UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

GUILHERME CARVALHO DA SILVA

PREDIÇÃO DOS ARRANQUES MÍNIMOS DE UM MANIPULADOR ROBÓTICO UTILIZANDO APRENDIZADO DE MÁQUINA PARA APLICAÇÕES EM CIRURGIAS MÉDICAS

FORTALEZA

2021

GUILHERME CARVALHO DA SILVA

PREDIÇÃO DOS ARRANQUES MÍNIMOS DE UM MANIPULADOR ROBÓTICO UTILIZANDO APRENDIZADO DE MÁQUINA PARA APLICAÇÕES EM CIRURGIAS MÉDICAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Orientadora: Prof. Dr. Laurinda Lúcia Nogueira dos Reis

FORTALEZA

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58p Silva, Guilherme Carvalho da.
Predição dos arranques mínimos de um manipulador robótico utilizando aprendizado de máquina para aplicações em cirurgias médicas / Guilherme Carvalho da Silva. – 2021.
49 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2021.
Orientação: Profa. Dra. Laurinda Lúcia Nogueira dos Reis.
Coorientação: Prof. Me. Darielson Araújo de Souza.

1. Robôs. 2. Trajetórioas. 3. Arranques. 4. Machine Learning. 5. Cirurgias Médicas. I. Título.
CDD 621.3

GUILHERME CARVALHO DA SILVA

PREDIÇÃO DOS ARRANQUES MÍNIMOS DE UM MANIPULADOR ROBÓTICO UTILIZANDO APRENDIZADO DE MÁQUINA PARA APLICAÇÕES EM CIRURGIAS MÉDICAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Laurinda Lúcia Nogueira dos Reis (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Msc. Darielson Araújo de Souza
Universidade Federal do Ceará (UFC)

Prof. Msc. Josias Guimarães Batista
Instituto Federal do Ceará (IFCE)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Aos meus colegas de curso, que assim como eu encerram uma difícil etapa da vida acadêmica.

AGRADECIMENTOS

À minha família, que me ensinou o valor da educação, me apoiou mesmo nos momentos de dificuldade e me deram suporte para conseguir alcançar meus sonhos.

À Prof.ª. Dr.ª. Laurinda Lúcia Nogueira dos Reis por me orientar e me dar a chance de realizar este trabalho.

Aos Profs. Msc. Darielson Araújo de Souza e Josias Guimarães Batista, pela coorientação e ensinamentos que vêm me proporcionando desde antes do começo deste trabalho.

Aos meus colegas do curso de Engenharia Elétrica: Diêgo Mezes Campos, Cláudio César Martins Magalhães Filho, Átila Mendonça Araújo Cavalcante, Gabriel Oliveira de Almeida, Gabriel Paulino de Silva Lima, Giovanni Oliveira Confalonieri, Guilherme Lawrence Rebouças Oliveira, Jessé do Nascimento Lopes, José Moacir Holanda Neto, Pedro Braga Luz, Márcio Venício Ramos Limaverde, Matheus Lima Sampaio, Endryo Moreira Rodrigues Feitosa, Gabriel Freitas Machado, Wesley Bezerra Nogueira, Mikael Lucas de Brito Souza, Rafael Santos Bertuzzi, Thiago Azevedo Campos Costa e Lindemberg Samuel de Brito Matias pelos momentos bons durante as dificuldades do curso.

Aos meus colegas do Grupo de Pesquisa em Automação, Controle e Robótica (GPAR): Dayse Maria Benevides de Queiroz, João Paulo Bezerra de Araújo, Felipe José de Souza Vasconcelos, Gabriel Feitas Machado, Ismael de Souza Bezerra, José Nogueira do Nascimento Júnior, Judá Teixera Santos, René Descartes Olimpo Pereira, pelos momentos bons e os ensinamentos advindos destes.

Aos Profs. Dr. Fabrício Gonzalez Nogueira e Bismark Claure Torrico, pela oportunidade de trabalhar no GPAR como aluno de iniciação científica.

