UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Paulo Augusto Alves Luz Viana

Identificação de Movimentos da Mão por Machine Learning Utilizando Eletromiografia de Superfície

São Carlos

2019

Paulo Augusto Alves Luz Viana

Identificação de Movimentos da Mão por Machine Learning Utilizando Eletromiografia de Superfície

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Alberto Cliquet Júnior

São Carlos

2019

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Viana, Paulo Augusto Alves Luz
V614i Identificação de Movimentos da Mão por Machine Learning Utilizando Eletromiografia de Superfície / Paulo Augusto Alves Luz Viana; orientador Alberto Cliquet Júnior. São Carlos, 2019.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2019.

1. Eletromiografia. 2. Aprendizado de Máquina. 3. Redes Neurais. 4. Regressão Logística. 5. k-Nearest Neighbors. 6. Reconhecimento de movimentos. 7. Processamento de sinais. 8. Sinais biomédicos. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Paulo Augusto Alves Luz Viana

Título: "Identificação de movimentos da mão por machine learning utilizando eletromiografia de superfície"

Trabalho de Conclusão de Curso defendido e aprovado em 21 / 11 / 2019,

com NOTA 10,0 (dez, zero), pela Comissão Julgadora:

Prof. Titular Alberto Cliquet Júnior - Orientador - SEL/EESC/USP

Dr. Renato Varoto - UNICAMP

Mestre Renata Manzano Maria - UNICAMP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Este trabalho é dedicado à meus amigos, família e à quem se inspira com a ciência e o conhecimento humano.

AGRADECIMENTOS

Agradeço ao Prof. Dr. Alberto Cliquet pelo incentivo dado nas aulas que culminou com meu interesse na área de Engenharia de Reabilitação, um artigo e este trabalho. Agradeço pelo passagem de conhecimento e oportunidades oferecidas, pelo suporte e disposição de equipamentos e laboratório. Agradeço aos professores da Escola de Engenharia de São Carlos, que são responsáveis pelo ensino das diversas áreas do conhecimento da Engenharia que tornaram possível a execução deste trabalho. Agradeço aos meus amigos pela companhia nos estudos e em inúmeras situações, e também pelos momentos de descontração que tornaram esses anos suportáveis possibilitando uma passagem saudável pela universidade. Agradeço à minha família e pelo suporte e incentivo durante esses anos, e à minha namorada pelo apoio incondicional.

"As convicções são inimigas mais perigosas da verdade do que as mentiras."
Friedrich Nietzsche

RESUMO

VIANA, P. **Identificação de Movimentos da Mão por Machine Learning Utilizando Eletromiografia de Superfície**. 2019. 83p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

Neste trabalho exploraram-se técnicas de processamento de sinais de eletromiografia e de reconhecimento de movimentos utilizando aprendizado de máquina. Como o sinal sofre com muitos ruídos e suas características variam muito de indivíduo para indivíduo a utilização de um bom processamento de sinal e de um algoritmo de reconhecimento robusto é importante. O sinal de eletromiografia utilizado provém da base de dados Ninapro e passou por um procedimento de *denoising* e filtragem, e após isso realizaram-se mais de 100 experimentos utilizando os algoritmos *k-Nearest Neighbors*, Regressão Logística e um modelo baseado em Redes Neurais para classificação de diversos números de movimentos. Nestes experimentos também é mostrada a importância da divisão correta dos conjuntos de treino, validação e teste para que se tenha uma estimativa correta da performance real do modelo. A comparação de modelos que utilizam algoritmos de estado da arte, como as redes neurais profundas, com algoritmos não tão complexos trouxeram conclusões interessantes. A regressão logística por vezes ultrapassava a performance da rede neural, e em um caso o algoritmo *k-NN* teve a melhor performance entre os três. Neste sentido, para diferentes quantidades de movimentos um ou outro modelo poderia ser considerado mais adequado. Conseguiu-se uma acurácia balanceada de até 98.1% ao se classificar 2 movimentos e de até 24.6% na classificação de 49 movimentos, e também observou-se que as performances da rede neural profunda e da regressão logística foram semelhantes.

Palavras-chave: Eletromiografia. Aprendizado de Máquina. Redes Neurais. Regressão Logística. k-Nearest Neighbors. Reconhecimento de movimentos. Processamento de sinais. Sinais biomédicos.

ABSTRACT

VIANA, P. **Hand Movement Recognition by Machine Learning Using Surface Electromyography**. 2019. 83p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

This work explored electromyographic signal processing and gesture recognition techniques using machine learning. As the signal suffers from a lot of noise and its characteristics vary greatly from individual to individual, the use of good signal processing and a robust recognition algorithm is important. The electromyography signal used comes from the Ninapro database and has undergone a denoising and filtering procedure, after which more than 100 experiments were performed using k-Nearest Neighbors, Logistic Regression and Neural Networks based algorithms for classification of several numbers of movements. It is also shown the importance of correct splitting of training, validation and testing sets in order to have a correct estimate of the actual performance of the model. The comparison of models using state-of-the-art algorithms, such as deep neural networks, with not so complex algorithms brought interesting conclusions. The logistic regression sometimes outperformed the neural network performance, and in one case the k-nearest neighbors algorithm outperformed the other three. In this sense, for different sets of movements different algorithms could be considered more appropriate. A balanced accuracy of up to 98.1% was achieved by classifying 2 movements and up to 24.6% in the 49 movement classification, and it was also observed that the deep neural network and logistic regression performances were similar.

Keywords: Electromyography. Machine Learning. Neural Networks. Logistic Regression. k-Nearest Neighbors. Movement recognition. Signals processing. Biomedic signals.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AVC	Acidente Vascular Cerebral
ANN	*Artificial Neural Network*
ARV	*Average Rectified Value*
BSS	*Blind Source Separation*
USPSC	Campus USP de São Carlos
CWT	*Continuous Wavelet Transform*
DWT	*Discrete Wavelet Transform*
EMG	Eletromiografia/Sinal eletromiográfico
FFT	*Fast Fourier Transform*
k-NN	*k-Nearest Neighbors*
MAV	*Mean Absolute Value*
MSE	*Mean Squared Error*
MUAP	*Motor unit action potential*
MAV	*Mean absolute value*
PA	*Peak Amplitude*
RMS	*Root Mean Square*
sEMG	*surface Electromyography*
SNR	*Signal to noise ratio*
SC	*Spectral centroid*
SVM	*Support Vector Machine*
TC	*Turn counting*
USP	Universidade de São Paulo
WAMP	*Willison Amplitude*
ZC	*Zero crossings*

SUMÁRIO

1 INTRODUÇÃO

Atividades executadas diariamente (como se alimentar, tomar banho, se vestir) estão fortemente ligadas a um controle específico e adequado dos membros superiores. Existem, entretanto, diversas condições patológicas que podem limitar o controle do indivíduo sobre seus braços e mãos como lesões na medula espinhal, acidente vascular cerebral (AVC), paralisia cerebral, bem como amputações e problemas congênitos (VIANA et al., 2019). Essas condições podem influenciar negativamente no senso de autonomia e impactar na qualidade de vida do indivíduo. Em investigações sobre a qualidade de vida de lesados medulares mostrou-se que o domínio da saúde física é um dos piores avaliados, chegando a prejudicar o desenvolvimento econômico do indivíduo (FRANÇA et al., 2013).

O controle de próteses, por indivíduos amputados, e órteses, por lesados medulares ou indivíduos com deficiências, é difundidamente realizada utilizando sinais eletromiográficos. Apesar deste conceito ter sido introduzido em 1940, apenas em 1970 próteses miolétricas começaram a ter algum impacto clínico significativo, quando próteses elétricas controladas por sinais miolétricos passaram a ser utilizadas em avaliações clínicas. Esse tipo de prótese tem muitas vantagens: poucos fios e aparelhos, não é invasivo, adaptabilidade e o fato de que atividades musculares pequenas são suficientes para obtermos sinais de controle (ENGLEHART; HUDGINS, 2003).

Atualmente indivíduos com amputações trans-radiais (maioria dos indivíduos com amputação de membros superiores) podem utilizar próteses controladas por sinais eletromiográficos. Entretanto na maioria dos casos os movimentos permitidos por esses equipamentos se limitam à abertura e fechamento da mão e os algoritmos de controle são rudimentares, se limitando a estratégias de controle sequencial. Métodos invasivos são reportados por terem excelente performance enquanto métodos não invasivos mostram acurácia entre 80 e 90%. Além disso muitos dos estudos feitos na área utilizam poucos indivíduos e poucos movimentos para a classificação, tornando seus resultados estatisticamente pouco relevantes (ATZORI et al., 2014).

Para traduzir a vasta e complexa gama de informações contidas nos sinais de eletromiografia de superfície (sEMG), seja para controle ou para identificação de problemas neuromusculares, precisa-se de técnicas avançadas de análise de dados e de aprendizado de máquina capazes de lidar com grande quantidade de dados. Com o advento dos “data papers” muitas bases de dados passaram a estar abertas para uso em pesquisa, o que ajuda no desenvolvimento de algoritmos baseados em aprendizado de máquina já que necessitam

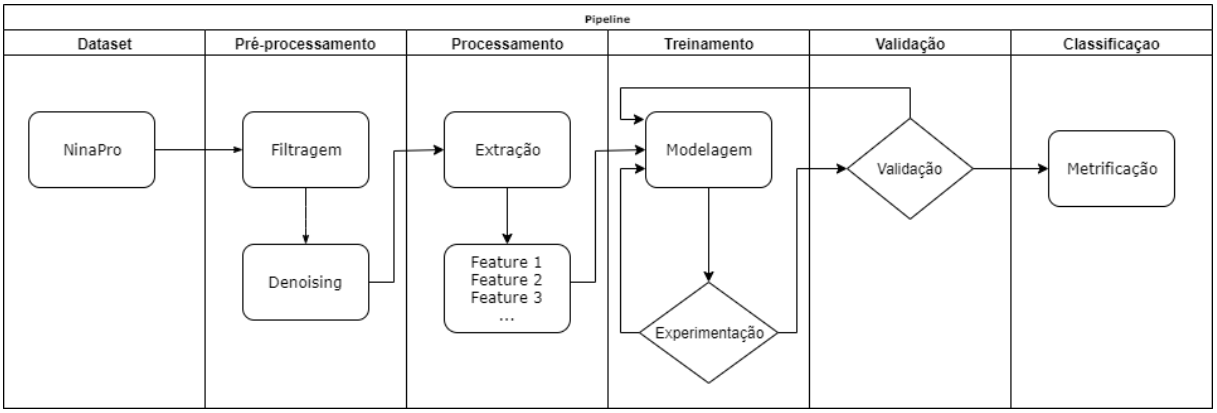
de um grande volume de dados. Essa grande quantidade de dados é necessária para se validar a robustez das técnicas desenvolvidas já que o reconhecimento dos movimentos utilizando EMG varia com o indivíduo, protocolo, equipamento de aquisição e processamento (PHINYOMARK; SCHEME, 2018).

A área de reconhecimento de padrões diz respeito à descoberta automática de regularidades em dados usando algoritmos computacionais e ao uso dessas regularidades para se tomar decisões em determinadas tarefas. Espera-se que um algoritmo de aprendizado de máquina consiga generalizar o conhecimento obtido durante seu treinamento, categorizando corretamente exemplos novos, diferentes dos exemplos com os quais ele foi treinado (BISHOP, 2006).

Utilizando conhecimentos da natureza dos sinais eletromiográficos e da grande área de aprendizado de máquina, neste trabalho serão construídos, implementados e validados três algoritmos de reconhecimento automático de padrões nestes sinais com o objetivo de classificá-los como um entre um conjunto de movimentos possíveis, dado que cada sinal foi extraído de uma pessoa durante a execução de um gesto.

O resumo dos passos realizados podem ser vistos na figura 1.

Figura 1: *Pipeline* de trabalho.



Fonte: Autoria própria.

1.1 Motivação

O algoritmo, depois de treinando para como realizar a classificação pode ser utilizado para o controle de dispositivos, próteses e computadores, podendo então melhorar a qualidade de vida de pessoas com movimentos limitados.

Dada uma janela temporal amostrada pelo eletrodo, a classificação pode ser feita e

baseando-se no movimento detectado o sistema pode enviar comandos para outro dispositivo. Por exemplo, pode-se escolher os movimentos que um indivíduo lesado medular tem mais dificuldade de realizar e utilizar os comandos enviados pelo sistema para o controle de uma prótese auxiliar. Ou pode-se escolher um conjunto de movimentos do braço e utilizá-lo para controlar o ponteiro em um computador, que é uma tarefa que exige certa coordenação motora da mão.

Além disso um bom sistema de reconhecimento de movimentos pode ser utilizado para operação remota de equipamentos pesados, equipamento médico, controle de robôs em locais de difícil acesso, entre outros. O reconhecimento de gestos por eletromiografia, no geral, pode ser utilizado como mais uma ferramenta de interfaceamento homem-máquina.

1.2 Objetivos

O objetivo geral deste trabalho é o treinamento de modelos de aprendizado de máquina para o reconhecimento e classificação de movimentos de membros superiores utilizando sinais de eletromiografia de superfície da base de dados Ninapro, e comparação da performance destes modelos entre si para avaliação de qual método é mais adequado em termos de acurácia balanceada, principalmente, mas também de outras métricas que serão discutidas.

Os objetivos específicos são:

- Pré-processamento de sinal mielétrico: Entender como se pode realizar filtragens e a limpeza do sinal sEMG para utilização posterior.
- Processamento de sinal miolétrico: Estudo de quais informações podem ser extraídas do sinal sEMG são úteis para a classificação de movimentos.
- Modelagem: Estudo e implementação de modelos de aprendizado de máquina visando a obtenção de algoritmos de classificação automática de movimentos baseando-se somente em eletromiografia de superfície.
- Comparação e avaliação dos modelos criados.

1.3 Organização

Este trabalho está organizado em 6 capítulos:

1. Introdução. Neste capítulo algumas noções do objeto de estudo e de desenvolvimento deste projeto são introduzidas ao leitor.

2. Fundamentação teórica. Neste capítulo é feita uma exposição de toda a teoria que é aplicada neste trabalho, e contém a revisão bibliográfica, listando os autores que realizaram trabalhos nos quais este foi baseado.

3. Metodologia. Dada a teoria, nesta seção explica-se como o conhecimento foi aplicado, com quais ferramentas, métodos, materiais, protocolos e tomadas de decisão.

4. Resultados. Explanação dos resultados obtidos ao se utilizar a metodologia proposta.

5. Discussão. Neste capítulo é feita uma avaliação e uma comparação dos resultados obtidos com os trabalhos citados na revisão bibliográfica.

6. Considerações finais. Reflexão sobre os resultados atingidos, considerações e proposição de trabalhos futuros.

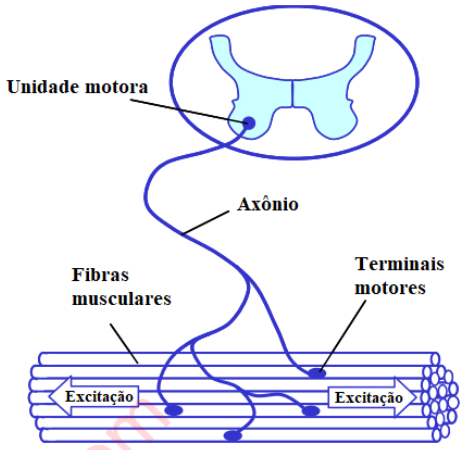
2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é exposto o conhecimento necessário para o entendimento deste trabalho. Há uma explicação sobre a fisiologia das células mióticas e sinais eletromiográficos, além do embasamento teórico sobre os métodos de pré-processamento utilizados nos sinais, o processamento realizados neles, as caracteríticas que podem ser extraídas, e por fim uma seção sobre aprendizado de máquina.

2.1 Fisiologia da atividade elétrica muscular

Uma unidade motora é o menor componente funcional responsável pelo processo de controle muscular. Na figura 2 pode-se ver a unidade motora, formada pelo corpo celular, dendritos e axônios de um neurônio motor, e as fibras musculares que estes inervam. Apesar de serem chamadas assim, todas se comportam "como uma" durante o processo de inervação (KONRAD, 2006).

Figura 2: Unidade motora.



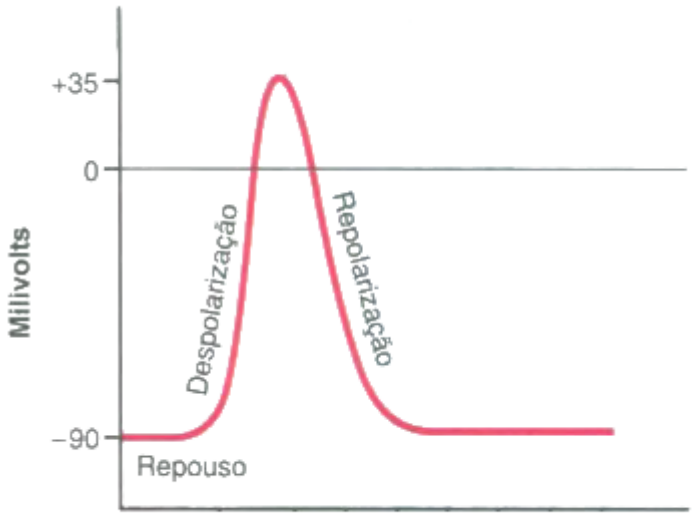
Fonte: Adaptado de Konrad (2006).

Existe uma diferença entre a quantidade de íons no interior e no exterior das células musculares. Estes íons são em sua maior parte sódio Na^+ e potássio K^+, e estão presentes dentro e fora das células em equilíbrio dinâmico. Há potenciais de difusão de entrada de íons Na^+ e de saída de íons K^+, devido à diferença de concentração destes. A bomba sódio-potássio também influencia no equilíbrio por realizar o transporte ativo de íons de sódio para fora e íons de potássio para dentro da célula (ARANHA, 2017; KONRAD, 2006).

Os sinais nervosos são transmitidos por potenciais de ação que atravessam as

membranas celulares das fibras musculares, que estão inicialmente polarizadas em aproximadamente -90mV. Devido ao potencial de ação a membrana da célula se torna muito permeável ao Na^+, fazendo com que estes íons entrem e o potencial aumente, ocorrendo então a despolarização da célula, como mostra a figura 3. Finalmente, com os canais de sódio se fechando, o equilíbrio volta a ser reestabelecido com muitos íons de potássio difundindo para o exterior da célula (GUYTON; HALL, 1988).

Figura 3: Despolarização e repolarização da célula muscular.



Fonte: Adaptado de Guyton e Hall (1988).

Uma unidade motora é composta de diversas fibras musculares e quando um neurônio motor é ativado diversas fibras são despolarizadas e repolarizadas. Medindo-se o potencial entre dois pontos das fibras musculares são observados todos esses potencias superpostos. Esse conjunto de sinais superpostos é chamado de "*Motor unit action potential*", ou MUAP. A superposição de diversas MUAP's resultam no sinal de eletromiografia (CHOWDHURY et al., 2013; KONRAD, 2006).

2.2 Eletromiografia de superfície

Com a devida preparação e amplificação os sinais podem ser capturados na superfície da pele. Sinais de eletromiografia de superfície (sEMG) tem por volta de 5mV (em atletas) e a maior quantidade de informação está entre 6 e 500Hz, com a maior parte da energia espectral entre 20 e 150Hz (KONRAD, 2006).

A qualidade dos eletrodos é o ponto mais crucial na captura dos sinais sEMG, pois é necessário uma distorção mínima e a maior relação sinal-ruído (SNR) possível. Para uma boa fidelidade de captura é necessário haver amplificação diferencial nos eletrodos, alta impedância de entrada, filtragem, estabilidade do eletrodo na pele, e uma boa localização e orientação dos eletrodos. É importante não colocar os eletrodos perto do tendão do músculo nem exatamente no ponto motor do músculo, que é onde o nervo motor inerva as fibras musculares (LUCA, 2002). A figura 4 mostra um exemplo de posicionamento dos eletrodos no antebraço. É importante salientar que os dois mecanismos mais importantes que influenciam as amplitude e densidade dos sinais são o recrutamento e a frequência de disparo das células nervosas, que por consequência controlam a força muscular a ser exercida (KONRAD, 2006).

