return np.sum(scaling\_factor \* psd / norm\_factor, axis=1) / N |

Os sinais do *dataset* contém uma frequência de amostragem de 200Hz, e segundo teorema de Nyquist, a frequência máxima que pode ser representada no domínio da frequência é de 100Hz. Como será utilizado apenas a banda de 1-50Hz para calcular as caraterísticas, um filtro passa-banda é aplicado nos intervalos utilizados. Dessa forma, é possível remover quaisquer artefatos de alta e baixa frequência (incluindo DC) do sinal.

Para gerar as coeficientes de FFT dos sinais foi utilizado a função fft da biblioteca Scipy. Como a implementação usa uma janela de tempo que representa uma fração do tempo total de um ensaio, o cálculo da FFT pode sofrer efeitos de janelamento, como o vazamento espectral (*Spectral Leakage*). Para minimizar os efeitos do vazamento espectral, é necessário multiplicar o sinal com uma função janela antes de realizar a FFT. Nesse caso, é utilizado uma janela Hann já que este é amplamente utilizado em trabalhos correlatos. A implementação da geração de características para um único canal é demonstrado no Quadro 12.

Quadro 12. Pre-processamento e geração de características

|  |
| --- |
| […]  strides = stride\_windows(df.loc[:, column\_name], win\_size, overlap\_size, axis=1)    b, a = band\_pass\_coefs(lowcut=1, highcut=50, sf=sf, order=5)  win = hann(win\_size)    strides\_filt = signal.filtfilt(b, a, strides)  strides\_fft = fft(win\*strides\_filt)  norm\_factor = sf \* (np.abs(win)\*\*2).sum()  psd\_band\_features = np.array([multi\_psd\_band(strides\_fft, N=win\_size, sf=sf, band=(band), norm\_factor=norm\_factor) for band in bands]).T  […] |

A implementação parcial no Quadro 12 utiliza uma função auxiliar *stride\_windows* que divide os dados de um canal em um vetor multidimensional de janelas do tamanho *win\_size* com sobreposição de *overlap\_size*. A vantagem dessa implementação, é que a função *stride\_windows* retorna uma estrutura de memória compartilhada, sem realizar uma cópia dos dados. A estrutura está num formato vetorizável, portanto as operações de filtragem, multiplicação e FFT podem ser calculadas eficientemente.

Para gerar as características de todos os ensaios é necessário percorrer o *dataset* por completo, porém, devido ao tamanho do *dataset* de aproximadamente 6,5GB não é possível carregá-lo completamente na memória com o equipamento utilizado. Portanto, é necessário ler e calcular de forma incremental. A implementação completa está no Apendice A.

* 1. Treinamento de algoritmos

Para o desenvolvimento dos algorítmos de aprendizado de máquina, foi utilizado a biblioteca Scikit-Learn. A biblioteca Scikit-Learn contém uma grande variedade de API’s para a modelagem e validação de algorítmos de IA, como também utilidades para a manipulação e a distribuição de dados. Nessa seção, será discutido as etapas realizadas para o treinamento e validação dos algoritmos utilizados.

* + 1. Considerações de organização do *dataset*

Para o treinamento de algorítmos de aprendizado de máquina, foi utilizado o padrão de divisão em 3 partes: treinamento, validação e teste.

* O *dataset* de treinamento é utilizado para realizar a etapa de aprendizado do algorítmo.
* Já o *dataset* de validação serve para validar os hiperparâmetros escolhidos na etapa de treinamento. Essa etapa serve para a escolha de diferentes hiperparâmetros do modelo utilizado, assim para obter o modelo com a melhor acurácia contra o *dataset* de validação. O principal motivo de ter um *dataset* de validação é para evitar quaisquer víes na escolha dos hiperparâmetros ao testar o *dataset*.
* O *dataset* de teste serve para avaliar a acurácia do modelo contra amostras não vistas.

A proporção das partes utilizadas foram de 60% para treinamento, 20% para validação e 20% para teste, onde as proporções estão agrupadas por ensaio. Também, as proporções das classes são estratificadas, isto é, cada *dataset* contém a mesma proporção de classes, desta forma os *datasets* não estão desiquilibrados em respeito das classes.

Antes de realizar a divisão do *dataset*, é comum embaralhar o *dataset* para evitar com que o modelo aprenda qualquer víes na ordenação das amostras. Porém, para o *dataset* deste trabalho, seria um erro embaralhar todas as amostras das características calculadas. Como o *dataset* está agrupado por ensaios, a mistura de amostras de diferentes ensaios irá fazer com que o modelo aprenda a variância do *dataset* de teste ou validação na etapa de treinamento, e consequentemente, irá reportar acurácias ótimistas que não são válidas. Portanto, o embaralhamento precisa ocorrer em nível de ensaio e não diretamente nas amostras.

* + 1. Modelos e hiperparâmetros

Modelos de aprendizado de máquina são inicializados a partir da escolha de diferentes hiperparâmetros. Os hiperparâmetros definem como um modelo irá funcionar na etapa de treinamento e precisam ser ajustados para adquirir o melhor modelo possível. No Quadro 13, são listados os modelos utilizados e os hiperparâmetros que foram variados para extrair o melhor modelo para a aplicação.

Quadro 13. Modelos e hiperparâmetros

|  |  |  |
| --- | --- | --- |
| **Modelo** | **Hiperparâmetros** | **Descrição** |
| Regressão Logística | C | Controla o nível de regularização. Valores menores restringem o modelo contra alta variância. |
| Rede Neural (MLP) | Nº de camadas e neurônios | Define o tamanho da rede neural. Quanto maior a rede mais complexidade e variância aprendida. |
| Alpha | Controla o nível de regularização. |
| SVM | Kernel | Define a complexidade dos hiperplanos. |
| C | Controla o nível de regularização. Valores menores restringem o modelo contra alta variância. |
| Gamma | Controle de regularização. Define a influência das amostras no qual os vetores de suporte se baseiam. |
| LDA | N/A | Modelo sem hiperparâmetros |
| Random Forest | Quantidade de arvores/estimadores na floresta | Define a quantidade de arvores de decisão que são utilizadas. |

Para encontrar o modelo com a maior acurácia, é preciso percorrer um espaço multidimensional de combinações de hiperparâmetros. Para cada combinação, é realizado o treinamento e medido a acurácia do modelo. Isto pode ser feito em alguns laços de repetição aninhados, porém, como o processo de treinamento é paralelizável, o tempo de busca dos melhores hiperparâmetros pode ser minimizado com a execução paralela de processos. Este funcionamento é implementado na classe *GridSearchCV* do Scikit-Learn, e sua utilização é detalhada na próxima seção.

* + 1. Validação cruzada

Como o *dataset* não é grande suficiente para ser uma população representativa do universo de amostras, o treinamento e validação de um modelo em uma divisão arbitrária do *dataset* faz com que o modelo não generaliza corretamente para amostras ainda não vistas. Para amenizar esse problema, métodos de validação cruzada (*cross validation*) são empregadas.

A validação cruzada consiste no particionamento do *dataset* em vários subconjuntos, onde é realizado o treinamento e validação do modelo. Existem diversas variações desta técnica, porém, a utilizada neste trabalho é a *K-Fold*.A validação cruzada *K-Fold* é realizada pela organização do *dataset* em K conjuntos do mesmo tamanho. Cada conjunto contém K particionamentos, onde K-1 dos particionamentos é utilizado para treinamento e uma para validação.

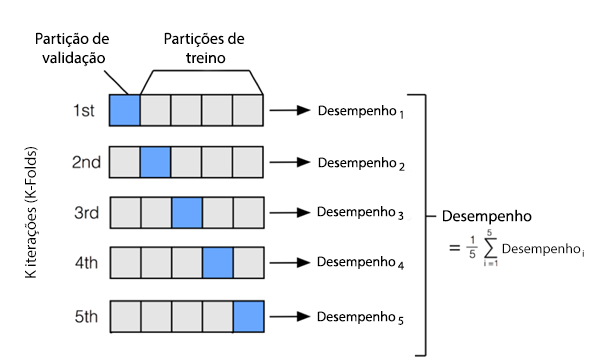


Figura 20. Validação cruzada *K-Fold*

Fonte: Adaptado de ETHEN ([s.d.])