Aos bibliotecários da Universidade Federal do Ceará: Francisco Edvander Pires Santos, Juliana Soares Lima, Izabel Lima dos Santos, Kalline Yasmin Soares Feitosa e Eliene Maria Vieira de Moura, pela revisão e discussão da formatação utilizada neste *template*.

Ao aluno Thiago Nascimento do curso de ciência da computação da Universidade Estadual do Ceará que elaborou o *template* do qual este trabalho foi adaptado para Universidade Federal do Ceará.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

E à Fundação Cearense de Apoio ao Desenvolvimento (Funcap), na pessoa do Presidente Tarcísio Haroldo Cavalcante Pequeno pelo financiamento da pesquisa de iniciação científica via bolsa de estudos.

“Transmita o que aprendeu. Força, maestria. Mas fraqueza, insensatez, fracasso também. Sim, fracasso acima de tudo. O maior professor, o fracasso é. Luke, nós somos o que eles crescem além. Esse é o verdadeiro fardo de todos os mestres.”

(Mestre Yoda)

RESUMO

A utilização de robôs para realização de tarefas automatizadas e/ou repetitivas vem se expandindo para fora do âmbito industrial, sendo esses utilizados atualmente em áreas como produção agrícola, setores de serviços e hospitalares: para garantir que a performance do robô seja a melhor possível, criou-se o campo de estudo focado em otimização de planejamento de trajetórias. Um dos maiores problemas enfrentados nesse campo é a modelagem dos arranques mínimos para que possa-se operar os robôs sem que estes entrem em ressonância e, consequentemente, sofram desgastes em seus componentes: porém, esse processo de modelagem comumente utiliza algoritmos de otimização, como *Particle Swarm Optmization* (PSO) e *Natural Language Processing* (NLP). Neste trabalho são avaliados métodos de modelagem, como Redes Neurais Artificiais (RNA) e algoritmos de *Machine Learning*, para determinar os arranques mínimos de um manipulador robótico com aplicação em cirurgias médicas, montado em uma simulação. O objetivo deste trabalho é comprovar que métodos mais simples e menos custosos computacionalmente podem ser utilizados para este propósito, com resultados tão bons quanto ou melhores que os dos métodos utilizados atualmente.

Palavras-chave: Robôs. Trajetórioas. Arranques. Modelagem, RNA, *Machine Learning*, Cirurgias Médicas.

ABSTRACT

The usage of robots to perform automated and/or repetitive tasks has been expanding beyond the industrial scope, and these are currently used in areas such as agricultural production, service sectors and hospitals: to ensure that the robot's performance is the best possible, the field of study focused on trajectory planning optimization was created. One of the biggest problems faced in this field is the modeling of minimum jerks so that robots can be operated without them going into resonance and, consequently, suffering wear in their components: however, this modeling process commonly uses optimization algorithms, such as PSO and NLP. In this work, modeling methods, such as Artificial Neural Networks (ANN) and *Machine Learning* algorithms, are evaluated to determine the minimum jerks of a robotic manipulator with application in medical surgery, assembled in a simulation. The focus of this work is to prove that simpler and less computationally costly methods can be used for this purpose, with results as good as or better than those of the methods currently used.

Keywords: Robots. Trajectories. Jerks. Modeling, RNA, Machine Learning, Medical Surgeries.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Ae	Área efetiva da antena
B	Largura de faixa em que o ruído é medido em Hertz
d	Distância em metros
E	Campo elétrico
FA	Fator da antena
Gr	Ganho de recepção
h	Altura efetiva ou comprimento efetivo de uma antena
I	Corrente elétrica
k	Constante de Boltzmann's
K	Eficiência de irradiação
M	Variação do patamar de ruído em função da RBW
N	Condutor de neutro
NF	Figura de ruído
Ni	Potência do ruído na entrada
No	Potência do ruído na saída
P	Potência
R	Resistência
Si	Potência do sinal na entrada
So	Potência do sinal na saída
t	Tempo
V	Tensão
ZL	Impedância da antena
Zo	Impedância de referência (50Ω)
λ	Comprimento de onda
Γ	Coeficiente de reflexão

