UNIVERSIDADE ESTADUAL PAULISTA

“JÚLIO DE MESQUITA FILHO”

Campus de Ilha Solteira

FABIO AGOSTINHO BORIS

ANÁLISE DE TÉCNICAS DE CLASSIFICAÇÃO DE SINAIS DE ELETROMIOGRAFIA PARA CONTROLE DE PRÓTESE DE MEMBRO SUPERIOR

Ilha Solteira

2020

FABIO AGOSTINHO BORIS

ANÁLISE DE TÉCNICAS DE CLASSIFICAÇÃO DE SINAIS DE ELETROMIOGRAFIA PARA CONTROLE DE PRÓTESE DE MEMBRO SUPERIOR

Dissertação apresentada à Faculdade de Engenharia de Ilha Solteira – UNESP como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de conhecimento: Automação.

Prof. Dr. Aparecido Augusto de Carvalho

Orientador

Ilha Solteira

2020

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

Boris, Fabio Agostinho.
B734a Análise de técnicas de classificação de sinais de eletromiografia para controle de prótese de membro superior / Fabio Agostinho Boris. -- Ilha Solteira: [s.n.], 2020
72 f. : il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2020

Orientador: Aparecido Augusto de Carvalho
Inclui bibliografia

1. Classificação de EMG. 2. Aprendizado de máquina. 3. Extração de Características. 4. Wavelets. 5. Controle de prótese. 6. Membro superior.

Raiane da Silva Santos
Supervisora Técnica de Seção
Seção Técnica de Referência, Atendimento ao usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação
CRB/8 - 9999

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Análise de Técnicas de Classificação de Sinais de Eletromiografia para Controle de Prótese de Membro Superior

AUTOR: FABIO AGOSTINHO BORIS
ORIENTADOR: APARECIDO AUGUSTO DE CARVALHO

Aprovado como parte das exigências para obtenção do Título de Mestre em ENGENHARIA ELÉTRICA, área: Automação pela Comissão Examinadora:

Prof. Dr. APARECIDO AUGUSTO DE CARVALHO
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira - UNESP

Prof. Dr. MARCELO AUGUSTO ASSUNÇÃO SANCHES
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira - UNESP

Prof. Dr. TONY INÁCIO DA SILVA
Departamento de Eletroeletrônica / Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso - IFMT

Ilha Solteira, 31 de julho de 2020

RESUMO

A classificação de sinais eletromiográficos é uma tarefa importante para o controle de próteses ativas de membros superiores. Este trabalho propõe analisar e avaliar técnicas e ferramentas para classificação de sinais eletromiográficos (EMG) de superfície, produzindo modelos preditivos viáveis para o controle de próteses de membros superiores. Os sinais eletromiográficos foram obtidos a partir de uma base de dados pública. Algoritmos de aprendizagem de máquina, transformadas discretas *wavelets* e sete características do sinal EMG foram usadas para classificar os sinais. Foram criadas amostras aleatórias para as tarefas de treinamento e teste em subconjuntos com 80% e 20% dos dados, respectivamente. Os algoritmos de aprendizado de máquina foram treinados com diferentes configurações, permitindo avaliação entre os classificadores e os parâmetros utilizados em cada modelo. Após a realização dos testes, observa-se que o processamento dos sinais e a extração de características são processos importantes para a obtenção de resultados acurados em sinais eletromiográficos. O melhor resultado produziu uma acurácia média de 97,22% com um classificador *Random Forest* e três características extraídas a partir de uma transformada discreta wavelet nos dois canais do sinal EMG.

Palavras chave: Classificação de EMG. Aprendizado de máquina. Extração de características. Wavelets. Controle de prótese. Membro superior.

ABSTRACT

The classification of surface electromyographic signals is an important task for the control of active upper-limb prostheses. This study aims to analyze and evaluate techniques and tools to classify surface electromyographic signals (EMG), creating viable models for the control of upper limb prostheses. The electromyographic signals were obtained from a public database. Machine learning algorithms, discrete wavelets transforms and seven features of the EMG signal were used to classify the signals. Random samples were created for the training and testing tasks in subsets with 80% and 20% of the data, respectively. Machine learning algorithms for classifying electromyographic signals were trained with different configurations, allowing evaluation between combinations of techniques and parameters. It was observed that signal processing and feature extraction are important processes for obtaining accurate results. The best result produced an average accuracy of 97,22% with a Random Forest classifier and three features extracted from discrete wavelet transform from surface electromyography signals of two channels.

Keywords: EMG classification. Machine learning. Features extraction. Wavelets. Prostheses control. Upper limb.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	*Artificial Neural Network* – Rede Neural Artificial
CWT	*Continuous Wavelet Transform* – Transformada Wavelet Contínua
DTREE	*Decision Tree* – Árvore de Decisão
DWT	*Discrete Wavelet Transform* – Transformada Wavelet Discreta
EMG	*Electromyography* – Eletromiografia
FDA	*Food and Drug Administration* – Administração de Alimentos e Medicamentos
iEMG	*Intramuscular Electromyography* – Eletromiografia Intramuscular
KNN	*K-Nearest Neighbors* – Vizinhos Próximos de K
LDA	*Linear Discriminant Analysis* – Análise do Discriminante Linear
LR	*Logistic Regression* – Regressão Logística
MAV	*Mean Absolute Value* – Média do Valor Absoluto
MLP	*Multilayer Perceptron*
MNF	*Mean Frequency* – Frequência Média
QDA	*Quadractic Discriminant Analysis* – Análise do Linear Quadrático
RF	*Random Forests* – Florestas Aleatórias
RMS	*Root Mean Square* – Raiz Quadrada Média
RSS	*Residual Sum of Squares* – Soma Residual dos Quadrados
SD	*Standard Deviation* – Desvio Padrão
sEMG	*Surface Electromyography* – Eletromiografia de Superfície
SSC	*Slope Sign Changes* – Mudanças do Sinal da Inclinação
SVM	*Support Vector Machine* – Máquina de Vetor de Suporte
VAR	*Variance* – Variância
WT	*Wavelet Transform* – Transformada Wavelet
ZC	*Zero Crossing* – Contagem de Passagens em Zero

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo são apresentadas e discutidas as justificativas para realização da pesquisa, o estado da arte, o objetivo e a organização do texto.

1.1 JUSTIFICATIVAS

Atualmente, sistemas para o reconhecimento de padrões em sEMG para o controle próteses de membro superior são raros no mercado. Segundo Calado, Soares e Matos (2019), existem apenas dois produtos comerciais aprovados pela FDA (*Food and Drug Administration*).

Xavier *et al*. (2015, 2016, 2017) desenvolveram uma prótese antropomórfica de membro superior de baixo custo capaz de reproduzir movimentos anatomicamente funcionais através de um sistema eletrônico monocanal para aquisição de sinais eletromiográficos de superfície (sEMG), uma luva equipada com sensores flexíveis, para gerar a base de movimentos de mão, e um mecanismo de classificação baseado na contagem de contrações musculares presentes no sinal, apresentando bons resultados com voluntários hígidos. Posteriormente, Xavier *et al*. (2019) apresentaram um sistema eletrônico de controle que funciona em três modos de operação, que permitem a execução e a configuração dos movimentos de mão reproduzidos pela prótese. O sistema foi testado por voluntários amputados e congênitos mostrando funcionamento adequado em ambos os casos.

De acordo com Sapsanis, Georgoulas e Tzes (2013), o sEMG apresenta-se como um candidato viável para classificação de padrões devido à assinatura distinta de cada movimento no sinal produzido. No campo da biomecatrônica, o uso desses sinais como entrada em sistemas de tomada de decisão é uma parte vital para o controle efetivo de um exoesqueleto robótico (mãos, braços e membros inferiores). Nos campos da saúde e reabilitação, análise e classificação de dados tornam-se uma ferramenta importante, auxiliando em diagnósticos e acelerando o processo de novas descobertas (GU *et al*., 2017; PIZZOLATO *et al*., 2017).

1.2 ESTADO DA ARTE

Sapsanis *et al*. (2013) apresentaram uma abordagem para o reconhecimento de padrões para a identificação de movimentos de mão básicos através de dados provenientes de sEMG. Os dados foram pré-processados e então submetidos a extração de oito características que foram testadas em diferentes subconjuntos por um classificador linear. Em um estudo posterior, Sapsanis, Georgoulas e Tzes (2013) submeteram o mesmo conjunto de dados a uma nova abordagem envolvendo duas características adicionais e dois métodos de redução de dimensionalidade, aumentando a acurácia de classificação geral, indicando que o conjunto de características utilizado é adequado. Os conjuntos de dados produzidos e utilizados foram disponibilizados de forma pública no *UCI Machine Learning Repository* (DHEERU; KARRA TANISKIDOU, 2017).

Negi, Kumar e Mishra (2016) avaliaram dezenove características de domínio do tempo extraídas de três canais sEMG para o controle de próteses de membros superiores através de gráficos de dispersão e um classificador linear com diferentes combinações das características comparando a acurácia entre dois métodos de redução de dimensionalidade.

Betthauser *et al*. (2016) propuseram um método de classificação robusto a partir do treinamento em condições construídas por representações esparsas dos dados compostos de cinco características de domínio do tempo. Três indivíduos hígidos executaram cinco movimentos de mão, que foram adquiridos por oito canais sEMG. Para demonstrar a robustez, foram aplicadas modificações na posição do membro as quais não foram explicitamente treinadas. A acurácia de classificação foi comparada por dois classificadores.

Masson *et al*. (2016) apresentaram um sistema que permite a integração de dispositivos de baixo custo para o controle de próteses mioelétricas com algoritmos para mapear diferentes respostas mioelétricas com comandos externos sem exigir conhecimento profundo ou específico de programação. O controle é realizado por um *software* com interface gráfica construído pelos autores. Oito indivíduos hígidos executaram cinco movimentos de mão, cujos sinais foram adquiridos por oito canais eletromiográficos de superfície. A classificação ocorre após selecionadas as características pelo índice Daves-Boldin.

Cene e Balbinot (2016) desenvolveram um método baseado em regressão logística regularizada para classificar dezessete movimentos de mão distintos através do processamento de doze canais sEMG, dos quais extraíram três características distintas. Os testes envolveram cinquenta indivíduos, incluindo dez amputados.

Jarrassé *et al*. (2017) investigaram a correlação de doze movimentos de dedos, mão e punho e a atividade muscular residual em cinco sujeitos amputados baseados na análise de sEMG obtidos por vinte e quatro eletrodos posicionados no membro residual. O estudo mostrou que com uma arquitetura de classificação estado da arte é possível classificar corretamente a atividade do membro fantasma.

Karabulut *et al*. (2017) compararam três características de domínio do tempo de sEMG na classificação de forças externas aplicadas a mãos humanas. Três voluntários hígidos realizaram os exercícios de contração. Após a extração das características, o processo de classificação foi aplicado para a predição das forças externas utilizando redes neurais artificiais. Apesar dos bons resultados, os autores apontam a natureza empírica do desenvolvimento dos modelos como principal desvantagem das redes neurais.

Tosin *et al*. (2017) propuseram a inserção de uma etapa no processo de classificação de sEMG que seleciona quais características serão usadas no processo de treinamento com o objetivo de melhorar a acurácia de classificação, obtendo resultados positivos, utilizando entre 39 e 53 das 144 características totais.

Bian, Li e Liang (2017) investigaram quatro classificadores para identificar oito movimentos de mão baseados no reconhecimento de padrões de sEMG. Após obtidos os sinais de oito canais, foram extraídas cinco características distintas e realizou-se redução de dimensionalidade. Os testes foram realizados com cinco voluntários hígidos.

Mayor *et al*. (2017) apresentaram o desenvolvimento de um novo método para reconhecimento de padrões em sEMG. O sinal foi obtido a partir de quatro canais. Foram extraídas 17 características distintas, que antes de serem classificadas por três classificadores, passaram por dois tipos de redução de dimensionalidade. O sistema foi testado em dez voluntários amputados e os resultados foram comparados com os obtidos por dez voluntários hígidos.

Bellingegni *et al*. (2017) realizaram uma análise comparativa em quatro classificadores sem a extração de características para o controle embarcado de próteses de membros superiores a partir dos sinais obtidos por seis canais sEMG coletados com trinta voluntários amputados realizando cinco movimentos distintos. Foram avaliadas a acurácia e o custo computacional de cada algoritmo.

Krasoulis *et al*. (2017) propuseram uma técnica para o controle de prótese de membro superior a partir do uso de sEMG e medidas inerciais. Os dados de seis movimentos de mão foram coletados a partir de vinte voluntários hígidos e dois amputados. A classificação foi realizada por um classificador linear a partir de quatro características distintas. Os autores apontaram que a inclusão das medidas inerciais melhorou sensivelmente a acurácia de classificação.

Hu, Kan e Li (2018) apresentaram uma nova abordagem de classificação de padrões a partir da extração de vinte e sete características e um classificador híbrido baseado em grafos acíclicos direcionais e máquinas de vetores de suporte. Os dados de seis movimentos executados por quinze voluntários hígidos foram obtidos a partir de quatro canais sEMG. O sinal foi pré processado com *wavelets* e foi realizada a redução de dimensionalidade. Para os autores, as descobertas podem ter aplicações como controle mioelétrico, bem como aplicações práticas com interação humano-computador.

Powar, Chemmangat e Figarado (2018) apresentaram uma nova etapa de pré-processamento de sEMG baseada no método da deconvolução por mínima entropia com o objetivo de melhorar o sinal para a extração de características resultando em melhores classificações de movimentos de membros superiores. Os dados de oito movimentos executados por sete voluntários hígidos foram obtidos a partir de dois canais sEMG, extraindo-se de dezoito características distintas e posteriormente a reduzindo-se a dimensionalidade. Os resultados práticos demonstraram a viabilidade da abordagem com aumento na precisão da classificação sem aumento significativo no tempo computacional em sete sujeitos.

Cognolato *et al*. (2018) avaliaram de forma quantitativa o desempenho do classificador de gestos manuais da braçadeira Myo em três voluntários amputados. Foi observada uma clara relação entre o tamanho do membro residual e a acurácia da classificação, quanto maior o membro residual, maior a acurácia.

Phukpattaranont *et al*. (2018) propuseram um sistema para classificação de catorze movimentos executados por dez indivíduos hígidos capturados por seis canais sEMG. Foram comparadas seis técnicas para extração de características combinadas a sete classificadores diferentes com o objetivo de encontrar o conjunto com melhor acurácia.

Wu *et al*. (2018) exploraram um método baseado em um único canal sEMG de superfície para a classificação de movimentos. Nesse método, extraem o envelope do sinal com um circuito de pré-processamento, e do envelope, extraem quinze características de cada segmento válido, descartando as características com menor coeficiente de correlação, realizam redução de dimensionalidade, e então a classificação com um algoritmo híbrido.

Yang *et al*. (2018) desenvolveram um sistema com três canais sEMG para adquirir sinais dos músculos de antebraços. As características do sinal foram extraídas com o método da densidade espectral de potência. Os resultados mostraram melhor acurácia aliada ao emprego do algoritmo genético para a otimização de um classificador de máquina de vetores de suporte.

Sui, Wan e Zhang (2019) propuseram resolver o problema de baixa acurácia de reconhecimento em próteses mioelétricas com três graus de liberdade e o longo tempo de treinamento. Nesse projeto, o sinal é decomposto com pacotes wavelet, e um novo classificador baseado no método do enxame de partículas e nas máquinas de vetores de suporte é utilizado para classificação. Os resultados mostraram que o algoritmo pode identificar efetivamente seis tipos de movimentos com melhor acurácia comparado ao algoritmo tradicional.

Pancholi e Joshi (2019a) apresentaram um sistema embarcado com baixo consumo de energia que reconhece movimento decodificando oito canais de sinal eletromiográfico de superfície. O sistema foi validado em tempo real com a execução de seis atividades por seis voluntários, incluindo um amputado. Em um estudo posterior, Pancholi e Joshi (2019b) propuseram a extração de um novo conjunto de características em momentos derivativos do tempo para melhorar a acurácia de classificação de movimentos em membros superiores. Os testes foram realizados com sinais gerados por oito voluntários hígidos de uma base de dados pública de sinais eletromiográficos de superfície. Foram comparados três classificadores diferentes, obtendo aumento de acurácia em ambos.

Li *et al*. (2019) propuseram um novo método de extração de características de sinais eletromiográficos de superfície baseado na região ativa do músculo. O experimento classificou quatro movimentos diferentes com o objetivo de provar que as novas características produzem melhor acurácia de classificação comparadas a outras características tradicionais. Os autores concluíram que o método pode melhorar a acurácia de classificação, entretanto esperam melhorar o resultado empregando eletrodos de alta densidade.

1.3 OBJETIVO

O objetivo deste trabalho é analisar e avaliar técnicas e ferramentas para classificação de sinais eletromiográficos, produzindo modelos preditivos viáveis para a ativação de próteses de membros superiores.

1.4 ORGANIZAÇÃO DO TEXTO

Este trabalho está dividido em 5 capítulos. No primeiro, apresenta-se uma discussão do estado da arte, as justificativas e o objetivo do trabalho. No Capítulo 2 apresenta-se a fundamentação teórica, desde uma introdução ao conceito de eletromiografia, bem como as técnicas empregadas para análise e classificação desses sinais. No capítulo 3 discute-se o desenvolvimento e a metodologia do trabalho, detalhando os processos empregados para a realização dos testes e avaliação dos resultados. No Capítulo 4, apresenta-se e discute-se os resultados obtidos. Por fim, as conclusões do estudo e ideias para trabalhos futuros são apresentadas no Capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica e os conceitos que inspiram este trabalho. Inicialmente, apresenta-se uma descrição sucinta sobre a eletromiografia de superfície, que é empregada na obtenção dos dados utilizados neste estudo. É apresentada também uma introdução sobre *wavelets*, que são empregadas no processamento dos dados analisados. Em seguida, discorre-se sobre técnicas de análise e classificação de dados.