Conforme demonstrado na Figura 20, validação cruzada realiza o treinamento e validação do modelo para cada um dos conjuntos. Em seguida, é cálculado a média da métrica de desempenho sendo medida, que no caso deste trabalho é a acurácia.

A validação cruzada *K-Fold* foi utlizada para determinar os melhores hiperparâmetros para um dado modelo. Porém, para testar a acurácia real, é preciso testar o melhor modelo encontrado contra amostras não vistas, portanto, é necessário usar um *dataset* de teste. Como a escolha do *dataset* de teste também é arbitrária, e pelos mesmos motivos que foram utilizados a validação cruzada *K-Fold*, uma segunda validação cruzada *K-Fold* é utilizada para testar a acurácia real dos modelos. Esta estrategia é denominada de validação cruzada aninhada (Nested Cross-Validation).

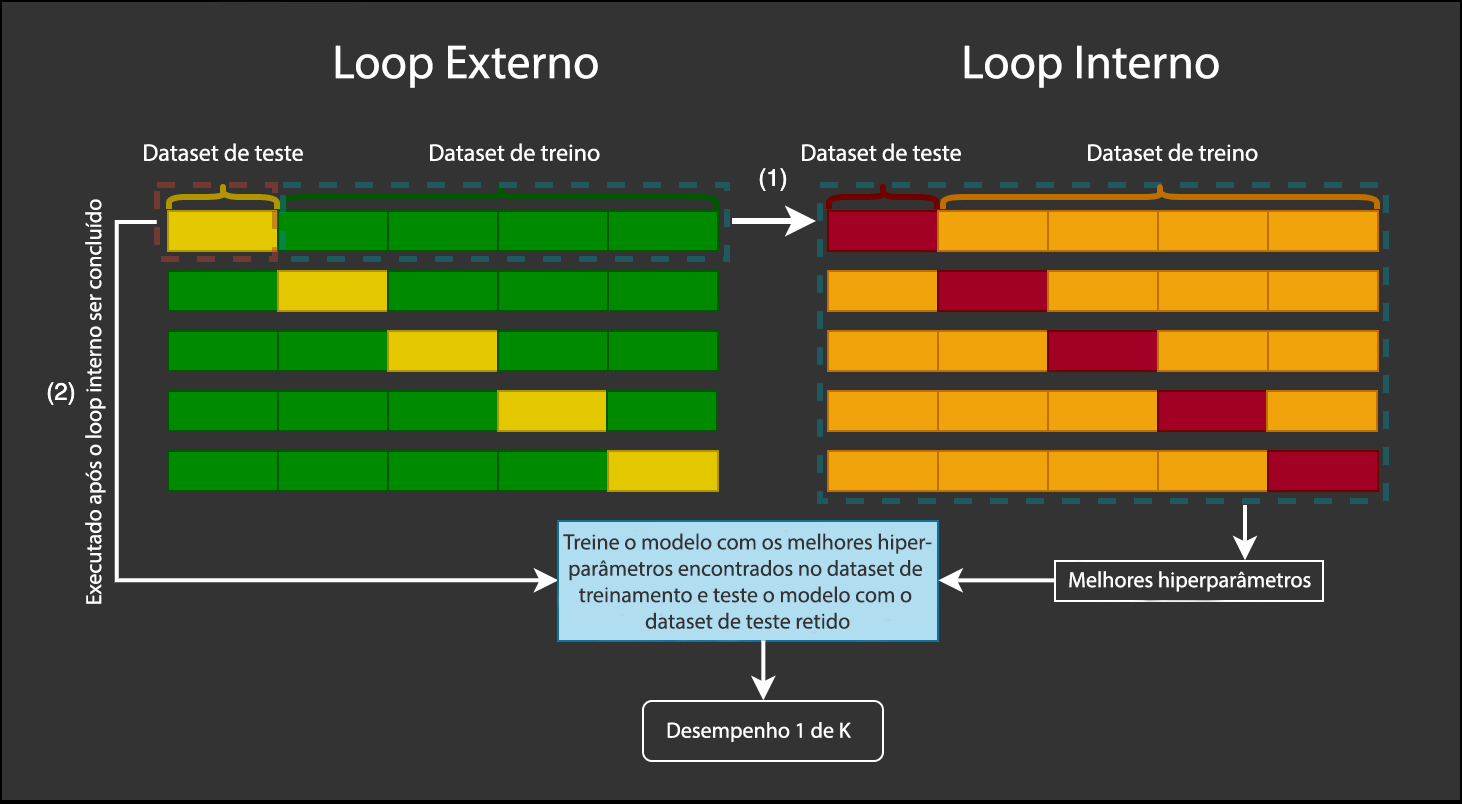


Figura 21. Validação cruzada aninhada.

Fonte: adaptado de HANSEN (2019)

Na Figura 21, é demonstrado a visão geral do método de validação cruzada. Com esta organização, é possível obter uma acurácia mais realistica do melhor modelo adquirido.

O Quadro 14 demonstra a implementação do treino e validação de vários modelos e seus hiperparâmetros. A classe *RepeatedStratifiedGroupKFold* implementa a validação cruzada estratificada, com suporte à agrupações (nesse caso, agrupados por ensaio). A implementação completa está no apendíce.

Quadro 14. Modelos, hiperparâmetros e validação cruzada

|  |
| --- |
| […]  models = {      'LinearDiscriminantAnalysis': LinearDiscriminantAnalysis(),      'RandomForestClassifier': RandomForestClassifier(),      'LogisticRegression': Pipeline([          ('scaling', StandardScaler()),          ('estimator', LogisticRegression()),      ]),      'MLPClassifier': Pipeline([          ('scaling', StandardScaler()),          ('estimator', MLPClassifier(solver='lbfgs')),      ]),      'LinearSVC': Pipeline([          ('scaling', StandardScaler()),          ('estimator', LinearSVC())      ]),      'SVC': Pipeline([          # ('feat', SelectKBest(mutual\_info\_classif, k=20)),          ('scaling', StandardScaler()),          ('estimator', SVC())      ])  }  params = {      'LinearDiscriminantAnalysis': {},      'LogisticRegression': {'estimator\_\_C': [0.01, 0.03, 0.1, 0.3, 1, 3]},      'MLPClassifier': {          'estimator\_\_hidden\_layer\_sizes': [(15,10), (10,5), (5,3),(30,15,10), (20,15,10), (15,10,5)],          'estimator\_\_alpha': np.logspace(-1, -7, num=7)      },      'RandomForestClassifier': { 'n\_estimators': [16, 32, 48, 100] },      'LinearSVC': [          {'estimator\_\_C': np.logspace(5, -5, base=2, num=10)},      ],      'SVC': [          {'estimator\_\_C': np.logspace(15, -15, base=2, num=31), 'estimator\_\_gamma': np.logspace(8, -8, base=10, num=17)},      ]  }  […]  cv\_grp = RepeatedStratifiedGroupKFold(n\_splits=5, n\_repeats=1)  outer\_cv\_grp = RepeatedStratifiedGroupKFold(n\_splits=5, n\_repeats=1)  helper = EstimatorSelectionHelper(models, params)  helper.fit(X, y, scoring='accuracy', n\_jobs=-1, cv=cv\_grp, outer\_cv=outer\_cv\_grp, verbose=10, groups=groups) |

Quadro 15. Classe *EstimatorSelectionHelper*

|  |
| --- |
| […]  class EstimatorSelectionHelper:  […]  def fit(self, X, y, cv=3, n\_jobs=3, verbose=1, scoring=None, refit=False, groups=None, outer\_cv=None, persist\_dir=None, randomSearchFor=None):  for key in self.keys:  model = self.models[key]  params = self.params[key]  if randomSearchFor != None and key in randomSearchFor:  SearchCV = RandomizedSearchCV  else:  SearchCV = GridSearchCV  gs = SearchCV(model, params, cv=cv, n\_jobs=n\_jobs,  verbose=verbose, scoring=scoring,  refit=True,  return\_train\_score=True)  if outer\_cv != None:  print("Running GridSearchCV for %s with nested cross validation." % key)  self.cross\_val\_results[key] = cross\_validate(gs, X, y, groups=groups, cv=outer\_cv, scoring=scoring,  return\_train\_score=True, return\_estimator=True, verbose=10, fit\_params={'groups': groups})  self.ran\_with\_outer\_cv = True  gs = max( |

No quadro 15 é exibido a implementação parcial da classe *EstimatorSelectionHelper*. Esta classe é utilizada para percorrer e treinar todos os modelos e salvar resultados de teste para futura consulta. Esta chama a classe *GridSearchCV* que realiza a busca de hiperparâmetros listados no Quadro 14, respeitando as partições da validação cruzada dada pela variável *cv*. A chamada à função *cross\_validate* realiza a validação cruzada exterior, dessa forma se obtêm o funcionamento de validação cruzada aninhada desejado.