Figura 4: Exemplo de posicionamento de eletrodos.

Fonte: Adaptado de Pizzolato et al. (2017).

Diversas fontes de interferência são inerentes à forma como o sinal EMG é gerado e capturado (CHOWDHURY et al., 2013). Entre elas estão:

- **Ruído de eletrodo:** depende das impedâncias dos eletrodos e do circuito que realiza a amplificação do sinal EMG. Eletrodos de alta impedância fornecem um valor melhor de SNR. Por sua vez, a impedância do eletrodo depende do material que é feito, do seu tamanho, e do circuito e instrumentos à qual está ligado.
- **Artefatos de movimento:** movimentos do cabo que conecta o eletrodo ao amplificador e da superfície do eletrodo em relação à pele criam os chamados artefatos de movimento. Eletrodos devidamente fixados e uma camada de gel condutor entre a pele e o eletrodo ajudam a evitar este problema. Entretanto quando os músculos são contraídos ou relaxados a pele, eletrodos e os próprios músculos se movem em relação à si mesmos, acarretando ruídos referentes às diferenças de potenciais em diferentes pontos da pele. O intervalo de frequência dos ruídos de movimento está normalmente entre 1 e 10Hz.
- **Ruído eletromagnético:** Todos equipamentos eletrônicos estão sujeitos à uma contínua e inevitável exposição à ruídos elétricos e magnéticos, entre os quais o mais preocupante é o ruído de linha (60Hz).

- ***Cross-talk**:* Ocorre quando um eletrodo captura um sinal de sEMG de um grupo muscular do qual ele à priori não deveria capturar. Aumenta com a quantidade de gordura subcutânea, e depende de parâmetros como a distância entre eletrodos e a área da superfície destes. A correlação cruzada entre os sinais EMG não é uma medida interessante de *cross-talking*, e ele não pode ser reduzido de maneira efetiva por filtros passa-alta.

- **Ruído interno:** Fatores fisiológicos, bioquímicos e anatômicos influenciam na quantidade de ruído. Alta quantidade de gordura corporal e tecido subcutâneo são os fatores que mais afetam o sinal EMG (diminuem a amplitude do sinal), bem como características elétricas da pele.

- **Instabilidade inerente ao sinal:** As amplitudes do sinal EMG são por natureza quase aleatórias. Entre 0 e 20Hz as componentes são instáveis e variam muito com a taxa de dispara das unidades motoras.

- **Artefatos eletrocardiográficos:** o chamado ECG é a uma das principais fontes de ruído nos sinais de eletromiografia de superfície. São ocasionados pela atividade elétrica do coração. Filtros passa alta são eficientes para removê-los.

Além desses, existem outros fatores que influenciam diretamente as informações contidas no sinal sEMG: a força de contração da unidade motora; o número de unidades motoras ativas; amplitude, duração e formato das MUAP's; interações mecânicas entre as fibras musculares; recrutamento e estabilidade das unidades motoras. (LUCA, 1997).

2.3 Pré-processamento

A função do pré-processamento do sinal é condicioná-lo à fase de processamento, cuja função é extrair as informações importantes deste. Estas informações são compostas tanto pelas informações de excitação da musculatura esquelética que está abaixo da pele, quanto pelo equipamento de aquisição. A forma como o sinal é coletado e também as características fisiológicas do local a ser instrumentado tem alta influência na amplitude e nas componentes espectrais do sinal (GERDLE et al., 1999).

2.3.1 Transformada Wavelet

O ramo da matemática que levou ao estudo das *wavelets* começou com Joseph Fourier e suas teorias sobre análise de frequência. Com a noção de análise em frequência, veio também a noção de análise em escala. A primeira menção ao termo *wavelet* foi feita na tese de A. Haar, em 1909 (GRAPS, 1995).

A transformada *wavelet* pode ser interpretada como uma rotação no espaço de funções para um outro domínio. Na transformada de Fourier esse domínio contém funções de base seno e cosseno, e no caso da transformada *Wavelet* essas funções de base são as chamadas *wavelets*, ou *mother wavelets* (*wavelets*-mãe). Funções de base são funções ortogonais que podem ser utilizadas para construir outras funções (GRAPS, 1995).

A principal diferença entre a transformada de Fourier (neste caso a FFT) e a transformada *wavelet* é que *wavelets* individuais estão localizadas no tempo, ao contrário das funções seno e cosseno que se repetem de maneira igual (são periódicas) por todo seu domínio. Por exemplo, ao analisarmos uma janela do sinal, pode-se dizer quais componentes seno e cosseno estão presentes, mas não é possível dizer exatamente onde elas estão presentes, o que não acontece com as *wavelets*.

Tendo apresentada a transformada *wavelet*, pode-se definir a base matematicamente, como mostra a equação 2.1. Observa-se que esta base tem os parâmetros s e l, que são respectivamente os fatores de escala e localização (ou translação). Pode-se ver nesta mesma equação que as funções da base são escaladas em potências de 2 (GRAPS, 1995).

$$\phi_{s,l}(x) = 2^{-\frac{s}{2}}\phi(2^{-s}x - l) \tag{2.1}$$

A função de escala da *wavelet*-mãe ϕ é definida como na equação 2.2. Os coeficientes c_k são os coeficientes *wavelet* e precisam satisfazer as condições dadas pelas equações 2.3 e 2.4, onde δ é a função delta.

$$W(x) = \sum_{k=-1}^{N-2} (-1)^k c_{k+1}\phi(2x + k) \tag{2.2}$$

$$\sum_{k=0}^{N-1} c_k = 2 \tag{2.3}$$

$$\sum_{k=0}^{N-1} c_k c_{k+2l} = 2\delta_{l,0} \tag{2.4}$$

Define-se a Transformada *Wavelet* Contínua (CWT) como na equação 2.5. Como é possível variar s e u de forma contínua, o resultado da transformação de um vetor unidimensional é uma matriz (ou uma "imagem") altamente redundante (STéPHANE,

2009).

$$W_{s,l}(f) = \langle f, \phi_{s,l} \rangle = \int_{-\infty}^{\infty} f(t)\phi^*_{s,l}(t)dt \tag{2.5}$$

A transformada discreta é definida de forma semelhante. Primeiro, define-se uma *wavelet*-mãe ϕ com suporte em $[-K/2, K/2]$, de tal forma que a sua versão escalada por a^j (onde $a = 2^{1/v}$), com $1 \leq a^j \leq NK^{-1}$ (onde N é a frequência de amostragem do sinal discreto) é dada pela equação 2.6 (STéPHANE, 2009). Pode-se perceber que essa equação é semelhante à sua forma contínua 2.1. Entretanto, o parâmetro de translação não está presente nesta equação, mas como será visto a seguir isto não será um problema.

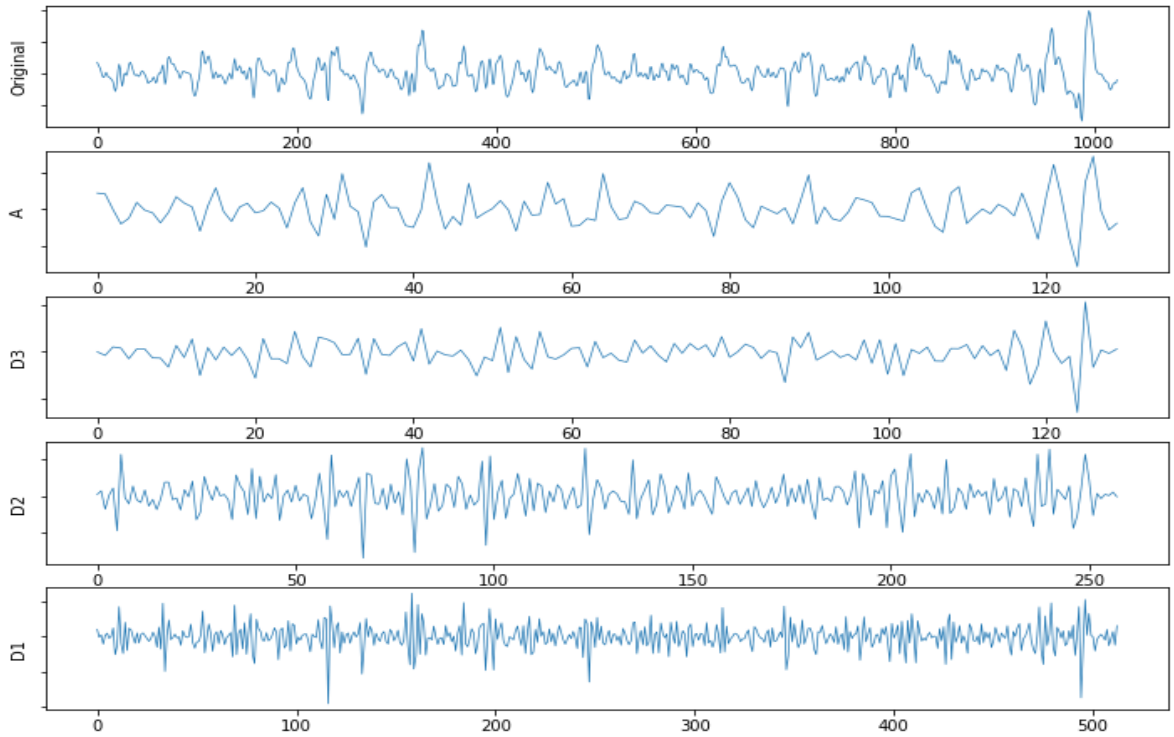
$$\phi_j[n] = \frac{1}{\sqrt{a^j}}\phi\left(\frac{n}{a^j}\right) \tag{2.6}$$

A Transformada *Wavelet* Discreta é dada pela equação 2.7, que é um convolução em uma dimensão. Tendo só o parâmetro a de escala consegue-se uma transformada semelhante à transformada contínua. A transformada *wavelet* discreta possui uma transformada inversa na forma $W^{-1}(W(f)) = f$. Detalhes sobre as propriedades das transformadas e das *wavelets* podem ser encontradas em Stéphane (2009).

$$W_j f[n] = \sum_{m=0}^{N-1} f[m]\phi_j[m-n] = f * \phi_j[-n] \tag{2.7}$$

Ao se utilizar diversas escalas (também chamadas de níveis) na transformada *wavelet* obtém-se mais ou menos vetores de saída. Por exemplo, ao se utilizar uma decomposição *wavelet* de nível 3 obtém-se 4 saídas: A, $D3$, $D2$ e $D1$. O primeiro vetor de coeficientes é composto pelas componentes de baixa frequência e é chamado de "aproximação". Os próximos níveis são chamados "detalhes'" e contém as componentes de frequência maior, como mostra a figura 5.

Figura 5: Níveis de decomposição de um sinal sEMG na transformada wavelet discreta utilizando a *wavelet*-mãe Daubechie 2.



Fonte: Autoria própria.

Como é possível realizar a reconstrução do sinal original utilizando a transformada inversa, pode-se modificar esses vetores para modificar o espectro de frequência do sinal original, o que significa que pode-se filtrar o sinal. É dito que o sinal sEMG observado é composto pelo sinal original mais um ruído: $s[n] = f[n] + e[n]$, onde s, f e e são o sinal sEMG, o sinal EMG original, e um ruído gaussiano, respectivamente (PHINYOMARK; LIMSAKUL; PHUKPATTARANONT, 2009).

2.3.2 *Denoising*

O *denoising*) pode ser dividido em três passos. Primeiro é necessário escolher a *wavelet*-mãe a ser utilizada. Como existem diversas *wavelets* conhecidas e possíveis de serem utilizadas, este passo é crucial. Após isto é necessário escolher o nível da decomposição a ser utilizado (pois isso influenciará nas componentes de frequência que cada nível conterá). Por último, diminui-se as componentes de alta frequência do sinal. Isso pode ser feito se aplicando um *threshold* nos coeficientes de detalhes (PHINYOMARK; LIMSAKUL; PHUKPATTARANONT, 2009).

Uma variedade de ruídos provenientes dos instrumentos de medição causam grandes

problemas na análise de sinais eletromiográficos, de tal forma que removê-los é um dos passos mais importantes. Enquanto ruído de linha pode ser removido utilizando filtragens convencionais, a eliminação de outras fontes ruidosas, como o ruído branco gaussiano, necessita de um processamento de sinal mais elaborado. Algoritmos de *denoising* utilizando *wavelets* foram utilizados nesta tarefa (PHINYOMARK; LIMSAKUL; PHUKPATTARANONT, 2009).

Em Phinyomark, Limsakul e Phukpattaranont (2011) avaliou-se o uso de diferentes métodos de aplicação da transformação por *threshold* no *denoising* do sinal sEMG. Para avaliar o resultado foi utilizado o erro quadrático médio (MSE) entre os coeficientes do sinal original e os coeficientes do sinal após o *denoising*. Conclui-se que a melhor combinação é a utilização conjunta da *wavelet* mãe Daubechie 2, a decomposição de quarta ordem, o chamado "*thresholding* universal" (equação 2.8) e a transformação "soft", descrita pela equação 2.9, onde cD_j é o coeficiente de detalhe de nível j.

$$THR_{UNI} = \sigma\sqrt{2log(N)}, \quad \sigma = \frac{\text{median}(|cD_j|)}{0.6745} \tag{2.8}$$

$$cD_j = \begin{cases} \text{sgn(cD}_j)(cD_j - THR_j), & \text{se } |cD_j| > THR_j \\ 0, & \text{c.c.} \end{cases} \tag{2.9}$$

Em Phinyomark, Limsakul e Phukpattaranont (2009) foram analisados os resultados de se realizar o *denoising* com 53 *wavelets* diferentes, com variados níveis de decomposição e diferentes transformações de *threshold*. Utilizando-se o MSE como métrica para avaliação da qualidade do *denoising* concluiu-se que as *wavelets*-mãe chamadas $db1$, $bior1.1$, $rbio1.1$ eram as melhores candidatas para este papel, com a $db2$ e a $db7$ também sendo marginalmente melhores que outras *wavelets*. A transformação por *soft-thresholding* novamente foi reportada como sendo a mais adequada nesta tarefa.

2.4 Processamento

Há diversas maneiras de descrever o sinal EMG, que podem ser separadas em variáveis de amplitude e de frequência. O cálculo dessas variáveis é útil pois diminui-se a quantidade de dados que precisam ser armazenados ou tratados. Entretanto uma simples medida de média do sinal resultará em zero, já que o sinal sEMG pode ser descrito por uma variável aleatória de média zero, sendo então necessário utilizar métodos de descrição e análise mais sofisticados. A seguir estão alguns métodos de extração de características

mostrados por Gerdle et al. (1999).

- Retificação: Já que uma média simples do sinal resulta em zero, uma abordagem válida é retificar o sinal sEMG. A retificação pode ser feita de duas maneiras: na primeira os valores negativos são truncados para 0, e na segunda o seu sinal é invertido, conservando a energia do sinal.

- *Average Rectified Value* (Valor médio retificado): o sinal retificado ainda varia aleatoriamente, entretanto as flutuações instantâneas podem ser suprimidas utilizando um filtro passa-baixa. O ARV é a integral da curva resultante, ou seja, a área debaixo do envelope do sinal retificado.

- EMG Integrado (iEMG): O iEMG é a integral do envelope do sinal retificado durante uma janela de tempo pré-definida τ. Como o envelope só contém valores positivos, o iEMG sempre aumenta conforme τ aumenta.

- O valor RMS mede a raiz da potência do sinal e extrai mais informações sobre ele que os métodos acima, já que contém uma informação fisiológica mais clara. Seu cálculo pode ser feito utilizando a equação 2.10.

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \tag{2.10}$$

- Amplitude de Pico (PA): É definida como a componente de frequência mais dominante no sinal.

- Largura de banda: É definida como o intervalo no qual as componentes de frequência caem 3dB a partir da PA.

- Mediana da frequência: É definida como a frequência que divide o espectro do sinal em duas áreas de energia igual.

- Média da frequência: valor médio da soma das componentes de frequência do sinal.

- *Zero-Crossings* (Passagens por zero): ZC é definido pelo número de vezes que o sinal cruza o zero dividido por dois (equação 2.11).

- *Turn counting* (TC): É definido pelo número de vezes que o sinal atinge um pico, dividido por dois.

Centroide espectral (SC) é o centro de massa do espectro de frequência do sinal. Neste trabalho a representação em frequência utilizada será a transformada de Fourier do sinal (FFT), e sua formulação pode ser vista na equação 2.12.

$$ZCC = \frac{1}{2}\sum_{i=2}^{N} |sgn(x_i) - sgn(x_{i-1})| \tag{2.11}$$

$$SC = \frac{\sum_0^{\omega_{max}} F(\omega)\omega}{\sum_0^{\omega_{max}} \omega} \tag{2.12}$$

Em Atzori et al. (2014) utilizou-se o valor RMS, estatísticas temporais (*Time domain statistics*), a Transformada de Wavelet marginal (usando a *wavelet* db7 e três níveis de decomposição) e o histograma (20 *bins*, limitados em 3δ). Em Hudgins, Parker e Scott (1993) utilizou-se o valor absoluto médio (MAV), de equação 2.13), *mean absolute value slope* MAVS, de equação 2.14), ZCC, TC e a *waveform length* (WL), de equação 2.15.

$$MAV = \frac{1}{N}\sum_{i=1}^{N} |x_i| \tag{2.13}$$

$$MAVS = \sum_{i=0}^{N-1} \dot{X}_{i+1} - \dot{X}_i \tag{2.14}$$

$$WL = \sum_{i=1}^{N} |\Delta x_i| \tag{2.15}$$

No trabalho de Veer e Sharma (2016) utilizou-se, além de algumas características discutidas acima, a energia do sinal sEMG (equação 2.16), a sua variância (que é igual ao quadrado do desvio padrão, de equação 2.22) e a amplitude de Willison (2.17). Este autor também usou coeficientes de auto-regressão de ordem 4, que são obtidos ao se otimizar os coeficientes a na equação 2.18, de tal forma que o x'_n obtido através dos coeficientes e dos p valores anteriores à x_n tenham o menor erro possível em relação ao seu valor real x_n.

$$\text{Energy} = \sum_{i=1}^{N} x_i^2 \tag{2.16}$$

$$\begin{aligned} \text{WAMP} &= \sum_{i=1}^{N} f(|x_n - x_{n+1}|) \\ f(x) &= \begin{cases} 1, \text{ se } x >= threshold \\ 0, \text{c.c.} \end{cases} \end{aligned} \tag{2.17}$$

$$x'_n = \sum_{i=1}^{p} a_i x_{n-i} + a_0 \tag{2.18}$$

Em Búrigo (2014) e no trabalho de Viana et al. (2019) outras estatísticas de alta ordem (*high order statistics*, ou HOS) foram utilizadas. O termo HOS vem do fato desses valores serem calculados utilizando-se as potências terceiras ou superiores de uma amostra. A curtose (ou *kurtosis*), mostrada na equação 2.20, é uma medida que caracteriza o achatamento de uma distribuição. A obliquidade (ou *skewness*) de uma distribuição é calculada através da equação 2.21, e representa a assimetria de uma distribuição em relação à média.

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{2.19}$$

$$Kurtosis = \frac{\frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^4}{\left(\frac{1}{N} \times \sum_{i=1}^{N} (x_i - \bar{x})^2\right)^2} \tag{2.20}$$

$$Skewness = \frac{\frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^3}{\left(\sqrt{\frac{1}{N} \times \sum_{i=1}^{N} (x_i - \bar{x})^2}\right)^3} \tag{2.21}$$

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{2.22}$$

Os quantis também podem ser utilizados, pois trazem uma informação sobre a distribuição e variação do sinal. Outra variável que traz uma informação semelhante é o histograma: cada *bin* de um histograma é a contagem de quantos valores estão dentro do intervalo definido por aquele *bin*.