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização do Tema: Robótica e suas Aplicações

Com o avanço da tecnologia na sociedade, a Robótica vêm se tornando cada vez mais comum no cotidiano, pois o uso de robôs vem permitindo a automatização de tarefas e processos diversos, sejam eles repetitivos e/ou de alto risco para os próprios trabalhadores: por isso, eles tem se tornado alternativas seguras e de garantia de produtividade, principalmente no setor industrial. O processo de digitalização da manufatura proporcionado pela Indústria 4.0 também vem contribuindo para a ascensão da robótica como facilitador dos processos trabalhistas: a integração das máquinas através de computadores e sistemas inteligentes vêm permitindo que, ao se coletar e analisar grandes volumes de dados com informações variadas, possam ser tomadas decisões que impactam diretamente em fatores como orçamento, manutenção das máquinas e otimização de processos (MARR, 2019).

Nas últimas décadas, as aplicações da força de trabalho formada por robôs se expandiram além das indústrias, sendo empregadas em áreas como a produção agrícola e o setor de serviços, tendo como exemplo o setor de serviços hospitalares. Pelo fato de robôs serem consideravelmente mais rápidos de treinar, mais baratos de se manter e poderem trabalhar turnos mais longos do que humanos, eles são considerados alternativas ideais para arcar com trabalho braçal, cuidar de pacientes com doenças crônicas e diminuir despesas com lares de idosos e cuidadores (QURESHI; SYED, 2014). Além do trabalho de apoio, eles podem ser empregados em outras aplicações relacionadas à medicina, como em cirurgias complexas, que exigem mais precisão, e automação de processos em laboratórios de testes, o que otimiza o tempo de obtenção dos resultados (OWEN-HILL, 2021).

A depender da estrutura mecânica, os robôs podem ser divididos entre: Manipuladores Robóticos, que apresentam base fixa; e Robôs Móveis, que possuem bases móveis. Os manipuladores, em específico, consistem em partes mecânicas rígidas interconectadas através de juntas, o que forma a imagem de um braço humano: o braço em si permite mobilidade; na ponta, um pulso para conferir destreza; e nos "dedos", a ferramenta com a qual o manipulador realizará a tarefa (*end-effector*) (SICILIANO *et al.*, 2010). Este tipo de robô é amplamente usado na indústria, especialmente na automobilística, por permitirem a montagem de peças industriais mais pesadas e complexas de forma mais rápida e precisa, sem por em risco à segurança dos operadores (SIMPLÍCIO; LIMA, 2016). Com isso, diminuiu-se os gastos com encargos trabalhistas

e permitiu a manutenção de turnos de trabalho mais longos, pelo fato das máquinas poderem operar por períodos mais longos. Manipuladores robóticos têm se tornado cada vez mais comuns fora das fábricas nos últimos tempos: podem ser encontrados na agricultura, ajudando, por exemplo, na automatização de tarefas em locais como estufas e lavouras (NEVES, 2020); na engenharia aeroespacial, para permitir maior segurança dos tripulantes em condições hostis, como é visto nas estações espaciais; e nos ramos de medicina e enfermagem, sendo empregados para cirurgias invasivas que exigem maiores graus de precisão do profissional ou para dar suporte a pacientes em fase de reabilitação física e/ou com limitações físicas que exigem acesso e mobilidade diferenciados.

1.2 Problemática

Em muitas tarefas envolvendo manipuladores robóticos, a precisão requerida para performá-las exigem fatores como acurácia, velocidade e suavidade cuidadosamente calculados (OLIVEIRA *et al.*, 2020). Devido à isso, tem-se aumentado o foco em estudos relacionados à ótimização de planejamento de trajetórias, de forma a se executarem movimentos similares aos dos braços humanos e garantir que esses movimentos levem em conta às restrições físicas do manipulador em estudo (LU *et al.*, 2017).