* + 1. Resultados e metricas de desempenho

A escolha da janela e sobreposição para o sistema foi realizada a partir de resultados preliminares de classificação. Foram experimentados vários tamanhos e sobreposições de janela conforme mostrado no Quadro 16. Os autores do *dataset* realizaram o processamento offline dos dados com uma janela de 4 segundos sem sobreposição. Já neste trabalho, foi utilizado uma janela de 4 segundos, com 2 segundos de sobreposição.

Quadro 16. Tamanho de janela e sobreposição

|  |  |  |
| --- | --- | --- |
| **Janela (segundos)** | **Sobreposição (segundos)** | **Acurácia média entre os modelos utilizados com 3 usuários de uma sessão** |
| 1 | 0 | 46,67% |
| 1 | 0,5 | 47,28% |
| 2 | 0 | 47,67% |
| 2 | 1 | 47,53% |
| 4 | 0 | 45,12% |
| **4** | **2** | **47,91%** |

Para resultados de classificação de EEG, existem diferentes métricas utilizadas em publicações. Devido a dinâmica da EEG, diferentes comparações são realizadas. Dentro das utilizadas em artigos, tem os resultados de sistemas sujeito dependentes e independentes.

Para resultados sujeito dependente, modelos de classificação são otimizados e classificados contra um único usuário por vez, sendo assim, existe um classificador por usuário. No Quadro 17 são listados os melhores modelos encontrados por usuário.

Quadro 17. Melhores modelos por usuário

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Usuário** | **LDA** | **Random Forest** | **Regressão Logística** | **Rede Neural (MLP)** | **SVM** |
| 1 | 40,03 | 44,41 | 40,49 | 39,19 | 41,31 |
| 2 | 64,36 | 65,92 | 62,83 | 60,62 | 63,78 |
| 3 | 46,22 | 44,6 | 43,47 | 42,24 | 44,9 |
| 4 | 59,56 | 67,28 | 61,42 | 61,31 | 64,49 |
| 5 | 52,86 | 56,09 | 50,41 | 50,02 | 50,85 |
| 6 | 64,58 | 68,92 | 67,97 | 66,18 | 67,56 |
| 7 | 58,53 | 61,71 | 59,93 | 54,74 | 60,84 |
| 8 | 60,31 | 62,07 | 62,39 | 60,01 | 61,66 |
| 9 | 55,17 | 52,82 | 45,17 | 53,66 | 52,56 |
| 10 | 52,22 | 56,93 | 54,39 | 52,97 | 52,88 |
| 11 | 49,71 | 45,09 | 48,2 | 43,4 | 49,8 |
| 12 | 42,11 | 46,5 | 41,32 | 42,33 | 44,29 |
| 13 | 54,97 | 52,16 | 52,03 | 49,17 | 54,03 |
| 14 | 52,68 | 55,69 | 54,55 | 52,44 | 53,95 |
| 15 | 73,67 | 75,93 | 76,19 | 71,23 | 76,18 |

A melhor acurácia média obtida para um usuário foi de 76,19% com o modelo de Regressão Logística (usuário 15), e a melhor acurácia média para todos os usuario de 57,07% com o modelo de Random Forest. Esse resultado é similar aos resultados dos autores do *dataset*, que obtiveram acurácia média de 57,70% para características de densidade espectral com um modelo SVM.

* 1. Desenvolvimento de software

O projeto de software originalmente propõe o desenvolvimento da solução completa de captura de sinais e controles de uma simulação, porém, após análise mais profunda das funcionalidades de softwares existentes utilizadas pela comunidade, foi visto que alguns já implementam as funcionalidades necessárias para dar suporte à aplicação. Dos softwares analisados, o mais notável foi o OpenBCI. O OpenBCI é um aplicativo de código aberto, que permite a captura de sinais, e permite a visualização de informações dos dados em tempo real. O software permite a criação de visualizações personalizadas a partir da criação de *widgets* customizados.

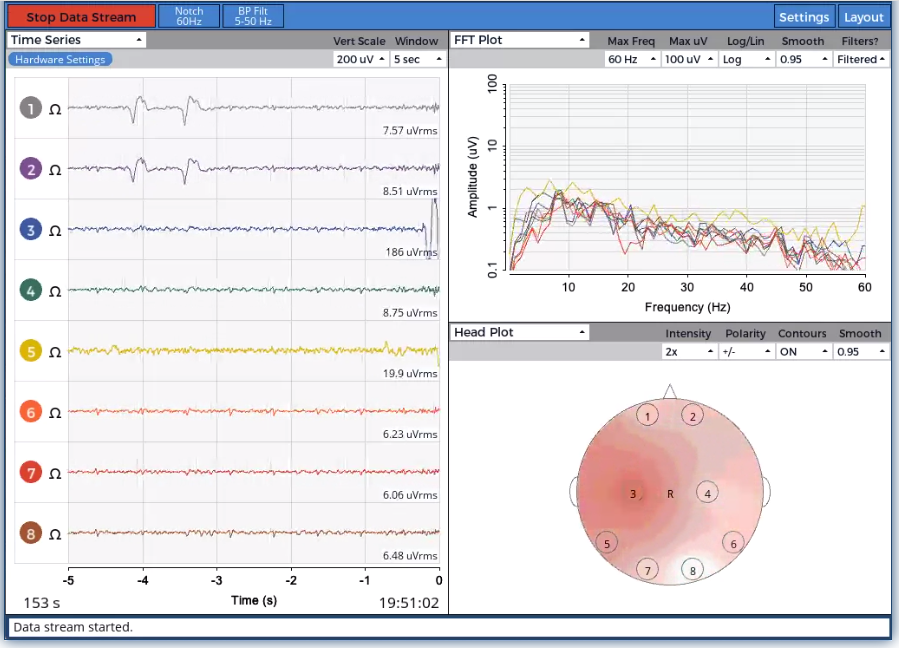


Figura 22. Interface gráfica do OpenBCI

Devido a ampla adoção do software OpenBCI na comunidade ciêntifica, e como esta já implementa alguns requisitos do software proposto, a aplicação foi desenvolvida na plataforma do OpenBCI. Dessa forma, trabalhos futuros podem usufruir das funcionalidades realizadas nesse projeto.

* + 1. Software OpenBCI

A OpenBCI está implementada na plataforma de Processing, que é um ambiente de aprendizado de programação no contexto de artes visuais, este também é de código aberto. A linguagem de programação utilizada para criar aplicações dentro plataforma Processing é a linguagem Java com API’s extras específicas à plataforma.

* + 1. Desenvolvimento do *widget* no OpenBCI

A interface criada é simples, e tem o simples objetivo de mostrar a classe sendo classificada no tempo de execução. A Figura 23 mostra a inteface desenvolvida para mostrar a classificação.