Em Tang et al. (2012) utilizou-se a energia relativa entre os canais sEMG, que é uma medida de quantas vezes mais energia um sinal de um canal tem em relação à um outro canal. Como o *dataset* utilizado neste trabalho possui 12 canais, existem 66 possibilidades de combinação $\frac{E_i}{E_j}$ com $i \neq j$, onde E_k é a energia do canal k. A formulação

pode ser vista na equação 2.23.

$$\begin{aligned} R =& [R_{1,2}, R_{1,3}, ..., R_{1,p}, R_{2,3}, R_{2,4}, ..., R_{2,p}, ..., R_{p-1,p}] \\ R_{i,j} =& \frac{\text{Energy}(S_i)}{\text{Energy}(S_j)} \end{aligned} \tag{2.23}$$

Uma outra medida relativa (entre canais) possível de ser utilizada é o coeficiente de correlação de Pearson. Este coeficiente (equação 2.24) mede a correlação entre duas variáveis de escala métrica.

$$\rho(x, y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} = \frac{\sigma_{X,Y}}{\sqrt{\sigma_X \cdot \sigma_Y}} \tag{2.24}$$

Outra medida de correlação é o ρ de Spearman, que mede a concordância entre postos. Neste trabalho ela foi utilizada entre canais, e é calculado entre os sinais u e v como na equação 2.25 (ZWILLINGER; KOKOSKA, 1999). Da mesma maneira utilizou-se o valor do teste de Wilcoxon , dada pela equação 2.26, com $(x_{1,i}, x_{2,i})$ sendo o i-ésimo par de amostras na qual $|x_{1,i} - x_{2,i}| \neq 0$ são diferentes (onde há N_r pares de x_1 e x_2 diferentes), e R_i é o posto entre o par i (WILCOXON, 1945).

$$\begin{aligned} r_S &= 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \\ d_i &= u_i - v_i \end{aligned} \tag{2.25}$$

$$W = \sum_{i=1}^{N_r}[sgn(x_{2,i} - x_{1,i}) \cdot R_i] \tag{2.26}$$

2.5 Aprendizado de máquina

O problema discutido neste trabalho é de classificação. Isto significa que deseja-se uma caracterização qualitativa de um dado a partir de suas características, ou seja, baseado somente no sinal sEMG deseja-se dizer qual o movimento o paciente realizou no momento em que o sinal foi capturado. Variáveis qualitativas também podem ser chamadas de categóricas.

Nesta seção serão explicados a lógica e a formalização matemática dos algoritmos de aprendizado de máquina que utilizou-se nas classificações dos movimentos. Nas equações

x (ou x_n) e ϕ se refere ao vetor de características de uma amostra, Y se refere à classe desta amostra, y (ou y_n) se refere à saída do modelo para a entrada x (ou x_n).

2.5.1 Regressão Logística

Primeiramente será abordado um caso de classificação no qual se quer estimar a probabilidade de x pertencer à uma classe 1 (C_1) ou 2 (C_2). A probabilidade da classe Y de x ser 1 é $p(x) = p(Y = 1|x)$ (é lido como "A probabilidade de Y ser igual à 1 dado x"). Pode-se convencionar que $p > 0.5$ implica que x pertence à classe 1 e caso contrário ele pertence à classe 2 James et al. (2013).

Usando a regra da soma $p(x) = \sum_j p(x|Y = j)$ e o Teorema de Bayes (que vem da regra da multiplicação $p(x|Y)p(Y) = p(Y|x)p(x)$), pode-se escrever a equação 2.27, onde C_1 é o evento no qual x pertence à classe 1 e C_2 é o evento no qual x pertence à classe 2.

$$p(C_1|x) = \frac{p(x, C_1)}{p(x)} = \frac{p(x|C_1)p(C_1)}{p(x)} = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)} \tag{2.27}$$

A primeira igualdade diz que a probabilidade do evento C_1 dado o evento x é dado pela probabilidade de x e C_1 acontecerem ao mesmo tempo (que é dado por $p(x|C_1)p(C_1)$, como visto na segunda igualdade), sobre a probabilidade de x ter acontecido. A equação 2.27 pode ser reescrita como em 2.28.

$$\frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)} = \sigma(a) \tag{2.28}$$

Na equação 2.28, $\sigma(x) = \frac{1}{1+e^{-x}}$ é a função chamada função logística ou sigmóide. Na mesma equação $a = ln\left(\frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}\right)$. No caso de $K > 2$ classes, utiliza-se a equação 2.27 generalizada, que toma a forma da 2.29, chamada de exponencial normalizada. Nesta equação $a_k = ln(p(x|C_k)p(C_k))$.

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)} = \frac{e^{a_k}}{\sum_j e^{a_j}} \tag{2.29}$$

Em Bishop (2006) é visto que $a_k(x)$ pode ser escrito na forma $a_k(x) = w_k^T x + w_{k_0}$ (onde w é um vetor linha) ao se assumir que as densidades de probabilidade $p(x|C_k)$ são de distribuição normal gaussiana, que todas as classes C_i compartilham a mesma matriz

de covariância e que x pode variar de maneira contínua.

Sendo ϕ um vetor de características, vale a equação $p(C_k|\phi) = y(\phi) = \frac{e^{a_k}}{\sum_j e^{a_j}}$, com $a_k = w_k^T\phi + b$. O objetivo, portanto, é encontrar os vetores w_k que dão a melhor estimativa de $p(C_k|\phi)$.

Para o caso de duas classes a função de verossimilhança pode ser escrita como na equação 2.30. A verossimilhaça é uma estimativa da probabilidade de um modelo ter gerado os dados observados, ou seja, mede até que ponto o modelo descreve a realidade dos dados. Nesta equação t é um vetor de tamanho N com as classes alvo t_n. Em um caso multi-classe t seria T, uma matriz de dimensão $N \times K$ onde cada linha t_n é um vetor de zeros, exceto pela posição t_nk (representando a classe real de ϕ_n) que é 1 e y_{nk} é a probabilidade predita pelo modelo de uma amostra ϕ_n pertencer à classe C_k, dado que $y_n = \{p(C_1|\phi_n), ..., p(C_K|\phi_n)\}$.

$$p(t|w_1, ..., w_K) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n} \tag{2.30}$$

O logaritmo negativo da função de verossimilhança, também chamada de entropia cruzada, toma a forma da equação 2.31. Essa função é usada como uma medida de de erro do valor da probabilidade encontrado pelos valores de w_k e os valores reais t_n (BISHOP, 2006).

$$E(w_1, ..., w_K) = -ln(p(t_n|w_1, ..., w_K)) = -\sum_{n=1}^{N}(t_n ln(y_n) + (1 - t_n)ln(1 - y_n)) \tag{2.31}$$

É possível diminuir esse erro ao alterar w_k. Pode-se utilizar o gradiente da função E em relação às variáveis de w para ter uma estimativa da "direção" do gradiente que leva ao mínimo de E, utilizando o fato de que $y_n = \sigma(a_n) = \sigma(\omega^T\phi_n)$. Derivando E em relação à w obtêm-se a equação 2.32.

$$\nabla E(w_1, ..., w_K) = \sum_{n=1}^{N}(y_n - t_n)\phi_n \tag{2.32}$$

Com o gradiente do erro pode-se atualizar os valores de w, "caminhando" por ele até encontrar um valor mínimo de erro, que será local ou global. Para o caso multi-classe

a equação é generalizada e toma a forma da equação 2.33, que é a entropia cruzada para o caso multi-classe. Detalhes sobre os cálculos podem ser vistos em Bishop (2006).

$$E(w_1, ..., w_K) = -ln(p(T|w_1, ..., w_K)) = -\sum_{n=1}^{N}\sum_{k=1}^{K}[t_{nk}ln(y_{nk})] \tag{2.33}$$

2.5.2 *k-Nearest Neighbors*

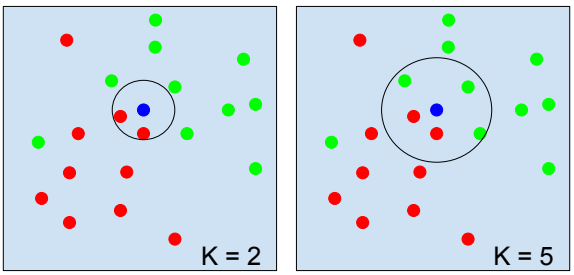
O algoritmo "*K-Nearest Neighbors*" é utilizado para se obter uma estimativa da probabilidade da classe de um dado x. No caso de uma classificação, sua saída será a probabilidade de um certo conjunto de características $x = (x_1, x_2, ..., x_p)$ pertencer à classe C_j (JAMES et al., 2013).

No algoritmo, primeiro os K pontos mais próximos de x (uma amostra a que se quer inferir a classe) que são do conjunto de treino são selecionados. Esse conjunto será chamado $\aleph_0$. Então a probabilidade de Y pertencer à C_j é estimada como a fração de pontos em $\aleph_0$ que pertencem à classe C_j, e pode ser calculada como na equação 2.34, onde x_i é um ponto do conjunto de treino que está próximo de x, C_i é sua classe ($C(x_i)$ é a classe de x_i) e $I(cond) = 1$ caso $cond$ seja verdadeira e 0 caso $cond$ seja falso (JAMES et al., 2013).

$$p(Y = C_j|x) = \frac{1}{K}\sum_{x_i \in \aleph_0} I(C(x_i) = C_j) \tag{2.34}$$

A escolha do K faz toda a diferença na hora de se criar o classificador. Com um K pequeno, o modelo aprenderá as micro características do conjunto de treino, e por consequência ocorrerá o *overfitting*. Pode-se dizer que o *limite de decisão* (do inglês *decision boundary*) se torna mais flexível, com pouco viés e uma variância alta. Na figura 6 pretende-se classificar o ponto azul, como um exemplo. À esquerda K=2, e o ponto seria classificado como pertencente à classe vermelha, enquanto na direita, com K=5, o ponto seria classificado como pertencente à classe verde (JAMES et al., 2013).

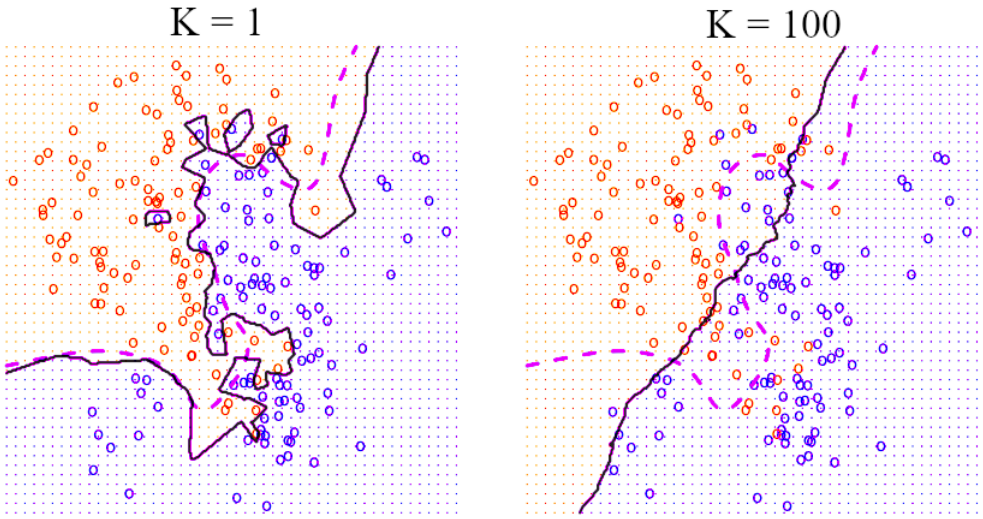
Figura 6: Exemplo do impacto da escolha do K.



Fonte: Autoria própria.

Por outro lado, com um K alto, o modelo se atentará às características macroscópicas do conjunto de treinamento, produzindo resultados próximos do linear. A figura 7 exemplifica o impacto de K na classificação. Na figura, os círculos representam os dados, os pontos representam o domínio e a cor representa a classe, com a linha pontilhada representando o limite de decisão real. Pode-se perceber que com o aumento de K o limite de decisão se torna menos complexo (JAMES et al., 2013).

Figura 7: Exemplo do impacto da escolha do K.



Fonte: Adaptado de James et al. (2013).

2.5.3 Redes Neurais

O nome "Rede Neural" vem das tentativas de se encontrar uma formulação matemática que representasse a forma como sistemas biológicos processam a informação, por exemplo, como o cérebro aprende e processa as informações que recebemos. Há discussão se as redes neurais realmente são representantes plausíveis de um sistema biológico real, pois um realismo biológico implicaria em muitas limitações desnecessárias (BISHOP, 2006).

O cérebro humano é constituído de neurônios, que são os menores elementos funcionais. Juntos eles formam uma rede neural complexa altamente não-linear, onde se comunicam um com os outros através de sinais elétricos que viajam através das sinapses. Diferenças nas informações passadas pelas sinapses afetam o comportamento dos neurônios subsequentes, de forma a influenciar a resposta de todo o conjunto ao estímulo original (HAYKIN, 1999).

Uma rede neural pode ser um circuito eletrônico ou um programa que usa uma extensa rede de neurônios conectados para realizar cálculos. O sistema recebe uma sequên-

cia de entradas, as quais através das conexões e ativações produzem uma saída. As redes neurais são treinadas com algumas informações, produzindo determinadas respostas para certas entradas, e após o treinamento ela deverá gerar saídas razoáveis para entradas que nunca viu, ou seja, deve generalizar o seu treino (HAYKIN, 1999).

Entre as vantagens de se utilizar uma rede neural pode-se apontar: não linearidade, adaptabilidade, tolerância à falhas, uniformidade de análise e *design* (diferentes redes podem ser implementadas se utilizando dos seus conceitos básicos simples) e sua analogia biológica (HAYKIN, 1999).

Em um neurônio, elemento básico de uma rede neural, podem ser identificados 3 componentes:

- Um conjunto de sinapses com pesos ω_{kj}, significando que um sinal do neurônio k é passado ao neurônio j multiplicado por ω_{kj} (ou o caminho contrário dependendo do contexto).
- Um somador, que soma as entradas do neurônio já multiplicadas pelos seus pesos sinápticos.
- Uma função de ativação, de tal forma que a saída do neurônio é a saída do somador aplicada à esta função.

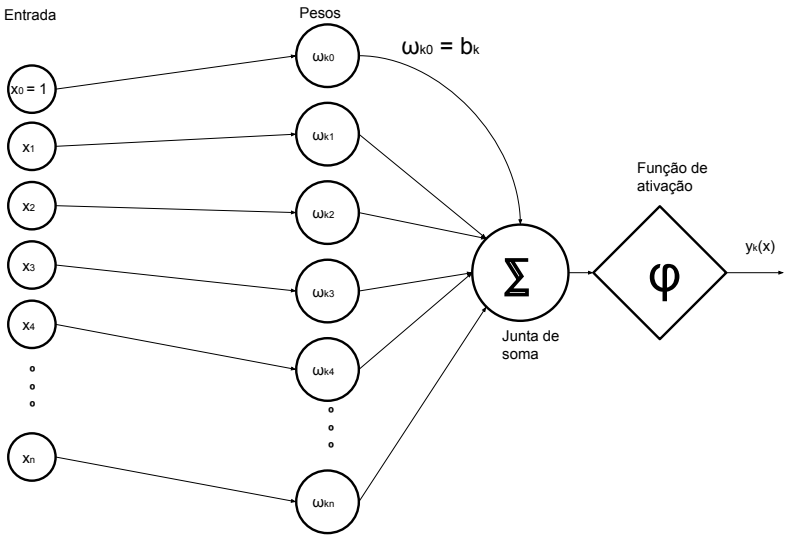
A figura 8 esquematiza esse modelo. Pode-se ver que uma das entradas do somador é fixa em 1. Na realidade x_0 é fixo, e o peso ω_{k0} é chamado de bias (denotado por b_k). Assim, a saída de um neurônio é dada pela equação 2.35 (HAYKIN, 1999).

$$y_k = \phi(\sum_{j=0}^{m} \omega_{kj} * x_j + b_k) \tag{2.35}$$

Caso tenha-se uma camada de neurônios nas quais todas entradas se conectam à todos os neurônios, a saída de cada neurônio será dada pela equação 2.35. Pode-se inserir mais camadas, de forma que a saída dos neurônios da camada anterior sejam entradas da próxima camada, podendo-se então montar uma rede neural com várias camadas de neurônios onde cada uma, sem contar a de saída, é chamada de camada escondida. Essa topologia de rede neural é comumente chamada de perceptron multicamada (MLP, do inglês).

Um exemplo de conexões de neurônios que formam uma rede neural é vista na figura 9. A figura representa a estrutura de uma rede neural com 8 entradas, 3 camadas

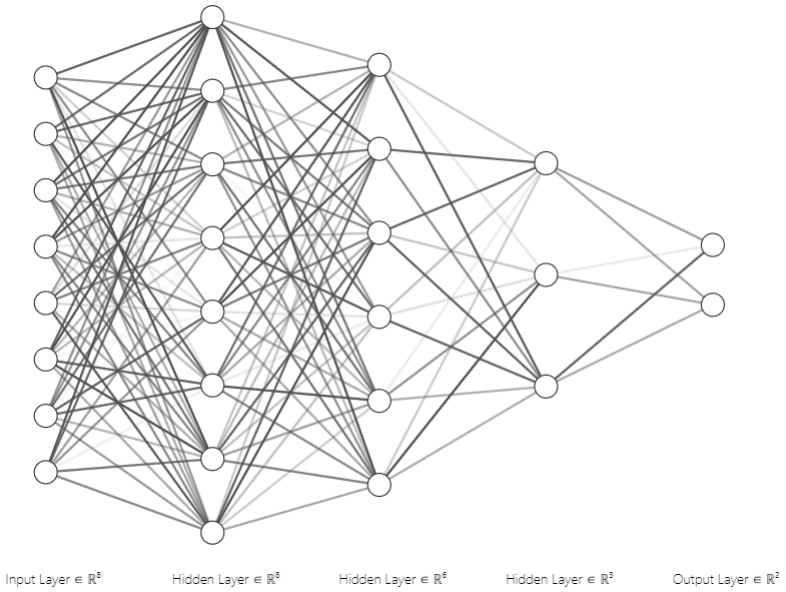
Figura 8: Modelo do neurônio.



Fonte: Autoria própria.

escondidas (com 8, 6 e 3 neurônios, respectivamente), e duas saídas. Em um contexto de aplicação, essa rede poderia ser um utilizada, por exemplo, para predizer o tipo e o estágio de uma doença (duas saídas), dada algumas características das células cancerosas (que nesse caso seriam 8, dado o número de neurônios na entrada da rede).

Figura 9: Exemplo de rede neural.



Fonte: Adaptado de LeNail (2019).

A popularidade do treinamento e uso dos MLP's foi beneficiado com o desenvolvimento do algoritmo de treinamento chamado *backpropagation*, por ser um método computacionalmente eficiente de treiná-los (HAYKIN, 1999). O algoritmo funciona da

seguinte maneira: dada uma saída esperada y_e para uma entrada x, o erro entre a saída esperada e saída real y (que se obtém ao se colocar x como entrada da rede) é calculado. A partir disso, as derivadas parciais do sinal de erro em relação aos pesos são calculadas, onde cada derivada parcial é da forma $\frac{E_o}{\partial \omega_{ij}}$, E_o é o erro da saída y_o e ω_{ij}^l é o peso da sinapse que vai do neurônio i da camada l para o neurônio j da camada $l+1$.

As derivadas parciais do sinal de erro, definido na equação 2.36, são calculadas em relação aos pesos das sinapses da rede. Este valor é multiplicado por um fator (chamado de taxa de aprendizado) e é iterativamente somado aos pesos de forma a diminuir o erro, fazendo com que a rede se aproxime do comportamento esperado.