2.1 ELETROMIOGRAFIA

O sinal eletromiográfico (EMG) é o sinal biológico que mede a atividade produzida pelos músculos esqueléticos durante a contração, representando atividades neuromusculares (SAPSANIS; GEORGOULAS; TZES, 2013). O EMG pode ser dividido em duas categorias principais: sinal eletromiográfico intramuscular (iEMG), no qual as medições são feitas através de agulhas inseridas no músculo e sinal eletromiográfico de superfície (sEMG), no qual as medições são feitas através de eletrodos na pele ao longo do músculo. Tendo como principal vantagem, ser não invasiva, a sEMG apresenta desvantagens em relação ao iEMG, pois o sinal em cada ponto de aquisição é composto de uma sobreposição das ativações das fibras musculares próximas, o que reduz a resolução do sinal. A alta impedância do tecido da pele também diminui consideravelmente a intensidade do sinal em comparação com o iEMG (CHOWDHURY *et al*., 2013).

2.2 WAVELETS

A transformada wavelet (WT) é uma ferramenta que divide dados ou funções ou operadores em diferentes componentes de frequência e, em seguida, estuda cada componente com uma resolução correspondente à sua escala (DAUBECHIES, 1992).

O método de transformada wavelet consiste em duas categorias, a transformada wavelet discreta (DWT) e a transformada wavelet contínua (CWT). A CWT apresenta a distribuição da amplitude e fase do sinal em duas variáveis, que são tempo e escala, onde as duas variáveis são discretizadas e usadas para quebrar uma função de tempo em uma pequena onda que é chamada de wavelet. A redundância

do CWT pode ser resolvida pelo método DWT, que é uma técnica para transformar o sinal de interesse em subconjuntos de coeficientes multiresolução ou multifrequência. Há duas seleções importantes que devem ser conduzidas e consideradas, que são a função wavelet e a profundidade de decomposição (BURHAN; KASNO; GHAZALI, 2016).

O benefício de usar uma função wavelet é que ela tem derivadas contínuas, o que permite decompor uma função contínua com mais eficiência e também evita sinais indesejados. Chowdhury, *et al*. (2013) concluiu que a análise de sinais sEMG usando a função de Daubechies, apresenta bons resultados, apontando como recomendação as funções db2, db4, db6, db44 e db45 no nível de decomposição 4. A Tabela 1 apresenta diferentes tipos de *wavelets* com suas famílias.

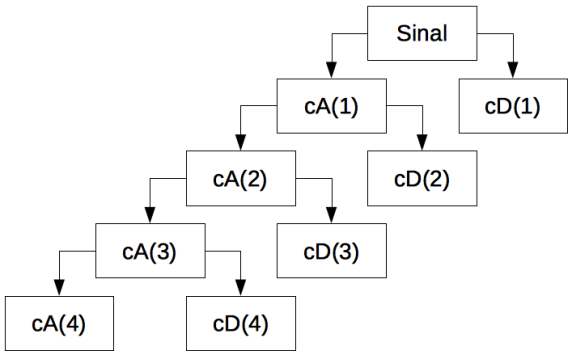
Tabela 1 – Lista de 324 funções wavelet de 15 famílias wavelet

Família Wavelet	**Subtipo Wavelet**	**Número**
Haar	db1	1
Daubechies	db2-db45	2-45
Coiflet	coif1-coif5	46-50
Morlet	morl	51
Complex Morlet	cmor	52-147
Discrete Meyer	dmey	148
Meyer	meyr	149
Mexican Hat	mexh	150
Shannon	shan	151-200
Frequency B-spline	fbsp	201-260
Gaussian	gaus	261-267
Complex Gaussian	cgaus	268-275
Biorthogonal	bior	276-290
Reverse Biorthogonal	rbio	291-305
Symlet	sym	306-324

Fonte: Adaptado de Chowdhury *et al*. (2013)

Na decomposição, o sinal é transformado em subconjuntos de coeficientes. No primeiro nível, o sinal original é filtrado pela função wavelet para obter um subconjunto do coeficiente de detalhes (cD) e um subconjunto do coeficiente de aproximação (cA). Em seguida, os dados são processados por vários níveis até o nível final escolhido, conforme ilustrado na Figura 1 (BURHAN; KASNO; GHAZALI, 2016).

Figura 1 – Representação gráfica da árvore de decomposição wavelet nível 4



Fonte: Adaptado de Burhan, Kasno e Ghazali (2016)

2.3 ANÁLISE DE DADOS E REGRESSÃO LINEAR SIMPLES

Análise de dados é um processo onde dados brutos são ordenados e organizados para serem utilizados em métodos que auxiliam em explicar o passado e prever o futuro. Não se trata exclusivamente números, mas também sobre elaborar e responder questões, desenvolver explicações e testar hipóteses, sendo um campo multidisciplinar que combina Ciência da Computação, Inteligência Artificial, Aprendizado de Máquina, Estatística, Matemática e conhecimento do Domínio (CUESTA, 2013).

Regressão linear é uma ferramenta útil para predizer respostas quantitativas. Várias abordagens de aprendizado estatístico podem ser vistas como generalizações ou extensões de regressão linear. Especificamente, regressão linear simples é uma abordagem direta para prever uma resposta quantitativa Y com base em um único preditor X, assumindo a existência aproximada de uma relação entre X e Y (JAMES *et al.*, 2013). Matematicamente pode-se descrever essa relação como:

$$Y \approx \beta_0 + \beta_1 X \tag{1}$$

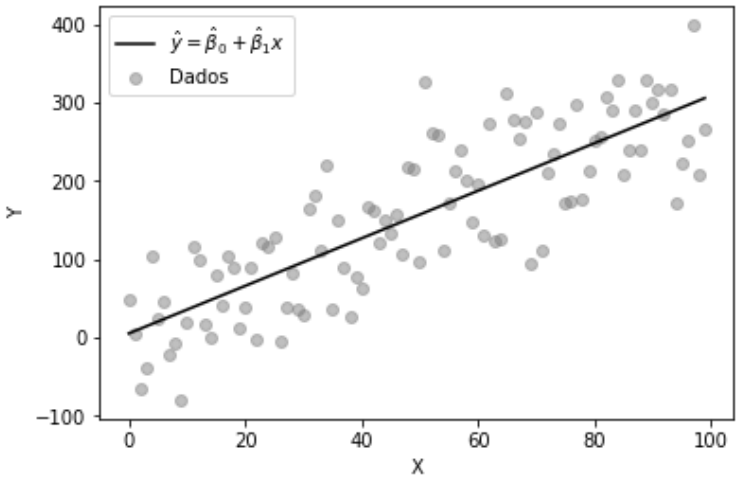
na qual β_0 e β_1 são duas constantes desconhecidas que representam os termos de intersecção e inclinação no modelo linear. Juntas, β_0 e β_1 são conhecidas como coeficientes ou parâmetros do modelo. Uma vez utilizado os dados de treinamento para estimar $\hat{\beta}_0$ e $\hat{\beta}_1$ como coeficientes do modelo, calcula-se

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x \quad (2)$$

na qual $\hat{y}$ indica uma predição de Y baseado em $X = x$. Usando-se o símbolo ^ para denotar o valor estimado para um parâmetro ou coeficiente desconhecido, ou para denotar o valor previsto da resposta.

Na prática β_0 e β_1 são desconhecidos. Portanto, antes de utilizar (1) para realizar previsões, deve-se estimar esses coeficientes. Seja $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ a representação de n pares observados, onde cada um consiste em um valor de X e um valor de Y, onde o objetivo é obter os coeficientes estimados β_0 e β_1 de modo que o modelo linear (1) apresente um bom ajuste aos dados, isto é, $y_i \approx \hat{\beta}_0 + \hat{\beta}_1 x_i$ para $i = 1, \ldots, n$. Em outras palavras, pretende-se encontrar a intersecção $\hat{\beta}_0$ e a inclinação $\hat{\beta}_1$ que resulte na linha mais próxima aos n pontos de dados conforme Figura 2. Existem várias maneiras de medir a proximidade, entretanto, de longe, a abordagem mais comum envolve a minimização do critério dos mínimos quadrados.

Figura 2 – Simulação de regressão linear simples



Fonte: Elaboração do próprio autor

Seja $y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ a previsão para Y baseada no i-ésimo valor de X. Portanto, $e_i = y_i - \hat{y}_i$ representa o i-ésimo residual – esta é a diferença entre o valor da resposta observada e o valor de resposta prevista pelo modelo linear. Define-se a soma residual dos quadrados (*residual sum of squares* RSS) como

$$RSS = e_1^2 + e_2^2 + \cdots + e_n^2$$

ou, de forma equivalente

$$RSS = \left(y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1\right)^2 + \left(y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2\right)^2 + \cdots + \left(y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n\right)^2 \quad (3)$$

A abordagem dos mínimos quadrados escolhe $\hat{\beta}_0$ e $\hat{\beta}_1$ para minimizar RSS. Realizando alguns cálculos, pode-se mostrar que os minimizadores são

$$\begin{aligned} \hat{\beta}_1 &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2} \\ \hat{\beta}_0 &= \bar{y} - \hat{\beta}_1 \bar{x} \end{aligned} \quad (4)$$

onde $\bar{y} \equiv \frac{1}{n}\sum_{i=1}^{n} y_i$ e $\bar{x} \equiv \frac{1}{n}\sum_{i=1}^{n} x_i$ são as amostras médias. Assim, (4) define as estimativas dos coeficientes de mínimos quadrados para regressão linear simples.

Pode-se assumir como verdadeira a relação entre X e Y na forma $Y = f(X) + \epsilon$ para uma função f desconhecida, onde ϵ é um termo de erro aleatório com média zero. Se f é aproximada por uma função linear, então pode-se escrever esta relação como

$$Y = \beta_0 + \beta_1 X + \epsilon \quad (5)$$

O termo de erro é um resumo do que se perde com esse modelo simples. O relacionamento verdadeiro provavelmente não é linear, podendo haver outras variáveis que causam variação em Y, e/ou erro de medição. Normalmente, assume-se que o termo de erro é independente de X.

O modelo dado por (5) define a linha de regressão populacional, que é a melhor aproximação linear para a verdadeira relação entre X e Y. As estimativas dos coeficientes de regressão dos mínimos quadrados (4) caracterizam a reta dos mínimos quadrados (3).

2.4 REGRESSÃO LINEAR MÚLTIPLA

A regressão linear simples é uma abordagem útil para prever uma resposta com base em um único preditor. No entanto, na prática, muitas vezes tem-se mais de um preditor (JAMES *et al.*, 2013).

Para estender uma análise de dados para acomodar duas ou mais variáveis preditivas, uma opção é executar regressões lineares simples separadas, cada uma com um preditor; entretanto a abordagem de ajustar um modelo de regressão linear

simples para cada preditor não é satisfatória, pois não fica claro como fazer uma predição simples nesse cenário, pois cada preditor está associado a uma equação diferente, e cada uma das equações de regressão ignora as outras variáveis preditivas na formação de estimativas para os coeficientes de regressão.

Uma abordagem melhor é estender o modelo de regressão linear simples apresentado na equação (5) para que ele possa acomodar diretamente múltiplas variáveis preditivas. Pode-se fazer isso dando a cada preditor um coeficiente de inclinação separado em um único modelo. Em geral, supõe-se p preditores distintos. Então, o modelo de regressão linear múltipla assume a equação

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon \tag{6}$$

assim, os coeficientes $\beta_0, \beta_1, \ldots, \beta_p$ em (6) são desconhecidos, e devem ser estimados. Neste caso, dadas as estimativas $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$, realiza-se as previsões com a seguinte equação

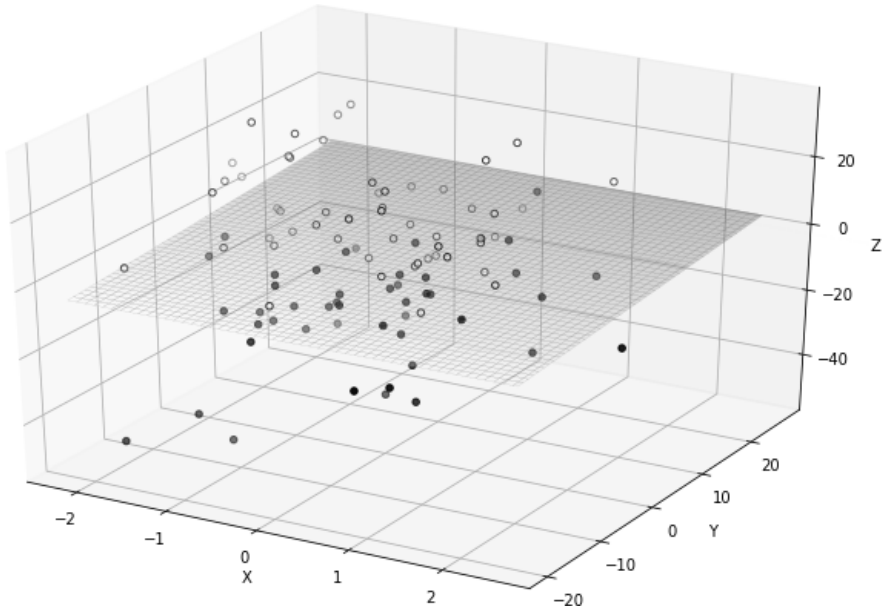
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p \tag{7}$$

e os parâmetros são estimados usando a mesma abordagem dos mínimos quadrados usada no contexto de regressão linear simples.

$$\begin{aligned} RSS &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ RSS &= \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip}\right)^2 \end{aligned} \tag{8}$$

Para fins ilustrativos, apresenta-se na Figura 3 uma simulação elaborada a partir de conjunto de dados tridimensional aleatório, com dois preditores e uma resposta. De forma análoga a Figura 2 onde a linha é calculada por (2), o hiperplano da Figura 3 é obtido por (7). Observa-se que os pontos sobre o hiperplano estão em branco, e os abaixo do hiperplano em preto.

Figura 3 – Simulação de regressão linear múltipla



Fonte: Elaboração do próprio autor

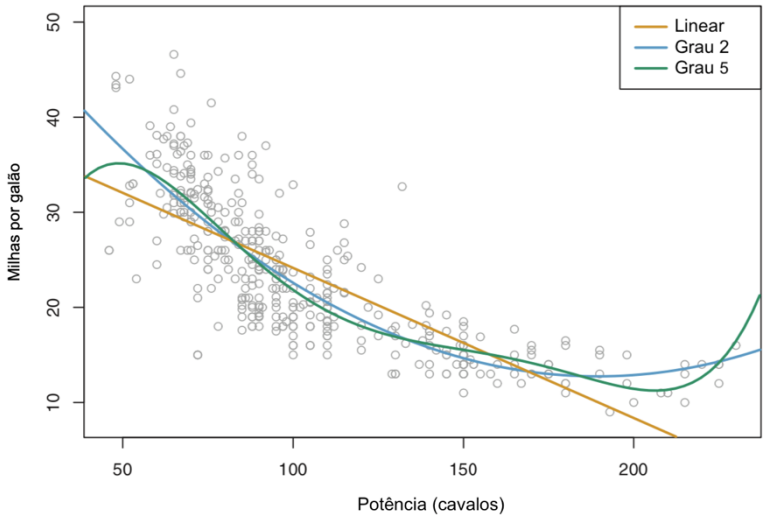
O modelo (6) assume uma relação linear entre a resposta e os preditores, entretanto, em alguns casos, essa relação deve ser não linear. Uma forma de estender o modelo linear para acomodar relações não lineares é o uso de regressão polinomial.

Considerando-se a Figura 4, com os atributos de consumo em milhas por galão e potência em cavalos, onde modelo é não linear, e os dados sugerem uma relação "curva", nesse caso, apresentadas em graus 2 e 5. Como exemplo, a forma quadrática pode ser obtida pelo modelo

$$consumo = \beta_0 + \beta_1 \times \text{potência} + \beta_2 \times \text{potência}^2 + \epsilon \quad (9)$$

entretanto, a equação (9), que prevê o atributo *consumo* usando uma função linear de *potência*, ainda é um modelo linear.

Figura 4 – Regressão linear polinomial



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

Esta abordagem descreve como estender o modelo linear para acomodar relações não lineares sendo conhecida como regressão polinomial, pois inclui funções polinomiais dos preditores no modelo de regressão.

2.5 CLASSIFICAÇÃO

Os modelos de regressão linear discutidos nos tópicos 2.2 e 2.4 assumem que a variável de resposta é quantitativa, entretanto, em muitas situações a variável de resposta poderia ser qualitativa. Segundo Cuesta (2013), dados quantitativos são medidas numéricas expressas em termos de números, enquanto dados qualitativos são medidas categóricas expressas em termos de linguagem natural. Para James *et al*. (2013), a tarefa de classificação consiste em prever respostas qualitativas, uma vez que atribui a observação para uma categoria, ou classe.

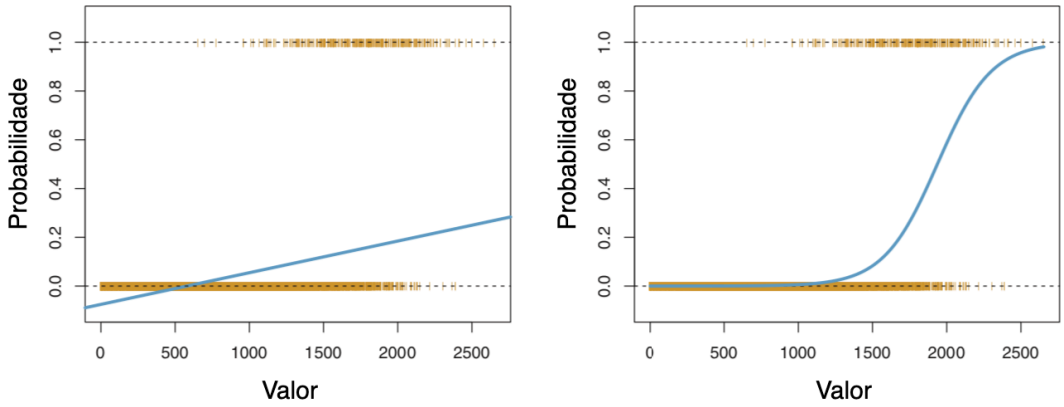
James *et al*. (2013) apontam que frequentemente os métodos usados para classificação, predizem primeiro a probabilidade de cada uma das categorias de uma variável qualitativa como a base para realizar a classificação. Nesse sentido, eles também se comportam como métodos de regressão.

Existem várias técnicas de classificação, ou classificadores, que podem ser usadas para prever uma resposta qualitativa, algumas delas serão discutidas posteriormente.

2.6 REGRESSÃO LOGÍSTICA

Considerando-se um conjunto de dados, em que a resposta cai em uma de duas categorias, “Sim” ou “Não”. Em vez de modelar essa resposta Y diretamente, a regressão logística (*logistic regression* – LR) modela a probabilidade de que Y pertence a uma categoria específica.