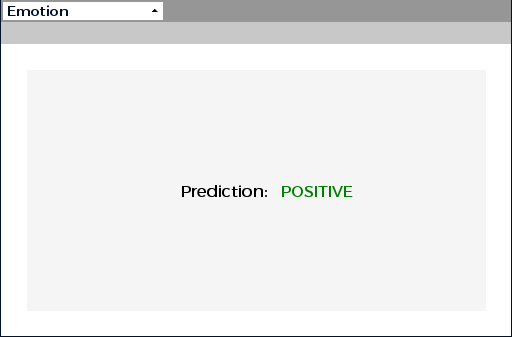


Figura 23. Interface gráfica do widget criado.

A organização de software da OpenBCI está dirigido a compartilhamento de variáveis globais, que são disponibilizadas para todos os *widgets* que tem no software. Apesar da documentação escassa para o desenvolvimento de *widgets*, após profunda inspeção do código fonte e sua execução, foi possível identificar duas váriaveis globais essenciais ao projeto, *isRunning* e *yLittleBuff\_uV*:

* *isRunning*: indica que novos dados dos canais estão disponíveis.
* *yLittleBuff\_uV*: buffer que contém as novas tensões captadas dos canais em uV.
  + 1. Comunicação entre Java e Python

Para enviar os dados para o contexto de execução do Python, foi configurado uma conexão TCP para o envio e recebimento de dados. Os dados são codificados e decodicados no formato JSON para obter interoperabilidade entre os diferentes contextos de execução ao transmitir e receber dados. O Quadro 18 mostra como é realizado a conexão, envio e recebimento dos dados no contexto de execução do Java.

Quadro 18. Comunicação entre contextos de execução de Java e Python.

|  |
| --- |
| […]  // configura uma conexão com o servidor  myClient = new Client(ourApplet, "127.0.0.1", 5204);  […]    if(isRunning) {  myClient.write(new org.json.JSONArray(yLittleBuff\_uV).toString());  }  if (myClient.available() > 0) {  JSONObject json = parseJSONObject(myClient.readString());  println(json.getInt("message"));  prediction = json.getInt("message");  }  […] |

Para o recebimento de dados no Python, é configurado um servidor TCP para receber e enviar dados da aplicação em Java. O Quadro 19 demonstra um resumo da configuração do servidor, recebimento e envio de dados.

Quadro 19. Servidor TCP no Python

|  |
| --- |
| HOST = '' # Symbolic name meaning all available interfaces  PORT = 5204 # Arbitrary non-privileged port  s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  s.bind((HOST, PORT))  s.listen(1)  conn, addr = s.accept()  print('Connected by', addr)  [...]  while True:  [...]  data = conn.recv(102400)  [...]  time\_signals = json.loads(data)  [...]  jsonResponse = {  "message": int(prediction[0])  }  conn.send(json.dumps(jsonResponse).encode('utf-8')) |

* 1. Medidas de Execução

Para verificar a possibilidade de execução em tempo real, foi medido o tempo de execução a partir da disponibilidade dos sinais em buffer na memória conforme a implementação em Python. No Quadro 20, foi realizado a captura de tempo de execução para um modelo SVM. O cálculo também inclui o tempo de execução para calcular as características para 6 canais de dados, com uma janela de 4 segundos. Para referência, os testes foram realizados em um computador com processador Intel Core i5-7400 @ 3.00Ghz.

Quadro 20. Medindo tempo de execução

|  |
| --- |
| […]  start\_time = time.perf\_counter()  features = calculateFeatures(buffer[:6])  prediction = model.predict(features)  print("--- {:.3f} ms ---".format((time.perf\_counter() - start\_time) \* 1000.0))  […] |

Na Figura 24, é demonstrado o tempo de execução do experimento. Após alguns segundos de execução, o sistema estabiliza e tem um tempo de processamento em torno de 1 milissegundo. Vale ressaltar que esta é uma análise superficial, com objetivo de ter uma referência de tempo de execução. Mais testes precisam ser realizados para obter uma análise completa de tempo de execução.

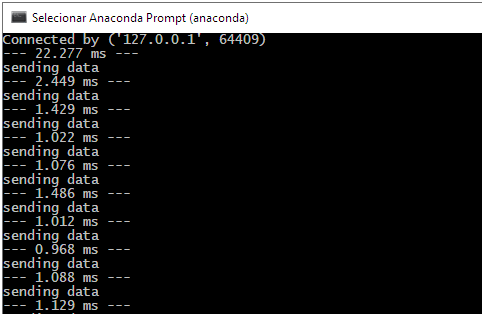


Figura 24. Tempos de execução em um experimento

1. Conclusões

Pesquisas envolvendo sistemas BCI realizados na UNIVALI e instituições parceiras demonstraram as possíveis vantagens que um sistema utilizando sinais de EEG pode proporcionar. No entanto, as pesquisas realizadas obtiveram resultados de forma *post factum*, havendo uma lacuna de tempo até que um resultado seja relatado. Para algumas pesquisas, essa limitação é indiferente, porém, para outras o *feedback* imediato é uma característica de alto interesse.

Para contribuir à projetos de pesquisa envolvendo sistemas BCI realizadas na UNIVALI, um sistema BCI online foi proposto, no âmbito de reconhecimento de emoções. A partir de uma fundamentação teórica, foram levantadas as principais características e formatos de reconhecimento emoções utilizadas por pesquisas recentes. Foram revisados métodos de processamento de sinais e de algoritmos de classificação para desenvolver um projeto de sistema BCI online.

O desenvolvimento do sistema BCI foi realizado pela plataforma do OpenBCI, já que esta é a utilizada em pesquisas e contém as ferramentas de controle para uma sessão. Foi realizada a integração da ferramenta desenvolvida no OpenBCI com o modelo de classificação, que permite a classificação dos sinais de forma online.

O sistema desenvolvido utiliza uma janela de 4 segundos com sobreposição de 2 segundos, portanto, a cada 2 segundos é realizado uma classificação sob dados de 4 segundos. Como as emoções não variam rapidamente, uma solução em tempo real não é necessária. Uma breve análise de tempo de execução foi realizada sobre o sistema para considerar trabalhos futuros com requisitos de tempo real.

Para estimar o desempenho do sistema BCI, vários modelos de aprendizado de máquina foram treinados e avaliados. O treinamento dos modelos de classificação foi realizado a partir do *dataset* SEED-IV, que também é utilizado em pesquisas correlatos envolvendo o reconhecimento de emoções de sinais de EEG. Os resultados obtidos variam de acordo com o usuário e sessão. A melhor acurácia média obtida para um usuário foi de 76,19%, e uma acurácia média entre todos os usuários de 57,07%. Os resultados obtidos são similares aos autores do *dataset* no quesito de classificação pela densidade espectral.

Foi visto que não há uma diferença significativa em relação ao desempenho dos diferentes modelos quando treinados para um usuário, porém existe uma diferença substancial do desempenho entre os usuários. Um dos motivos desta diferença pode ser pela presença de artefatos nas mesmas bandas de frequência que foram calculadas as características. Isso significa que o um filtro passa-banda não é suficiente para filtrar estes artefatos, e outros métodos de processamento precisam ser abordados para a remoção destes artefatos.

Algoritmos de remoção de artefatos podem melhorar significantemente o desempenho do sistema. Estes algoritmos não foram abordados neste trabalho por restrições de tempo, mas trabalhos futuros podem abordar o tema com a base desenvolvida neste projeto.

Além do reconhecimento de emoções, o projeto desenvolvido pode ser utilizado para outras aplicações, com alterações mínimas devido a modularidade do projeto. Portanto, trabalhos futuros podem utilizar a base desenvolvida para criar soluções BCI que requerem um feedback online, sem ter que desenvolver um sistema do zero.

Referências

BAMFORD, A. M.; NOBBS, J. H. Use and application of the PAD scale in the study of colour emotion. **AIC2007 Proceedings**, n. November, p. 46–123, 2007.

BRAGANÇA, J. **Roda de emoções**. Disponível em: <https://pt.wikipedia.org/wiki/Ficheiro:Roda\_Das\_Emocoes.png>. Acesso em: 6 nov. 2019.