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \parallel y(x_n, w) - t_n \parallel^2 \tag{2.36}$$

O algoritmo funciona como a seguir: dada uma entrada X e uma saída esperada T, sendo Y a saída da rede para a entrada X, que é obtida utilizando o algoritmo de *feed-forward*, e a_i^j o i-ésimo neurônio da camada j calcula-se as derivadas parciais de $E(w)$ (onde w são os pesos da rede, já que a saída depende deles) em relação aos próprios pesos da rede. A derivada do erro em relação à saída dos neurônios da camada de saída é dada por $\frac{\partial E}{\partial a_i^L} = t_i - y_i$. Para as camadas seguintes, da camada $L-1$ (penúltima) até a primeira camada:

- Para cada neurônio a_i^l na camada l calcula-se a sua contribuição para o erro com $\frac{\partial E}{\partial a_j^l} = \sum_{i=0}^{N(l)} \sigma'(a_i^{l+1}) \frac{\partial E}{\partial a_i^{l+1}} \omega_{ij}^l$, onde ω_{ij}^l é o peso sináptico que conecta o neurônio a_i^j ao neurônio a_j^{l+1}, $N(l)$ é o número de neurônios da camada l e σ' é a derivada da função de ativação (nesse caso σ se refere à função sigmoide, mas pode ser qualquer função derivável).

- Para cada peso sináptico, calcula-se a derivada do erro em relação ao peso usando a regra da cadeia: $\frac{\partial E}{\partial \omega_{ij}^l} = \frac{\partial E}{\partial a_i^{l+1}} \frac{\partial a_i^{l+1}}{\partial \omega_{ij}^l}$

- Atualiza-se os pesos utilizando a equação $\omega_{ij}^l = \omega_{ij}^l + \eta \frac{\partial E}{\partial \omega_{ij}^l}$, onde η é a chamada taxa de aprendizagem, ou *learning rate*. Isso fará a derivada da função de erro tender à zero, o mínimo da função.

No caso de $K > 2$ classes, é possível aplicar uma rede neural com K saídas. Considerando que a saída de cada neurônio da camada de saída representa uma probabilidade, que uma entrada x_n tem uma saída desejada t_n, e que as classes são independentes, têm-se que a distribuição condicional das saídas desejadas t_n é representada pela equação 2.37

(BISHOP, 2006).

$$p(T|x,w) = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}(x,w)^{t_{nk}} \tag{2.37}$$

O erro total, calculado para N entradas $x_1, x_2, ..., x_N$ pode ser calculado pela equação 2.38, que é o logaritmo negativo da função de verosimilhança (essa função também é chamada de entropia cruzada). Na equação $y_nk(w)$ é a k-ésima saída da rede para a entrada x_n, e t_{nk} é a saída esperada no k-ésimo neurônio de saída para a entrada x_n.

$$E(w) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} * ln(y_{nk}(w)) \tag{2.38}$$

Na literatura há a convenção de se utilizar a minimização da função de verossimilhança ao invés da minimização da função de erro, entretanto os dois métodos são equivalentes (BISHOP, 2006). Na prática um otimizador é utilizado para minimizar essa função erro, como o *Adam* (KINGMA; BA, 2014).

2.5.4 Trabalhos anteriores

Em Glette et al. (2008) foram implementados diferentes algoritmos de classificação para identificar 8 movimentos da mão (mão aberta, mão fechada, contração, flexão, desvio ulnar, desvio radial, pronação e supinação). Foram utilizados: *K-Nearest Neighbors*, árvores de decisão, Máquina de Vetores de Suporte (SVM), *Embedded Cartesian Programming Evolvable Hardware* e *Functional Unit Row Evolvable Hardware Architecture*. A tabela 1 mostra as taxas de erro de cada classificador. Como o experimento foi feito em 3 dias, a tabela mostra o resultado de cada dia. No experimento "*2 of 3*" os dois primeiros dias foram usados como treino e o último como teste, enquanto nos outros foi realizada uma validação cruzada *leave-one-out*.

Tabela 1: Resultados de Glette et al. (2008)

	Day 1 [%]	Day 2 [%]	Day 3 [%]	Day 1 - 3 [%]	2 of 3 [%]
k-NN	3.5	4.6	4.6	4.5	5.6
DT	9.7	11.3	10.5	9.0	15.9
SVM	4.2	4.0	2.6	4.5	5.4
EHW1	9.8	4.0	5.3	9.0	10.6
EHW2	9.0	4.6	4.0	4.9	8.4

Em Boschmann et al. (2009) foram usados 4 pares de eletrodos e uma SVM como o algoritmo de classificação de 11 movimentos (extensão de punho, flexão de punho, desvio ulnar, desvio radial, pronação, supinação, abrir a mão, fechar a mão, preensão "chave", *pinch*, e extensão do indicador). A tabela 2 mostra a taxa de acerto para cada quantidade de movimentos utilizada no modelo.

Tabela 2: Resultados de Boschmann et al. (2009).

Número de gestos	11	10	9	8	7	6	5	4-2
Taxa de acerto [%]	91.3	94.5	96.1	97.5	98.1	98.6	99.7	100

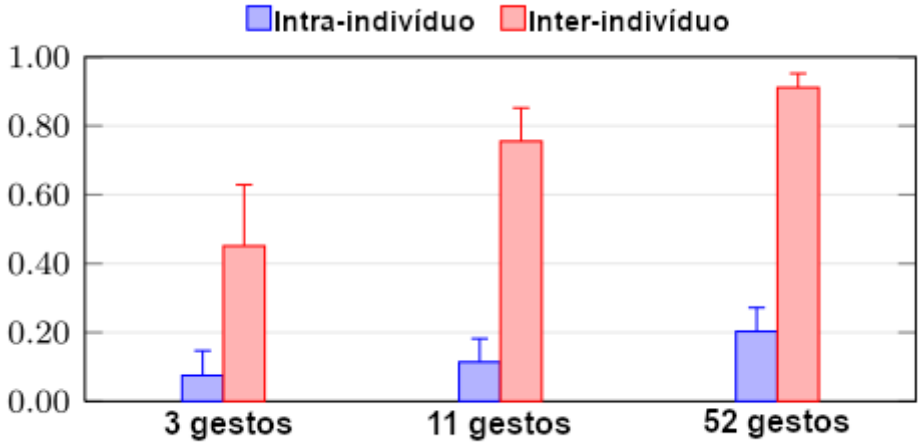
Em Búrigo (2014) utilizou-se a base de dados NinaPro, e três técnicas de aprendizado de máquina foram utilizadas: Regressão Logística, Redes Neurais e SVM. Foram realizados diversos experimentos variando o número de eletrodos, o número de movimentos a serem classificados, entre outras configurações de treinamento. A tabela 3 mostra os seus resultados utilizando configurações de 8, 4 e 2 eletrodos, ao se classificar 6 movimentos.

Tabela 3: Resultados de Búrigo (2014)

	Number of electrodes		
	2	4	8
ANN	77.9%	84.6%	96.3%
SVM	70.3%	76.7%	92.2%
LR	76.9%	84%	96.3%

Em Atzori et al. (2012) foram utilizados dados de 27 indivíduos (a *database* 1 do NinaPro) no total para o treinamento e teste dos modelos, e utilizou-se o algoritmo *Least Squares SVM* com *kernel* RBF. Uma busca de hiperparâmetros foi realizada para escolher o melhor modelo, e o treino e teste foram separados aleatoriamente pelo número da repetição do movimento. O gráfico da figura 10 mostra o erro da acurácia balanceada para diferentes quantidades de movimentos, onde a barra azul representa o erro ao se utilizar os mesmos indivíduos no treino e teste, e a barra vermelha ao se utilizar indivíduos diferentes, mostrando que os dados de um indíviduo não são muito representativos do comportamentos de outros indivíduos.

Figura 10: Resultados de Atzori et al. (2012) (erro da acurácia).

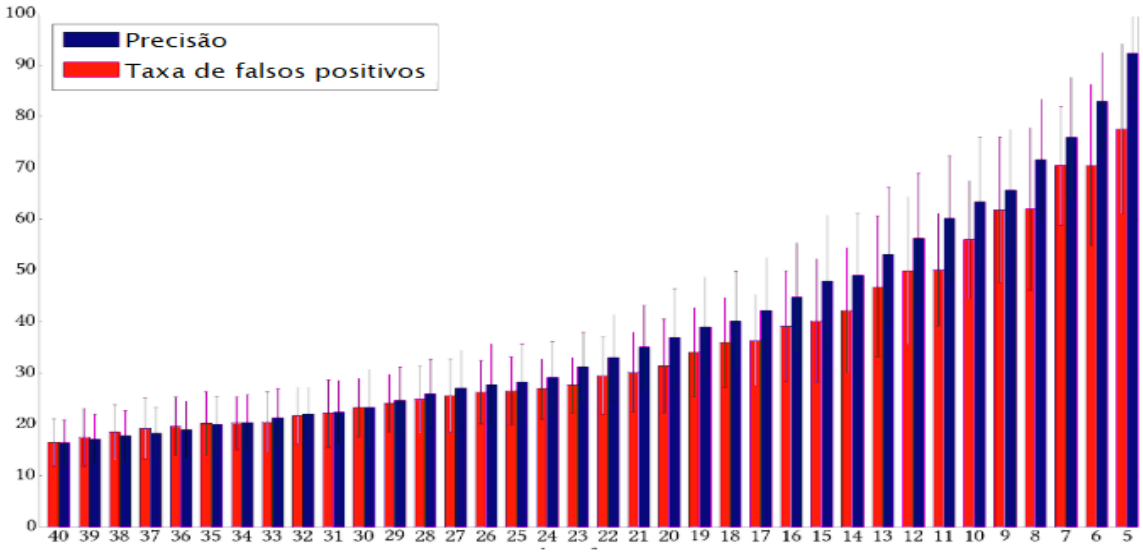


Fonte: Adaptado de Atzori et al. (2012).

Em Atzori et al. (2014) utilizou-se os métodos de *K-Nearest Neighbors*, SVM, *Random Forests* e LDA (*Linear Discriminant Analysis*). Na classificação de 50 movimentos utilizou-se poucas *features* (transformada de Wavelet marginal, valor RMS e histograma) onde obteve-se por volta de 68% de acurácia com o k-NN, 75% com a SVM e pouco mais de 75% com a *Random Forest*. Para o LDA não foi mostrada acurácia ao se treinar com todas as *features* anteriores. Estes resultados foram obtidos utilizando-se a mesma base de dados utilizada neste trabalho.

Em Kerber, Puhl e Krüger (2017) foram treinados modelos de SVM utilizando os sinais eletromiográficos de 14 indivíduos. No total o *dataset* continha 40 movimentos e as *features* utilizados foram: valor RMS, *mean absolute value*, *relação de energia*, histograma, amplitude de Willison, variância, entre outros. Foram treinados diversos modelos com números diferentes de movimentes a serem classificados, começando de um algoritmo que classificasse os 40 movimentos e progressivamente se retirando classes com menor precisão e com maior taxa de falsos positivos até que se sobrasse 5 classes. Na figura 11 estão as acurácias dos modelos em função do número de classes. Foi utilizado a validação cruzada para achar os hiperparâmetros da SVM.

Figura 11: Acurácia do classificador em função do número de movimentos.



Fonte: Adaptado de Kerber, Puhl e Krüger (2017).

Em Viana et al. (2019) foi utilizado uma Rede Neural de 2 camadas tendo como entrada 13 *features* de 2 eletrodos, totalizando 26 *features*. Utilizando 2/3 do *dataset* como treino e 1/3 dele como teste, obteve-se no melhor caso 88.89% de acurácia na classificação de 6 movimentos de 6 indivíduos.

3 METODOLOGIA

Neste capítulo serão mostrados os métodos utilizados (baseando-se na literatura) para se executar cada etapa do projeto, desde o tratamento e pré-processamentos dos sinais e escolha de *features* até as bibliotecas utilizadas para o treinamento dos modelos. Além disso, são abordadas algumas decisões técnicas tomadas durante o desenvolvimento do projeto.

Utilizou-se o Google Colaboratory para realizar a maior parte dos experimentos e processamentos descritos neste trabalho. No Colaboratory se têm a disposição uma máquina virtual na nuvem com recursos virtuais de *hardware*. Utilizou-se a *runtime* com GPU, pois nela era oferecida mais espaço de disco e mais memória, além de a utilização da GPU acelerar o treinamento dos modelos baseados em Redes Neurais.

As especificações estão descritas abaixo:

Sistema de arquivos:

```
Filesystem      Size  Used Avail Use% Mounted on
overlay         359G   58G  283G  17% /
tmpfs            13G     0   13G   0% /dev
tmpfs            13G     0   13G   0% /sys/fs/cgroup
tmpfs            13G   12K   13G   1% /var/colab
/dev/sda1       365G   61G  305G  17% /opt/bin
shm             6.0G   24K  6.0G   1% /dev/shm
tmpfs            13G     0   13G   0% /sys/firmware
drive           359G   90G  269G  26% /content/drive
```

Núcleos de processamento:

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 63
model name      : Intel(R) Xeon(R) CPU @ 2.30GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2300.000
cache size      : 46080 KB
```

```
processor       : 1
vendor_id       : GenuineIntel
cpu family      : 6
model           : 63
model name      : Intel(R) Xeon(R) CPU @ 2.30GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2300.000
cache size      : 46080 KB

processor       : 2
vendor_id       : GenuineIntel
cpu family      : 6
model           : 63
model name      : Intel(R) Xeon(R) CPU @ 2.30GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2300.000
cache size      : 46080 KB

processor       : 3
vendor_id       : GenuineIntel
cpu family      : 6
model           : 63
model name      : Intel(R) Xeon(R) CPU @ 2.30GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2300.000
cache size      : 46080 KB
```

Memória RAM:

```
MemTotal:       26753416 kB
MemFree:          754120 kB
MemAvailable:   22529476 kB
```

Graphical Processing Unit:

```
Model:          Tesla P100-PCIE-16GB
IRQ:            39
GPU UUID:       GPU-11b7941f-220c-db06-bda8-a280b094687b
Video BIOS:     86.00.4d.00.01
```