Dentre as diversas limitações, como velocidade, aceleração e tempo de movimento das juntas, o arranque tem sido uma das mais problemáticas de se modelar: para que as trajetórias otimizadas possam garantir o funcionamento a longo prazo do manipulador e a segurança dos operadores, deve-se considerar durante o planejamento da trajetória as vibrações do robô, advindas da suavidade dos movimentos do mesmo. Como movimentos em altas velocidades tendem a reduzir a suavidade e a acurácia dos movimentos, desgastar as juntas do braço robótico mais rapidamente e por em risco a segurança dos operadores, surgiu o princípio de estudos para se achar o *arranque mínimo*, que seria a mínima derivada da aceleração das juntas ao longo do tempo para que não aja ressonância das vibrações das juntas. Essa estratégia comprovadamente permitiu observar que o arranque produzido no movimento afeta diretamente na performance do algoritmo de trajetória e reduz os erros de detecção da posição das juntas (KYRIAKOPOULOS; SARIDIS, 1988; KYRIAKOPOULOS; SARIDIS, 1991; KYRIAKOPOULOS *et al.*, 1994).

1.3 Justificativa

Uma das estratégias que mais tem se apresentado eficiente para estimação dos arranques minímos nas últimas décadas é a utilização de algoritmos de Inteligência Computacional, em destaque, algoritmos de otimização para prever valores de arranque que atendam os critérios de minimização. O uso de métodos e algoritmos de inteligência computacional são vantajosos por serem capazes de performarem bem em relação a sistemas dinâmicos e a sistemas e restrições lineares e não lineares, o que permite a estimação, para controladores e atuadores, de hiperparamentros mais próximos de situações reais. Algoritmos de *Particle Swarm Optmization* (PSO) podem ser utilizados para hiperparametrização de controladores PID para que haja arranque mínimo (OLIVEIRA *et al.*, 2020); Métodos de Pseudo-Espectro podem ser utilizados para transcrever problemas de otimização dos arranques em programas de dimensão náo-lineares (NLP), o que permite que métodos numéricos mais simples possam resolvé-los (FREEMAN, 2012).

Porém, há pouco material relacionado à otimização de arranques por métodos como Redes Neurais Artificiais (RNA) e métodos de aprendizado de máquina, como modelos baseados em árvores, o que levanta a hipótese: esses métodos podem ser tão eficazes para a determinação de arranques mínimos quanto os métodos de otimização mais utilizados na literatura? Para essa hipótese, foi montanda uma simulação de um manipulador robótico aplicável para cirurgias médicas, baseando-se em dados de um manipulador simulado.

1.4 Objetivos

Com o intuito de solucionar o problema proposto, foram determinados, para este trabalho, os seguintes objetivos:

1.4.1 Objetivos Gerais

Estimar arranques mínimos para as juntas de um manipulador robótico com aplicações em cirurgias médicas utilizando diferentes métodos de inteligência computacional e compararar seus resultados.

1.4.2 Objetivos Específicos

- Apresentar um estudo da estrutura do manipulador robótico em análise;
- Apresentar estudos sobre a problemática da minimização dos arranques do manipulador e seus impactos na longevidade do mesmo;
- Utilizar métodos de Aprendizado de Máquina e Redes Neurais Artificiais para estimar arranques mínimos para as juntas;
- Listar e comparar os resultados de cada método e selecionar o mais eficiente;

1.5 Estrutura do Trabalho

No Capítulo 2, serão apresentadas as fundamentações teóricas sobre: estrutura e cinemática de manipuladores robóticos; os métodos de aprendizado de máquina e redes neurais utilizados na literatura que serão utilizados e comparados; e as aplicações de robôs com esses métodos em cirurgias médicas.