BREAZEAL, C.; ARYANANDA, L. Recognition of affective communicative intent in robot-directed speech. **Autonomous Robots**, v. 12, n. 1, p. 83–104, 2002.

BUECHEL, S.; HAHN, U. Emotion analysis as a regression problem-dimensional models and their implications on Emotion representation and metrical evaluation. **Frontiers in Artificial Intelligence and Applications**, v. 285, n. February 2019, p. 1114–1122, 2016.

CARVALHO, A. P. DE L. F. DE. **Redes Neurais Artificiais**. Disponível em: <http://conteudo.icmc.usp.br/pessoas/andre/research/neural/#hist>. Acesso em: 27 out. 2019.

CATECATI, T. **AVALIAÇÃO DA SATISFAÇÃO DO USUÁRIO EM TEMPO REAL POR MEIO DO USO DE ELETROENCEFALOGRAFIA**. [s.l.] UDESC, 2018.

COHEN, M. X. **Analyzing Neural Time Series Data: Theory and Practice**. [s.l.] The MIT Press, 2014.

COOLEY, J. W.; LEWIS, P. A. W.; WELCH, P. D. Historical Notes on the Fast Fourier Transform. **IEEE Transactions on Audio and Electroacoustics**, v. 15, n. 2, p. 76–79, 1967.

COURSERA.ORG. **Linear Regression with Multiple Variables - Lecture Notes**. Disponível em: <https://www.coursera.org/learn/machine-learning/resources/QQx8l>. Acesso em: 27 out. 2019a.

COURSERA.ORG. Logistic Regression. [s.d.].

DATA SCIENCE ACADEMY. **Deep Learning Book, 2019**. Disponível em: <http://deeplearningbook.com.br/cross-entropy-cost-function/>. Acesso em: 28 out. 2019.

DEVLIN, J. **​​​​28 Jupyter Notebook Tips, Tricks, and Shortcuts**. Disponível em: <https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/>. Acesso em: 10 jun. 2020.

EKMAN, P.; FRIESEN, W. V. Facial Action Coding System: A Technique for the Measurement of Facial Movement. **Journal of Personality and Social Psychology**, v. 17, n. 2, p. 124–129, 1971.

ETHEN. **Model Selection**. Disponível em: <http://ethen8181.github.io/machine-learning/model\_selection/model\_selection.html>. Acesso em: 19 jun. 2020.

FOX, E. Perspectives from affective science on understanding the nature of emotion. **Brain and Neuroscience Advances**, v. 2, p. 239821281881262, 2018.

FOXALL, G. R. The emotional texture of consumer environments: A systematic approach to atmospherics. **Journal of Economic Psychology**, v. 18, n. 5, p. 505–523, 1997.

GRANER, J. **FMRI scan during working memory tasks**. Disponível em: <https://commons.wikimedia.org/wiki/File:FMRI\_scan\_during\_working\_memory\_tasks.jpg>. Acesso em: 6 nov. 2019.

HAGER, J. C. **Facial Action Coding System Affect Interpretation Dictionary (FACSAID)**. Disponível em: <https://web.archive.org/web/20110520164308/http://face-and-emotion.com/dataface/facsaid/description.jsp>. Acesso em: 16 out. 2019.

HANSEN, C. **Nested Cross-Validation Python Code**. Disponível em: <https://mlfromscratch.com/nested-cross-validation-python-code/#/>. Acesso em: 19 jun. 2020.

HERZ, R. S. et al. Neuroimaging evidence for the emotional potency of odor-evoked memory. **Neuropsychologia**, v. 42, n. 3, p. 371–378, 2004.

HOU, X. et al. CogniMeter: EEG-based Emotion, Mental Workload and Stress Visual Monitoring. **Proceedings - 2015 International Conference on Cyberworlds, CW 2015**, p. 153–160, 2016.

HU, M.; LIU, B. **Mining and summarizing customer reviews**. KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. **Anais**...2004

KOŁAKOWSKA, A. et al. Modeling emotions for affect-aware applications. n. January, 2015.

KOOLAGUDI, S. G.; RAO, K. S. Emotion recognition from speech: A review. **International Journal of Speech Technology**, v. 15, n. 2, p. 99–117, 2012.

LAN, Z. et al. Real-time EEG-based User ’ s Valence Monitoring. 2015.

LARSEN, J. T.; NORRIS, C. J.; CACIOPPO, J. T. Effects of positive and negative affect on electromyographic activity over zygomaticus major and corrugator supercilii. **Psychophysiology**, v. 40, n. 5, p. 776–785, 2003.

LIU, Y. J. et al. Real-time movie-induced discrete emotion recognition from EEG signals. **IEEE Transactions on Affective Computing**, v. 9, n. 4, p. 550–562, 2018.

LOGOTHETIS, N. K. et al. Neurophysiological investigation of the basis of the fMRI signal. **Nature**, v. 412, n. 6843, p. 150–157, 12 jul. 2001.

LOZANO-MONASOR, E. et al. **Facial Expression Recognition from Webcam Based on Active Shape Models and Support Vector Machines**. IWAAL 2014. **Anais**...14 nov. 2014

MEHRABIAN, A., & RUSSELL, J. A. **An Approach to Environmental Psychology**. [s.l.] MIT Press, 1974.

MEHRABIAN, A. **Basic dimensions for a general psychological theory**. [s.l: s.n.].

MOST, T.; AVINER, C. Auditory, visual, and auditory - Visual perception of emotions by individuals with cochlear implants, hearing aids, and normal hearing. **Journal of Deaf Studies and Deaf Education**, v. 14, n. 4, p. 449–464, 2009.

NEUROSCIENTIFICALLY CHALLENGED. **Know your brain: Cerebral cortex — Neuroscientifically Challenged**. Disponível em: <https://www.neuroscientificallychallenged.com/blog/know-your-brain-cerebral-cortex>. Acesso em: 20 out. 2019.

NIELSEN, M. A. **Neural Networks and Deep Learning**Determination Press, , 2015.

NOGUEIRA, K. A. Estudo de respostas emocionais às cores no contexto de cartazes de cinema. **Design e Tecnologia**, v. 8, n. 15, p. 1, 2018.

OLDONI, M. L. et al. Working Memory Load Measurement on Aphasia patients through a Brain Computer Interface : Theoretical Considerations and Preliminary Results. p. 1–11, 2019.

PANKSEPP, J. **Affective Neuroscience: The Foundations of Human and Animal Emotions (Series in Affective Science)**. [s.l.] Oxford University Press, USA, 2004. v. 4

PICARD, R. W. **Affective Computing**. [s.l: s.n.].

PICARD, R. W.; VYZAS, E.; HEALEY, J. Toward machine emotional intelligence: Analysis of affective physiological state. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 23, n. 10, p. 1175–1191, 2001.

PLUTCHIK, R. The nature of emotions: Human emotions have deep evolutionary roots. **American Scientist**, v. 89, n. 4, p. 344–350, 2001.

POSNER, J. et al. The neurophysiological bases of emotion: An fMRI study of the affective circumplex using emotion-denoting words. **Human Brain Mapping**, v. 30, n. 3, p. 883–895, 2009.

PURNAMASARI, P. D.; RATNA, A. A. P.; KUSUMOPUTRO, B. EEG based patient emotion monitoring using relative wavelet energy feature and Back Propagation Neural Network. **Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS**, v. 2015- Novem, p. 2820–2823, 2015.

RACHED, T. S.; PERKUSICH, A. Emotion Recognition Based on Brain-Computer Interface Systems. **School of Enviromental Sciences**, 2012.

RAO, R. P. N. **Brain-computer interfacing: An introduction**. [s.l: s.n.].

RUSSELL, J. A. A circumplex model of affect. **Journal of Personality and Social Psychology**, v. 39, n. 6, p. 1161–1178, 1980.

RUSSELL, J. A.; PRATT, G. A Description of the Affective Quality Attributed to Environments A Description of the Affective Quality Attributed to Environments. **Journal of Personality and Social Psychology**, v. 38, n. August, p. 311–322, 1980.