Figura 5 – Estimativas obtidas com regressão linear e regressão logística



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

A Figura 5 ilustra de forma comparativa as estimativas obtidas com regressão linear (lado esquerdo) com as estimativas obtidas com regressão logística (lado direito), usando um intervalo entre 0 e 1. Nota-se que algumas estimativas realizadas com regressão linear são negativas. Os pontos em laranja indicam os valores codificados como 0 ou 1.

Para exemplificar, a probabilidade de Y dado X, escrita como $\Pr(Y = Sim|X)$, onde seus valores são abreviados como $p(X)$, estando no intervalo entre 0 e 1, então para cada valor de X, uma previsão pode ser feita por Y. Por exemplo, uma previsão para $Y = Sim$ sendo obtida para cada instância onde $p(X) > 0{,}5$.

Procurando representar um modelo em uma relação entre $p(X) = Pr(Y = 1|X)$ e X (convenientemente usa-se a codificação genérica 0/1 para a resposta), com um modelo de regressão linear representado por $p(X) = \beta_0 + \beta_1 X$, observado na Figura 5 (lado esquerdo), percebe-se alguns problemas como valores de X próximos de zero, produzindo probabilidades negativas, ou ainda, para valores mais altos de X, produzindo probabilidades maiores que 1.

Sempre que uma linha reta é ajustada a uma resposta binária codificada como 0 ou 1, em princípio pode-se predizer $p(X) < 0$ para alguns valores de X e $p(X) > 1$ para outros (a menos que o intervalo de X seja limitado).

Pra evitar este problema, modela-se $p(X)$ usando uma função que produz saídas entre 0 e 1 para todos valores de X. Muitas funções atendem essa descrição. Em regressão logística, utiliza-se a função logística.

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \tag{10}$$

Os coeficientes β_0 e β_1 em (10) são desconhecidos, e devem ser estimados baseados nos dados de treinamento disponíveis. James *et al*. (2013) apontam que o principal método para essa atividade é método da máxima verossimilhança (*maximum likelihood*) por apresentar melhores propriedades estatísticas. Para isso busca-se prever β_0 e β_1 de forma que a probabilidade prevista $\hat{p}(x_i)$ de X para cada instância usando o modelo (10) corresponda o mais próximo possível das instâncias observadas, de acordo com a função de verossimilhança:

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i\prime:y_{i\prime}=1} \left(1 - p(x_{i\prime})\right) \tag{11}$$

As estimativas $\hat{\beta}_0$ e $\hat{\beta}_1$ são escolhidas para maximizar esta função de verossimilhança. Os detalhes matemáticos da máxima verossimilhança estão além do escopo deste estudo, entretanto, regressão logística e outros modelos são facilmente ajustados por meio de *softwares* estatísticos.

Assim como em regressão linear múltipla, pode-se obter um modelo generalizado para regressão logística múltipla, onde $X = (X_1, \dots, X_p)$ são preditores, pode-se reescrever o modelo (10) como

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_P}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_P}} \tag{12}$$

e da mesma forma, usa-se o método da máxima verossimilhança para estimar $\beta_0, \beta_1, \dots, \beta_p$.

2.7 O TEOREMA DE BAYES

Conforme apresentado por James *et al*. (2013), para classificar uma observação de uma das classes K, onde $K \geq 2$, em outras palavras, a variável de resposta qualitativa Y pode assumir K valores possíveis, distintos e não ordenados. Seja π_k a representação da probabilidade global ou anterior de que uma observação escolhida aleatoriamente da $k_{ésima}$ classe. Essa é a probabilidade de que uma dada observação esteja associada à $k_{ésima}$ categoria da variável de resposta Y. Seja $f_k(X) \equiv \Pr(X = x|Y = k)$ a função de densidade de X para uma observação que vem da $k_{ésima}$ classe, em outras palavras, $f_k(x)$ é relativamente grande se houver uma alta probabilidade de que uma observação na $k_{ésima}$ classe tenha $X \approx x$, e $f_k(x)$ seja pequena se for muito improvável que uma observação na $k_{ésima}$ classe tenha $X \approx x$. Então, o teorema de Bayes afirma que

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \tag{13}$$

2.8 ANÁLISE DO DISCRIMINANTE LINEAR (LDA)

A análise do discriminante linear (LDA, *linear discriminant analysis*) é um método estatístico usado para encontrar uma combinação linear de recursos, que pode ser usada como um classificador linear. É frequentemente usado como uma etapa de redução de dimensionalidade antes de uma classificação complexa (CUESTA, 2013).

Nazmi *et al*. (2016) comparam LDA com outras técnicas de classificação, e apresentam bons resultados em suas comparações. Scheme e Englehart (2011) mencionam LDA como uma técnica popular para a classificação de eletromiografia, e apontam como principais vantagens a simplicidade de implementação (especialmente em sistemas embarcados) e a facilidade no treinamento.

Algumas vantagens do LDA com relação a regressão logística que merecem ser apontadas, como quando as classes são bem separadas, as estimativas dos parâmetros no modelo de regressão logística são instáveis, e o modelo LDA não sofre deste problema; também se n for pequeno e a distribuição dos preditores X for aproximadamente normal em cada uma das classes, o modelo LDA é novamente mais

estável do que o modelo de regressão logística; além do modelo LDA ser popular com mais de duas classes de resposta (JAMES *et al*., 2013).

Em linhas gerais, LDA assume que as observações dentro de cada classe são tiradas de uma distribuição gaussiana multivariada com um vetor médio específico da classe e uma matriz de covariância que é comum a todas as classes K (JAMES *et al*., 2013).

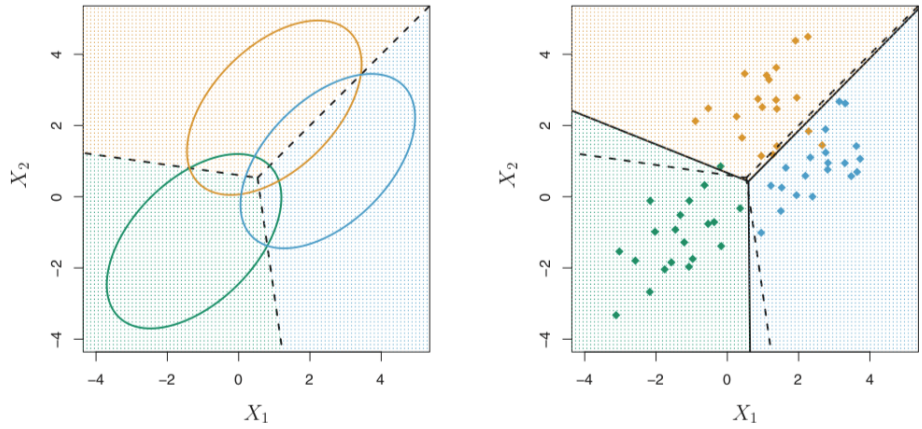
A função de estimativa do discriminante $\delta_k(x)$ é dada por

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^t\Sigma^{-1}\mu_k + \log \pi_k \quad (14)$$

considerando p o número total de classes, onde $k = 1 \dots p$, x o novo vetor de características a ser classificado, π_k a probabilidade anterior estimada da classe k, μ_k a média estimada da classe k e Σ^{-1} a matriz inversa de covariância agrupada estimada.

A Figura 6 apresenta um exemplo de classificação LDA com três classes, onde as observações de cada classe são tiradas de uma distribuição Gaussiana multivariada com $p = 2$, um vetor médio específico da classe e uma matriz de covariância comum. Do lado esquerdo, a elipses apresentam 95% de probabilidade para cada classe, e as linhas tracejadas os limites de decisão de Bayes. Na direita, 20 observações foram geradas para cada classe, e os limites de decisão LDA correspondentes são apresentados com as linhas sólidas; os limites de decisão de Bayes são novamente apresentados por linhas tracejadas.

Figura 6 – Exemplo de LDA com três classes



Fonte: (JAMES *et al*., 2013)

2.9 ANÁLISE DO DISCRIMINANTE QUADRÁTICO (QDA)

Assim como LDA, a análise do discriminante quadrático (QDA, *quadratic discriminant analysis*) resulta da suposição de que as observações de cada classe são extraídas de uma distribuição gaussiana e de inferir estimativas para os parâmetros no teorema de Bayes, a fim de realizar a previsão. No entanto, ao contrário do LDA, o QDA assume que cada classe tem sua própria matriz de covariância (JAMES *et al*., 2013).

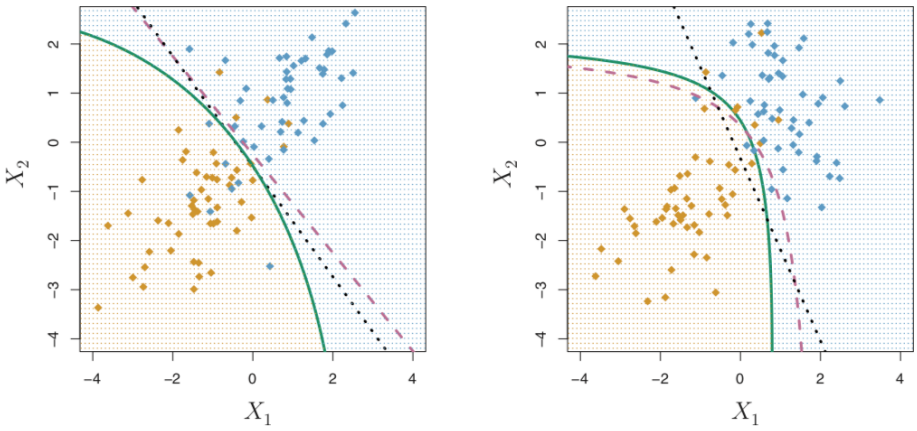
A função de estimativa do discriminante $\delta_k(x)$ é dada por

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\Sigma_k| + \log\pi_k \\ &= -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\log|\Sigma_k| + \log\pi_k\end{aligned} \tag{15}$$

onde k e x são como em (14), e $\Sigma_k(\Sigma_k^{-1})$ é a matriz de covariância estimada (inversa) da classe k.

A Figura 7 ilustra os desempenhos de LDA e QDA em dois cenários. À esquerda, apresenta-se Bayes tracejado, LDA pontilhado, e QDA sólido com limite de decisão para um problema com duas classes com $\Sigma_1 = \Sigma_2$. Como os limites de decisão de Bayes são lineares é mais precisamente aproximado por LDA que por QDA. À direita, os detalhes são fornecidos no painel à esquerda, exceto que $\Sigma_1 \neq \Sigma_2$. Como o limite de decisão de Bayes é não-linear, é mais precisamente aproximado por QDA do que por LDA.

Figura 7 – Aplicações de LDA e QDA em dois cenários

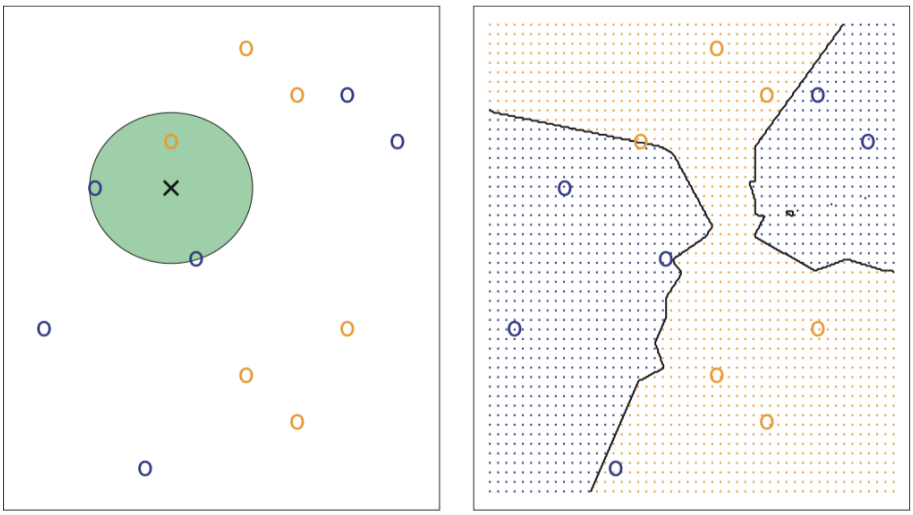


Fonte: (JAMES *et al*., 2013)

2.10 VIZINHOS MAIS PRÓXIMOS DE K (KNN)

KNN (*K-Nearest Neighbors*) é um dos poucos algoritmos que fazem previsões numéricas em funções complexas e ainda assim é fácil de interpretar. Nessa abordagem, toma-se um novo item para o qual deseja uma previsão numérica e o compara com um conjunto de itens para os quais ele já possui valores, encontrando os mais semelhantes ao item em questão e calculando a média de seus valores para obter um valor previsto (SEGARAN, 2007).

Figura 8 – Abordagem KNN, usando $K = 3$

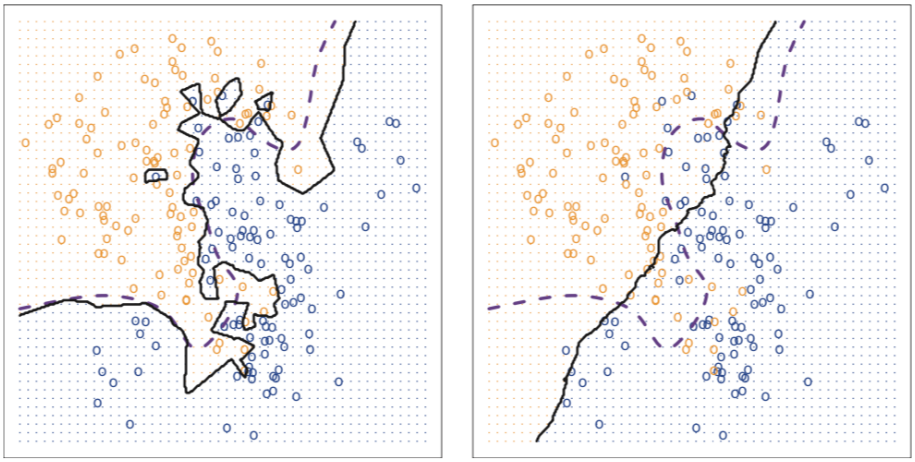


Fonte: (JAMES *et al*., 2013)

A Figura 8 ilustra a abordagem KNN, usando $K = 3$, em uma situação com seis observações azuis e seis laranjas. Na esquerda, uma observação teste em que uma classe previsível desejada é mostrada como uma cruz preta, os três pontos mais próximos são identificados, e é previsto que esta observação pertence a classe de maior ocorrência, neste caso, azul. Na direita, o limite de decisão KNN para este exemplo é mostrado em preto, a grade azul indica a região na qual uma observação de teste será atribuída à classe azul e a grade laranja indica a região na qual ela será atribuída à classe laranja.

James *et al*. (2013) apontam que a escolha de K tem um efeito drástico no classificador KNN obtido. A Figura 9 mostra dois ajustes de KNN para dados simulados, usando $K = 1$ e $K = 100$. À medida que K cresce, o método se torna menos flexível e produz um limite de decisão próximo de linear. Isso corresponde a um classificador de baixa variância, mas alto-viés. O limite de decisão de Bayes é exibido na linha tracejada.

Figura 9 – Ajustes do valor de K no classificador KNN



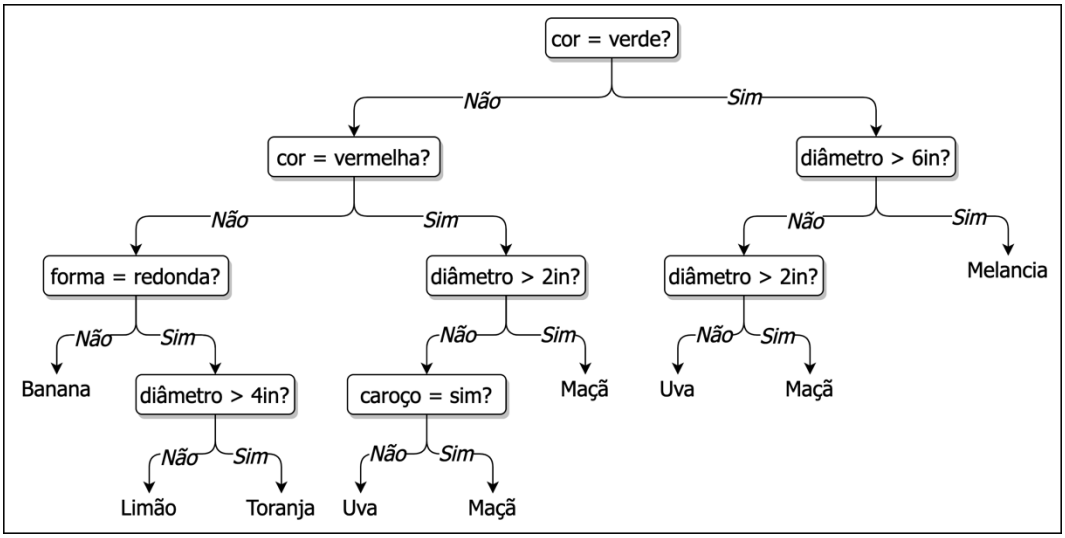
Fonte: (JAMES *et al*., 2013)

É importante notar que, como apontado por Segaran (2007), pelo fato de KNN se tratar uma técnica *online*, adicionar novos dados não requerem que estes sejam novamente treinados, entretanto sua maior fraqueza é requerer que todos os dados de treinamento estejam presentes para fazer previsões.

2.11 ÁRVORES DE DECISÃO

Segundo Segaran (2007), árvores de decisão (*decision trees* – DTREE) são um dos métodos mais simples de aprendizado de máquina. Trata-se de um método completamente transparente de classificar as observações, onde, após o treinamento, parecem uma série de declarações *if-then* organizadas em uma árvore. A Figura 10 mostra um exemplo de uma árvore de decisão para classificar frutas.

Figura 10 – Uma árvore de decisão exemplo



Fonte: Adaptado de Segaran, 2007

O processo de construção de uma árvore de regressão, a grosso modo, é constituído por dois passos:

1. Divide-se o espaço preditor – isto é, o conjunto de valores possíveis para $X_1, X_2, \ldots, X_p$ – em J regiões distintas e não sobrepostas, $R_1, R_2, \ldots, R_J$.
2. Para cada observação que cai na região R_j, faz-se a mesma previsão, que é simplesmente a média dos valores de resposta para as observações de treinamento em R_j.