No Capítulo 3, é apresentada a metodologia utilizada pelo autor, a planta de estudo, a base de dados a ser utilizada e como cada método foi projetado para a planta em questão.

No Capítulo 4, são apresentados os resultados da aplicação de cada método e o comparativo entre eles.

No Capítulo 5, são apresentadas as considerações finais do trabalho e sugestões de trabalhos futuros.

2 FUNDAMENTAÇÃO

Neste capítulo, será realizada a fundamentação teórica de conceitos básicos envolvendo os estudos de manipuladores robóticos. Em seguida, apresentam-se os conceitos e a formulação de modelos estatísticos de Inteligência Computacional Aplicada. Por fim, apresenta-se estudos e considerações acerca de projetos de Inteligência Computacional Aplicada em Manipuladores Robóticos, com enfâse em projetos para aplicações médicas.

2.1 Manipuladores Robóticos

2.1.1 Introdução a Manipuladores Robóticos e Degrees of Freedom (DOF)

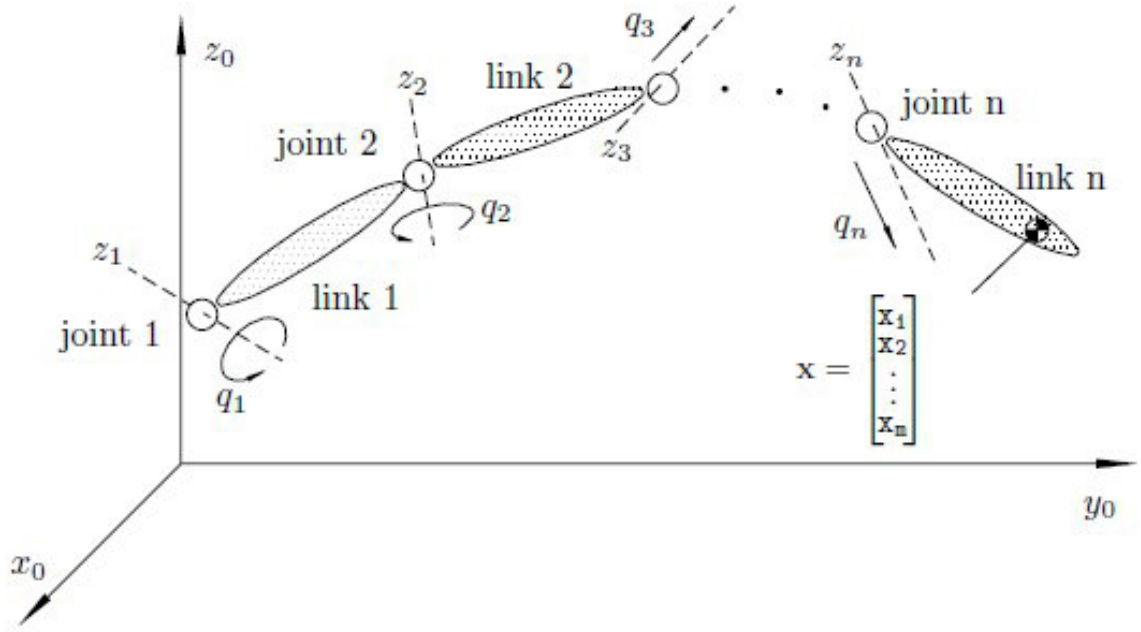
Manipuladores robóticos são, em sua essência, formados por múltiplas conexões de juntas que interligam estruturas rígidas entre si, denominadas de elos, de forma a apresentar similaridades de formato e, como consequência de sua montagem, a mobilidade suave e estável do braço humano. Essas interações entre as juntas e os elos são denominadas *Kinematic Chains* ou Cadeias Cinemáticas (SPONG *et al.*, 2006). As juntas podem ser classificadas, de forma geral, de duas formas: juntas rotacionais (R), que permite que dois elos conectados possam rotacionar em um eixo; e juntas prismáticas (P), que permitem movimentos lineares. Essas notações serão utilizadas durante o resto do trabalho