SAMPLER, T. M. **Convex analysis**. Disponível em: <https://msampler.wordpress.com/2009/07/08/convex-analysis/>. Acesso em: 28 out. 2019.

SCARANTINO, A. **The Circumplex, and the Psychological Construction of Emotions**. Disponível em: <http://emotionresearcher.com/the-circumplex-and-the-psychological-construction-of-emotions/>. Acesso em: 1 out. 2019.

SCHACTER, D. L. et al. **Psychology**. Second Eur ed. [s.l: s.n.].

SCHACTER, D. L.; GILBERT, D. T.; WEGNER, D. M. **Introducing psychology**. [s.l.] Worth Publishers, 2013.

SHELLEY, K.; SHELLEY, S. Pulse Oximeter Waveform: Photoelectric Plethysmography. **Clinical Monitoring: Practical Applications for Anesthesia and Critical Care**, n. May, p. 420–423, 2001.

SKLAR, M. **tDCS – Electrode Positioning – The 10-20 System**. Disponível em: <https://blog.adafruit.com/2017/06/19/tdcs-electrode-positioning-the-10-20-system/>. Acesso em: 20 out. 2019.

SMITH, W. **The Scientist and Engineer’s Guide to Digital Signal Processing**. [s.l: s.n.].

STANFORD.EDU. **Linear Regression - Lecture Notes**. Disponível em: <https://lagunita.stanford.edu/c4x/HumanitiesScience/StatLearning/asset/linear\_regression.pdf>. Acesso em: 27 out. 2019.

WILSON, R. L. **Ethical Issues of Brain Computer Interfaces (BCI)**. IACAP 2013. **Anais**...College Park, Maryland: 2013Disponível em: <http://www.iacap.org/proceedings\_IACAP13/paper\_41.pdf>

ZANGENEH SOROUSH, M. et al. A Review on EEG Signals Based Emotion Recognition. **International Clinical Neuroscience Journal**, v. 4, n. 4, p. 118–129, 2017.

ZHENG, W. L. et al. EmotionMeter: A Multimodal Framework for Recognizing Human Emotions. **IEEE Transactions on Cybernetics**, v. 49, n. 3, p. 1110–1122, 2019.

1. Códigos-fonte

Quadro 21. Geração de Características

|  |
| --- |
| %load\_ext autoreload  %autoreload  import scipy.io as spio  import pandas as pd  import matplotlib.pyplot as plt  from tsfresh import extract\_features, extract\_relevant\_features, select\_features  import numpy as np  import re  from bottleneck import move\_mean  import os  from matplotlib.mlab import stride\_windows  from scipy.fft import fft  from scipy.signal.windows import hann  from feature\_calculators import multi\_psd\_band, multi\_diferential\_entropy  from constants import electrode\_order\_names, subjects, electrodes, trials  from scipy import signal  from entropy.entropy import perm\_entropy, spectral\_entropy, svd\_entropy, app\_entropy, sample\_entropy, petrosian\_fd, katz\_fd, higuchi\_fd, detrended\_fluctuation  electrode\_names = electrode\_order\_names  electrode\_names\_indices = [i for i, x in enumerate(electrode\_order\_names) if x in electrode\_names]  bands = [  (1, 4), # 'delta'  (4, 8), # 'theta'  (8, 12), # 'alpha'  (12, 30), # 'beta'  (30, 50), # 'gamma'  ]  session\_labels = {  '1': [1,2,3,0,2,0,0,1,0,1,2,1,1,1,2,3,2,2,3,3,0,3,0,3],  '2': [2,1,3,0,0,2,0,2,3,3,2,3,2,0,1,1,2,1,0,3,0,1,3,1],  '3': [1,2,2,1,3,3,3,1,1,2,1,0,2,3,3,0,2,3,0,0,2,0,1,0]  }  # 0 - neutral  # 1 - sad  # 2 - fear  # 3 - happy  allowed\_session\_labels = [0, 1, 3]  # allowed\_session\_labels = [1, 3]  # Sampling Frequency  sf = 200  def find\_regex\_index(regex, dictionary):  return [index for index, value in dictionary.items() if re.match(regex, index)][0]  def band\_pass\_coefs(lowcut, highcut, sf, order):  nyq = 0.5 \* sf  low = lowcut / nyq  high = highcut / nyq    return signal.butter(order, [low, high], btype='band')  def generate\_trial\_features(df, label, electrodes, sf, win\_size, overlap\_size):  df\_array\_cols = []  for index, column\_name in enumerate(electrodes):    # Memory efficient striding  strides = stride\_windows(df.loc[:, column\_name], win\_size, overlap\_size, axis=1)  win = hann(win\_size)  b, a = band\_pass\_coefs(lowcut=1, highcut=50, sf=sf, order=5)    strides\_filt = signal.filtfilt(b, a, strides)    # strides\_filt = move\_mean(strides\_filt, 20, axis=1, min\_count=1)  strides\_fft = fft(win\*strides\_filt)  df\_columns = []  # norm\_factor = sf \* (np.abs(win)\*\*2).sum()  psd\_band\_features = np.array([multi\_psd\_band(strides\_fft, N=win\_size, sf=sf, band=band) for band in bands]).T  df\_columns = df\_columns + [ column\_name + '\_psd\_' + band for band in ['delta','theta', 'alpha', 'beta', 'gamma']]  # psd\_band\_features = np.array([multi\_psd\_band(strides\_fft, N=win\_size, sf=sf, band=(1, 50), norm\_factor=norm\_factor)]).T  # df\_columns = df\_columns + [ column\_name + '\_psd\_' + 'all']    # Calculate DE features  # de\_band\_features = np.array([multi\_diferential\_entropy(strides\_fft, N=win\_size, sf=sf, band=band) for band in bands]).T  # df\_columns = df\_columns + [ column\_name + '\_de\_' + band for band in ['delta','theta', 'alpha', 'beta', 'gamma']]  # de\_band\_features = np.array([(0.5\*np.log(2\*np.pi\*np.e\*np.var(sig, axis=1))) for sig in band\_signals]).T  # df\_columns = df\_columns + [ column\_name + '\_de\_' + band for band in ['delta','theta', 'alpha', 'beta', 'gamma']]  # de\_band\_features = np.array([multi\_diferential\_entropy(strides\_fft, N=win\_size, sf=sf, band=(1, 50))]).T    # df\_columns = df\_columns + [ column\_name + '\_de\_' + 'all']  df\_array\_cols.append(  pd.DataFrame(psd\_band\_features, columns=df\_columns)  # pd.DataFrame(de\_band\_features, columns=df\_columns)  )  df = pd.concat(df\_array\_cols, axis=1)  df.insert(0, column='label', value=int(label))    return df  def read\_dataset(dataset\_dir, session\_list=None, user\_list=None, trial\_list=None):  column\_names = ['session', 'user', 'trial'] + electrode\_names  df\_eeg\_session\_trial\_list = []  with os.scandir(dataset\_dir) as folders:  for folder in list(folders):  session = folder.name  if session\_list != None and int(session) not in session\_list:  continue    with os.scandir(folder) as files:  for file in list(files):    user = file.name.split('\_')[0]  if user\_list != None and int(user) not in user\_list:  continue    mat = spio.loadmat(file.path)  for trial in trials:  if session\_labels[session][trial] not in allowed\_session\_labels:  continue  if trial\_list != None and int(trial) not in trial\_list:  continue    trial\_key = find\_regex\_index(r'^.\*eeg{}$'.format(trial+1), mat)  mat\_subset = [val for i, val in enumerate(mat[trial\_key]) if i in electrode\_names\_indices]    df\_eeg\_session\_trial = pd.DataFrame(np.array(mat\_subset).T)  df\_eeg\_session\_trial.columns = electrode\_names  df\_eeg\_session\_trial.insert(0, column='session', value=int(session))  df\_eeg\_session\_trial.insert(1, column='user', value=int(user))  df\_eeg\_session\_trial.insert(2, column='trial', value=int(trial))    df\_eeg\_session\_trial\_list.append(df\_eeg\_session\_trial)  return pd.concat(df\_eeg\_session\_trial\_list).reset\_index(drop=True)  def generate\_session\_features(dataset\_dir, win\_size, overlap\_size, session\_list=None, user\_list=None, trial\_list=None):  # Actually used columns on dataframe  column\_names = ['session', 'user', 'trial'] + electrode\_names    trial\_feature\_list = []  with os.scandir(dataset\_dir) as folders:  for folder in list(folders):  session = folder.name  if session\_list != None and int(session) not in session\_list:  continue  with os.scandir(folder) as files:  for file in list(files):  mat = spio.loadmat(file.path)  user = file.name.split('\_')[0]  if user\_list != None and int(user) not in user\_list:  continue  for trial in trials:  if session\_labels[session][trial] not in allowed\_session\_labels:  continue  if trial\_list != None and int(trial) not in trial\_list:  continue  trial\_key = find\_regex\_index(r'^.\*eeg{}$'.format(trial+1), mat)  mat\_subset = [val for i, val in enumerate(mat[trial\_key]) if i in electrode\_names\_indices]      df\_eeg\_session\_trial = pd.DataFrame(np.array(mat\_subset).T)  df\_eeg\_session\_trial.columns = electrode\_names  trial\_features = generate\_trial\_features(  df=df\_eeg\_session\_trial,  label=session\_labels[session][trial],  electrodes=electrode\_names,  sf=sf,  win\_size=win\_size,  overlap\_size=overlap\_size  )  trial\_features.insert(0, column='session', value=int(session))  trial\_features.insert(1, column='user', value=int(user))  trial\_features.insert(2, column='trial', value=int(trial))    trial\_feature\_list.append(trial\_features)  return pd.concat(trial\_feature\_list).reset\_index(drop=True) |