James *et al*. (2013) apontam como principais vantagens das árvores de decisão com relação a abordagens mais clássicas:

- São fáceis de explicar para as pessoas;
- Algumas pessoas acreditam que árvores de decisão são próximas a decisões tomadas por pessoas;

- Podem ser exibidas graficamente, e facilmente interpretadas por leigos (especialmente se forem pequenas);
- Podem lidar facilmente com preditores qualitativos sem a necessidade de criar variáveis fictícias.

Como desvantagem, árvores de decisão geralmente não apresentam o mesmo nível de acurácia nas previsões como em outras abordagens. Por outro lado, essa acurácia pode ser substancialmente melhorada ao agregar outros métodos como *bagging*, *random forests*, e *boosting*.

2.12 FLORESTAS ALEATÓRIAS

Técnicas como ensacamento (*bagging*), florestas aleatórias (*random forests* – RF) e impulsionamento (*boosting*) usam árvores para construir modelos preditivos mais poderosos (JAMES *et al*., 2013).

Alguns estudos apontam bons resultados através da aplicação de florestas aleatórias comparando com outras técnicas de classificação em conjuntos de sinais EMG (GU *et al*., 2017; PIZZOLATO *et al*., 2017).

As florestas aleatórias ganharam popularidade nas aplicações de aprendizado de máquina durante a última década devido ao bom desempenho de classificação, escalabilidade e facilidade de uso. Intuitivamente, uma floresta aleatória pode ser considerada como um conjunto de árvores de decisão. A ideia por trás da aprendizagem conjunta é combinar aprendizes fracos para construir um modelo mais robusto, um aprendiz forte, que tenha um erro de generalização melhor e seja menos suscetível a sobreajuste (RASCHKA, 2016).

Nas florestas aleatórias constrói-se várias árvores de decisão em amostras de treinamento inicializadas. Ao construir essas árvores de decisão, cada vez que uma divisão em uma árvore é considerada, uma amostra aleatória de m preditores é escolhida como candidatos divididos do conjunto completo de preditores p. A divisão tem permissão para usar apenas um desses m preditores. Uma amostra nova de m preditores é obtida em cada divisão, e normalmente escolhe-se $m \approx \sqrt{p}$ – ou seja, o número de preditores considerados em cada divisão é aproximadamente igual à raiz quadrada do número total de preditores.

2.13 MÁQUINAS DE VETORES DE SUPORTE (SVM)

A fundamentação teórica do SVM está no trabalho de Vladimir Vapnik e na teoria da aprendizagem estatística desenvolvida na década de 1970 (CUESTA, 2013). Trata-se de uma abordagem para classificação que foi desenvolvida pela comunidade da ciência da computação nos anos 1990, e que tem crescido em popularidade desde então (JAMES *et al*., 2013).

Segundo Chowdhury *et al*. (2013), SVM está estre os classificadores mais populares para sinais EMG, apresentando bons resultados, entretanto, em alguns casos, apresenta baixa velocidade no treinamento.

Os SVMs são altamente usados no reconhecimento de padrões de séries temporais, bioinformática, processamento de linguagem natural e visão computacional (CUESTA, 2013).

Para teorizar o funcionamento das SVM, supõe-se o produto interno de dois r-vetores a e b, definido por $\langle a, b\rangle = \sum_{i=1}^{r} a_i b_i$. Assim, o produto interno de duas observações $x_i, x_{i'}$ é dado por

$$\langle x_i, x_{i'}\rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{16}$$

e um classificador de vetor de suporte linear representado como

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i\rangle \tag{17}$$

onde os n parâmetros $\alpha_i, i = 1, \ldots, n$, um por observação de treinamento.

Para estimar os parâmetros α_i e β_0 utiliza-se dos produtos internos $\langle x_i, x_{i'}\rangle$ de $\binom{n}{2}$ entre todos os pares de observações de treinamento. Nesse caso, se a observação de treinamento não for um vetor de suporte, então α_i é igual a zero. Então se $\mathcal{S}$ é a coleção de índices dos pontos de suporte, pode-se escrever a função (17) na forma

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i\rangle \tag{18}$$

que tipicamente envolve menos termos.

Agora, supondo que toda vez que o produto interno (16) apareça na representação (17), ou em um cálculo para a solução do classificador do vetor de suporte, substitui-se com uma generalização do produto interno na forma

$$K(x_i, x_{i'}) \tag{19}$$

onde K é alguma função que será referenciada como *kernel* (núcleo). O *kernel* é uma função que quantifica a similaridade de duas observações.

A partir disso, obtém-se a equação *kernel linear*

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{20}$$

A equação *kernel polinomial de grau* d

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d \tag{21}$$

E ainda, a equação *kernel radial*

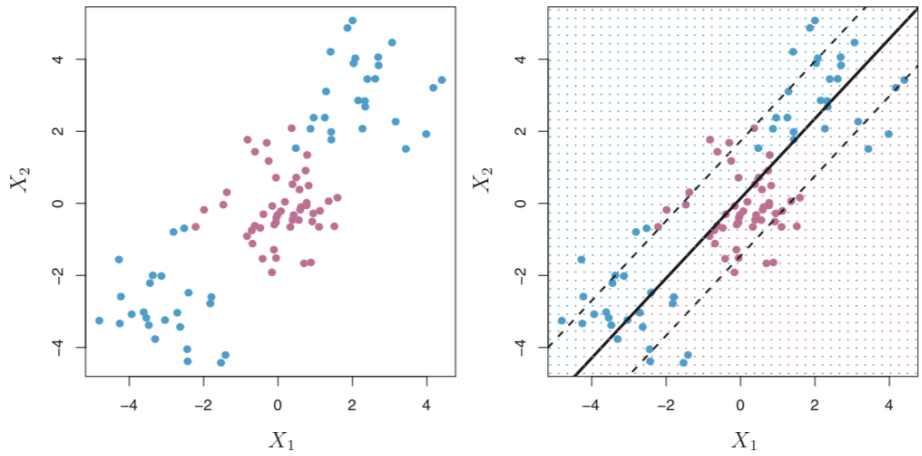
$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2\right) \tag{22}$$

Quando o classificador do vetor de suporte é combinado com um *kernel* não linear, como em (21) e (22) o classificador resultante é conhecido como máquina de vetor de suporte (JAMES *et al*., 2013). Neste caso, a função tem a forma

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i) \tag{23}$$

A Figura 11 apresenta à esquerda, observações que caíram em duas classes com limite não linear entre elas. À direita o classificador de vetor de suporte encontra um limite linear, e consequentemente, funciona muito mal.

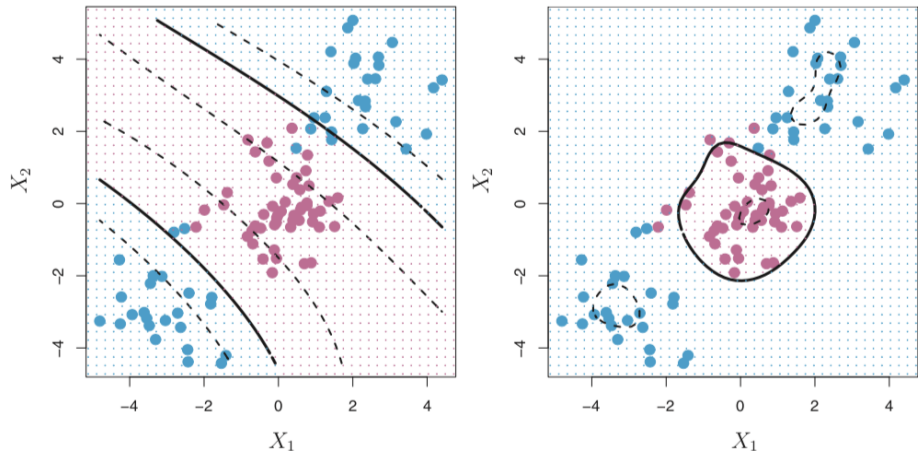
Figura 11 – Classificador de vetor de suporte linear em classes não lineares



Fonte: (JAMES *et al*., 2013)

À esquerda da Figura 12, um SVM com *kernel* polinomial de grau três é aplicado aos dados da Figura 11, resultando em uma regra de decisão muito mais apropriada. À direita, é aplicado um SVM com *kernel* radial. Neste exemplo, o *kernel* é capaz de capturar o limite de decisão.

Figura 12 – SVM com *kernels* polinomial de grau 3 e radial



Fonte: (JAMES *et al*., 2013)

2.14 REDES NEURAIS ARTIFICIAIS (ANN)

Embora as redes neurais artificiais tenham ganhado muita popularidade nos últimos anos, os primeiros estudos de redes neurais remontam aos anos 1940, quando Warren McCulloch e Walter Pitt descreveram pela primeira vez como os

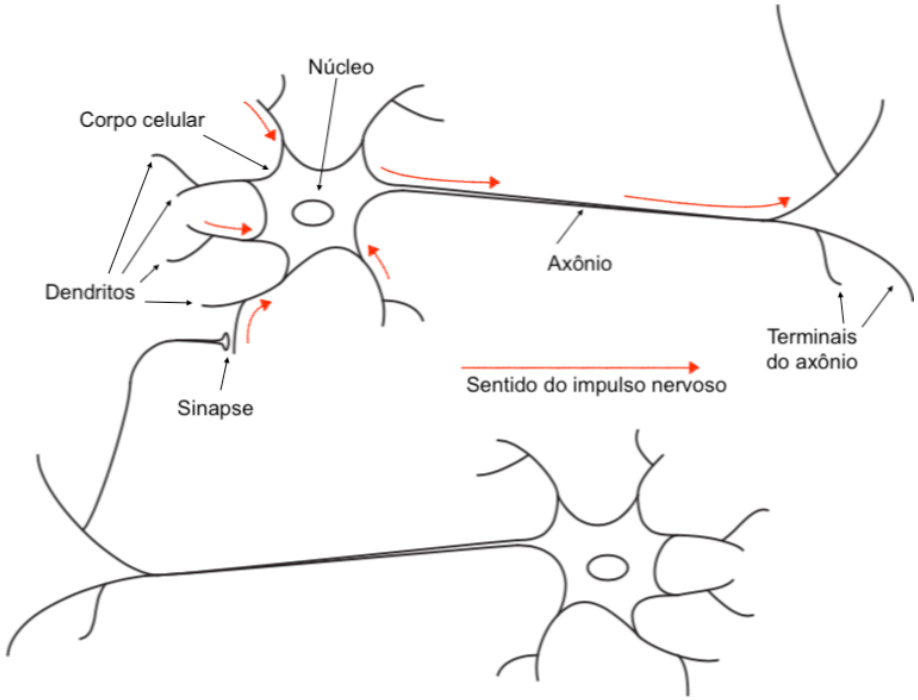
neurônios podiam funcionar. No entanto, nas décadas que se seguiram à primeira implementação do modelo de neurônios *McCulloch-Pitt*, o *perceptron* de Rosenblatt nos anos 50, muitos pesquisadores e praticantes de aprendizado de máquina lentamente começaram a perder o interesse em redes neurais, já que ninguém tinha uma boa solução para treinar uma rede neural com várias camadas. Eventualmente, o interesse em redes neurais foi reacendido em 1986, quando D.E. Rumelhart, G.E. Hinton e R.J. Williams envolveram-se na (re)descoberta e popularização do algoritmo de retropropagação (*backpropagation*) para treinar redes neurais de forma mais eficiente (RASCHKA, 2016).

2.14.1 Neurônios

De um ponto de vista biológico, o estudo de redes neurais é sobre o modelo matemático para as operações do cérebro. Os elementos aritméticos correspondem aos neurônios, e a rede como um todo corresponde a uma coleção de neurônios interconectados. Por esse motivo, as redes são chamadas de redes neurais (RUSSELL; NORVIG, 1995).

Sabe-se que um neurônio, uma célula nervosa, é a unidade funcional fundamental de todo o tecido do sistema nervoso, incluindo o cérebro (RUSSELL; NORVIG, 1995). Esses neurônios possuem três componentes principais: os dendritos, o corpo celular e o axônio. Os dendritos são redes receptivas, semelhantes a árvores, de fibras nervosas que transportam sinais elétricos para o corpo celular. O corpo da célula efetivamente soma e limita esses sinais de entrada. O axônio é uma única fibra longa que transporta o sinal do corpo celular para outros neurônios. E o ponto de contato entre um axônio de uma célula e um dendrito de outra célula é chamado de sinapse. É o arranjo dos neurônios e as forças das sinapses individuais, determinadas por um processo químico complexo, que estabelece a função da rede neural (HAGAN *et al*., 2014). A Figura 13 representa um diagrama simplificado de dois neurônios biológicos.

Figura 13 – Representação de dois neurônios biológicos

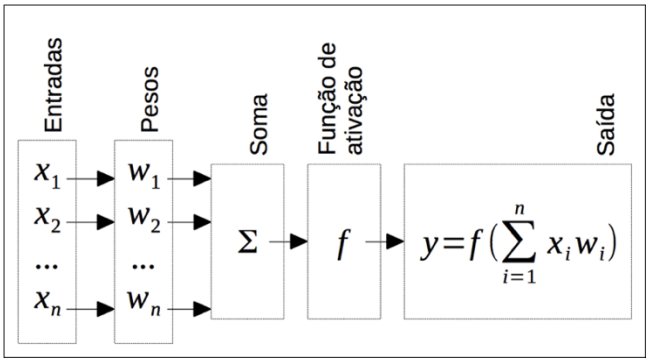


Fonte: Adaptado de Hagan *et al*. (2014)

Redes neurais artificiais não se aproximam da complexidade do cérebro biológico. Existem, no entanto, duas semelhanças importantes entre redes neurais biológicas e artificiais. Primeiro, os blocos de construção de ambas as redes são dispositivos computacionais simples (embora os neurônios artificiais sejam muito mais simples que os neurônios biológicos), altamente interconectados. Em segundo lugar, as conexões entre os neurônios determinam a função da rede (HAGAN *et al*., 2014).

A Figura 14 representa de forma simplificada um neurônio artificial. Em sua arquitetura, um único neurônio tem um número n de entradas x_i onde $(i = 1, \dots, n)$, e uma saída y. Associado a cada entrada há um peso w_i, podendo haver um parâmetro adicional w_0 do neurônio chamado *bias* (viés), que pode ser visto como o peso associado a uma entrada x_0, que é permanentemente definida como 1. Um único neurônio é um mecanismo *feedforward* – as conexões são direcionadas das entradas para a saída do neurônio (MACKAY, 2005).

Figura 14 – Representação simplificada de um neurônio artificial



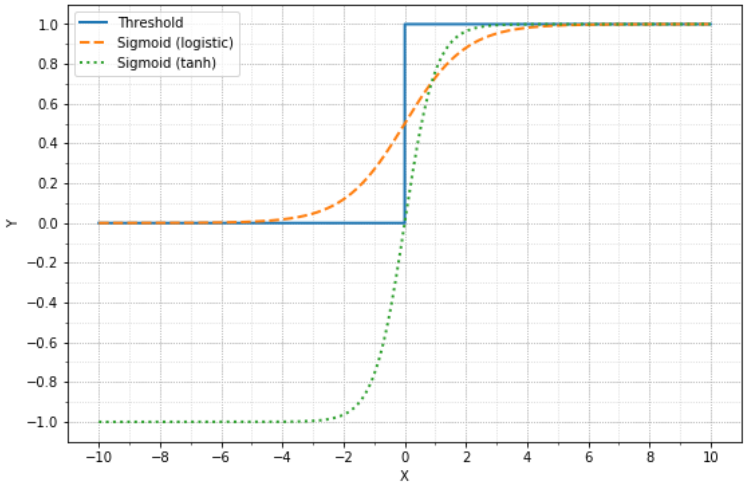
Fonte: Elaboração do próprio autor

A regra de ativação é realizada em duas etapas: primeiro, em resposta às entradas impostas x, calcula-se a ativação do neurônio,

$$z = \sum_i x_i w_i \tag{24}$$

onde a soma é sobre $i = 0, \ldots, n$ se houver um viés e $i = 1, \ldots, n$ caso contrário. Em segundo lugar, a saída y é definida como uma função $f(z)$ da ativação. A saída também é conhecida como atividade do neurônio, e não deve ser confundida com a ativação z. Existem várias funções de ativação, algumas das mais comuns são apresentadas graficamente conforme a Figura 15.

Figura 15 – Funções de ativação *threshold*, logística e tangente hiperbólica



Fonte: Elaboração do próprio autor

ou na forma de equações, conforme (25), (26) e (27).

Sigmoide
(Logística)

$$y(z) = \frac{1}{1 + e^{-z}} \tag{25}$$

Sigmoide simétrica
(Tangente hiperbólica)

$$y(z) = \tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{26}$$

Threshold
(limite rígido)

$$y(z) = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases} \tag{27}$$

2.14.2 Projetos de redes neurais artificiais

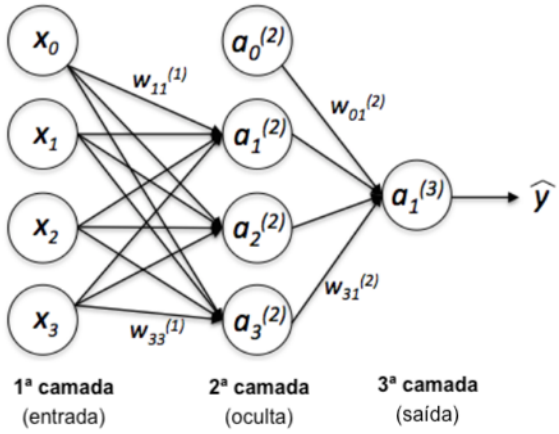
Uma rede neural é composta de um número de nós, ou unidades, conectados por links. Cada link tem um peso numérico associado a ele. Os pesos são o principal meio de armazenamento em redes neurais, e o aprendizado geralmente ocorre com a atualização os pesos. Algumas das unidades estão conectadas ao ambiente externo e podem ser designadas como unidades de entrada ou saída (RUSSELL; NORVIG, 1995).

Redes neurais são adequadas para modelar dados não lineares e são capazes de cobrir distinções entre diferentes condições. Os requisitos para projetar uma ANN para uma dada aplicação incluem: primeiro, determinar a arquitetura da rede; segundo, definir o número total de camadas, o número de unidades ocultas nas camadas intermediárias e o número de unidades nas camadas de entrada e saída; e terceiro, o algoritmo de treinamento durante a fase de aprendizado (CHOWDHURY *et al.*, 2013).