A Figura 1 apresenta um diagrama de uma configuração genérica de um manipulador robótico, abstraído para n-*Degrees of Freedom* (DOF) ou Graus de Liberdade em que o robô pode operar dependem diretamente da quantidade de juntas que o mesmo possui. Para se obter um modelo matemático geral para um manipulador, insere-se o mesmo dentro de um espaço de referência cartesiano tridimensional com origem na base do mesmo (KELLY *et al.*, 2006). Os elos são numerados desde a base (elo 0) até o *end-effector* (elo i), e as juntas, de forma a (i-1)-ésima junta se conecte com a i-ésima junta e cada uma delas sejam acionadas independentemente por atuadores próprios.

Como as juntas são os elementos que proporcionam a movimentação dos elos e, consequentemente, do manipulador em si, a quantidade de DOFs. No geral, é requerido que o manipulador possua, no mínimo, seis DOFs independentes para que o braço possa se mover no seu ambiente de forma arbitrária (SPONG *et al.*, 2006).

A Figura 2 apresenta o esquema de um robô com 4-DOF. Considerando z_i como o

Figura 1 – Diagrama Abstrato de um Manipulador Robótico de n-DOF



Fonte: (KELLY *et al.*, 2006)

eixo de movimento das juntas, e q_i a coordenada de deslocamento da junta em torno de z_i, para robores com n-DOF. o vetor de posições das juntas q possui n elementos (KELLY *et al.*, 2006):

$$q_{n,1} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \tag{2.1}$$

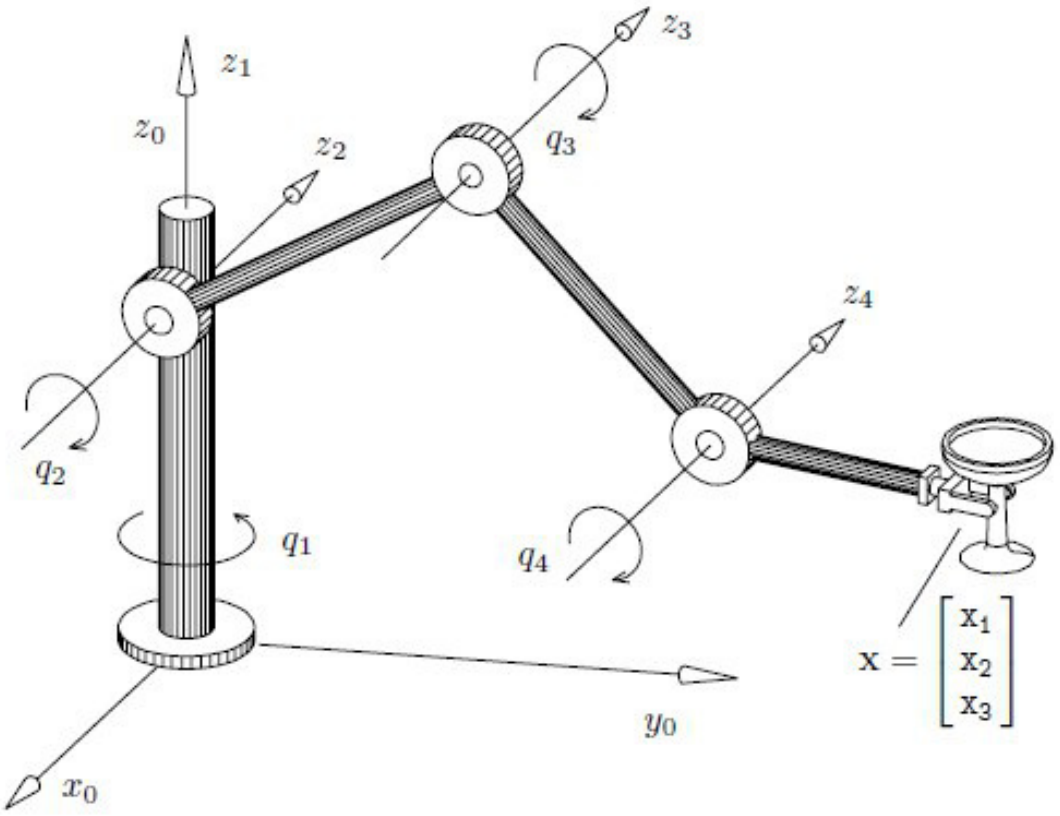
i.e. $q \in \mathbb{R}^n$ (KELLY *et al.*, 2006).