```
Bus Type:          PCI
DMA Size:          47 bits
DMA Mask:          0x7fffffffffff
Bus Location:      0000:00:04.0
Device Minor:      0
Blacklisted:       No
```

Como pode-se ver, contou-se com cerca de 360Gb de disco rígido, 25Gb de memória RAM, 4 núcleos de processamento, e também é possível verificar que se contou com 1 GPU. Todo o código foi desenvolvido em Python 3.6 devido à existência de bibliotecas como *sklearn* (PEDREGOSA et al., 2011) e *Keras* (CHOLLET et al., 2015), reduzindo muito o tempo de prototipação, experimentação e desenvolvimento dos modelos.

3.1 *Dataset*

O dataset usado neste trabalho foi o DB2 da base de dados Ninapro, que é descrita em ATZORI et al.. A base de dados conta com dados de 40 indivíduos 28 homens, 12 mulheres, 34 destros, 6 canhotos, idade 29.9±3.9 anos, e contém dados de 49 movimentos mais o descanso (os movimentos estão mostradas na figura 12). A atividade muscular foi medida usando ou eletrodos de sEMG duplo diferenciais OttoBock ou Delsys. Na configuração deste dataset (o segundo descrito no artigo) foram utilizados 12 eletrodos Trigno Wireless (Delsys, Inc, www.delsys.com), cada um equipado com baterias recarregáveis, 40m de alcance e ruído de *baseline* de menos de 750nV RMS. Além dos eletrodos havia um acelerômetro tri-axial (ATZORI et al., 2014).

Figura 12: Movimentos da *database* 2 do projeto Ninapro.

Fonte: Adaptado de Atzori et al. (2012).

Os movimentos exercidos pelos voluntários foram escolhidos baseados na literatura de taxonomia, robótica e reabilitação de forma a cobrir a maioria das atividades comuns ao dia-a-dia. Os indivíduos então repetiam os movimentos vistos na tela de um laptop, onde cada movimento durava 5 segundos, sempre seguidos de 3 segundos de descanso para evitar fadiga muscular. Os movimentos não foram aleatorizados (ATZORI et al., 2014).

3.1.1 Divisão

É importante avaliar a performance de um modelo depois que seu treinamento foi completo, seja para o compararmos com outros modelos ou para estimar o seu erro real. Uma maneira de fazer isso para um tarefa de regressão é avaliar o erro quadrático médio (MSE, de equação 3.1), enquanto para uma tarefa de classificação pode-se avaliar o valor da entropia cruzada. No caso do MSE, o valor obtido indica o erro quadrático médio das previsões do modelo em relação aos valores reais.

$$\frac{1}{N}\sum_{i=1}^{n}(y_i - t_i)^2 \tag{3.1}$$

O MSE que o modelo obtém no treino não interessará tanto, pois os parâmetros do modelo foram otimizados de forma à diminuírem este erro. Por exemplo, caso queira-se um modelo que prediz o valor da bolsa de valores pode-se treiná-lo em valores passados dela. Mas pouco importa o quão bem ele predisse valores passados, importa o quão bem ele irá predizer o valor de amanhã. Ou seja, uma métrica melhor do que o erro do modelo no conjunto de treino é o erro de regressão/classificação em um conjunto de teste, com dados que o modelo nunca viu (JAMES et al., 2013).

Uma forma de fazer isso consiste em dividir o conjunto de dados em k grupos diferentes. O modelo é então treinado em $k-1$ grupos, e avaliado no último. Assim a estimativa do MSE da validação cruzada é a média dos MSEs: $CV_k = \frac{1}{k}\sum_{i=1}^{k} MSE_i$.

Entretanto há um *trade-off* entre viés e variância dependendo do valor de k utilizado. Para $k = N$ (caso especial chamado de *Leave-One-Out-CV*) a estimativa terá muito pouco viés, pois ele é treinado em quase todo o *dataset*. Entretanto, como tira-se uma média das saídas de k modelos que são pouco correlacionadas umas com as outras, a variância da métrica será alta, o que prejudica as conclusões sobre a performance do modelo (JAMES et al., 2013).

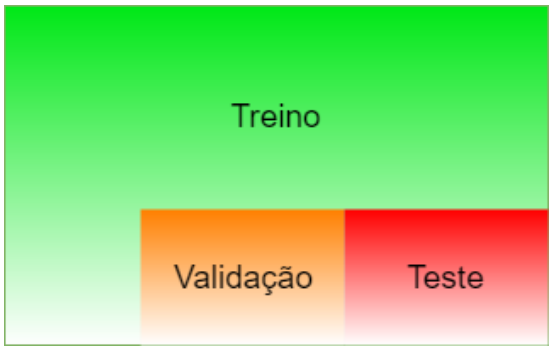
Por exemplo, sabe-se que redes neurais podem “decorar” o conjunto de dados no

qual elas foram treinadas. Caso o treinamento continue por muitas épocas esse efeito tende a ser agravado pois o modelo continua sendo otimizado no conjunto de treino. Neste caso é possível utilizar um outro conjunto, chamado de validação, para dizer quando se deve parar o treinamento, e esse momento é quando o modelo começa a piorar sua performance neste conjunto. Outro exemplo é no *K-Nearest Neighbors*, onde se quer saber qual o K ótimo: criam-se diversos modelos com diferentes valores de K, ele é treinado, verifica-se os resultados para o conjunto de validação, e toma-se como K ótimo o valor que maximizou alguma métrica neste conjunto, como por exemplo a acurácia.

Ainda assim, como validou-se vários modelos neste conjunto de validação, eles ficaram de certa forma enviesados a serem o melhor modelo para esse conjunto. Por isso existe o 3º conjunto, o de teste, e é nele que realmente a performance do modelo é avaliada e onde todas as métricas são calculadas, pois este conjunto além de nunca ter sido "visto" pelo modelo, também não foi utilizado em momento algum para verificar se os parâmetros do modelo estavam corretos, simulando assim um caso real de aplicação do algoritmo.

Neste trabalho serão utilizados os conjuntos de treino, validação e teste. A forma como esses conjuntos são separados dependerá do experimento (como será explicado posteriormente), mas as proporções serão: 70% das janelas no conjunto de treino, 15% no conjunto de validação e 15% no conjunto de teste. É importante que existam indivíduos diferentes nos conjuntos, pois assim pode-se verificar se o modelo não está sendo enviesado por características peculiares de cada pessoa. Como foi visto na seção 2.2, podem haver diferenças no sinal EMG devido ao posicionamento dos eletrodos, características fisiológicas, ruído de movimento. Assim espera-se que o modelo escolhido seja o mais robusto e o que melhor generalizou o treinamento.

Figura 13: Proporções dos conjuntos de treino, validação e teste.



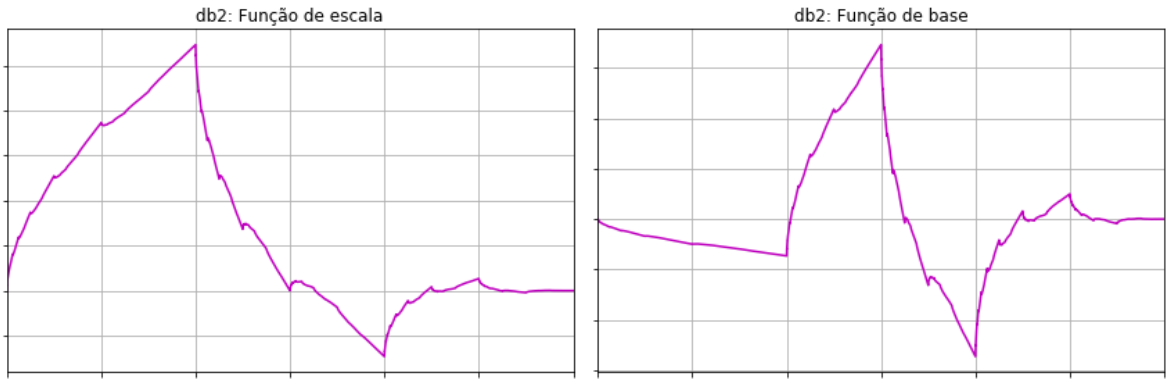
Fonte: Autoria própria.

3.2 Pré-processamento

A filtragem foi feita em *software* utilizando a biblioteca *scipy* disponível para Python. O filtro utilizado é um filtro IIR, da família Butterworth, de ordem 4. O filtro Butterworth foi escolhido por ter uma resposta plana em frequência dentro de sua banda passante. A alta ordem torna a resposta de sinais de frequência fora da banda ainda menores, assegurando que o sinal filtrado conterá apenas frequências interessantes à aplicação. A banda de passagem é entre 5 e 240Hz, garantindo que o sinal conterá as informações importantes sobre o sinal EMG, como visto em na seção 2.2.

O método de *denoising* a ser utilizado é o descrito por Phinyomark, Limsakul e Phukpattaranont (2009) e Phinyomark, Limsakul e Phukpattaranont (2011), como abordado na seção 2. Utilizou-se as decomposições de nível 4, utilizando a *wavelet* mãe *db2* (mostrada na figura 14), *soft-thresholding* e o *threshold* utilizado foi o chamado *universal threshold* por nível, dado pela equação 2.8. As funções de transformação *wavelet* discreta e sua inversa foram utilizadas da biblioteca PyWavelets versão 1.0.3 (LEE et al., 2019).

Figura 14: Função de escala (esquerda) e função base (direita) da wavelet Daubechie2.



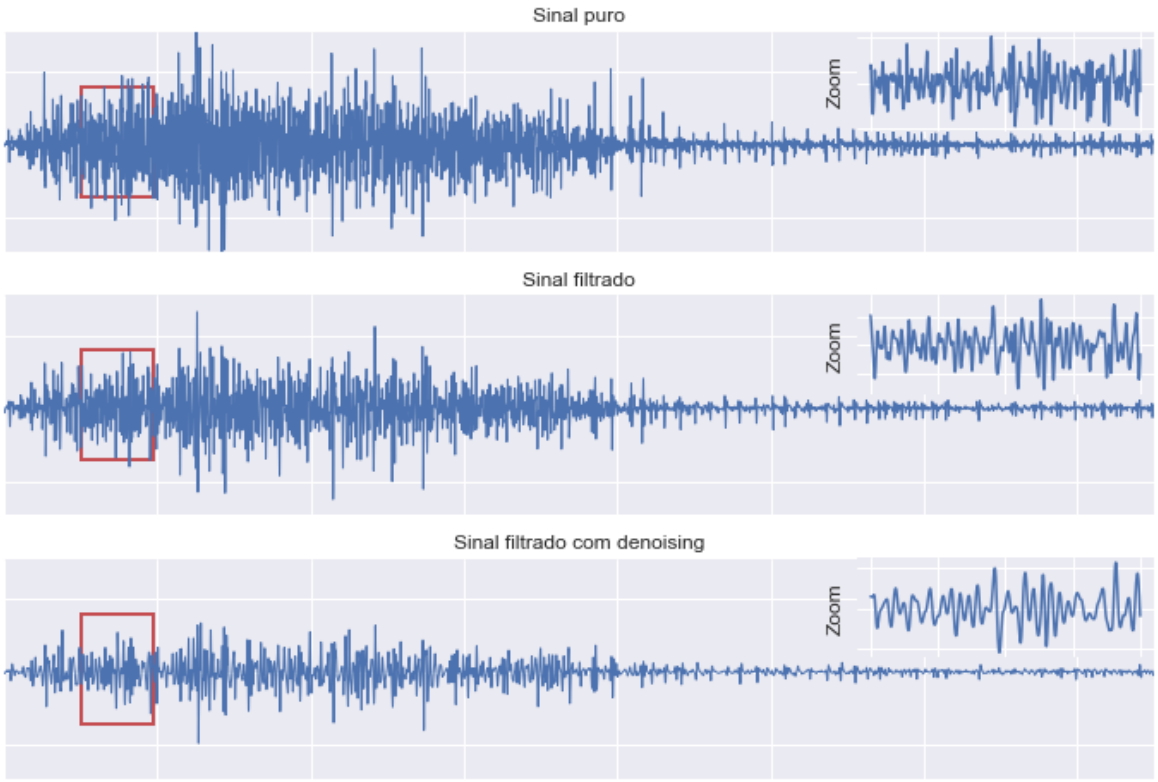
Fonte: Autoria própria.

A figura 15 mostra um exemplo de uma secção de um sinal eletromiográfico puro, um exemplo com apenas a filtragem utilizando o filtro passa-banda e outro exemplo com a filtragem seguida do *denoising*. O retângulo vermelho indica o intervalo do sinal de onde se retirou o *zoom* exibido.

Após a extração do vetor de características de cada amostra temporal, foi realizada a normalização dos dados. Essa normalização foi feita em duas etapas.

Primeiro realizou-se uma normalização com o *Robust Scaler* da biblioteca *sklearn*.

Figura 15: Exemplo de filtragem e *denoising*.



Fonte: Autoria própria.

Esta normalização recebe um vetor de características X e aplica a transformação da equação 3.2, onde IQR é o *inter-quartile range*, que é a diferença entre o 1º e o 3º quartil da *feature*. Note que essa transformação é realizada independentemente para cada atributo.

$$x' = \frac{x - \bar{x}}{IQR} \tag{3.2}$$

Então realiza-se uma transformação nos dados implementada pela classe *PowerTransformer* da biblioteca *sklearn*, que é a chamada transformação de Yeo-Johnson, regida pela equação 3.3, onde x_i é a *feature* i e λ é o parâmetro aprendido na transformação. Essa transformação tem como objetivo mapear os dados o tão bem quanto possível para uma distribuição normal, inclusive tendo média zero e desvio padrão 1 (SKLEARN. . . , Acesso em: 23 de setembro de 2019). Essa normalidade dos dados é desejada para a convergência das redes neurais.

$$x_i^{(\lambda)} = \begin{cases} [(x_i+1)^{\lambda}-1]/\lambda & \text{if } \lambda \neq 0,\ x_i \geq 0, \\ ln(x_i)+1 & \text{if } \lambda = 0,\ x_i \geq 0, \\ -[(x_i+1)^{2-\lambda}-1]/(2-\lambda) & \text{if } \lambda \neq 2,\ x_i 0, \\ -ln(-x_i+1) & \text{if } \lambda = 2,\ x_i < 0 \end{cases} \tag{3.3}$$

A primeira transformação é feita para que a segunda seja executada da forma correta (em experimentos feitos, a transformação de Yeo-Johnson falhava quando havia números muito grandes, indicando uma instabilidade númerica). Portanto a transformação importante é a segunda, mas as duas são feitas em ordem.

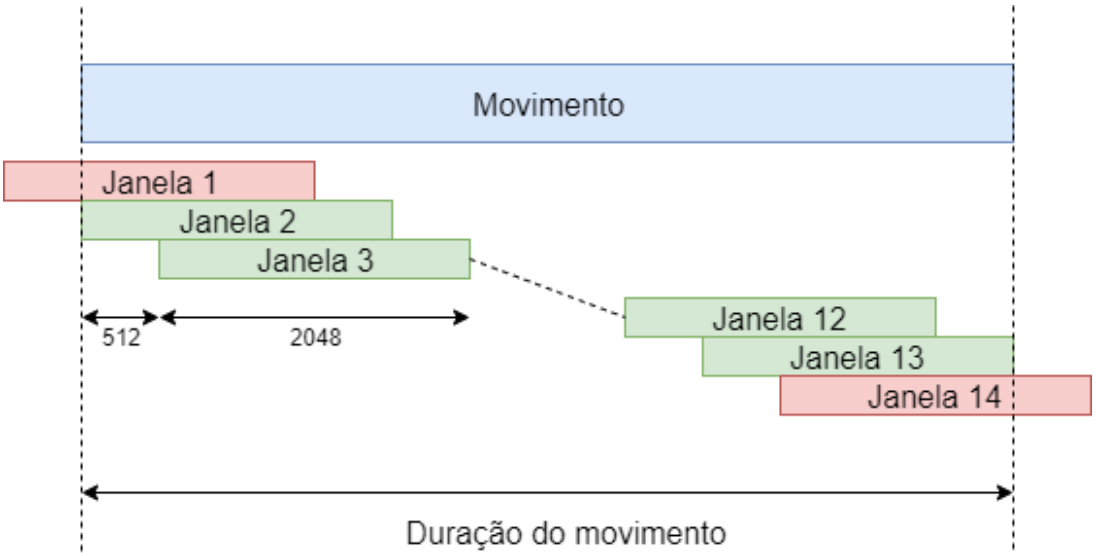
3.3 *Features* e processamento

As *features* no domínio do tempo que foram utilizadas estão listadas na tabela 4. Elas foram calculadas após a realização da filtragem, seguida do *denoising*, e foi feita em janelas de tamanho 2048 (aproximadamente 1 segundo em 2kHz). O janelamento foi feito separando-se janelas de tamanho 2048 e andando 512 amostras para a frente no tempo, e apenas as janelas totalmente dentro do intervalo de movimento foram utilizadas, exatamente como mostra a figura 16 (na figura, as janelas 1 e 14 não são utilizadas por estarem fora do intervalo de duração do movimento). Não foram utilizadas janelas de *resting* (quando não há movimento).

Tabela 4: Características temporais

Curtose	Obliquidade	Desvio padrão
Variância	Média	Média absoluta
Valor RMS	*Zero crossing count*	Amplitude de Willison
Energia	Histograma**	Percentis**
Coeficiente de Pearson*	Energia relativa*	W do teste de Wilcoxon*
Correlação de Spearman*	Energia (Wavelet)**	

Figura 16: Janelamento dos movimentos.



Fonte: Autoria própria.

No domínio da frequência foram extraídas as *features* da tabela 5

Tabela 5: Características espectrais

Centroide espectral	Curtose	Obliquidade
Desvio padrão	Variância	Média
Histograma**	Percentis*	Correlação*

As *features* com asterisco simples são calculadas entre canais, dois a dois, portanto cada uma resulta em $\frac{n_c*(n_c-1)}{2}$ valores, onde n_c é o número de canais sendo utilizados (se n_c for 12, o número de *features* criadas é 66). As que tem asterisco duplo são calculadas por canal, mas resultam em mais de um valor. Por exemplo, foram usados 10 *bins* para o histograma, 10 valores de percentis, e há 8 níveis da transformada wavelet discreta, resultando em 9 valores (A + 8D) de energia por canal (108 no total).

No total, há 1044 valores, de 12 canais. Os experimentos realizados utilizavam todas ou algumas *features*, que foram escolhidas seguindo um critério de correlação com os classes das amostras.

3.4 Aprendizado de máquina

Todos os modelos foram implementados em Python (o código está disponível no *github* <https://github.com/Kotzly/TCC_EMG.git>), e executados no Google Colab, a plataforma em nuvem da Google (COLAB. . . , Acesso em: 9 de setembro de 2019).

3.4.1 k-Nearest Neighbors

O algoritmo k-NN foi implementado em Python, sem utilização de bibliotecas. O algoritmo implementado utiliza um peso para cada ponto vizinho que "vota" para a classe do ponto que está sendo classificado. Este peso é dado pelo inverso da distância do ponto votante para o ponto com classe desconhecida, assim pontos mais próximos pesam mais na inferência da classe.

O algoritmo calcula uma versão modificada da equação 2.34, que toma na verdade a forma da equação 3.4. Nesta equação o voto de cada ponto vizinho de X é ponderado pelo inverso de sua distância até ele. Desta forma pontos mais próximos influenciarão mais na decisão da classe de X. Após o cálculo das probabilidades $p(Y = C_j|X)$ para cada classe C_j, o ponto X é classificado como pertencente à classe de maior probabilidade, ou seja, $argmax(p(Y = C_1|X), p(Y = C_2|X), ..., p(Y = C_p|X))$. Na equação, M é a quantidade de classes.

$$\begin{aligned} p_j &= \sum_{x_i \in \aleph_0} \frac{I(C(x_i) = C_j)}{dist(x, x_i)} \\ p(Y = &C_j|x) = \frac{p_j}{\sum_{i=1}^{M} p_i} \end{aligned} \tag{3.4}$$

3.4.2 Regressão Logística

Utilizou-se a biblioteca Keras para modelagem da regressão logística. Keras oferece API's simples e consistentes, sendo fácil de aprender e usar. Apesar de simples a flexibilidade não é reduzida, pois ele é integrado com bibliotecas de baixo-nível (como o TensorFlow), permitindo a integração de qualquer rotina escrita nesta linguagem (KERAS. . . , Acesso em: 17 de setembro de 2019).

O modelo de Regressão Logística criado teve como base uma arquitetura de rede neural com uma camada e função de ativação "softmax". Em tal arquitetura, no caso de uma entrada X de dimensão M e saída Y de dimensão K, é dado que a entrada do k-ésimo neurônio da saída é $\omega_k^T * X + \omega_{k_0}$. Como se está utilizando a função *softmax*, a k-ésima saída da rede toma exatamente a forma da equação 2.29 que é exatamente o modelo da regressão logística.

Esta rede neural contém, portanto:

- M neurônios de entrada (M será 1044 ou 500, dependendo do experimento).

- Função de ativação “softmax”.
- Regularização L2 (colocada para facilitar a generalização do modelo).
- K neurônios de saída (K é o número de classes que se está classificando).

3.4.3 Rede Neural

Assim como na regressão logística, para a modelagem da rede neural também se utilizou a biblioteca Keras. As redes neurais utilizada nos experimentos são construídas a partir de um padrão, apenas tendo sua entrada e sua saída variando de acordo com as características do experimento. Em experimentos utilizando 500 *features* a camada de entrada tem 500 neurônios, e caso estivesse se classificando 30 movimentos a camada de saída teria 30 neurônios. Fora as camadas de entrada e saída, as camadas deste padrão contavam com:

- 200 neurônios na primeira camada escondida.
- 150 neurônios na segunda camada escondida.
- 100 neurônios na terceira camada escondida (portanto três camadas escondidas).
- Inicialização Xavier dos pesos (GLOROT; BENGIO, 2010).
- Função de ativação *selu* em todas camadas (3.5), exceto a última que utiliza a função *softmax*.

$$selu(x) = \lambda \begin{cases} x, \text{ se } x > 0 \\ \alpha e^x - \alpha, \text{ c.c.} \end{cases} \tag{3.5}$$

Para o treinamento a configuração foi:

- Regularização L2 em todas as camadas, sendo $l = 1e-3$ para os *bias* e $l = 1e-4$ para os pesos.
- *Dropout* de 30% na primeira camada, 20% na segunda e 0 para o resto.
- Norma máxima de 0.5 dos pesos incidentes em cada neurônios (vetores com normas acima disso são re-escalados de volta ao tamanho máximo).
- Otimizador: Adam, com a taxa de aprendizado variando com o experimento (levando em conta o tempo que se levava para treinar, a acurácia obtida, entre outros) e valor máximo da norma do gradiente em 0.2 (evita gradientes muito grandes que afetam a convergência).

- Função de erro: Entropia cruzada multi-classe.
- Critério de parada: erro do conjunto de validação não diminuiu 30 iterações.
- Taxa de aprendizado diminui se ocorrerem 10 iterações sem melhora no erro da validação.

3.5 Métricas

A Matriz de Confusão C é uma tabela que mostra quanto um classificador "confunde" uma classe com as demais. Cada elemento C_{ij} representa o número de observações que pertencem ao grupo i mas foram classificados como pertencentes ao grupo j.

A tabela 6 mostra como a matriz é construída. Para o caso binário: C_{00} é chamado *True positive Rate* (taxa de verdadeiros positivos), C_{01} é chamado *False Negative Rate* (taxa de falsos negativos), C_{10} é chamado *False Positive Rate* (taxa de falsos positivos) e C_{11} é chamado de *True Negative Rate* (taxa de verdadeiros negativos). No caso binário, portanto, é necessário definir uma classe como sendo a classe positiva e a outra será a classe negativa, onde geralmente a classe positiva é a classe de interesse, como por exemplo na predição da existência de células cancerosas em amostras laboratoriais.

Tabela 6: Matriz de confusão

		Predição	
		1	0
Classe	1	TP	FN
	0	FP	TN

A acurácia (em inglês *accuracy*) é calculada utilizando a equação 3.6. É um método para se verificar o quão certas as predições de um modelo são. Quando a quantidade de amostras de cada uma das classes estão desbalanceadas a métrica mais interessante de se utilizar é a acurácia balanceada, que leva em conta a quantidade de exemplos pertencentes a cada classe. Na equação 3.7 m é o número de classes e A_i é a acurácia da classe i.

$$accuraccy = \frac{TP + TN}{N} \tag{3.6}$$

$$balanced\_accuracy = \frac{1}{m}\sum_{i=0}^{m} A_i \tag{3.7}$$

Caso a acurácia balanceada seja normalizada para o intervalo $[\frac{1}{1-m}, 1]$ obtém-se o chamada J de *Youden*, ou *informedness*, no qual uma predição aleatória terá uma pontuação de 0.
A Precisão (em inglês *precision*) e o *Recall* são métricas utilizadas para saber a qualidade das predições de um classificador. O *recall* informa a acurácia do classificador para a classe positiva, enquanto a precisão informa o quanto das predições de classe positiva realmente estavam certas.

Por exemplo, caso existam 50 exemplos da classe negativa e 40 foram preditas como negativa, 40 exemplos da classe positiva e 20 foram preditas como positiva, a precisão será de $\frac{30}{30+10} = 75\%$ e o *recall* será de $\frac{30}{30+20} = 60\%$. A métrica *F1 Score* é a média harmônica entre o *recall* e a precisão, e é importante pois geralmente deve-se verificar as duas métricas em conjunto, já que no caso de um classificador que diz que todos os exemplos são da classe positiva o seu *recall* é de 1, em depreciação da precisão.

Os cálculos da precisão, do *recall* e da métrica F1 estão nas equações 3.8, 3.9 e 3.10, respectivamente.

$$precision = \frac{TP}{TP + FP} \tag{3.8}$$

$$recall = \frac{TP}{TP + FN} \tag{3.9}$$

$$f1 = 2 * \frac{precision * recall}{precision + recall} \tag{3.10}$$

O coeficiente de Cohen Kappa κ é uma medida de concordância de dois avaliadores categóricos. Ou seja, se houverem dois classificadores dizendo se um conjunto de figuras representam cachorros ou gatos, utiliza-se o κ para se ter uma medida quantitativa de quanto os dois classificadores concordaram entre si. No contexto deste trabalho um dos classificadores é o modelo de *Machine Learning*, e o outro é a classe real do dado, assim o κ será uma medida de concordância entre o classificador e a realidade.

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{3.11}$$

A equação do $Kappa$ é mostrada na equação 3.11, onde p_0 é a acurácia do classificador e p_e é a chance hipotética de concordância aleatória entre os dois classificadores.

3.6 Experimentos

Foram realizados 4 grandes conjuntos de experimentos neste trabalho, que seguem pelos prefixos:

- **INTER**: Neste conjunto de experimentos os *datasets* foram separados por indivíduo, assim como foi discutido em 3.1. Desta maneira o modelo é treinado com dados de um conjunto de indivíduos, e sua performance é avaliada em conjuntos com indivíduos diferentes. Esta é a maneira correta de divisão dos *datasets*, já que desta forma é possível tirar conclusões sobre como o modelo aprende com dados de determinado conjunto de pessoas e como ele generaliza para outro conjunto de indivíduos.
- **INTER_500**: Igual ao conjunto INTER, mas apenas 500 *features* foram utilizadas.
- **INTRA**: Neste conjunto de experimentos os *datasets* são separados por amostras de maneira aleatória: depois de feito o pré-processamento, o janelamento e a extração de *features* estas amostras são divididas aleatoriamente de forma a haver 70% dos dados para treinamento, 15% para validação e 15% para teste. Este não é o melhor protocolo de divisão de *dataset* para aplicações biomédicas, entretanto estes experimentos foram realizados para mostrar a grande diferença de performance do modelo nesta condição, e para se haver uma base de comparação com a literatura (muitos autores usam este protocolo).
- **BACKWARD**: Neste conjunto o treinamento começa criando modelos que classificam 49 classes, e gradativamente classes são retiradas até que sobrem cinco. O protocolo é o seguinte: os modelos de regressão logística e rede neural são criados e treinados, classificando 49 gestos. Após isso, os 4 gestos que tiveram a pior acurácia de classificação (pela rede neural) no conjunto de validação são retirados. A seguir são criados e treinados dois modelos (de regressão logística e rede neural) que classificam os 45 gestos restantes. Isto se repete até sobrarem 5 movimentos.

A não ser pelo experimento BACKWARD, os outros conjuntos começam na primeira rodada criando modelos que classificam 49 gestos. Após isto treinou-se modelos que classificam 45 gestos, e depois 41, até 5 gestos, depois 4, 3 e finalmente 2. Isto foi feito pois como alguns treinamentos poderiam durar muitas horas foi necessário diminuir o número de modelos treinados.

Nos experimentos INTER, INTER_500 e INTRA os gestos foram escolhidos a cada rodada baseando-se na suas correlações lineares com as *features* utilizadas: A correlação linear de cada classe em relação à cada *feature* foi calculada, e após isso calcula-se o valor absoluto destas correlações já que apenas o módulo destes valores importa (correlações

fortemente negativas ou fortemente positivas são interessantes). Depois a média destas correlações em relação à cada *feature* é calculada, resultando em um valor para cada classe. Esse valor é a média do valor absoluto da correlação desta classe com todas as *features*.

Assim, na rodada de classificação de Q gestos eram utilizados as Q classes que tinham este valor maior. Este procedimento é realizado para aumentar as chances de que nas rodadas com poucas classes os modelos teriam facilidade de convergir e aprender, já que as classes que eles estão aprendendo são de fato correlacionadas com as entradas dos modelos. Entretanto ele pode falhar caso estas classes tenham grande confusão entre si (caso a distribuição de *features* de cada classe seja parecida e os modelos confundam muito uma classe pela outra).

Como há 3 algoritmos sendo utilizados, serão utilizados sufixos para se referir à eles. Será utilizado **NEURAL** para as redes neurais, **LOGISTIC** ou **LOGISTIC_REGRESSION** para a regressão logística e **KNN** para *k-Nearest Neighbors*. Para facilitar a referência a cada modelo e cada protocolo eles serão referidos pelo nome do algoritmo seguido pelo nome do experimento. Por exemplo, "NEURAL_INTER" se refere aos modelos baseados em redes neurais que utilizaram o protocolo INTER de separação dos conjuntos de treino, validação e teste. A tabela 7 e 8 mostra todos os conjuntos de experimentos feitos.

Tabela 7: Experimentos realizados (parte 1).

	INTER	INTER_500
Reg. logística	LOGISTIC_INTER	LOGISTIC_INTER_500
Rede Neural	NEURAL_INTER	NEURAL_INTER_500
k-NN	KNN_INTER	KNN_INTER_500

Tabela 8: Experimentos realizados (parte 2).

	INTRA	BACKWARD
Reg. logística	LOGISTIC_INTRA	LOGISTIC_BACKWARD
Rede Neural	NEURAL_INTRA	NEURAL_BACKWARD
k-NN	KNN_INTRA	———

4 RESULTADOS

Neste capítulo são expostos os resultados dos experimentos que foram propostos para o trabalho, com os resultados de performance dos modelos e dos protocolos de treinamento abordados.

Para cada conjunto de experimentos INTER, INTER_500, INTRA e BACKWARD os três algoritmos utilizados serão comparados utilizando um gráfico de barras. Como as métricas foram calculadas por indivíduo, há a barra que indica a média da métrica em questão, e a linha preta na extremidade superior da barra que indica o desvio padrão da métrica. É importante lembrar que estas métricas foram calculadas utilizando o conjunto de teste. As tabelas mostram as métrica multiplicadas por um fator de 100.

4.1 INTER

A tabela 9 mostra todas as métricas para o modelo de regressão logística, para todas as quantidades de classes, a tabela 10 mostra estes resultados obtidos com o modelo de *k-Nearest Neighbors*, e a tabela 11 contém os resultados da rede neural. A figura 17 mostra a acurácia balanceada para cada modelo com cada número de movimentos sendo classificados.

Tabela 9: Resultados de LOGISTIC_INTER.

N	Balanced Accuracy	F1 Score	Cohen Kappa	AUC	Precision	Recall
2	98.1 ± 2.2	98.2 ± 1.9	96.4 ± 3.9	99.9 ± 0.2	98.4 ± 1.7	98.3 ± 1.9
3	92.7 ± 6.6	92.8 ± 5.7	89.0 ± 8.3	98.9 ± 1.3	93.7 ± 4.7	93.0 ± 5.3
4	80.9 ± 6.1	75.7 ± 10.8	69.7 ± 12.1	93.5 ± 3.0	80.2 ± 7.7	77.5 ± 9.3
5	67.3 ± 6.1	60.8 ± 9.4	53.8 ± 9.5	88.9 ± 3.9	67.1 ± 6.7	63.1 ± 8.1
9	57.9 ± 6.4	54.3 ± 6.6	49.4 ± 6.0	91.7 ± 1.5	63.0 ± 7.2	55.1 ± 5.3
13	51.3 ± 7.4	48.7 ± 6.0	45.5 ± 6.0	91.8 ± 1.9	56.4 ± 8.0	49.8 ± 5.4
17	46.3 ± 5.3	42.4 ± 3.7	40.5 ± 4.1	91.0 ± 2.0	51.5 ± 4.8	44.1 ± 3.7
21	40.4 ± 4.6	35.5 ± 3.4	33.7 ± 3.8	89.1 ± 1.6	44.8 ± 3.3	37.0 ± 3.5
25	37.5 ± 3.6	32.4 ± 2.3	31.2 ± 2.7	89.5 ± 2.2	41.5 ± 2.8	34.0 ± 2.5
29	35.2 ± 3.2	30.1 ± 2.3	29.7 ± 2.5	89.1 ± 1.9	38.1 ± 1.8	32.2 ± 2.3
33	32.5 ± 4.1	27.6 ± 3.3	27.6 ± 3.5	88.3 ± 1.5	35.3 ± 3.4	29.8 ± 3.4
37	29.8 ± 4.2	25.3 ± 2.7	25.4 ± 3.6	87.8 ± 1.7	32.1 ± 2.5	27.4 ± 3.5
41	28.9 ± 5.6	24.2 ± 3.3	24.3 ± 4.7	87.6 ± 2.1	31.0 ± 3.1	26.2 ± 4.5
45	25.9 ± 4.2	21.0 ± 2.6	21.5 ± 3.6	86.3 ± 2.3	27.0 ± 2.9	23.3 ± 3.5
49	24.6 ± 4.4	19.8 ± 2.4	20.3 ± 3.4	85.8 ± 2.2	26.4 ± 3.2	21.9 ± 3.3

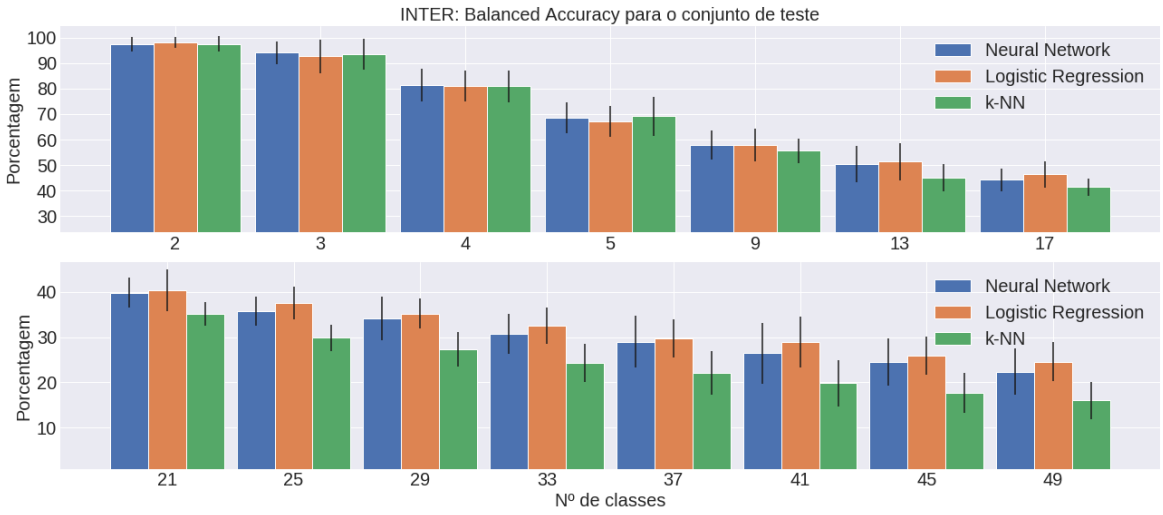
Tabela 10: Resultados de KNN_INTER.

N	Balanced Accuracy	F1 Score	Cohen Kappa	AUC	Precision	Recall
2	97.5 ± 3.0	97.6 ± 3.0	95.0 ± 6.1	97.5 ± 3.0	97.7 ± 2.8	97.6 ± 3.0
3	93.5 ± 6.0	93.9 ± 5.7	90.6 ± 8.8	94.9 ± 4.7	94.9 ± 4.6	93.9 ± 5.7
4	80.9 ± 6.1	77.4 ± 8.6	70.5 ± 10.3	84.7 ± 5.3	82.3 ± 6.6	78.2 ± 7.7
5	69.1 ± 7.7	64.9 ± 10.0	56.7 ± 11.4	77.9 ± 5.8	69.2 ± 9.2	65.7 ± 9.3
9	55.6 ± 4.7	52.8 ± 6.2	48.5 ± 5.6	74.2 ± 2.9	63.4 ± 6.3	54.5 ± 4.9
13	45.1 ± 5.2	42.2 ± 6.3	40.1 ± 5.8	70.1 ± 3.0	48.9 ± 4.0	45.0 ± 5.3
17	41.3 ± 3.5	38.0 ± 3.9	36.2 ± 3.6	68.1 ± 1.9	45.7 ± 3.3	40.2 ± 3.3
21	35.1 ± 2.6	31.3 ± 2.3	29.8 ± 2.5	64.9 ± 1.3	39.1 ± 2.4	33.4 ± 2.3
25	29.8 ± 2.9	26.3 ± 2.3	25.1 ± 2.1	62.5 ± 1.1	33.6 ± 3.1	28.2 ± 2.0
29	27.3 ± 3.8	22.9 ± 2.7	22.6 ± 2.9	61.3 ± 1.4	29.9 ± 5.1	25.4 ± 2.7
33	24.3 ± 4.2	20.8 ± 3.1	20.4 ± 3.6	60.2 ± 1.8	27.1 ± 4.3	23.0 ± 3.5
37	22.0 ± 4.9	18.6 ± 3.9	18.4 ± 4.5	59.2 ± 2.2	25.5 ± 5.4	20.8 ± 4.3
41	19.8 ± 5.1	16.7 ± 4.2	16.5 ± 4.7	58.2 ± 2.3	23.4 ± 5.9	18.7 ± 4.5
45	17.7 ± 4.4	14.7 ± 4.1	14.7 ± 4.3	57.3 ± 2.1	20.7 ± 6.3	16.7 ± 4.1
49	16.0 ± 4.1	13.1 ± 3.5	13.1 ± 3.8	56.6 ± 1.9	18.6 ± 5.2	15.0 ± 3.7

Tabela 11: Resultados de NEURAL_INTER.

N	Balanced Accuracy	F1 Score	Cohen Kappa	AUC	Precision	Recall
2	97.5 ± 2.8	97.7 ± 2.4	95.3 ± 5.0	99.9 ± 0.2	97.9 ± 2.2	97.7 ± 2.4
3	94.1 ± 4.4	94.1 ± 3.8	91.0 ± 5.8	98.9 ± 1.2	94.7 ± 3.4	94.2 ± 3.7
4	81.3 ± 6.3	74.8 ± 12.5	69.9 ± 12.5	92.8 ± 3.4	83.4 ± 6.7	77.5 ± 9.7
5	68.5 ± 6.1	61.5 ± 10.1	54.9 ± 9.7	89.3 ± 3.4	67.7 ± 7.0	64.0 ± 8.2
9	57.9 ± 5.6	53.8 ± 5.7	49.5 ± 5.3	92.1 ± 1.2	62.9 ± 6.8	55.2 ± 4.6
13	50.3 ± 7.1	46.5 ± 5.5	44.4 ± 6.4	92.3 ± 2.1	53.6 ± 5.7	48.8 ± 5.8
17	44.2 ± 4.5	38.9 ± 3.0	37.6 ± 3.4	91.3 ± 2.2	48.3 ± 4.2	41.4 ± 3.1
21	39.8 ± 3.4	34.0 ± 2.4	32.9 ± 2.2	89.4 ± 2.1	43.9 ± 4.8	36.2 ± 2.0
25	35.7 ± 3.1	30.1 ± 1.2	29.8 ± 1.7	89.2 ± 2.2	38.8 ± 2.6	32.7 ± 1.5
29	34.2 ± 4.8	28.5 ± 2.3	28.6 ± 3.5	89.0 ± 1.8	37.4 ± 4.1	31.1 ± 3.3
33	30.8 ± 4.3	25.6 ± 2.5	26.0 ± 3.6	88.6 ± 1.7	33.5 ± 4.2	28.3 ± 3.4
37	29.0 ± 5.7	24.2 ± 4.1	24.6 ± 5.3	88.7 ± 2.1	31.4 ± 4.4	26.7 ± 5.1
41	26.4 ± 6.7	21.8 ± 4.4	22.4 ± 5.6	88.0 ± 2.3	29.5 ± 4.9	24.4 ± 5.4
45	24.5 ± 5.2	20.0 ± 3.8	20.7 ± 4.7	87.3 ± 2.3	26.8 ± 3.6	22.5 ± 4.6
49	22.3 ± 5.1	17.6 ± 3.7	18.4 ± 4.6	86.6 ± 2.2	24.7 ± 3.2	20.1 ± 4.5

Figura 17: Comparação da métrica *balanced accuracy* no conjunto de teste para INTER.



Fonte: Autoria própria.

4.2 INTER_500

A tabela 12 mostra todas as métricas para o modelo de regressão logística, para todas as quantidades de classes, a tabela 13 mostra estes resultados obtidos com o modelo de *k-Nearest Neighbors*, e a tabela 14 contém os resultados da rede neural. A figura 18 mostra a acurácia balanceada para cada modelo com cada número de movimentos sendo classificados.