Atualmente existem várias arquiteturas de redes neuras. Neste trabalho será discutido a arquitetura de redes neurais conhecida como *Multi-layer Perceptron* (MLP). A Figura 16 explica o conceito de uma ANN MLP que consiste em três camadas: uma camada de entrada, uma camada oculta e uma camada de saída. As unidades na camada oculta são totalmente conectadas à camada de entrada e a camada de saída é totalmente conectada à camada oculta, respectivamente. Se essa rede tiver mais de uma camada oculta, pode-se chama-la de rede neural artificial profunda. Conforme

o modelo apresentado, denota-se a i-ésima unidade de ativação na l-ésima camada como $a_i^{(l)}$ e as unidade de ativação $a_0^{(1)}$ e $a_0^{(2)}$ são as unidades de *bias*, respectivamente, com valores iguais a 1.

Figura 16 – *Multi-layer* perceptron de três camadas



Fonte: Adaptado de Raschka (2016)

Pode-se adicionar um número arbitrário de camadas ao MLP para criar arquiteturas mais profundas. Na prática, pode-se pensar no número de camadas e unidades de uma ANN como *hiperparâmetros* adicionais que podem ser otimizados para um dado problema com o uso de uma técnica conhecida como *cross-validation*. No entanto, os gradientes de erro calculados por *backpropagation* se tornarão cada vez menores à medida que mais camadas forem adicionadas a uma rede. Esse problema torna o aprendizado do modelo mais desafiador. Por isso, algoritmos especiais foram desenvolvidos para testar essas estruturas de ANN profundas, o que é conhecido por *deep learning* (RASCHKA, 2016).

2.14.3 *Treinamento de rede neural com backpropagation*

Segundo Raschka (2016), pode-se resumir o procedimento de aprendizado do MLP em três etapas:

1. Na camada de entrada, encaminhar a propagação dos padrões dos dados de treinamento através da rede para gerar uma saída.
2. Com base na saída da rede, calcular o erro que se pretende minimizar usando uma função de custo.

3. Retropropagar o erro, calcular sua derivada em relação a cada peso na rede e atualizar o modelo.

Finalmente, após repetir essas etapas por várias vezes e aprender os pesos do MLP, usa-se a *forward propagation* para calcular a saída da rede e aplicar uma função *threshold* para obter a classe prevista.

Para a realização do treinamento com *backpropagation*, primeiramente aplica-se *forward propagation* para obter a ativação da camada de saída, formulada a seguir:

Entrada da camada oculta

$$\boldsymbol{Z}^{(2)} = \boldsymbol{W}^{(1)}\left[\boldsymbol{A}^{(1)}\right]^{T} \tag{28}$$

Ativação da camada oculta

$$\boldsymbol{A}^{(2)} = \phi\left(\boldsymbol{Z}^{(2)}\right) \tag{29}$$

Entrada da camada de saída

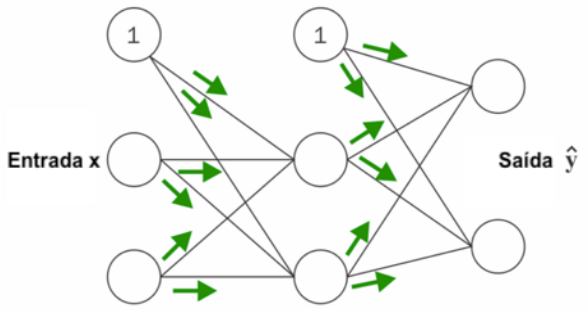
$$\boldsymbol{Z}^{(3)} = \boldsymbol{W}^{(2)}\boldsymbol{A}^{(2)} \tag{30}$$

Ativação da camada de saída

$$\boldsymbol{A}^{(3)} = \phi\left(\boldsymbol{Z}^{(3)}\right) \tag{31}$$

A Figura 17 ilustra o processo *forward* propagation, onde as características de entrada são propagadas através das conexões da rede.

Figura 17 – *Forward propagation*



Fonte: Adaptado de Raschka (2016)

No *backpropagation*, propaga-se o erro da direita para a esquerda, calculando o vetor de erros da camada de saída:

$$\delta^{(3)} = a^{(3)} - y \tag{32}$$

onde y é o vetor das classes verdadeiras. Em seguida, calcula-se o termo de erro da camada oculta:

$$\delta^{(2)} = \left(\boldsymbol{W}^{(2)}\right)^T \delta^{(3)} \frac{\partial \phi\left(z^{(2)}\right)}{\partial z^{(2)}} \tag{33}$$

onde o termo $\frac{\partial \phi\left(z^{(2)}\right)}{\partial z^{(2)}}$ é a derivada parcial da função de ativação Sigmoide:

$$\frac{\partial \phi\left(z^{(2)}\right)}{\partial z^{(2)}} = \left(a^{(2)}\left(1 - a^{(2)}\right)\right) \tag{34}$$

Em seguida, acumula-se a derivada parcial para cada j-ésimo nó na camada l e o i-ésimo erro do nó na camada $l + 1$:

$$\Delta_{i,j}^{(l)} = \Delta_{i,j}^{(l)} + a_j^{(l)} \delta_i^{(l+1)} \tag{35}$$

Como é necessário calcular $\Delta_{i,j}^{(l)}$ para cada amostra do conjunto de treinamento, torna-se mais fácil implementar uma versão vetorizada na forma:

$$\Delta^{(l)} = \Delta^{(l)} + \delta^{(l+1)}\left(\boldsymbol{A}^{(l)}\right)^T \tag{36}$$

Após acumuladas as derivadas parciais, adiciona-se o termo de regularização, exceto para o termo *bias*, na forma:

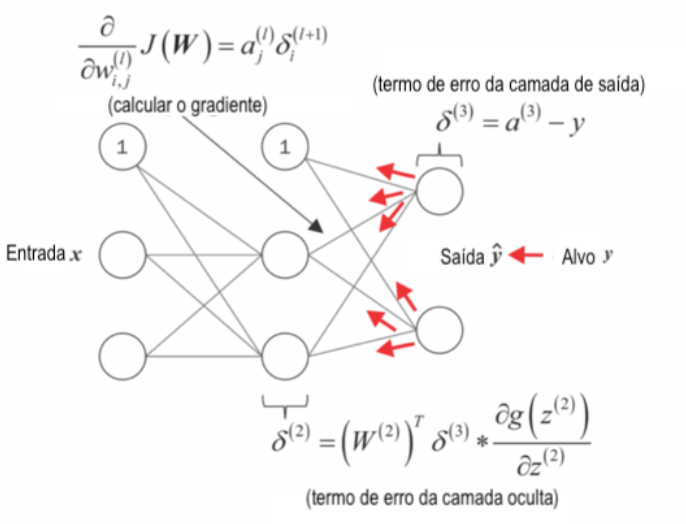
$$\Delta^{(l)} = \Delta^{(l)} + \lambda^{(l)} \tag{37}$$

e finalmente, após calcular os gradientes, atualiza-se os pesos dando um passo em direção oposta ao gradiente:

$$\boldsymbol{W}^{(l)} = \boldsymbol{W}^{(l)} - \eta \Delta^{(l)} \tag{38}$$

Observa-se na Figura 18 a sumarização do processo de *backpropagation*.

Figura 18 – O processo de *backpropagation*



Fonte: Adaptado de Raschka (2016)

Uma desvantagem das ANN é que não há regras definitivas para a escolha da taxa de treinamento e tamanho da rede para um problema específico. Esta decisão geralmente requer uma boa quantidade de experimentação. A escolha de uma taxa de treinamento muito alta significa que a rede pode generalizar excessivamente em dados ruidosos, e escolher uma que seja muito baixa significa que ela talvez nunca aprenda (SEGARAN, 2007).

3 MATERIAIS E MÉTODOS

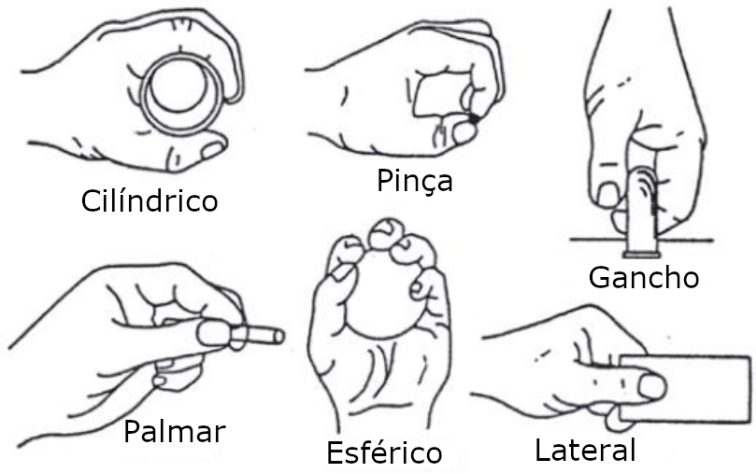
3.1 LEVANTAMENTO BIBLIOGRÁFICO

A pesquisa bibliográfica para o levantamento do estado da arte foi realizada nas bases Web of Science, Scopus, Science Direct, IEEE Xplorer, PUBMED, BIOMED e IOP a partir do mecanismo de busca Google Acadêmico. Escolheu-se o idioma inglês, e os critérios de pesquisa foram classificação, eletromiografia de superfície e prótese de membro superior. Foram usados como critérios de exclusão publicações relacionadas com membros inferiores, eletroencefalografia, eletrocardiografia, miografia de força, sinais intramusculares ou intraneurais, exoesqueletos, plataformas virtuais, e demais trabalhos cujos escopos divergem deste estudo.

3.2 CONJUNTO DE DADOS PARA EXPERIMENTOS

Neste estudo utilizou-se de uma base de dados publicada no *UCI Machine Learning Repository* (DHEERU; KARRA TANISKIDOU, 2017) descrita por Sapsanis, Georgoulas e Tzes (2013). Trata-se de 900 instâncias obtidas a partir de 5 indivíduos saudáveis (dois do gênero masculino e 3 do gênero feminino) com faixa etária entre 20 e 22 anos de idade, conduzindo 6 movimentos de preensão manual, sendo eles, Cilíndrico, para segurar objetos cilíndricos, Pinça, para segurar objetos pequenos, Gancho, para suportar carga pesada, Palmar, para agarrar com a palma voltada para o objeto, Esférico, para segurar objetos esféricos, e Lateral, para segurar objetos finos e planos, conforme apresentado na Figura 19, por 30 vezes cada um, com tempo de medição de 6 segundos. A velocidade e a força das preensões foram deixadas intencionalmente à vontade dos indivíduos. Usou-se dois eletrodos sEMG no antebraço, Flexor Ulnar do Carpo e Extensor Radial do Carpo, Longo e Breve, fixados por cintas elásticas, e o eletrodo de referência no meio, para obter informações sobre a ativação muscular.

Figura 19 – Movimentos de preensão manual

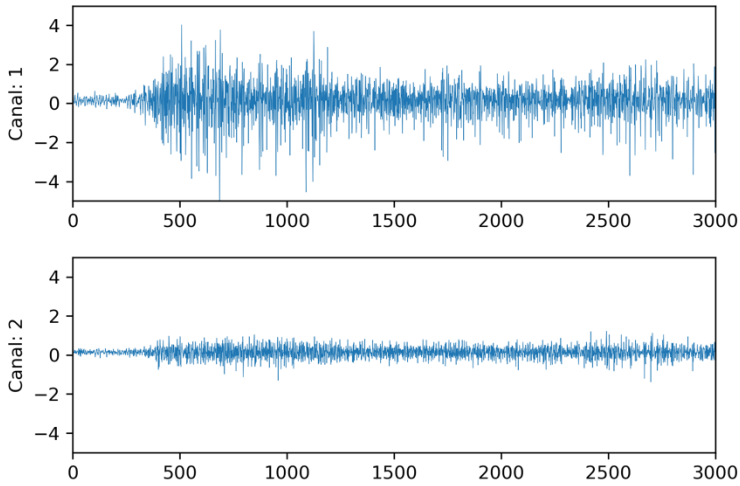


Fonte: Adaptado de Dheeru, Karra e Taniskidou (2017)

Os dados foram coletados a uma taxa de amostragem de 500 Hz, utilizando-se o *National Instruments Labview*. Os sinais passaram por um filtro passa-banda *Butterworth Band Pass* com corte baixo e alto a 15 Hz e 500 Hz, respectivamente, e um filtro *notch* a 50 Hz para eliminar interferência de linha. O equipamento utilizado foi uma placa de conversão analógica/digital *NI USB-009* montada em um microcomputador. O sinal foi obtido por dois sensores diferenciais sEMG e transmitidos para um sistema sEMG de 2 canais *Delsys Bagnoli™ Handheld EMG Systems* (SAPSANIS; GEORGOULAS; TZES, 2013).

O conjunto de dados foi disponibilizado em arquivos do *MATLAB* no formato ".mat" (MATLAB, 2018), de maneira que as amostras correspondentes aos cinco indivíduos se encontram em arquivos separados; um arquivo por indivíduo. A Figura 20 representa graficamente uma instância aleatória do sinal original.

Figura 20 – Representação gráfica de uma instância do sinal original



Fonte: Elaboração do próprio autor

Cada um dos arquivos possui, além dos meta-dados, 12 matrizes, que correspondem aos dados obtidos pelos 2 canais durante cada um dos 6 movimentos de preensão.

Cada uma das matrizes possui 30 linhas, que correspondem as repetições realizadas para cada movimento de preensão; e cada linha possui 3000 colunas, que correspondem aos 6 segundos de leitura a 500 Hz.

3.3 FERRAMENTAS DE SOFTWARE

Neste trabalho, os experimentos foram realizados em um computador com processador Intel Core i5-3210M 2,50 GHz, e 16 GB de memória RAM DDR3 1600 MHz.

Como conjunto de *software*, optou-se pela utilização de ferramentas abertas (*open source*) principalmente para evitar custos com licenças, além de questões como portabilidade, interoperabilidade e documentação. Nesse sentido, utilizou-se a linguagem de programação *Python* (VAN ROSSUM; DRAKE, 2009), que tem se mostrado bastante popular em publicações científicas, principalmente no campo de análise e classificação de dados. Por conveniência, utilizou-se a distribuição de *software Anaconda* (ANACONDA, 2020), que fornece um amplo conjunto de *software*

para computação científica, fornecendo, dentre outras, a linguagem de programação Python, bem como os módulos e ferramentas gerais.

Os módulos Python usados para processamento numérico, análise de dados e visualização de dados foram *NumPy* (OLIPHANT, 2006), *SciPy* (HUNTER, 2007), *Matplotlib* (JONES *et al*., 2001) e *Pandas* (MCKINNEY, 2010); o módulo utilizado para a criação dos modelos de aprendizagem de máquina e previsão foi o *scikit-learn* (PEDREGOSA *et al*., 2011), e para o pré-processamento de sinais foi utilizado o módulo *PyWavelets* (LEE *et al*., 2006).

3.4 EXPERIMENTOS

Para realizar os experimentos, a primeira tarefa realizada foi a de formatação do conjunto dos dados originais, de forma que pudessem ser utilizados como amostras para o treinamento e os testes dos modelos.

3.4.1 Formatação do conjunto de dados

Conforme apontado em 3.2, o conjunto de dados foi disponibilizado em arquivos separados, e em matrizes separadas nesses arquivos.

Nessa tarefa, todas as linhas de todas as matrizes de todos os arquivos, foram reunidas em uma única matriz, e uma coluna foi adicionada ao final, para identificar a classe (movimento de preensão) a qual cada uma das instâncias (linhas) pertence.

A matriz resultante possui 900 linhas (5 indivíduos $\times$ 6 classes $\times$ 30 repetições) por 6001 colunas (6 segundos $\times$ 500Hz $\times$ 2 canais $+$ 1 classe), e foi armazenada em um arquivo para uso posterior.

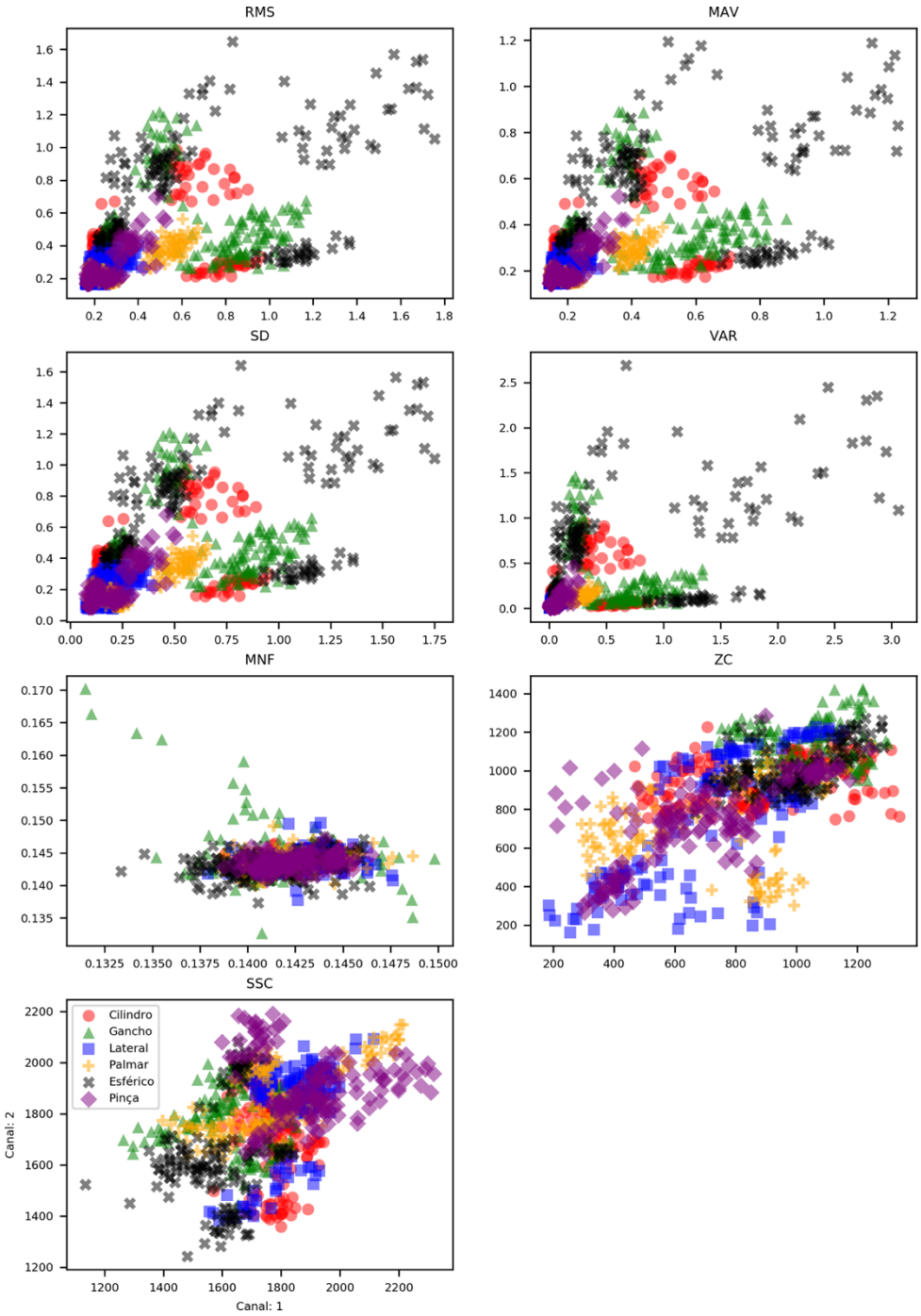
3.4.2 Extração de características

Inicialmente, os dados foram analisados sem nenhum tipo de pré-processamento. Nessas circunstâncias, em testes preliminares, percebeu-se baixa acurácia média dos resultados com todos os classificadores analisados.