Para o *end-effector*, a posição e a orientação do mesmo são medidas em referências ao plano cartesiano $\{x_0, y_0, z_0\}$ e nos Ângulos de Euler $\{\theta_0, \phi_0, \psi_0\}$ (CRAIG, 2009), sendo essas informações coletadas no vetor x:

$$x_{m,1} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \tag{2.2}$$

i.e. $x \in \mathbb{R}^n$ e $m < n$.

Figura 2 – Exemplo de um manipulador de 4-DOF



Fonte: (KELLY *et al.*, 2006)

2.1.2 Workspace

O *Workspace* ou Ambiente de Trabalho de um manipulador é toda a região tridimensional em que a ponta do manipulador, o *end-effector* pode alcançar considerando todas as movimentações possíveis realizadas pelo mecanismo, e seu tamanho varia conforme a geometria de configuração da estrutura é formada e das limitações mecânicas das juntas. Segundo Spong *et al.* (2006), o ambiente de trabalho comumente é divido em dois: o Ambiente Alcançável (*Reachable Workspace*), que consiste em todo o conjunto de pontos em que a estrutura completa pode alcançar; e o Ambiente Destro (*Dextrous Workspace*), que é um subconnjunto de pontos do Ambiente Desejável onde a mobilidade do *end-effector* permite que o manipulador alcance mais pontos.

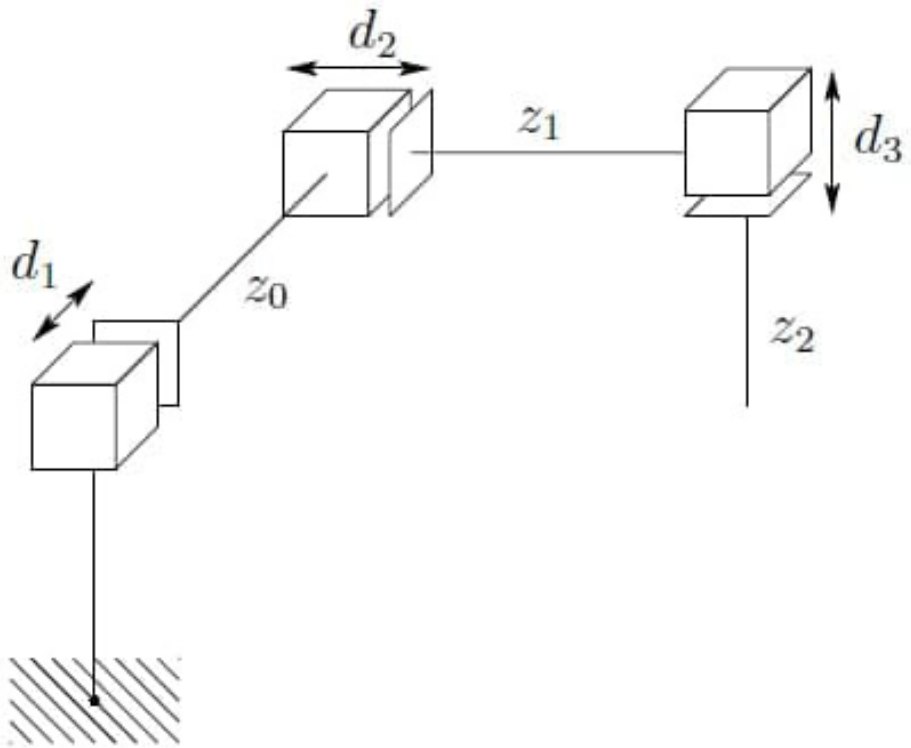
2.1.3 Tipos de Manipuladores Robóticos

Manipuladores robóticos podem ser classificados em vários critérios, como a fonte de energia. a área de atuação, o tipo de controlador utilizado, dentre outras. O critério mais