Quadro 22. Geração de características de várias janelas e sobreposições

|  |
| --- |
| %%time  import os.path  win\_overlaps = [(800, 400), (800, 0), (400, 200), (400, 0), (200, 100), (200, 0)]  dataset\_dir = 'D:\\facul\\eeg\_raw\_data\\'  for window\_size, overlap\_size in win\_overlaps:      df\_features = generate\_session\_features(dataset\_dir, window\_size, overlap\_size)      file\_path = f'D:/facul/features/features\_win\_size\_{window\_size}\_overlap\_size\_{overlap\_size}.pkl'      if not os.path.isfile(file\_path):          df\_features.to\_pickle(file\_path) |

Quadro 23. Classe RepeatedStratifiedGroupKFold

|  |
| --- |
| from collections import ChainMap  from sklearn.model\_selection import GridSearchCV, RandomizedSearchCV, cross\_val\_score, cross\_validate  import pandas as pd  import numpy as np  from collections import Counter, defaultdict  from sklearn.utils import check\_random\_state, shuffle  import matplotlib.pyplot as plt  from sklearn.model\_selection import learning\_curve  import pickle  from copy import deepcopy  # To be merged in https://github.com/scikit-learn/scikit-learn/issues/13621  class RepeatedStratifiedGroupKFold():  def \_\_init\_\_(self, n\_splits=5, n\_repeats=1, random\_state=None):  self.n\_splits = n\_splits  self.n\_repeats = n\_repeats  self.random\_state = random\_state  # Implementation based on this kaggle kernel:  # https://www.kaggle.com/jakubwasikowski/stratified-group-k-fold-cross-validation  def split(self, X, y=None, groups=None):  k = self.n\_splits  def eval\_y\_counts\_per\_fold(y\_counts, fold):  y\_counts\_per\_fold[fold] += y\_counts  std\_per\_label = []  for label in range(labels\_num):  label\_std = np.std(  [y\_counts\_per\_fold[i][label] / y\_distr[label]  for i in range(k)]  )  std\_per\_label.append(label\_std)  y\_counts\_per\_fold[fold] -= y\_counts  return np.mean(std\_per\_label)  rnd = check\_random\_state(self.random\_state)  for repeat in range(self.n\_repeats):  labels\_num = np.max(y) + 1  y\_counts\_per\_group = defaultdict(lambda: np.zeros(labels\_num))  y\_distr = Counter()  for label, g in zip(y, groups):  y\_counts\_per\_group[g][label] += 1  y\_distr[label] += 1  y\_counts\_per\_fold = defaultdict(lambda: np.zeros(labels\_num))  groups\_per\_fold = defaultdict(set)  groups\_and\_y\_counts = list(y\_counts\_per\_group.items())  rnd.shuffle(groups\_and\_y\_counts)  for g, y\_counts in sorted(groups\_and\_y\_counts, key=lambda x: -np.std(x[1])):  best\_fold = None  min\_eval = None  for i in range(k):  fold\_eval = eval\_y\_counts\_per\_fold(y\_counts, i)  if min\_eval is None or fold\_eval < min\_eval:  min\_eval = fold\_eval  best\_fold = i  y\_counts\_per\_fold[best\_fold] += y\_counts  groups\_per\_fold[best\_fold].add(g)  all\_groups = set(groups)  for i in range(k):  train\_groups = all\_groups - groups\_per\_fold[i]  test\_groups = groups\_per\_fold[i]  train\_indices = [i for i, g in enumerate(  groups) if g in train\_groups]  test\_indices = [i for i, g in enumerate(  groups) if g in test\_groups]  yield train\_indices, test\_indices  def get\_n\_splits(self, X=None, y=None, groups=None):  return self.n\_splits |

Quadro 24. Cálculo de Densidade Espectral

|  |
| --- |
| def multi\_psd\_band(fft\_coefs, N, sf, band):  # get band boundaries  low, high = band  # get frequency axis  freqs = fftfreq(N, 1/sf)  # get indexes for the given frequnecy band  idx\_band = np.logical\_and(freqs >= low, freqs <= high)  # calculate the psd  psd = np.abs(fft\_coefs[:, idx\_band])\*\*2  return np.sum(psd, axis=1) / (high-low) |

Quadro 25. Modelos e parâmetros

|  |
| --- |
| from sklearn.svm import LinearSVC, SVC  from sklearn.ensemble import RandomForestClassifier  from sklearn.ensemble import AdaBoostClassifier  from sklearn.ensemble import GradientBoostingClassifier  from sklearn.pipeline import Pipeline  from sklearn.preprocessing import StandardScaler, RobustScaler, Normalizer, PowerTransformer, MinMaxScaler  from sklearn.feature\_selection import f\_classif, SelectKBest, chi2, mutual\_info\_classif  from sklearn.decomposition import PCA  from helpers import EstimatorSelectionHelper  import numpy as np  from sklearn.naive\_bayes import GaussianNB  from xgboost import XGBClassifier  from sklearn.model\_selection import StratifiedShuffleSplit  from sklearn.linear\_model import LogisticRegression  from sklearn.neural\_network import MLPClassifier  from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  from datetime import datetime  import pickle  models = {  'LinearDiscriminantAnalysis': LinearDiscriminantAnalysis(),  'RandomForestClassifier': RandomForestClassifier(),  'LogisticRegression': Pipeline([  ('scaling', StandardScaler()),  ('estimator', LogisticRegression()),  ]),  'MLPClassifier': Pipeline([  ('scaling', StandardScaler()),  ('estimator', MLPClassifier(solver='lbfgs')),  ]),  'LinearSVC': Pipeline([  ('scaling', StandardScaler()),  ('estimator', LinearSVC())  ]),  'SVC': Pipeline([  # ('feat', SelectKBest(mutual\_info\_classif, k=20)),  ('scaling', StandardScaler()),  ('estimator', SVC())  ])  }  params = {  'LinearDiscriminantAnalysis': {},  'RandomForestClassifier': { 'n\_estimators': [16, 32, 48, 100] },  'LogisticRegression': {'estimator\_\_C': [0.01, 0.03, 0.1, 0.3, 1, 3]},  'MLPClassifier': {  'estimator\_\_hidden\_layer\_sizes': [(15,10), (10,5), (5,3),(30,15,10), (20,15,10), (15,10,5)],  'estimator\_\_alpha': np.logspace(-1, -7, num=7)  },  'LinearSVC': [  {'estimator\_\_C': np.logspace(5, -5, base=2, num=10)},  ],  'SVC': [  {'estimator\_\_C': np.logspace(15, -15, base=2, num=31), 'estimator\_\_gamma': np.logspace(8, -8, base=10, num=17)},  ]  } |