Para produzir melhores resultados, conforme apresentado por Jahan *et al*. (2015) foram extraídas: raiz quadrada média (RMS), média do valor absoluto (MAV),

desvio padrão (SD), variância (VAR), frequência média (MNF), contagem de passagens em zero (ZC) e mudança do sinal da inclinação (SSC), e produziu-se um gráfico de dispersão (Figura 21), que ilustra o resultado de cada uma das características aplicadas em cada canal de dados, separando as classes (preensões) esperadas por forma e cor.

Figura 21 – Gráficos de dispersão das características do sinal



Fonte: Elaboração do próprio autor

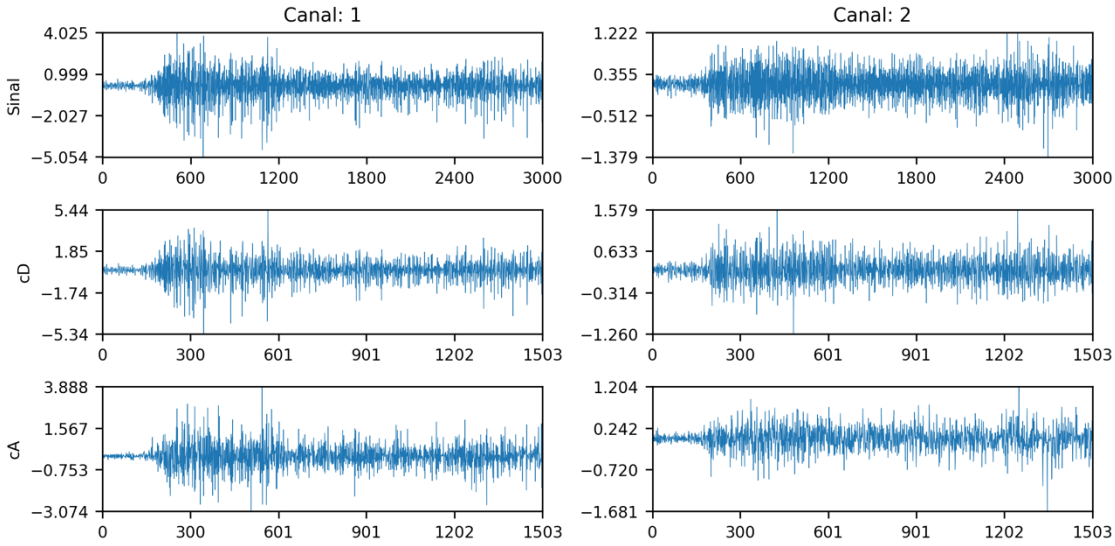
Percebe-se que as características RMS, MAV, SD e VAR apresentam valores visualmente semelhantes, e que a característica MNF apresenta pouca variação entre as classes de dados. Por estes motivos, decidiu-se experimentar uma nova configuração de características, mantendo-se apenas RMS, ZC e SSC, por apresentarem configurações distintas de valores.

3.4.3 Pré-processamento dos sinais

Realizou-se novos testes através de duas configurações de pré-processamento, as DWT através da função *dwt()*, e a decomposição *wavelet* multinível através da função *wavedec()* do módulo *PyWavelets* (LEE *et al*., 2006).

A Figura 22 ilustra uma instância do sinal obtido pelas DWT da família Daubechies com um nível de decomposição, e os coeficientes de detalhes (cD) e de aproximação (cA).

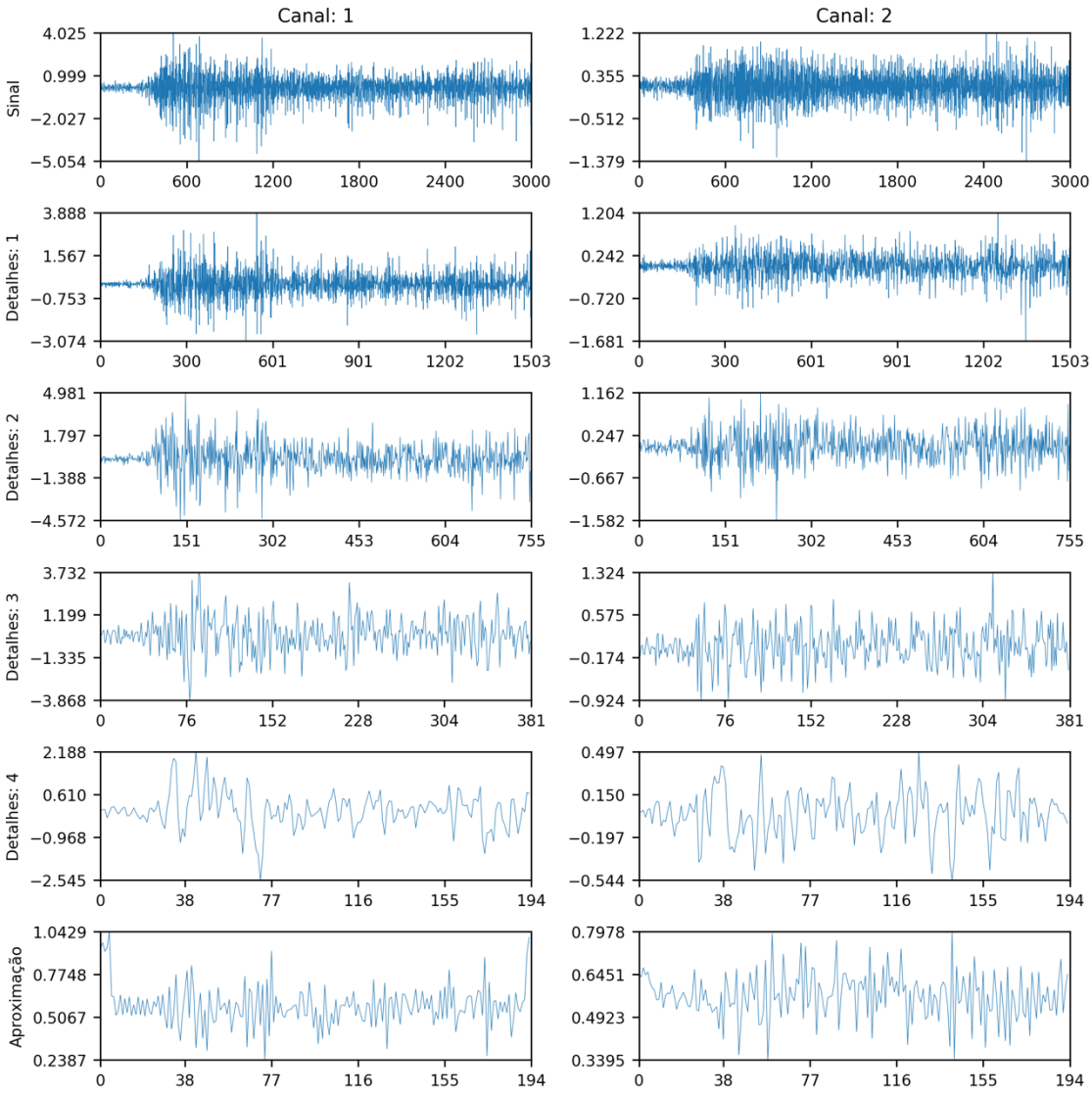
Figura 22 – Representação gráfica da DWT



Fonte: Elaboração do próprio autor

Examinou-se também a decomposição wavelet multinível com 4 níveis de decomposição, conforme ilustrado na Figura 23.

Figura 23 – Representação gráfica da decomposição *wavelet* multinivel

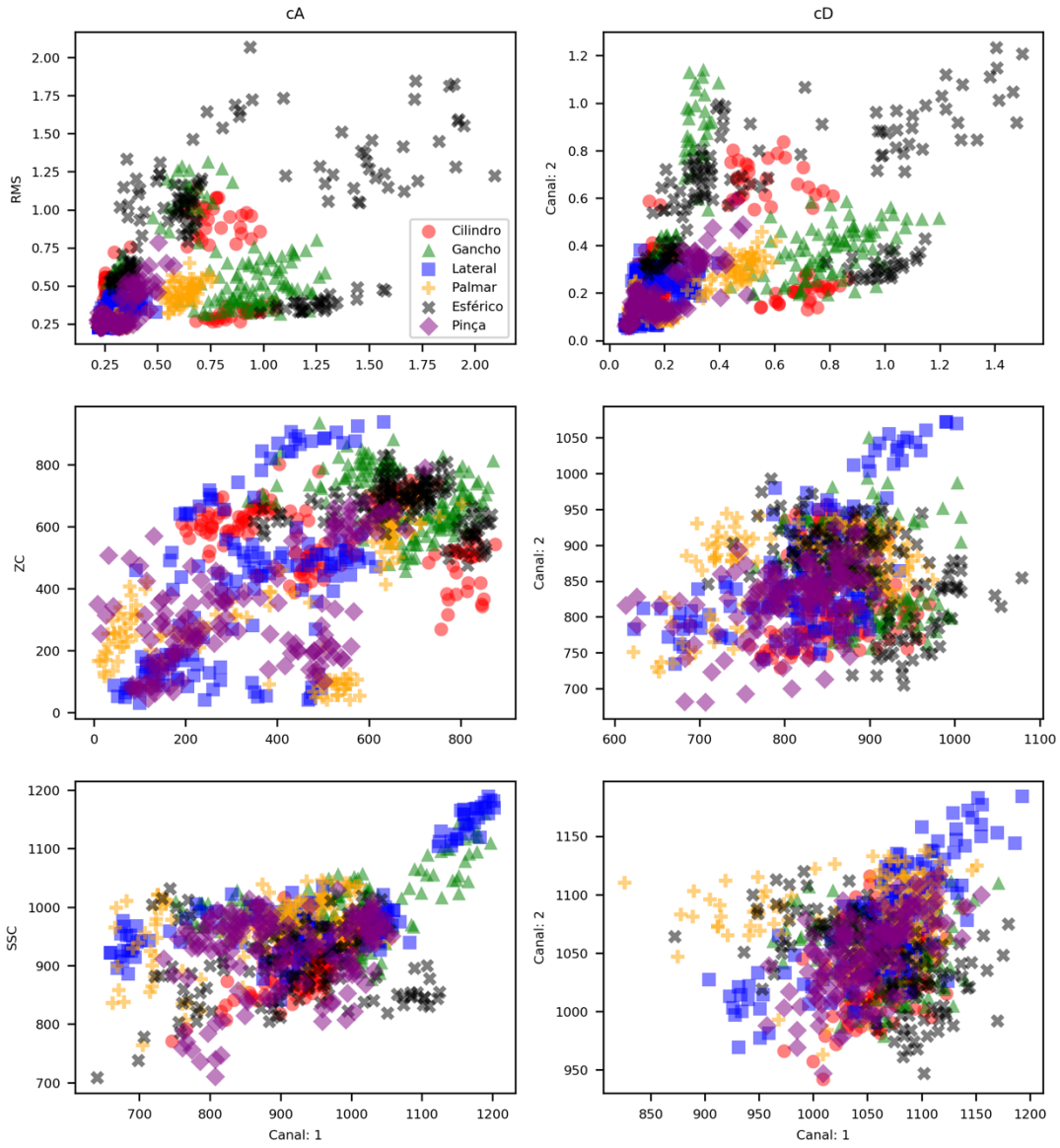


Fonte: Elaboração do próprio autor

A relação entre as características e as classes produzidas pela DWT pode ser observada nos gráficos de dispersão da Figura 24. A primeira coluna ilustra os valores do coeficiente de aproximação e a segunda os valores do coeficiente de detalhes. Cada uma das três linhas representa uma das características RMS, ZC e SSC.

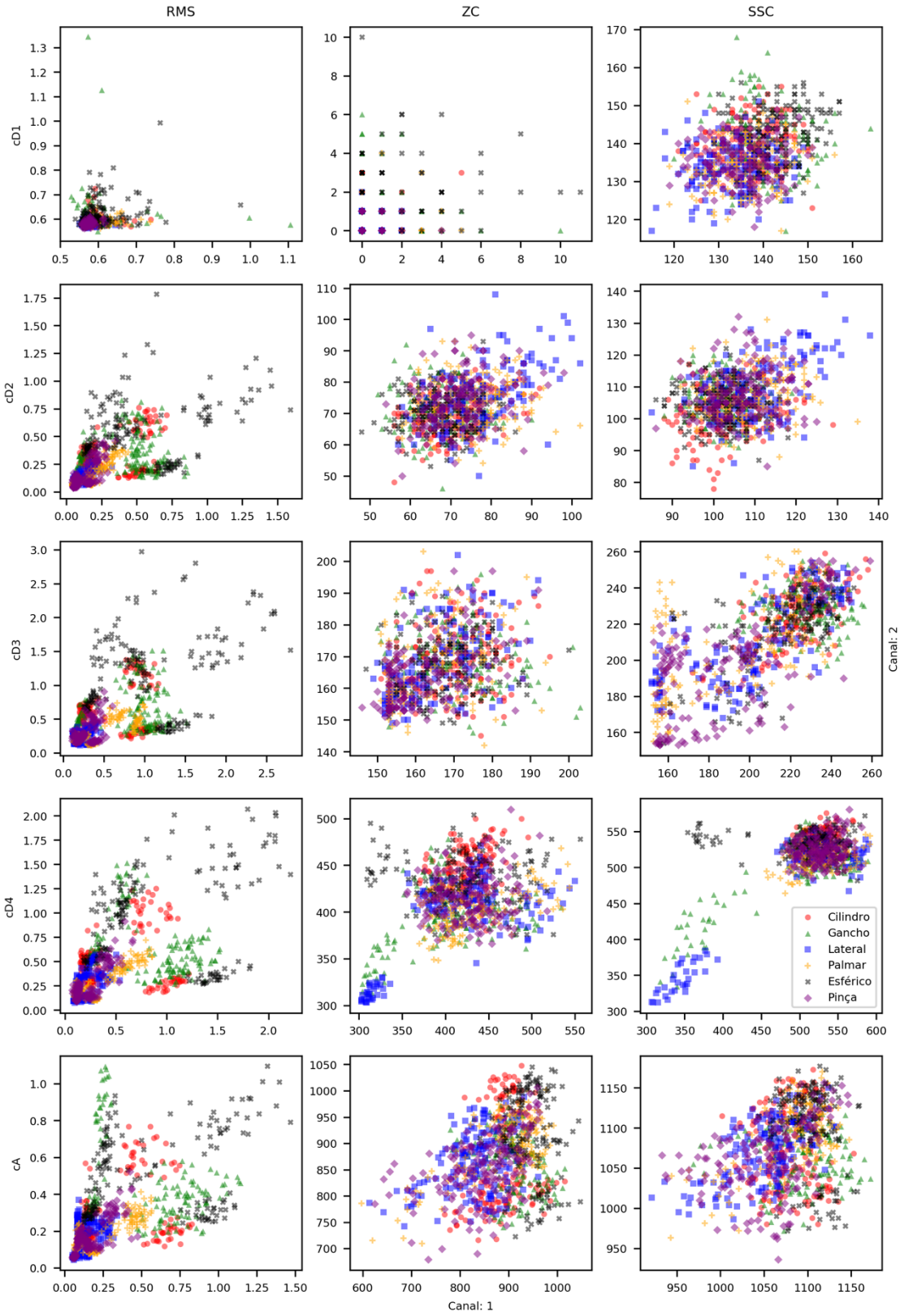
A relação entre as características e as classes produzidas pela decomposição *wavelet* da família Daubechies nível 4 é ilustrada nos gráficos de dispersão da Figura 25. As três colunas representam as características RMS, ZC e SSC, e cada uma das 5 linhas representa um dos coeficientes.

Figura 24 – Gráficos de dispersão das características da DWT



Fonte: Elaboração do próprio autor

Figura 25 – Gráficos de dispersão das características da decomposição *wavelet*



Fonte: Elaboração do próprio autor

3.4.4 Definição das amostras de treinamento e teste

As amostras para treinamento e testes foram geradas com a função *train_test_split()* do módulo *sklearn.model_selection* em subconjuntos aleatórios a partir dos dados originais com o atributo de semente fixado em zero para fins de reprodutibilidade. O tamanho das amostras foi definido em 80% das instâncias para treinamento e 20% para os testes.