Tabela 12: Resultados de LOGISTIC_INTER_500.

N	Balanced Accuracy	F1 Score	Cohen Kappa	AUC	Precision	Recall
2	98.2 ± 2.2	98.4 ± 2.0	96.7 ± 4.0	99.9 ± 0.2	98.5 ± 1.8	98.4 ± 1.9
3	95.1 ± 4.2	95.1 ± 3.8	92.5 ± 5.9	98.7 ± 1.7	95.5 ± 3.4	95.1 ± 3.8
4	82.2 ± 5.0	76.6 ± 10.9	71.2 ± 10.9	93.7 ± 2.2	82.7 ± 5.1	78.5 ± 8.5
5	68.0 ± 5.5	61.3 ± 9.5	54.0 ± 9.0	90.1 ± 2.3	68.8 ± 5.6	63.3 ± 7.6
9	56.3 ± 5.7	52.8 ± 6.4	48.0 ± 6.0	91.8 ± 1.4	62.5 ± 6.6	53.9 ± 5.2
13	48.4 ± 6.5	44.7 ± 5.7	42.4 ± 5.9	91.4 ± 2.5	52.3 ± 7.2	46.9 ± 5.3
17	44.5 ± 6.9	39.1 ± 5.0	37.7 ± 5.9	90.9 ± 2.9	48.0 ± 5.2	41.5 ± 5.4
21	39.0 ± 6.7	32.6 ± 4.3	31.5 ± 5.2	89.0 ± 2.7	42.6 ± 4.2	34.9 ± 4.8
25	35.3 ± 5.8	29.7 ± 4.0	28.9 ± 4.5	88.8 ± 3.0	37.7 ± 4.2	31.8 ± 4.2
29	34.0 ± 6.1	27.9 ± 3.9	28.0 ± 4.4	88.4 ± 2.6	36.1 ± 4.3	30.5 ± 4.2
33	31.2 ± 6.3	25.4 ± 4.5	25.9 ± 5.1	87.6 ± 2.4	32.6 ± 4.0	28.2 ± 4.9
37	28.4 ± 6.1	23.1 ± 4.3	23.7 ± 5.3	87.1 ± 2.5	30.7 ± 4.2	25.8 ± 5.1
41	26.3 ± 6.3	21.4 ± 4.2	21.7 ± 5.3	87.0 ± 2.6	28.8 ± 4.3	23.7 ± 5.1
45	24.1 ± 6.0	19.0 ± 4.1	19.7 ± 5.0	86.0 ± 2.7	25.3 ± 4.2	21.5 ± 4.8
49	22.1 ± 5.8	17.3 ± 3.9	17.8 ± 4.7	85.2 ± 2.5	23.9 ± 3.5	19.6 ± 4.6

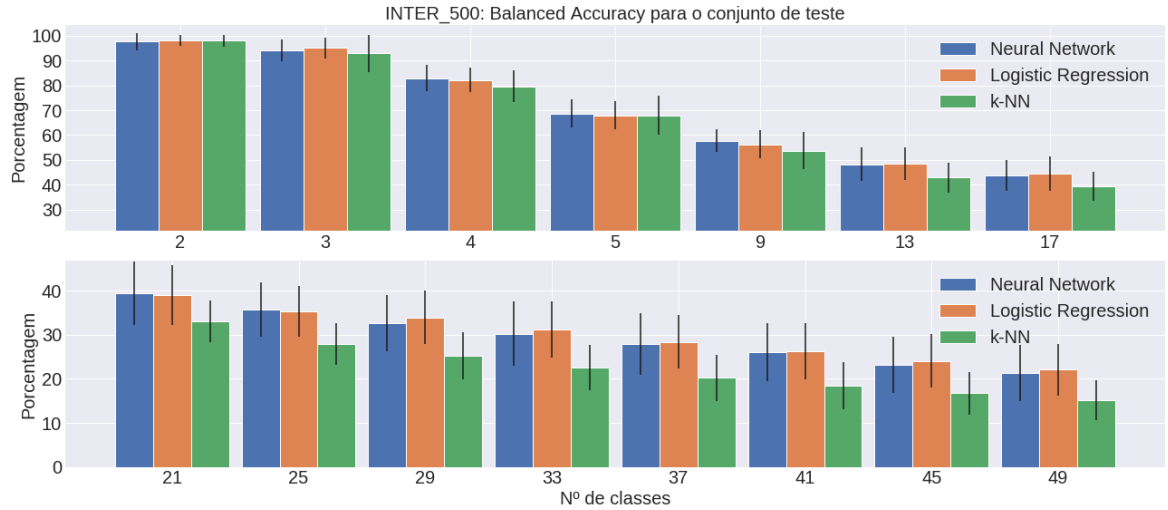
Tabela 13: Resultados de KNN_INTER_500.

N	Balanced Accuracy	F1 Score	Cohen Kappa	AUC	Precision	Recall
2	98.0 ± 2.5	98.0 ± 2.5	95.9 ± 5.1	98.0 ± 2.5	98.1 ± 2.4	98.0 ± 2.5
3	92.8 ± 7.4	93.1 ± 7.1	89.3 ± 10.9	94.3 ± 5.7	94.6 ± 5.1	93.1 ± 7.1
4	79.6 ± 6.3	75.2 ± 9.4	68.2 ± 10.4	83.6 ± 5.3	82.9 ± 5.3	76.4 ± 8.0
5	67.9 ± 7.9	63.1 ± 10.1	55.0 ± 11.0	77.1 ± 5.5	69.3 ± 7.7	64.2 ± 9.3
9	53.7 ± 7.4	50.0 ± 9.6	46.1 ± 9.0	73.1 ± 4.6	59.9 ± 8.7	52.4 ± 8.0
13	43.0 ± 6.0	39.7 ± 7.3	37.6 ± 7.1	68.8 ± 3.6	47.0 ± 4.6	42.6 ± 6.6
17	39.4 ± 5.8	36.0 ± 5.9	34.1 ± 5.8	67.0 ± 2.9	43.0 ± 5.9	38.1 ± 5.4
21	33.1 ± 4.8	29.2 ± 4.2	27.6 ± 4.5	63.8 ± 2.3	36.0 ± 3.5	31.3 ± 4.2
25	27.9 ± 4.7	24.1 ± 4.1	23.0 ± 4.1	61.5 ± 2.1	30.8 ± 5.0	26.2 ± 3.9
29	25.3 ± 5.4	21.3 ± 3.8	20.5 ± 4.2	60.3 ± 2.1	27.3 ± 5.7	23.4 ± 4.0
33	22.6 ± 5.1	19.1 ± 3.5	18.4 ± 4.0	59.2 ± 2.0	24.4 ± 5.3	21.0 ± 3.9
37	20.3 ± 5.2	16.9 ± 3.7	16.3 ± 4.2	58.2 ± 2.1	22.5 ± 5.2	18.7 ± 4.1
41	18.4 ± 5.3	15.3 ± 3.7	14.8 ± 4.3	57.4 ± 2.2	20.1 ± 4.4	17.0 ± 4.2
45	16.7 ± 4.9	13.5 ± 3.3	13.2 ± 3.9	56.6 ± 1.9	18.0 ± 4.3	15.2 ± 3.8
49	15.1 ± 4.5	12.1 ± 2.9	11.9 ± 3.6	55.9 ± 1.8	16.2 ± 3.7	13.8 ± 3.5

Tabela 14: Resultados de NEURAL_INTER_500.

N	Balanced Accuracy	F1 Score	Cohen Kappa	AUC	Precision	Recall
2	97.5 ± 3.5	97.8 ± 3.0	95.5 ± 6.2	99.8 ± 0.3	98.0 ± 2.7	97.9 ± 2.9
3	94.0 ± 4.3	94.0 ± 3.9	90.6 ± 6.0	98.8 ± 1.2	94.4 ± 3.6	94.0 ± 3.9
4	82.8 ± 5.3	77.1 ± 11.4	72.2 ± 11.3	94.6 ± 2.3	84.5 ± 5.4	79.2 ± 8.8
5	68.7 ± 5.7	62.0 ± 9.8	55.0 ± 9.4	90.3 ± 2.4	69.3 ± 5.1	64.1 ± 7.9
9	57.8 ± 4.7	53.0 ± 6.0	49.4 ± 5.3	92.1 ± 0.6	61.2 ± 6.5	55.1 ± 4.6
13	48.3 ± 6.8	44.2 ± 5.8	42.6 ± 6.3	91.4 ± 2.1	49.9 ± 6.7	47.2 ± 5.7
17	43.8 ± 6.2	38.1 ± 4.7	37.2 ± 5.6	91.0 ± 2.4	47.9 ± 6.2	41.0 ± 5.1
21	39.4 ± 7.1	33.3 ± 4.5	32.2 ± 5.7	89.0 ± 2.4	42.7 ± 6.2	35.5 ± 5.3
25	35.6 ± 6.2	29.4 ± 4.7	29.3 ± 5.0	89.0 ± 2.4	38.8 ± 6.8	32.2 ± 4.7
29	32.6 ± 6.5	26.1 ± 3.8	26.8 ± 4.6	88.5 ± 2.1	35.2 ± 4.9	29.4 ± 4.4
33	30.2 ± 7.3	24.5 ± 4.9	25.1 ± 5.9	87.9 ± 2.1	32.7 ± 3.9	27.4 ± 5.7
37	27.8 ± 7.0	22.2 ± 5.0	23.2 ± 6.2	87.6 ± 2.3	30.0 ± 4.5	25.3 ± 6.0
41	26.0 ± 6.5	20.8 ± 4.6	21.8 ± 5.6	87.4 ± 2.3	27.8 ± 4.2	23.7 ± 5.4
45	23.2 ± 6.4	18.1 ± 4.5	19.2 ± 5.6	86.6 ± 2.5	25.3 ± 3.0	21.0 ± 5.4
49	21.3 ± 6.4	16.5 ± 4.8	17.4 ± 5.7	86.1 ± 2.4	24.1 ± 4.6	19.1 ± 5.5

Figura 18: Comparação da métrica *balanced accuracy* no conjunto de teste para INTER_500.



Fonte: Autoria própria.

4.3 INTRA

A tabela 15 mostra todas as métricas para o modelo de regressão logística, para todas as quantidades de classes, a tabela 16 mostra estes resultados obtidos com o modelo de *k-Nearest Neighbors*, e a tabela 17 contém os resultados da rede neural. A figura 19 mostra a acurácia balanceada para cada modelo com cada número de movimentos sendo classificados.

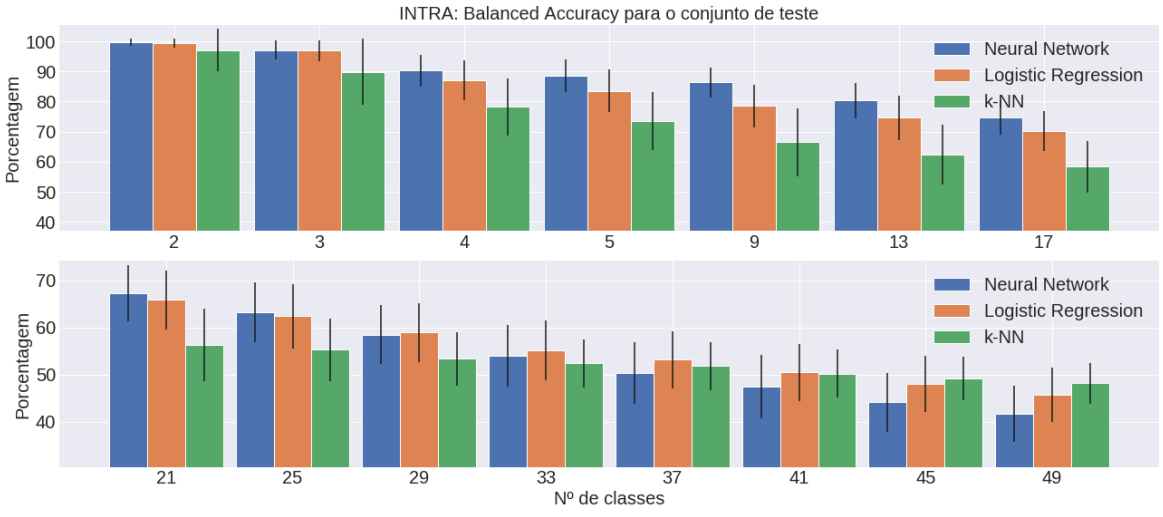
Tabela 15: Resultados de LOGISTIC_INTRA.

N	Balanced Accuracy	F1 Score	Cohen Kappa	AUC	Precision	Recall
2	99.5 ± 1.5	99.5 ± 1.6	99.0 ± 3.3	100.0 ± 0.0	99.6 ± 1.4	99.5 ± 1.7
3	96.9 ± 3.5	96.7 ± 3.8	94.6 ± 6.4	99.5 ± 1.5	97.2 ± 3.2	96.6 ± 3.9
4	87.0 ± 6.6	84.7 ± 8.6	79.2 ± 11.7	96.0 ± 3.5	86.7 ± 7.1	84.9 ± 8.2
5	83.6 ± 7.0	81.2 ± 8.3	76.4 ± 10.4	95.9 ± 2.7	83.5 ± 7.4	81.5 ± 8.1
9	78.5 ± 7.0	76.5 ± 7.6	73.6 ± 8.4	97.1 ± 1.5	79.2 ± 6.8	76.8 ± 7.4
13	74.6 ± 7.4	72.4 ± 7.2	70.2 ± 7.9	94.5 ± 15.2	75.1 ± 6.5	72.8 ± 7.2
17	70.1 ± 6.6	67.6 ± 6.4	65.6 ± 6.9	91.8 ± 21.1	71.1 ± 5.7	67.9 ± 6.4
21	65.9 ± 6.2	62.2 ± 6.6	60.5 ± 6.8	91.0 ± 20.9	65.7 ± 6.0	62.6 ± 6.5
25	62.4 ± 6.8	59.0 ± 7.0	57.6 ± 7.1	90.9 ± 20.9	63.1 ± 6.5	59.5 ± 6.8
29	58.9 ± 6.2	55.7 ± 6.5	54.6 ± 6.5	90.8 ± 20.9	59.7 ± 6.2	56.4 ± 6.3
33	55.2 ± 6.3	52.0 ± 6.8	51.1 ± 6.8	90.5 ± 20.8	56.0 ± 6.8	52.7 ± 6.6
37	53.2 ± 6.0	50.0 ± 6.5	49.2 ± 6.6	90.2 ± 20.8	54.1 ± 6.5	50.7 ± 6.4
41	50.5 ± 6.1	47.3 ± 6.5	46.7 ± 6.5	89.8 ± 20.7	51.4 ± 6.7	48.1 ± 6.4
45	48.1 ± 5.9	45.1 ± 6.5	44.7 ± 6.5	89.6 ± 20.6	49.2 ± 6.6	46.1 ± 6.3
49	45.7 ± 5.9	43.0 ± 6.4	42.5 ± 6.4	89.3 ± 20.6	47.1 ± 6.5	43.8 ± 6.3

Tabela 16: Resultados de KNN_INTRA.

N	Balanced Accuracy	F1 Score	Cohen Kappa	AUC	Precision	Recall
2	97.1 ± 7.0	97.2 ± 6.0	93.9 ± 13.9	97.1 ± 7.0	97.5 ± 5.8	97.3 ± 5.6
3	89.8 ± 10.9	90.0 ± 11.2	84.4 ± 17.2	92.0 ± 8.5	92.8 ± 8.6	90.2 ± 10.6
4	78.2 ± 9.4	74.5 ± 12.4	67.7 ± 14.6	83.3 ± 7.5	80.8 ± 12.1	76.6 ± 10.5
5	73.5 ± 9.7	69.7 ± 12.4	64.0 ± 14.3	81.8 ± 7.3	76.8 ± 11.4	71.8 ± 11.2
9	66.4 ± 11.3	62.8 ± 12.5	60.6 ± 12.6	80.3 ± 6.3	70.3 ± 12.9	65.4 ± 11.0
13	62.5 ± 10.0	60.4 ± 11.0	58.6 ± 10.8	77.5 ± 13.5	68.2 ± 10.8	62.3 ± 9.8
17	58.3 ± 8.7	56.7 ± 8.8	55.6 ± 8.9	74.1 ± 17.6	65.8 ± 7.5	58.7 ± 8.3
21	56.3 ± 7.7	53.9 ± 8.0	53.2 ± 8.0	72.9 ± 17.2	62.6 ± 7.6	55.8 ± 7.5
25	55.3 ± 6.6	53.1 ± 6.5	52.9 ± 6.5	72.7 ± 17.0	62.6 ± 7.2	55.1 ± 6.2
29	53.4 ± 5.7	51.5 ± 5.9	51.5 ± 5.6	72.1 ± 16.8	61.9 ± 6.8	53.5 ± 5.3
33	52.4 ± 5.1	50.6 ± 5.4	50.8 ± 5.4	71.8 ± 16.7	61.1 ± 5.8	52.5 ± 5.2
37	51.8 ± 5.0	50.1 ± 5.4	50.3 ± 5.2	71.5 ± 16.6	60.5 ± 6.1	51.9 ± 5.0
41	50.2 ± 5.1	48.7 ± 5.3	49.0 ± 5.2	70.9 ± 16.5	59.1 ± 5.7	50.5 ± 5.0
45	49.3 ± 4.6	47.6 ± 5.1	48.1 ± 5.0	70.5 ± 16.4	58.4 ± 5.4	49.4 ± 4.9
49	48.2 ± 4.3	46.6 ± 5.0	47.2 ± 4.8	70.0 ± 16.2	56.9 ± 5.3	48.5 ± 4.7

Figura 19: Comparação da métrica *balanced accuracy* no conjunto de teste para INTRA.



Fonte: Autoria própria.

Tabela 17: Resultados de NEURAL_INTRA.

N	Balanced Accuracy	F1 Score	Cohen Kappa	AUC	Precision	Recall
2	99.7 ± 1.2	99.6 ± 1.3	99.3 ± 2.6	100.0 ± 0.0	99.7 ± 1.1	99.6 ± 1.3
3	97.1 ± 3.1	96.9 ± 3.4	95.0 ± 5.7	99.6 ± 1.0	97.3 ± 3.1	96.9 ± 3.5
4	90.2 ± 5.3	88.9 ± 6.3	84.7 ± 8.6	97.6 ± 2.1	90.1 ± 5.7	89.0 ± 6.2
5	88.5 ± 5.4	87.0 ± 6.4	83.5 ± 8.1	97.9 ± 1.5	88.2 ± 6.0	87.1 ± 6.2
9	86.4 ± 4.9	84.8 ± 5.9	82.8 ± 6.4	98.6 ± 0.9	86.4 ± 5.0	85.0 ± 5.6
13	80.3 ± 5.8	78.8 ± 6.3	77.0 ± 6.8	95.8 ± 15.4	81.1 ± 5.8	79.0 ± 6.2
17	74.7 ± 5.8	72.3 ± 6.5	70.8 ± 6.8	92.7 ± 21.3	76.0 ± 5.3	72.7 ± 6.4
21	67.4 ± 6.0	63.3 ± 7.1	62.0 ± 7.0	91.7 ± 21.1	68.6 ± 5.5	64.0 ± 6.7
25	63.2 ± 6.4	58.9 ± 6.9	58.2 ± 6.8	91.6 ± 21.0	64.2 ± 6.9	60.1 ± 6.5
29	58.5 ± 6.3	54.8 ± 7.0	54.3 ± 6.9	91.3 ± 21.0	60.5 ± 6.9	56.1 ± 6.6
33	54.0 ± 6.6	50.1 ± 7.6	50.1 ± 7.5	90.9 ± 20.9	55.9 ± 7.9	51.8 ± 7.3
37	50.4 ± 6.6	46.5 ± 7.4	46.7 ± 7.3	90.5 ± 20.8	52.8 ± 7.2	48.3 ± 7.1
41	47.5 ± 6.8	43.4 ± 7.1	43.8 ± 7.1	90.1 ± 20.7	49.5 ± 6.8	45.4 ± 6.9
45	44.1 ± 6.3	40.1 ± 6.6	40.7 ± 6.7	89.7 ± 20.6	46.8 ± 6.0	42.2 ± 6.6
49	41.7 ± 6.0	37.8 ± 6.4	38.5 ± 6.6	89.5 ± 20.6	44.4 ± 6.0	39.9 ± 6.4

4.4 BACKWARD

A tabela 18 mostra todas as métricas para o modelo de regressão logística, para todas as quantidades de classes, e a tabela 19 mostra estes resultados obtidos com o modelo de rede neural. A figura 20 mostra a acurácia balanceada para cada modelo com cada número de movimentos sendo classificados. Neste experimento não foi utilizado o modelo de *k-Nearest Neighbors*.

Tabela 18: Resultados de LOGISTIC_BACKWARD.

N	Balanced Accuracy	F1 Score	Cohen Kappa	AUC	Precision	Recall
49	23.9 ± 3.9	19.7 ± 2.4	19.9 ± 3.3	85.8 ± 2.1	26.5 ± 3.1	21.5 ± 3.2
45	26.5 ± 4.6	21.9 ± 2.5	22.1 ± 3.9	86.5 ± 2.2	29.7 ± 2.8	23.8 ± 3.8
41	27.7 ± 4.3	22.8 ± 1.9	23.2 ± 3.3	87.2 ± 2.0	29.5 ± 2.8	25.2 ± 3.1
37	30.6 ± 4.7	25.6 ± 2.3	25.7 ± 3.6	88.4 ± 2.2	33.3 ± 2.3	27.8 ± 3.5
33	33.1 ± 4.8	28.8 ± 3.0	28.4 ± 4.0	89.1 ± 1.5	37.1 ± 2.6	30.6 ± 3.9
29	36.8 ± 4.1	32.2 ± 2.6	31.9 ± 3.3	89.4 ± 1.5	40.2 ± 2.6	34.4 ± 3.1
25	40.5 ± 4.2	36.2 ± 1.8	35.3 ± 3.3	90.2 ± 1.4	44.7 ± 2.1	38.0 ± 3.1
21	44.1 ± 4.2	40.0 ± 2.4	38.7 ± 3.4	90.8 ± 1.4	48.2 ± 1.2	41.7 ± 3.3
17	49.4 ± 7.8	45.6 ± 7.2	43.4 ± 8.1	91.5 ± 2.9	53.6 ± 7.1	46.9 ± 7.6
13	58.3 ± 6.6	54.5 ± 6.8	51.7 ± 7.3	93.7 ± 2.3	62.1 ± 6.2	55.6 ± 6.7
9	63.7 ± 7.3	60.7 ± 7.4	56.9 ± 8.0	94.2 ± 2.5	69.5 ± 6.3	62.0 ± 7.1
5	71.2 ± 6.2	68.9 ± 7.7	62.1 ± 8.2	94.1 ± 3.2	75.7 ± 5.4	70.0 ± 6.5

Tabela 19: Resultados de NEURAL_BACKWARD.

N	Balanced Accuracy	F1 Score	Cohen Kappa	AUC	Precision	Recall
49	22.6 ± 5.1	18.1 ± 3.8	18.8 ± 4.6	86.9 ± 2.2	26.4 ± 2.9	20.5 ± 4.5
45	23.9 ± 4.7	19.5 ± 3.5	20.0 ± 4.3	87.3 ± 2.0	26.0 ± 4.2	21.9 ± 4.2
41	25.9 ± 5.8	21.3 ± 4.9	21.9 ± 5.5	87.8 ± 2.0	28.9 ± 4.8	23.9 ± 5.4
37	28.7 ± 6.4	23.7 ± 4.5	24.5 ± 5.6	89.0 ± 2.1	31.1 ± 3.4	26.6 ± 5.4
33	31.0 ± 6.1	26.6 ± 4.6	26.8 ± 5.6	89.6 ± 1.8	34.4 ± 3.7	29.1 ± 5.4
29	36.2 ± 6.8	31.1 ± 5.4	31.2 ± 6.2	90.3 ± 2.0	39.8 ± 3.9	33.7 ± 5.9
25	40.6 ± 6.1	35.7 ± 3.8	35.6 ± 5.0	90.5 ± 1.4	45.1 ± 3.8	38.3 ± 4.8
21	45.2 ± 6.6	40.7 ± 4.2	39.9 ± 5.5	91.3 ± 1.8	49.3 ± 3.1	42.9 ± 5.2
17	49.0 ± 7.0	45.6 ± 6.9	43.7 ± 7.6	91.9 ± 2.7	53.6 ± 6.8	47.2 ± 7.1
13	57.8 ± 6.2	54.4 ± 7.1	51.7 ± 7.3	93.9 ± 2.1	62.2 ± 6.0	55.7 ± 6.7
9	64.6 ± 7.5	62.1 ± 8.2	58.4 ± 9.1	94.7 ± 2.4	70.2 ± 6.4	63.3 ± 8.0
5	70.6 ± 10.4	68.6 ± 11.7	62.1 ± 13.0	94.2 ± 3.7	75.7 ± 8.2	70.1 ± 10.2

Figura 20: Comparação da métrica *balanced accuracy* no conjunto de teste para BACKWARD.

Fonte: Autoria própria.

5 DISCUSSÃO

Neste capítulo serão expostos alguns pontos relativos aos resultados obtidos, como comparação entre os experimentos, entre modelos, entre quantidades de movimentos sendo classificados e comparação com trabalhos anteriores.

5.1 Comparação entre modelos

Pode-se ver os resultados dos experimentos de INTER na figura 17. A figura mostra como as regressões logísticas e a rede neural se comportam de forma semelhante até aproximadamente 13 movimentos, e partir disso a regressão logística tem uma média maior que a da rede neural. Entretanto os dois modelos ainda ficariam empatados considerando o desvio padrão.

O conjunto de experimentos de INTER_500 mostraram resultados semelhantes aos de INTER. A média da acurácia inclusive é maior em INTER_500 para poucos movimentos. Isso mostra que mesmo retirando quase metade das *features* os modelos conseguiam realizar o treinamento de forma semelhante, indicando que nem todas as *features* utilizadas eram realmente necessárias para o aprendizado.

Para os experimentos INTRA a maior acurácia classificando os 49 movimentos é do modelo de k-NN. Ainda pode-se ver que todas as acurácias são melhores em comparação aos experimentos INTER, e o mesmo acontece para todas as outras métricas, como se pode ver nas tabelas. Isso se dá porque o modelo treinou com dados de todos os indivíduos, portanto o modelo aprendeu a características de todos.

Fica evidente como a seleção de movimentos utilizando o protocolo BACKWARD pode aumentar a acurácia do modelo. O modelo de regressão logística alcançou 58.3% de acurácia para 13 movimentos, enquanto que a maior acurácia obtida no experimento INTER para esta quantidade foi 51%, quase ultrapassando o limite do desvio padrão. Para muitos movimentos o desempenho é semelhante aos modelos dos experimentos de INTER (para 49 movimentos o procedimento de BACKWARD converge para o mesmo procedimento de INTER).

5.2 Comparação com trabalhos anteriores

Como a maioria desses trabalhos vistos na subseção 2.5.4 utilizam uma divisão de *datasets* sem considerar cada indivíduo, eles serão comparados aos resultados dos modelos treinados seguindo o protocolo INTRA. Os trabalhos que realizaram a divisão mantendo indivíduos diferentes nos conjuntos de treino e teste serão comparados com os resultados dos experimentos INTER. Muitos autores também utilizaram uma quantidade de movimentos classificados diferente do que se realizou neste trabalho, portanto nestes casos ele será comparado com o experimento com o número de movimentos mais próximo.

Em Glette et al. (2008) conseguiu-se até 3.5% de taxa de erro na acurácia utilizando o algoritmo k-NN para classificação de 8 movimentos, enquanto que para 9 movimentos conseguiu-se até 33.6% de erro no experimento KNN_INTRA.

Em Boschmann et al. (2009) foi utilizada uma SVM para classificar de 2 à 11 movimentos. As acurácias obtidas para 5 e 9 movimentos foram 99.7% e 96.1%, respectivamente. Para 5 e 9 movimentos o experimento com o melhor desempenho foi o NEURAL_INTRA, com 88.5% e 86.4% respectivamente.

Contando com 27 indivíduos, Atzori et al. (2012) conseguiu um erro próximo à 43% utilizando o protocolo INTER para 3 movimentos, e aproximadamente 7% utilizando o INTRA. Para 11 movimentos teve-se aproximadamente 77% de erro no protocolo INTER e perto de 10% no INTRA. Agora para os 52 movimentos obteve-se mais de 90% de erro utilizando o protocolo INTER, e 20% utilizando o INTRA. Neste trabalho obteve-se 2.9% de erro no experimento NEURAL_INTRA para 3 movimentos, e obteve-se 4.9% de erro no experimento LOGISTIC_INTER_500. Para 13 movimentos obteve-se 19.7% de erro no experimento NEURAL_INTRA, e 41.7% no experimento LOGISTIC_BACKWARD, que segue a mesma divisão que INTER. Para 49 movimentos LOGISTIC_INTER com 75.4%, e 51.8% no experimento KNN_INTRA.

Em Kerber, Puhl e Krüger (2017) criaram-se modelos para classificar de 5 à 40 movimentos, como descrito na subseção 2.5.4. Para 40 movimentos obteve-se pouco mais de 15% de acurácia, e para 5 movimentos obteve-se em torno de 92% de acurácia. Conseguiu-se 88.5% de acurácia com NEURAL_INTRA para 5 movimentos e 50.5% de acurácia para 41 movimentos no experimento LOGISTIC_INTRA.

A acurácia obtida também ficou próxima à obtida em Viana et al. (2019) para 6 movimentos, entretanto aqui realizou-se 1 movimento a menos, obtendo-se 88.5% no