Quadro 26. Classe EstimatorSelectionHelper

|  |
| --- |
| class EstimatorSelectionHelper:  def \_\_init\_\_(self, models, params):  print(params.keys())  if not set(models.keys()).issubset(set(params.keys())):  missing\_params = list(set(models.keys()) - set(params.keys()))  raise ValueError(  "Some estimators are missing parameters: %s" % missing\_params)  self.models = models  self.params = params  self.keys = list(models.keys())  self.grid\_searches = {}  self.cross\_val\_results = {}  self.ran\_with\_outer\_cv = False  @staticmethod  def load(persist\_dir):  with open(persist\_dir, 'rb') as f:  instance = pickle.load(f)  return instance  def save(self, persist\_dir):  with open(persist\_dir, 'wb') as f:  pickle.dump(self, f)  print("Object persisted")  def fit(self, X, y, cv=3, n\_jobs=3, verbose=1, scoring=None, refit=False, groups=None, outer\_cv=None, persist\_dir=None, randomSearchFor=None):  for key in self.keys:  model = self.models[key]  params = self.params[key]  if randomSearchFor != None and key in randomSearchFor:  SearchCV = RandomizedSearchCV  else:  SearchCV = GridSearchCV  gs = SearchCV(model, params, cv=cv, n\_jobs=n\_jobs,  verbose=verbose, scoring=scoring, refit=True,  return\_train\_score=True)  if outer\_cv != None:  print("Running GridSearchCV for %s with nested cross validation." % key)  self.cross\_val\_results[key] = cross\_validate(gs, X, y, groups=groups, cv=outer\_cv, scoring=scoring, error\_score='raise',  return\_train\_score=True, return\_estimator=True, verbose=10, fit\_params={'groups': groups})  self.ran\_with\_outer\_cv = True  gs = max(  self.cross\_val\_results[key]['estimator'], key=lambda item: item.best\_score\_)  else:  self.ran\_with\_outer\_cv = False  print("Running GridSearchCV for %s." % key)  gs.fit(X, y, groups=groups)  self.grid\_searches[key] = gs  if persist\_dir != None:  self.save(persist\_dir)  # print("Score: ", gs.best\_score\_)  def score\_summary(self, sort\_by='mean\_score'):  if self.ran\_with\_outer\_cv:  rows = []    for k in self.keys:  cv\_results\_params = deepcopy(self.grid\_searches[k].cv\_results\_['params'])  # convert list values into string, e.g. (1, 2) = > '(1,2)'  for dictionary in cv\_results\_params:  for key in dictionary:  dictionary[key] = str(dictionary[key])  test\_scores = self.cross\_val\_results[k]['test\_score']  train\_scores = self.cross\_val\_results[k]['train\_score']  row = pd.DataFrame({  'estimator': [k],  'min\_score': test\_scores.min(),  'max\_score': test\_scores.max(),  'mean\_score': test\_scores.mean(),  'std\_score': test\_scores.std(),  'train\_mean\_score': train\_scores.mean(),  \*\*dict(ChainMap(\*cv\_results\_params)),  })  rows.append(row)  df = pd.concat(rows)  return df  else:  def row(key, scores, params):  d = {  'estimator': key,  'min\_score': min(scores),  'max\_score': max(scores),  'mean\_score': np.mean(scores),  'std\_score': np.std(scores),  }  return pd.Series({\*\*params, \*\*d})  rows = []  for k in self.grid\_searches:  # [{'n\_estimators': 16}, {'n\_estimators': 32}]  params = self.grid\_searches[k].cv\_results\_['params']  scores = []  for i in range(self.grid\_searches[k].n\_splits\_):  key = "split{}\_test\_score".format(i)  r = self.grid\_searches[k].cv\_results\_[key] # [0.96694215 0.95041322]  scores.append(r.reshape(len(params), 1))  all\_scores = np.hstack(scores)  for p, s in zip(params, all\_scores):  rows.append((row(k, s, p)))  df = pd.concat(rows, axis=1).T.sort\_values([sort\_by], ascending=False)  columns = ['estimator', 'min\_score','mean\_score', 'max\_score', 'std\_score']  columns = columns + [c for c in df.columns if c not in columns]  return df[columns] |

Quadro 27. Treino de modelos sujeito dependente

|  |
| --- |
| %%time  %load\_ext autoreload  %autoreload  from helpers import RepeatedStratifiedGroupKFold  from sklearn.svm import LinearSVC, SVC  from sklearn.ensemble import RandomForestClassifier  from sklearn.ensemble import AdaBoostClassifier  from sklearn.ensemble import GradientBoostingClassifier  from sklearn.pipeline import Pipeline  from sklearn.preprocessing import StandardScaler, RobustScaler, Normalizer, PowerTransformer, MinMaxScaler  from sklearn.feature\_selection import f\_classif, SelectKBest, chi2, mutual\_info\_classif  from sklearn.decomposition import PCA  from helpers import EstimatorSelectionHelper  import numpy as np  from sklearn.naive\_bayes import GaussianNB  from xgboost import XGBClassifier  from sklearn.model\_selection import StratifiedShuffleSplit  from sklearn.linear\_model import LogisticRegression  from sklearn.neural\_network import MLPClassifier  from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  from datetime import datetime  import pickle  cols = ['session', 'trial', 'user', 'label', 'T7', 'T8', 'FT7', 'FT8', 'TP7', 'TP8']  df\_features = read\_features\_file('D:\\facul\\features', 800, 400)  df\_features\_cols = df\_features.loc[:, df\_features.filter(regex=r"({})".format('|'.join(cols)), axis=1).columns]  df\_user\_scores = pd.DataFrame()  folder = './models/user-dependent-same-days'  for day in range(1, 4):  user\_helper\_list = []  for user in range(1, 16):  cv\_grp = RepeatedStratifiedGroupKFold(n\_splits=5, n\_repeats=1)  outer\_cv\_grp = RepeatedStratifiedGroupKFold(n\_splits=5, n\_repeats=1)  print(outer\_cv\_grp.get\_n\_splits())  helper = EstimatorSelectionHelper(models, params)  user\_helper\_list.append(helper)  user\_day\_index = (df\_features\_cols['user'].isin([user])) & (df\_features\_cols['session'] == day)  X = df\_features\_cols.loc[user\_day\_index, ~df\_features\_cols.columns.isin(['session', 'trial', 'user', 'label'])]    y = df\_features\_cols.loc[user\_day\_index, 'label'].astype(int)  groups = df\_features\_cols.loc[user\_day\_index, 'trial']  # helper.fit(X, y, scoring='f1', n\_jobs=2, cv=cv\_grp, outer\_cv=outer\_cv\_grp, groups=df\_features\_cols['trial'])  helper.fit(X, y, scoring='accuracy', n\_jobs=-1, cv=cv\_grp, outer\_cv=outer\_cv\_grp, verbose=10, groups=groups, persist\_dir=f"{folder}/user\_{user}\_day\_{day}\_{datetime.now().strftime('%Y-%m-%dT%H-%M-%S.pkl')}", randomSearchFor=['MLPClassifier', 'SVC'])  temp\_scores = helper.score\_summary(sort\_by='mean\_score')  temp\_scores.insert(0, 'user', user)  temp\_scores.insert(0, 'day', day)  df\_user\_scores = df\_user\_scores.append(temp\_scores)  with open(f"{folder}/all\_users\_day\_{day}\_{datetime.now().strftime('%Y-%m-%dT%H-%M-%S.pkl')}", "wb") as f:  pickle.dump(user\_helper\_list, f)  df\_user\_scores |