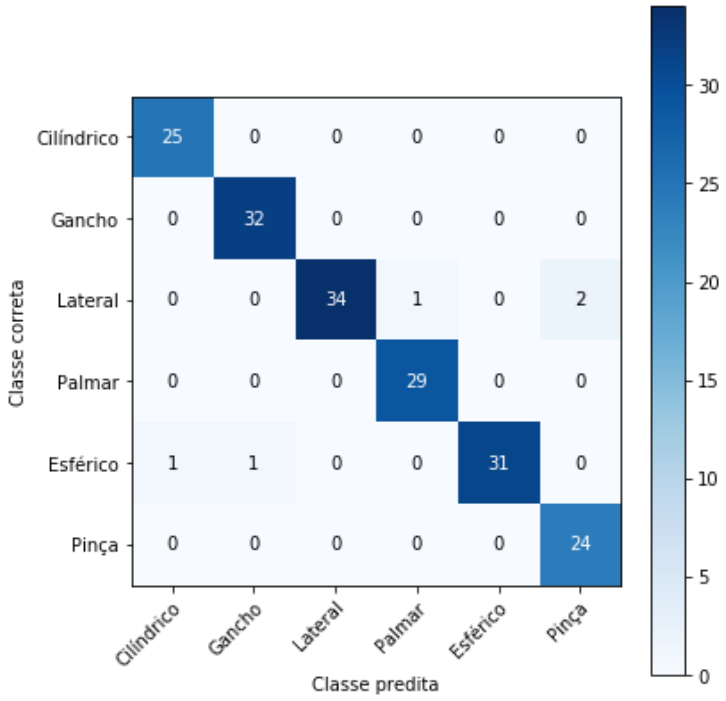
3.4.5 Ajustes e treinamento dos classificadores

Após definidos os conjuntos de treinamento e de teste, ajusta-se os parâmetros dos classificadores, e realiza-se o treinamento do modelo com a amostra de treinamento.

Com os modelos treinados, realiza-se as previsões com as amostras de testes. Cada classificador possui configurações de ajustes próprios, inerentes a sua natureza. Realizou-se um total de 1.728 testes, correspondente ao produto de 144 configurações de classificadores em 12 conjuntos de dados e características, conforme Tabela 3 e Tabela 4.

A acurácia dos resultados dos testes pode ser aferida, dentre outras maneiras, visualizando uma matriz de confusão (JAMES *et al*., 2013), que pode ser gerada com a função *confusion_matrix()* do módulo *sklearn.metrics*. Uma matriz de confusão é uma matriz quadrada que relata o desempenho de um algoritmo de aprendizado, mostrando as contagens das previsões verdadeiro positivo, verdadeiro negativo, falso positivo e falso negativo de um classificador (RASCHKA, 2016). A Figura 26 apresenta uma matriz de confusão gerada para uma classificação que apresentou 97,22% de acurácia. Nesse caso, observa-se que a diagonal onde o número da linha é igual ao número da coluna (intersecção entre previsão e observação), concentram-se as previsões que foram assertivas. Em um caso hipotético com 100% de acurácia, todas outras células com exceção dessa diagonal teriam valor igual a zero.

Figura 26 – Matriz de confusão: classificador RF com 97,22% de acurácia



Fonte: Elaboração do próprio autor

A relação dos classificadores do módulo Python *sklearn* avaliados neste trabalho é apresentada na Tabela 2.

Tabela 2 – Classificadores do módulo Python sklearn

Nome	**Classificador**
Regressão logística	sklearn.linear_model.SGDClassifier()
LDA	sklearn.discriminant_analysis.LinearDiscriminantAnalysis()
QDA	sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis()
KNN	sklearn.neighbors.KNeighborsClassifier()
Árvores de decisão	sklearn.tree.DecisionTreeClassifier()
Florestas aleatórias	sklearn.ensemble.RandomForestClassifier()
SVM	sklearn.svm.SVC()
MLP	sklearn.neural_network.MLPClassifier()

Fonte: Elaboração do próprio autor

A Tabela 3 apresenta os parâmetros das configurações usados nos classificadores durante a criação dos modelos. Os parâmetros foram escolhidos após a realização de testes preliminares.

Tabela 3 – Parâmetros de configuração dos classificadores testados

Classificador	Parâmetros testados
LR *logistic regression*	• loss=log: *logistic regression*
LDA *linear discriminant analysis*	• solver=svd: *singular value decomposition*, não calcula a matriz de covariância • solver=lsqr: método dos quadrados mínimos
QDA *quadratic discriminant analysis*	Nenhum
KNN *k-nearest neighbor*	• n_neighbors: número de vizinhos valores testados: 1, 3, 5, 7, 9
DTREE *decision tree*	• criterion: função para medir a qualidade de uma divisão valores testados: gini, entropy
RF *random forest*	• n_estimators: número de árvores na floresta valores testados: 10, 100, 1000 • oob_score=True: *utilizar amostras "out-of-bag" para estimar acurácia de generalização*
SVM *suport vector machine*	• C: parâmetro de penalidade do termo de erro valores testados: 0.5, 1, 10, 100 • kernel=rbf: *radial basis function*
MLP *multi-layer perceptron*	• hidden_layer_sizes: dimensão das camadas ocultas valores testados: (4,4), (4,4,4), (8,8), (8,8,8), (16,16), (16,16,16), (32,32), (32,32,32), (64,64), (64,64,64), (100,100), (200,200), (300,300), (600,600) • activation: função de ativação valores testados: relu, tanh, logistic • solver=adam: otimizador estocástico baseado em gradiente • solver=lbfgs: otimizador da família dos métodos Quasi-Newton • solver=sgd: gradiente descendente estocástico • max_iter=1000: número máximo de iterações do algoritmo *solver*

Fonte: Elaboração do próprio autor

A Tabela 4 apresenta os conjuntos de dados e as características utilizados para a realização dos treinamentos e dos testes. Todos estes conjuntos possuem a mesma quantidade de instâncias conforme descrito no item 3.4.1, entretanto a quantidade de informações por instância é definida de acordo com as colunas Sinal, RMS, ZC e SSC, que indicam se os dados correspondentes foram incluídos nos conjuntos de dados ou não.

Tabela 4 – Conjuntos de dados e características avaliados

#	Conjunto	Sinal	Características		
			RMS	ZC	SSC
1	Original	Sim	Sim	Sim	Sim
2	Original	Sim	Não	Não	Não
3	Original	Não	Sim	Sim	Sim
4	Original	Não	Sim	Não	Não
5	DWT	Sim	Sim	Sim	Sim
6	DWT	Sim	Não	Não	Não
7	DWT	Não	Sim	Sim	Sim
8	DWT	Não	Sim	Não	Não
9	Decomposição Wavelet nível 4	Sim	Sim	Sim	Sim
10	Decomposição Wavelet nível 4	Sim	Não	Não	Não
11	Decomposição Wavelet nível 4	Não	Sim	Sim	Sim
12	Decomposição Wavelet nível 4	Não	Sim	Não	Não

Fonte: Elaboração do próprio autor

O código produzido em linguagem Python para execução dos experimentos é apresentado no APÊNDICE A.

4 DISCUSSÃO DE RESULTADOS

Os resultados produzidos pelos experimentos são apresentados e discutidos nesse capítulo, considerando-se acurácia de classificação e tempo computacional durante as tarefas de treino e teste. Na Tabela 5 são apresentados os vinte melhores resultados gerais, o melhor resultado de cada classificador é apresentado na Tabela 6 e o melhor resultado de cada conjunto de dados e características é apresentado na Tabela 7.

4.1 RESULTADOS GERAIS

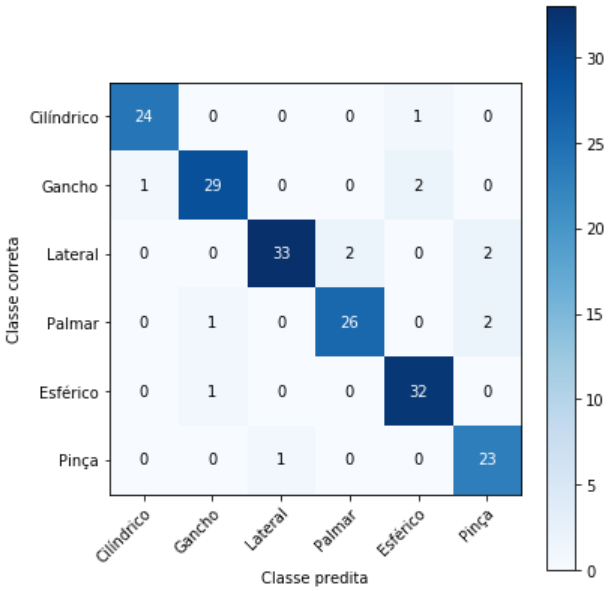
A Tabela 5 apresenta os 20 melhores resultados obtidos dentre todas as 1728 configurações testadas (144 combinações parametrizadas de classificadores, por 12 conjuntos de dados e características). Ordenou-se com os maiores valores na coluna Acurácia, e os menores valores nas colunas Teste e Treino, respectivamente. Dentre os 20 melhores resultados apresentados, a acurácia variou de 97,22% a 92,78%, configurando bons resultados. Nota-se que os conjuntos de dados 7 e 5, construídos com DWT produziram seis dos melhores resultados, cada um. Os conjuntos 12 e 11, construídos com decomposição *wavelet* nível 4, produziram respectivamente outros 4 e 2 dos melhores resultados. Dos conjuntos que não são baseados em técnicas de pré-processamento, aparece apenas o conjunto 3, duas vezes e os conjuntos 1, 2 e 4 não aparecem. O melhor resultado foi obtido por um classificador RF, com acurácia de 97,22%, cuja matriz de confusão foi previamente apresentada na Figura 26. O classificador MLP aparece na posição 20 com acurácia de 92,78%, conforme ilustrado na Figura 27. Percebe-se que os classificadores RF, SVM, KNN e MLP produzem acurácias satisfatórias com as devidas parametrizações e combinações de processamento de sinal e características.

Tabela 5 – 20 melhores resultados nos testes realizados

#	Classificador	Parâmetros	Sinal	Acurácia	Teste (s)	Treino (s)
1	RF	• n_estimators=1000	7	97,22%	0,309664	1,791518
2	RF	• n_estimators=100	7	96,67%	0,104969	0,275166
3	SVM	• C=10.0	5	96,11%	0,752776	5,702162
4	SVM	• C=100.0	5	96,11%	1,084578	6,554091
5	RF	• n_estimators=1000	11	96,11%	0,214576	1,858105
6	KNN	• n_neighbors=3	7	95,56%	0,115467	0,000768
7	RF	• n_estimators=100	3	95,00%	0,102898	0,283315
8	RF	• n_estimators=100	11	95,00%	0,103949	0,275647
9	RF	• n_estimators=1000	3	95,00%	0,208702	1,665807
10	KNN	• n_neighbors=3	5	95,00%	0,216293	0,110113
11	KNN	• n_neighbors=7	5	95,00%	0,221156	0,108330
12	SVM	• C=1.0	5	95,00%	0,780739	5,647353
13	KNN	• n_neighbors=5	5	94,44%	0,223242	0,110880
14	KNN	• n_neighbors=7	7	94,44%	0,112613	0,000760
15	KNN	• n_neighbors=5	7	94,44%	0,120305	0,000775
16	RF	• n_estimators=100	12	93,89%	0,103177	0,279320
17	KNN	• n_neighbors=1	7	93,89%	0,110690	0,000715
18	KNN	• n_neighbors=5	12	93,33%	0,114105	0,000743
19	RF	• n_estimators=1000	12	93,33%	0,310677	1,770980
20	MLP	• hidden_layer_sizes=(16,16) • transfer=tanh • solver=lbfgs	12	92,78%	0,000460	0,851575

Fonte: Elaboração do próprio autor

Figura 27 – Matriz de confusão: classificador MLP com 92,78% de acurácia



Fonte: Elaboração do próprio autor

4.2 CLASSIFICADORES

A Tabela 6 apresenta o melhor desempenho de cada classificador avaliado. Observa-se que os classificadores RF, SVM, KNN e MLP produziram resultados com acurácia superiores a 90%. Ainda, os classificadores DTREE e QDA produziram resultados com acurácia superior a 80%, entretanto, os classificadores LDA e LR produziram resultados com acurácia de 67,78% e 37,22% respectivamente. Dentre os classificadores com acurácia superior a 90%, observando o tempo levado para a realização do teste de cada classificador, o classificador MLP destaca-se com um tempo mais de 1000 vezes mais rápido com relação ao classificador RF, que tem a melhor acurácia.

Tabela 6 – Melhor desempenho de cada classificador

#	Classificador	Parâmetros	Sinal	Acurácia	**Teste** (s)	**Treino** (s)
1	**RF**	• n_estimators=1000	7	97,22%	0,309664	1,791518
3	**SVM**	• C=10.0	5	96,11%	0,752776	5,702162
6	**KNN**	• n_neighbors=3	7	95,56%	0,115467	0,000768
20	**MLP**	• hidden_layer_sizes=(16,16) • transfer=tanh • solver=lbfgs	12	92,78%	0,000460	0,851575
93	**DTREE**	• criterion=entropy	7	87,22%	0,000243	0,010935
135	**QDA**	–	11	83,89%	0,000792	0,003562
287	**LDA**	• solver=lsqr	11	67,78%	0,000130	0,004840
620	**LR**	–	5	37,22%	0,003144	0,108549

Fonte: Elaboração do próprio autor

4.3 CONJUNTOS DE DADOS E CARACTERÍSTICAS

A Tabela 7 apresenta o melhor desempenho de cada conjunto de dados e características utilizados para o treinamento e os testes dos classificadores. Percebe-se que os conjuntos 2, 6 e 10, que não utilizam nenhuma das características (RMS, ZC e SSC) tiveram os piores desempenhos dentre os classificadores testados, produzindo acurácias abaixo de 50%. Observa-se que, em conjunto com essas características, as Wavelets melhoram a acurácia dos resultados.

Tabela 7 – Melhor desempenho de cada conjunto de dados e características

#	Classificador	Parâmetros	Sinal	Acurácia	**Teste** (s)	**Treino** (s)
1	RF	• n_estimators=1000	**7**	97,22%	0,309664	1,791518
5	RF	• n_estimators=1000	**11**	96,11%	0,214576	1,858105
4	SVM	• C=10.0	**5**	96,11%	0,752776	5,702162
7	RF	• n_estimators=100	**3**	95,00%	0,102898	0,283315
16	RF	• n_estimators=100	**12**	93,89%	0,103177	0,27932
51	KNN	• n_neighbors=3	**8**	90,00%	0,113415	0,000745
52	KNN	• n_neighbors=5	**1**	90,00%	0,216144	0,122024
139	SVM	• C=10.0	**9**	83,89%	0,74811	5,361081
171	MLP	• hidden_layer_sizes=(32,32) • transfer=tanh • solver=lbfgs	**4**	80,00%	0,000595	1,737401
469	RF	• n_estimators=1000	**10**	47,78%	0,308811	17,909666
521	RF	• n_estimators=1000	**2**	43,89%	0,309482	12,360353
572	RF	• n_estimators=100	**6**	40,00%	0,110257	1,814592

Fonte: Elaboração do próprio autor

5 CONCLUSÃO

Neste trabalho foi apresentado um estudo sobre análise e implementação de modelos preditivos para classificação de sinais sEMG através do uso de ferramentas de *software* abertas e uma base de dados de sinais pública. Foram realizadas as tarefas de pré-processamento do sinal através de DWT, extração das características RMS, ZC e SSC, e parametrização e testes com os algoritmos classificadores RF, SVM, KNN, MLP, DTREE, QDA, LDA e LR para comparar os resultados.

A comparação dos resultados foi importante para identificar as melhores configurações em termos de acurácia média e tempo computacional entre as combinações dos algoritmos e suas configurações, as características extraídas e os tipos de pré-processamento analisados no estudo, podendo colaborar ou mesmo nortear o desenvolvimento de estudos futuros.

O melhor resultado apresentou acurácia média de 97,2%, obtido por um classificador RF com as características RMS, ZC e SSC extraídas a partir dos coeficientes da DWT, e pode ser considerado um bom resultado, se comparado com o resultado de 89,2% de acurácia geral, obtido por Sapsanis *et al*. (2013), que utilizaram a mesma base de dados de sEMG. Entretanto, é importante apontar que o classificador MLP, com acurácia média de 92,8%, destaca-se por apresentar um tempo computacional mais de 1000 vezes mais rápido com relação ao classificador RF na tarefa de teste, podendo ser o indicado para ambientes embarcados com menor poder computacional.

Considerando os resultados obtidos, fica evidente que a extração de características do sinal, bem como o pré-processamento com as *wavelets*, contribuem significativamente para obtenção de melhores resultados.

A classificação de sinais bioelétricos é um campo de pesquisa atrativo, tendo em vista o número de publicações, neste campo, nos últimos anos, conforme constatado na revisão bibliográfica. Alguns trabalhos futuros podem ser propostos, como a implementação de sistemas de controle de próteses ativas e a implementação de sistemas de simulação com interação humano-computador.

REFERÊNCIAS

ANACONDA Software Distribution. [*S.l.*]: Anaconda Inc, 2020. Disponível em: https://www.anaconda.com. Acesso em: 07 set. 2020.

BELLINGEGNI, A. D. *et al*. NLR, MLP, SVM, and LDA: a comparative analysis on EMG data from people with trans-radial amputation. **Journal of NeuroEngineering and Rehabilitation**, London, v. 14, n. 1, p. 82, dez. 2017.

BETTHAUSER, J. L. *et al*. Limb-position robust classification of myoelectric signals for prosthesis control using sparse representations. *In*: ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY, EMBC, 38, 2016, Orlando. **Proceedings of the** [...]. Orlando:IEEE, 2016.

BIAN, F.; LI, R.; LIANG, P. SVM based simultaneous hand movements classification using sEMG signals. *In*: IEEE INTERNATIONAL CONFERENCE ON MECHATRONICS AND AUTOMATION, ICMA, 2017, Takamatsu. **Proceedings of the** [...]. Takamatsu: IEEE, 2017.

BURHAN, N.; KASNO, M.; GHAZALI, R. Feature extraction of surface electromyography (sEMG) and signal processing technique in wavelet transform: a review. *In*: IEEE International Conference On Automatic Control And Intelligent Systems, I2CACIS, Selangor. **Proceedings of the** [...]. Selangor: IEEE, 2016. p. 141–146.

CALADO, A.; SOARES, F.; MATOS, D. A review on commercially available anthropomorphic myoelectric prosthetic hands, pattern-recognition-based microcontrollers and sEMG sensors used for prosthetic control. *In*: IEEE INTERNATIONAL CONFERENCE ON AUTONOMOUS ROBOT SYSTEMS AND COMPETITIONS, ICARSC, 2019, Porto. **Proceeding of the** [...] Lisboa: IEEE, 2019.

CENE, V. H.; BALBINOT, A. Optimization of features to classify upper-limb movements through semg signal processing. **Brazilian Journal of Instrumentation and Control**, v. 4, n. 1, p. 14, 12 nov. 2016.