experimento NEURAL_INTRA.

Deve-se avaliar, além das métricas obtidas, diversos fatores que influenciam na acurácia dos modelos. Neste trabalho utilizou-se 40 indivíduos, mais do que se utiliza na maioria dos trabalhos vistos. Os que mais se aproximam são os trabalhos de Atzori et al. (2012) (2012) e Atzori et al. (2014) (2014), pois além de utilizarem quase a mesma quantidade de indivíduos a base de dados é a mesma. Neste quesito conseguiu-se resultados melhores que o primeiro mas piores que o segundo, para todas as quantidades de movimentos. Conseguiram-se resultados competitivos com os alcançados em Kerber, Puhl e Krüger (2017), principalmente para experimentos utilizando uma grande quantidade de movimentos.

5.3 Discussão de resultados

No experimento INTER os modelos alcançam por volta de 98% de acurácia ao classificar apenas 2 movimentos. A acurácia cai rapidamente conforme o modelo tem que classificar mais movimentos, principalmente à partir de 4 gestos. Observa-se que a regressão logística e a rede neural tem resultados muito semelhantes, chegando a ser (numa análise superficial) estatisticamente iguais se considerarmos o desvio padrão. Pode-se ver que o método k-NN é marginalmente menos acurado a partir de 13 movimentos. Para 49 movimentos a maior acurácia balanceada foi da regressão logística com uma média de 24.6%, seguida da rede neural com 22.3%, e do k-NN com 16.0%. Isto mostra que o k-NN teve seu resultado mais afetado pela grande quantidade de classes do que os outros modelos.

No experimento INTER_500 o comportamento é similar ao do experimento INTER. Entretanto, aqui os resultados da acurácia balanceada para 49 classes tiveram uma queda marginal na sua média. Tem-se 22.1% para de regressão logística, 21.3 % para a rede neural, e 15.1% para o k-NN.

No experimento INTRA obteve-se uma acurácia perto dos 100% para 2 e 3 movimento, e que decai conforme a quantidade de movimentos aumenta. Deve-se notar que a acurácia cai muito mais lentamente que no experimento INTER. É interessante notar que para uma grande quantidade de movimento o algoritmo k-NN obteve a melhor acurácia balanceada, chegando à uma média de 48.2%, seguido da regressão logística com 45.7% e da rede neural com 41.7%.

No experimento BACKWARD têm-se resultados similares ao experimento INTER quando se tem 49 movimentos. Os valores de acurácia balanceada para a regressão logística

e para a rede neural são, respectivamente, 23.9% e 22.6%. Entretanto, para 5 classes o resultado é superior, quase ultrapassando o desvio padrão: para a regressão a acurácia balanceada é de 71.2% contra 67.3% do experimento INTER, e para a rede neural é 70.6%, contra 68.1% do experimento INTER.

6 CONSIDERAÇÕES FINAIS

Os métodos de pré-processamento utilizados foram importantes para se garantir uma qualidade de informação contida nos sinais eletromiográficos. De maneira similar os dados da base NinaPro foram de valiosíssimo uso, mostrando como bons instrumentos de aquisição, bons protocolos e rigoroso controle de qualidade são importantes para algoritmos robustos. A existência de uma extensa literatura sobre o assunto contribuiu grandemente para se entender o problema a ser resolvido e os meios necessários para tal.

A comparação de modelos que utilizam algoritmos de estado da arte, como as redes neurais profundas, com os algoritmos mais antigos trouxeram conclusões interessantes. A regressão logística por vezes ultrapassava a performance da rede neural, e em um caso o algoritmo *k-NN* foi o melhor entre os três. Isso mostra como o algoritmo de aprendizado de máquina a ser utilizado depende da área que esta sendo aplicado. Neste caso, para diferentes quantidades de movimentos um ou outro poderia ser considerado mais adequado.

É interessante analisar como várias áreas do conhecimento podem ser utilizadas para melhorar resultados previamente existentes. Utilizou-se conhecimentos de fisiologia do corpo humano para se ter um bom protocolo de aquisição de dados, que por sua vez necessita de uma excelente instrumentação eletrônica para aquisição do sinal. Com conhecimentos de processamento de sinais este sinal é limpo, e por métodos estatísticos e computacionais ele é utilizado para se reconhecer os gestos que o indivíduo realizou durante o procedimento.

Pode-se indicar algumas continuações de pesquisa que seriam complementares ou consequentes a este projeto:

- Utilização de diferentes formas de pré-processamento dos sinais. Seguiu-se certos os procedimentos de filtragem e *denoising*, que não necessariamente são os melhores. Existem outros métodos, por exemplo técnicas de BSS (*Blind Source Separation*) que podem ser utilizadas para se diminuir o *cross-talking* entre sensores.

- Pode-se pesquisar a aplicação de redes convolucionais (CNN's) no reconhecimento de gestos. As redes convolucionais podem utilizar informações temporais do sinal, não só estatísticas previamente calculadas. Redes convolucionais são conhecidas por aprenderem automaticamente as *features* importantes dos seus dados de entrada, sem a necessidade de uma extração de características anterior. Isso pode poupar

custo computacional, pois diminuiria a necessidade de um pré-processamento refinado.

- Há a possibilidade de utilização de redes recorrentes, que são normalmente utilizadas para séries temporais de dados. Redes recorrentes tem a capacidade de guardar informações vistas há tempo para as utilizarem depois. A combinação de uma rede recorrente com uma rede convolucional tem uma grande probabilidade de trazer um alto ganho de performance do modelo. Infelizmente há poucos casos na literatura de uso destas duas técnicas para reconhecimento de movimentos, o que mostra que há bastante pesquisa a ser feita na área.

- Implementação de casos de uso dos modelos. Neste trabalho se explorou processamento de sinais e algoritmos de aprendizado de máquina, e a continuação lógica da teoria é a aplicação, que seria a combinação dos modelos criados com um sistema real de controle. Este controle pode ser de uma prótese, de um ponteiro em um computador, de um braço robótico, de máquinas elétricas.
- Utilização de mais indivíduos para o treino dos modelos. Apesar dos modelos terem sido treinados com mais indivíduos do que a maioria dos trabalhos da literatura, este número ainda está longe de ser satisfatório. Modelos de *deep learning* necessitam utilizar massas gigantescas de dados para realizar seu aprendizado, e quanto mais dados de qualidade os são fornecidos melhor a performance deles. Além disso, é necessária uma grande amostra populacional de validação para se garantir de forma estatística que os resultados encontrados sejam reais. Se tratando de um contexto de uma aplicação médica, modelos confiáveis e de alta qualidade são necessários pois eles impactarão diretamente a vida dos indivíduos que utilizarem o sistema.
- Também pode-se investigar o *trade-off* entre quantidade de eletrodos utilizados e a performance dos modelos treinados. Idealmente não se deve ter nenhum tipo de acessório acoplado ao indivíduo para garantir seu conforto, mas como isso não é possível deve-se utilizar a menor quantidade possível. Com muitos eletrodos muita informação muscular pode ser obtida, porém o sistema se torna menos prático e menos confortável para quem o estiver utilizando.

REFERÊNCIAS

ARANHA, J. Sistema de acionamento bidirecional de motor DC para órtese de joelho através de sinais mioelétricos. 2017.

ATZORI, M. et al. Electromyography data for non-invasive naturally-controlled robotic hand prostheses. **Scientific Data**, v. 1, p. 1–13, 2014. ISSN 20524463.

_____. Building the ninapro database: A resource for the biorobotics community. In: **2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)**. IEEE, 2012. Disponível em: <https://doi.org/10.1109/biorob.2012.6290287>.

BISHOP, C. **Pattern Recognition And Machine Learning.pdf**. 2006.

BOSCHMANN, A. et al. Towards multi-movement hand prostheses: Combining adaptive classification with high precision sockets. In: **In Proceedings of second European conference technically assisted rehabilitation**. [S.l.: s.n.], 2009.

BúRIGO, A. Classificação de Movimentos da Mão Utilizando Eletromiografia de Superfície , Regressão Logística , Redes Neurais, Máquina de Vetores de Suporte e Base de Dados NinaPro. 2014. Monografia (Bacharel em Engenharia de Computação), UFRGS (Universidade Federal do Rio Grande do Sul), Porto Alegre, Brasil.

CHOLLET, F. et al. **Keras**. 2015. <https://keras.io>.

CHOWDHURY, R. et al. Surface electromyography signal processing and classification techniques. **Sensors**, MDPI AG, v. 13, n. 9, p. 12431–12466, set. 2013. Disponível em: <https://doi.org/10.3390/s130912431>.

COLAB webpage. Acesso em: 9 de setembro de 2019. <https://colab.research.google.com/notebooks/welcome.ipynb>.

ENGLEHART, K.; HUDGINS, B. A Robust, Real-Time Control Scheme for Multifunction Myoelectric Control. **IEEE Transactions on Biomedical Engineering**, v. 50, n. 7, p. 848–854, 2003. ISSN 15582531.

FRANÇA, I. S. X. de et al. Qualidade de vida em pacientes com lesão medular. **Revista Gaúcha de Enfermagem**, FapUNIFESP (SciELO), v. 34, n. 1, p. 155–163, mar. 2013. Disponível em: <https://doi.org/10.1590/s1983-14472013000100020>.

GERDLE, B. et al. Acquisition, processing and analysis of the surface electromyogram. In: **Modern Techniques in Neuroscience Research**. Springer Berlin Heidelberg, 1999. p. 705–755. Disponível em: <https://doi.org/10.1007/978-3-642-58552-4_26>.

GLETTE, K. et al. Comparing evolvable hardware to conventional classifiers for electromyographic prosthetic hand control. **Proceedings of the 2008 NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2008**, p. 32–39, 2008.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. v. 9, p. 249–256, 2010.

GRAPS, A. An introduction to wavelets. **IEEE Comput. Sci. Eng.**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 2, n. 2, p. 50–61, jun. 1995. ISSN 1070-9924. Disponível em: <https://doi.org/10.1109/99.388960>.

GUYTON; HALL. **Tratado de Fisiologia Médica 11ªed**. [S.l.: s.n.], 1988.

HAYKIN, S. **Neural Networks and Learning Machines**. [S.l.: s.n.], 1999. ISBN 9780131471399.

HUDGINS, B.; PARKER, P.; SCOTT, R. N. A New Strategy for Multifunction Myoelectric Control. **IEEE Transactions on Biomedical Engineering**, IEEE, v. 40, n. 1, p. 82–94, 1993. ISSN 1558-2531.

JAMES, G. et al. **An Introduction to Statistical Learning**. Springer New York, 2013. Disponível em: <https://doi.org/10.1007/978-1-4614-7138-7>.

KERAS documentation. Acesso em: 17 de setembro de 2019. <https://keras.io/why-use-keras/>.

KERBER, F.; PUHL, M.; KRüGER, A. User-independent real-time hand gesture recognition based on surface electromyography. In: **Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI '17**. ACM Press, 2017. Disponível em: <https://doi.org/10.1145/3098279.3098553>.

KINGMA, D. P.; BA, J. **Adam: A Method for Stochastic Optimization**. 2014. Disponível em: <http://arxiv.org/abs/1412.6980>.

KONRAD, P. **The ABC of EMG**. [S.l.: s.n.], 2006.

LEE, G. R. et al. Pywavelets: A python package for wavelet analysis. **Journal of Open Source Software**, abr. 2019. Disponível em: <https://doi.org/10.21105/joss.01237>.

LENAIL, A. NN-SVG: Publication-ready neural network architecture schematics. **Journal of Open Source Software**, The Open Journal, v. 4, n. 33, p. 747, jan. 2019. Disponível em: <https://doi.org/10.21105/joss.00747>.

LUCA, C. J. D. Surface Electromyography: Detection and Recording. **DelSys Incorporated**, v. 10, n. 2, p. 1–10, 2002.

LUCA, C. J. de. The use of surface electromyography in biomechanics. **Journal of Applied Biomechanics**, v. 13, n. 2, p. 135–163, 1997. ISSN 10658483.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PHINYOMARK, A.; LIMSAKUL, C.; PHUKPATTARANONT, P. A comparative study of wavelet denoising for multifunction myoelectric control. In: **2009 International Conference on Computer and Automation Engineering**. IEEE, 2009. Disponível em: <https://doi.org/10.1109/iccae.2009.57>.

_____. Application of wavelet analysis in EMG feature extraction for pattern classification. **Measurement Science Review**, Walter de Gruyter GmbH, v. 11, n. 2, jan. 2011. Disponível em: <https://doi.org/10.2478/v10048-011-0009-y>.

PHINYOMARK, A.; SCHEME, E. EMG pattern recognition in the era of big data and deep learning. **Big Data and Cognitive Computing**, MDPI AG, v. 2, n. 3, p. 21, ago. 2018. Disponível em: <https://doi.org/10.3390/bdcc2030021>.

PIZZOLATO, S. et al. Comparison of six electromyography acquisition setups on hand movement classification tasks. **PLOS ONE**, Public Library of Science (PLoS), v. 12, n. 10, p. e0186132, out. 2017. Disponível em: <https://doi.org/10.1371/journal.pone.0186132>.

SKLEARN documentation. Acesso em: 23 de setembro de 2019. <https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-transformer>.

STéPHANE, M. Acknowledgments. In: STéPHANE, M. (Ed.). **A Wavelet Tour of Signal Processing (Third Edition)**. Third edition. Boston: Academic Press, 2009. ISBN 978-0-12-374370-1. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780123743701000215>.

TANG, X. et al. Hand motion classification using a multi-channel surface electromyography sensor. **Sensors**, MDPI AG, v. 12, n. 2, p. 1130–1147, jan. 2012. Disponível em: <https://doi.org/10.3390/s120201130>.

VEER, K.; SHARMA, T. A novel feature extraction for robust EMG pattern recognition. **Journal of Medical Engineering and Technology**, v. 40, n. 4, p. 149–154, 2016. ISSN 1464522X.

VIANA, P. et al. An artificial neural network for hand movement classification using surface electromyography. In: **Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies**. SCITEPRESS - Science and Technology Publications, 2019. Disponível em: <https://doi.org/10.5220/0007404201850192>.

WILCOXON, F. Individual comparisons by ranking methods. **Biometrics Bulletin**, JSTOR, v. 1, n. 6, p. 80, dez. 1945. Disponível em: <https://doi.org/10.2307/3001968>.

ZWILLINGER, D.; KOKOSKA, S. **CRC Standard Probability and Statistics Tables and Formulae**. CRC Press, 1999. Disponível em: <https://doi.org/10.1201/9781420050264>.