CHOWDHURY, R. *et al*. Surface electromyography signal processing and classification techniques. **Sensors**, Basel, v. 13, n. 9, p. 12431–12466, 2013.

COGNOLATO, M. *et al*. Hand gesture classification in transradial amputees using the myo armband classifier*. *In*: IEEE INTERNATIONAL CONFERENCE ON BIOMEDICAL ROBOTICS AND BIOMECHATRONICS, BIOROB, 7, 2018, Enschede. **Proceedings of the** [...]. Enschede: IEEE, 2018.

CUESTA, H. **Practical data analysis**: transform, model, and visualize your data through hands-on projects, developed in open source tools. Birmingham: Packt Publ, 2013.

DAUBECHIES, I. *et al*. **Ten lectures on wavelets**. Philadelphia: Society for Industrial and Applied Mathematics, 1992.

DHEERU, D.; KARRA TANISKIDOU, E. **UCI machine learning repository**. Disponível em: http://archive.ics.uci.edu/ml. Acesso em: 07 set. 2020.

GU, Z. *et al*. **Multi-class classification for basic hand movements**. 2017. Disponível em: https://pdfs.semanticscholar.org/6b05/cda2076c7702da10521eed1eb8acce0293f4.pdf. Acesso em: 07 set. 2020.

HAGAN, M. T. *et al*. **Neural network design**. 2. ed. Boston: Martin Hagan, 2014.

HU, X.; KAN, J.; LI, W. Classification of surface electromyogram signals based on directed acyclic graphs and support vector machines. **Turkish Journal of Electrical Engineering & Computer Sciences**. n. 26, p. 732–742, 2018. Disponível em: https://journals.tubitak.gov.tr/elektrik/issues/elk-18-26-2/elk-26-2-9-1705-63.pdf. Acesso em: 07 set. 2020.

HUNTER, J. D. Matplotlib: a 2D graphics environment. **Computing In Science & Engineering**, Los Alamitos, v. 9, n. 3, p. 90–95, 2007.

JAHAN, M. *et al*. Feature extraction and pattern recognition of EMG-based signal for hand movements. INTERNATIONAL SYMPOSIUM ON ADVANCED COMPUTING AND COMMUNICATION, ISACC, 7, 2015, Madrid. **Proceeedings of the** [...]. Madrid: ICPEAC, 2015. p. 49–52.

JAMES, G. *et al*. **An introduction to statistical learning**. New York: Springer, 2013.

JARRASSÉ, N. *et al*. Classification of phantom finger, hand, wrist, and elbow voluntary gestures in transhumeral amputees with sEMG. **IEEE Transactions on Neural Systems and Rehabilitation Engineering**, Piscataway, v. 25, n. 1, p. 68–77, 2017.

JONES, E. *et al*. **SciPy**: open source scientific tools for Python. Disponível em: http://www.scipy.org. Acesso em: 07 set. 2020.

KARABULUT, D. *et al*. Comparative evaluation of EMG signal features for myoelectric controlled human arm prosthetics. **Biocybernetics and Biomedical Engineering**, Amsterdam, v. 37, n. 2, p. 326–335, 2017.

KRASOULIS, A. *et al*. Improved prosthetic hand control with concurrent use of myoelectric and inertial measurements. **Journal of NeuroEngineering and Rehabilitation**, London, v. 14, n. 1, p. 71, dez. 2017.

LEE, G. *et al*. PyWavelets: A Python package for wavelet analysis. **Journal of Open Source Software**, New York, v. 4, n. 36, p. 1237, 12 abr. 2019.

LI, G. *et al*. A novel feature extraction method for machine learning based on surface electromyography from healthy brain. **Neural Computing and Applications**, London, v. 31, n. 12, p. 9013–9022, 2019.

MACKAY, D. J. C. **Information theory, inference, and learning algorithms**. Cambridge: Cambridge University Press, 2005.

MASSON, S. *et al*. Integrating Myo armband for the control of myoelectric upper limb prosthesis. *In*: CONGRESSO BRASILEIRO DE ENGENHARIA BIOMÉDICA, 25, Foz do Iguaçú. **Anais** [...]. Foz do Iguaçú: SBEB, 2016. Disponível em: http://www.sbeb.org.br/cbeb2016/pt/anais-e-certificados. Acesso em: 07 set. 2020.

MATLAB. **Natick.** Massachusetts: The Mathworks Inc., 2018.

MAYOR, J. J. V. *et al*. Dexterous hand gestures recognition based on low-density sEMG signals for upper-limb forearm amputees. **Research on Biomedical Engineering**, Rio de Janeiro, v. 33, n. 3, p. 202–217, 2017. Disponível em: https://www.scielo.br/pdf/reng/v33n3/2446-4740-reng-2446-474008516.pdf. Acesso em: 07 set. 2020.

MCKINNEY, W. Data structures for statistical computing in Python. *In*: PROCEEDINGS OF THE PYTHON IN SCIENCE CONFERENCE, 9, 2010, Austin. **Proceedings of the** [...]. Austin: [s.n.], 2009. p. 56-61. Disponível em: : https://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf. Acesso em: 07 set. 2020.

NAZMI, N. *et al*. A Review of Classification Techniques of EMG Signals during Isotonic and Isometric Contractions. **Sensors**, Basel, v. 16, n. 8, p. 1304, 17 ago. 2016.

NEGI, S.; KUMAR, Y.; MISHRA, V. M. Feature extraction and classification for EMG signals using linear discriminant analysis. *In*: 2016 2ND INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING, COMMUNICATION, & AUTOMATION, ICACCA, FALL, 2, 2016, Bareilly. **Proceedings of the** [...]. Bareilly: IEEE, set. 2016.

OLIPHANT, T. E. **Guide to NumPy**. Kaneohe: Trelgol Publishing, 2006.

PANCHOLI, S.; JOSHI, A. M. Electromyography-based hand gesture recognition system for upper limb amputees. **IEEE Sensors Letters**, Piskataway, v. 3, n. 3, p. 1–4, mar. 2019a.

PANCHOLI, S.; JOSHI, A. M. Time Derivative moments based feature extraction approach for recognition of upper limb motions using EMG. **IEEE Sensors Letters**, Piscataway, v. 3, n. 4, p. 1–4, 2019b.

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

PHUKPATTARANONT, P. *et al*. Evaluation of feature extraction techniques and classifiers for finger movement recognition using surface electromyography signal. **Medical & Biological Engineering & Computing**, Stevenage, v. 56, n. 12, p. 2259–2271, 2018.

PIZZOLATO, S. *et al*. Comparison of six electromyography acquisition setups on hand movement classification tasks. **Plos One**, California, v. 12, n. 10, p. e0186132, 2017.

POWAR, O. S.; CHEMMANGAT, K.; FIGARADO, S. A novel pre-processing procedure for enhanced feature extraction and characterization of electromyogram signals. **Biomedical Signal Processing and Control**, Oxford, v. 42, p. 277–286, abr. 2018.

RASCHKA, S. **Python machine learning: unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics**. Birmingham Mumbai: Packt Publishing Open Source, 2016.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence**: a modern approach. Upper Saddle River: Prentice-Hall, 1995.

SAPSANIS, C. *et al*. Improving EMG based classification of basic hand movements using EMD. *In*: ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY, EMBC, 35, 2013, Osaka. **Proceedings of the** [...]. Osaka: IEEE, 2013.

SAPSANIS, C.; GEORGOULAS, G.; TZES, A. EMG based classification of basic hand movements based on time-frequency features. *In*: MEDITERRANEAN CONFERENCE ON CONTROL & AUTOMATION, 21, 2013, Chania. **Proceeding of the** [...]. Chania: IEEE, 2013. p. 716–722. Disponível em: https://www.researchgate.net/publication/261313999_EMG_based_classification_of_basic_hand_movements_based_on_time-frequency_features. Acesso em: 07 set. 2020.

SCHEME, E.; ENGLEHART, K. Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use. **The Journal of Rehabilitation Research and Development**, Washington, v. 48, n. 6, p. 643, 2011.

SEGARAN, T. **Programming collective intelligence**. Beijing: O'Reilly, 2007.

SUI, X.; WAN, K.; ZHANG, Y. Pattern recognition of SEMG based on wavelet packet transform and improved SVM. **Optik**, Munchen, v. 176, p. 228–235, jan. 2019.

TOSIN, M. C. *et al*. sEMG feature selection and classification using SVM-RFE. *In*: 2017 39th ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY, EMBC, 39, 2017, Seogwipo. **Proceedings of the** [...]. Seogwipo: IEEE, 2017.

VAN ROSSUM, G.; DRAKE, F. L. **Python 3 Reference Manual**. Scotts Valley, CA, CreateSpace, 2009.

WU, Y. *et al*. Gesture recognition method based on a single-channel sEMG envelope signal. **EURASIP Journal on Wireless Communications and Networking**, Heidelberg, v. 2018, n. 1, p. 35, dez. 2018.

XAVIER, R. T. *et al*. Desenvolvimento de uma mão biônica e de um sistema eletrônico para estudo dos movimentos da mão. *In*: CONGRESO IBEROAMERICANO DE TECNOLOGIAS DE APOYO A LA DISCAPACIDAD, 8, 2015, Punta Arenas. **Anais** [...] Punta Arenas: Cyted, 2015. p. 211–214.

XAVIER, R. T. *et al*. Prótese de membro superior com movimentos pré-definidos pelo usuário. *In*: CONGRESSO BRASILEIRO DE ENGENHARIA BIOMÉDICA, 25, 2016, Foz do Guaçú. **Anais** [...]. Foz do Guaçú. SBEB, 2016. p. 856–859, 2016.

XAVIER, R. T. *et al*. A importância de preensões funcionais em próteses de membro superior de baixo custo. *In*: MEMORIAS CONGRESO IBEROAMERICANO DE TECNOLOGÍAS DE APOYO A LA DISCAPACIDAD, 9, 2017, Rosário. **Memórias** [...]. Rosário: AITADIS, 2017. p. 277–283, 2017. Disponível em: https://www.escuelaing.edu.co/escuela/iberdiscap2017/pdf/Memorias-Iberdiscap-2017_v2.0.pdf. Acesso em: 07 set. 2020.

XAVIER, R. T. *et al*. Upper limb prosthesis for patients with congenital or acquired deformity. *In*: BRAZILIAN CONGRESS ON BIOMEDICAL ENGINEERING, 26, 2019, Singapore. **Proceedings of the** [...]. Singapore: Springer Singapore, 2019. v. 70/1. p. 723–728.

YANG, S. *et al*. Hand motion recognition based on ga optimized svm using semg signals. *In*: INTERNATIONAL SYMPOSIUM ON COMPUTATIONAL INTELLIGENCE AND DESIGN, ISCID, 11, 2018, Hangzhou. **Proceedings of the** [...]. China: IEEE, 2018.

APÊNDICE A – Código Python desenvolvido no trabalho

```python
# ##############################################################################
# Arquivo: array_from_matlab.py

import scipy.io
import numpy as np

def array_from_matlab(base_dir):
    files = ['female_1', 'female_2', 'female_3', 'male_1', 'male_2']
    labels = ['cyl', 'hook', 'lat', 'palm', 'spher', 'tip']
    n_rows = 900  # 5 subjects * 6 classes * 30 iterations
    n_cols = 6001 # 2 channels * 3000 entries + the class
    data = np.zeros((n_rows, n_cols))
    n_row = 0
    for file in files:
        mat = scipy.io.loadmat(base_dir + file)
        for i in range(30): # iterations
            for j, label in enumerate(labels):
                data[n_row][:-1] = np.concatenate((
                        mat[label + '_ch1'][i],
                        mat[label + '_ch2'][i]))
                data[n_row][-1] = j
                n_row += 1
    return data


# ##############################################################################
# Arquivo: features.py

import numpy as np

def rms(a):
    return np.sqrt(np.mean(np.square(a)))

def ssc(a):
    return len(np.where(np.diff(np.sign(np.diff(a))))[0])

def zc(a):
    return len(np.where(np.diff(np.signbit(a)))[0])


# ##############################################################################
# Arquivo: pre_processing.py

import numpy as np

from features import rms, ssc, zc

def signal(ch1, ch2, fn, params):
    n_rows = np.shape(ch1)[0]
    n_cols = 2 * np.sum([np.shape(a) for a in fn(ch1[0], **params)])
    data = np.zeros((n_rows, n_cols))
    for i in range(n_rows):
        data[i][:] = np.hstack(np.hstack([
                fn(ch1[i], **params),
                fn(ch2[i], **params)]))
    return data
```

```
def features(ch1, ch2, fts, fn, params):
    n_rows = np.shape(ch1)[0]
    n_cols = 2 * len(fts) * len(fn(ch1[0], **params))
    data = np.zeros((n_rows, n_cols))
    for i in range(n_rows):
        c1 = fn(ch1[i], **params)
        c2 = fn(ch2[i], **params)
        f = []
        for j in range(len(c1)):
            if 'rms' in fts:
                f.extend([rms(c1[j]), rms(c2[j])])
            if 'ssc' in fts:
                f.extend([ssc(c1[j]), ssc(c2[j])])
            if 'zc' in fts:
                f.extend([zc(c1[j]), zc(c2[j])])
        data[i][:] = f
    return data


# ##############################################################################
# Arquivo: classify.py

import time
import numpy as np
import pywt
import pandas as pd
from sklearn import metrics
from sklearn.linear_model import SGDClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split

from array_from_matlab import array_from_matlab
from pre_processing import signal, features

data = array_from_matlab('../sEMG_Basic_Hand_movements_upatras/Database 1/')

n_rows = np.shape(data)[0]
n_cols = (np.shape(data)[1] - 1) // 2

ch1 = data[:, :n_cols]
ch2 = data[:, n_cols:-1]

y = data[:, -1:]
y = np.reshape(y, n_rows)

datasets = []
for sign in ['raw', 'dwt', 'wavedec']:
    datasets.append({'sign': sign, 'add_sign': False, 'fts': ['rms']})
    datasets.append({'sign': sign, 'add_sign': False, 'fts': ['rms', 'ssc', 'zc']})
    datasets.append({'sign': sign, 'add_sign': True, 'fts': []})
    datasets.append({'sign': sign, 'add_sign': True, 'fts': ['rms', 'ssc', 'zc']})

fn_map = {'raw': {'fn': lambda a: np.array([a]), 'params': {}},
          'dwt': {'fn': pywt.dwt, 'params': {'wavelet': 'db4'}},
          'wavedec': {'fn': pywt.wavedec, 'params': {'wavelet': 'db4', 'level': 4}}}
```

```
results = []
for ds in datasets:
    X_signal = signal(ch1, ch2,
        fn_map[ds['sign']]['fn'],
        fn_map[ds['sign']]['params'])

    X_features = features(ch1, ch2, ds['fts'],
        fn_map[ds['sign']]['fn'],
        fn_map[ds['sign']]['params'])

    if ds['add_sign'] and len(ds['fts'] == 0):
        X = X_signal
    elif ds['add_sign'] and len(ds['fts'] > 0):
        X = np.hstack(X_signal, X_features)
    else:
        X = X_features

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=0)

    classifiers = []

    # LR
    cl = SGDClassifier(loss='log', random_state=0, n_jobs=-1)
    classifiers.append(
        {'name': 'reg-log', 'params': '', 'classifier': cl})

    # LDA
    for solver in ['svd', 'lsqr']:
        cl = LinearDiscriminantAnalysis(solver=solver)
        classifiers.append(
            {'name': 'lda', 'params': solver, 'classifier': cl})

    # QDA
    cl = QuadraticDiscriminantAnalysis()
    classifiers.append(
        {'name': 'qda', 'params': '', 'classifier': cl})

    # KNN
    for n_neighbors in [1, 3, 5, 7, 9]:
        cl = KNeighborsClassifier(n_neighbors=n_neighbors, n_jobs=-1)
        classifiers.append(
            {'name': 'knn', 'params': str(n_neighbors), 'classifier': cl})

    # DTree
    for criterion in ['gini', 'entropy']:
        cl = DecisionTreeClassifier(criterion=criterion, random_state=0)
        classifiers.append(
            {'name': 'dtree', 'params': criterion, 'classifier': cl})

    # RF
    for n_estimators in [10, 100, 1000]:
        cl = RandomForestClassifier(
            n_estimators=n_estimators, max_features='auto', oob_score=True,
            random_state=0, n_jobs=-1)
        classifiers.append(
            {'name': 'rf', 'params': str(n_estimators), 'classifier': cl})
```

```
    # SVM
    for C in [0.5, 1., 10., 100.]:
        cl = svm.SVC(C=C, kernel='rbf', random_state=0)
        classifiers.append(
            {'name': 'svm', 'params': 'rbf,C=' + str(C), 'classifier': cl})

    # MLP
    layers = [(4, 4), (4, 4, 4), (8, 8), (8, 8, 8), (16, 16), (16, 16, 16),
              (32, 32), (32, 32, 32), (64, 64), (64, 64, 64),
              (100, 100), (200, 200), (300, 300), (600, 600)]
    for layer in layers:
        for activation in ['relu', 'tanh', 'logistic']:
            for solver in ['adam', 'sgd', 'lbfgs']:
                cl = MLPClassifier(
                    hidden_layer_sizes=layer, activation=activation, solver=solver,
                    max_iter=1000, random_state=0)
                s_params = str(layer) + ',' + activation + ',' + solver
                classifiers.append(
                    {'name': 'mlp', 'params': s_params, 'classifier': cl})

    for cl in classifiers:
        expected = y_test

        start_time = time.time()
        cl['classifier'].fit(X_train, y_train)
        training_time = time.time() - start_time
        start_time = time.time()
        predicted = cl['classifier'].predict(X_test)
        predicting_time = time.time() - start_time

        s_fts = ','.join(
            ds['fts'] if not ds['add_sign'] else [ds['sign'], *ds['fts']]),

        results.append({'classifier': cl['name'],
                        'params': cl['params'],
                        'signal': ds['sign'],
                        'features': s_fts,
                        'accuracy': metrics.accuracy_score(expected, predicted),
                        'training_time': training_time,
                        'predicting_time': predicting_time})

df = pd.DataFrame(results).sort_values(
        by=['accuracy', 'predicting_time', 'training_time'],
        ascending=[False, True, True])

df.to_csv('../resultados/resultados.csv', sep='\t')
```