comumente utilizado para classificá-los, no entanto, é quanto a sua geometria ou estrutura cinemática. Levando-se em conta as três primeiras juntas desde a base, pode-se classificá-los em: Articulada, Esféricas, SCARA, Cilíndrica e Cartesiana. Essas estruturas cinemáticas determinam o *Workspace* do manipulador.

Em específico, os manipuladores robóticos do tipo Planar ou Cartesiano apresentam as três primeiras juntas com relação à base do tipo prismática, ou seja, possuem apenas um DOF para poderem serem usadas para descrever movimentos translacionais. Manipuladores cartesianos possuem como variáveis das juntas as coordenadas cartesianas dos *end-effectors* e, portanto, são os que possuem as descrições cinemáticas mais simples (SPONG *et al.*, 2006). A Figura 3 apresenta a estrutura de um manipulador cartesiano:

Figura 3 – Manipulador Robótico Cartesiano



Fonte: (SPONG *et al.*, 2006)

2.1.4 Cinemática

Um dos enfoques da montagem de manipuladores mecânicos é o planejamento de trajetórias para que cada junta do robô se desloque de forma que ele possa executar movimentos. Para isso, é necessário obter-se informações acerca da cinemárica da estrutura.

O modelo de *Cinemática Direta* do robô descreve a relação entre a posição q das juntas e as coordenadas x do *end-effector* (KELLY *et al.*, 2006). Eles determinam como será a coordenada x do *end-effector* dada que a junta está na posição q:

$$x = \Psi(q) \tag{2.3}$$

onde $\Psi : \mathbb{R}^n \to \mathbb{R}^n$. Modelos cinemáticos diretos costumam ser metódicos e geralmente, em casos de manipuladores com poucos DOF, podem ser modelados a partir de equações trigonométricas.

Já modelos de *Cinemática Inversa* computam as posições q das juntas dado as coordenadas x do *end-effector*:

$$q = \Psi(x)^{-1} \tag{2.4}$$

Modelos cinemáticos inversos podem apresentar múltiplas soluções conforme a quantidade de DOFs do manipulador e, consequentemene, a complexidade do mesmo aumenta (KELLY *et al.*, 2006).

2.2 Inteligência Computacional Aplicada (ICA)

2.2.1 Introdução à ICA

O conceito de Inteligência Computacional Aplicada como subárea da Inteligência Artificial (AI) é até hoje bastante debatido, mas sem um consenso, devido ao grande aumento de volume de dados que vêm ocorrendo com a melhoria do acesso a informações, principalmente devido à internet, e a necessidade de organizar áreas novas e existentes acerca de Inteligência Artificial. Uma forma de se definir Inteligência Artificial é explicá-la como a ciência que estuda o processo de criação de máquinas e programas inteligentes para realizar tarefas que normalmente requerem uso de inteligência humana para serem realizadas. Esse ramo da ciência busca compreender o comportamento de máquinas caso lhes fossem dadas formas de raciocinar (TURING, 1950) e, em posse desse raciocínio, possam receber estímulos do ambiente e tomar ações em função deles (RUSSEL *et al.*, 2013). Desses conceitos iniciais, vários sub-ramos surgiram de forma a tentar aplicá-los, sendo um deles a Inteligência Computacional.

Segundo Wlodzislaw (2007), uma definição geral de Inteligência Computacional é de que esta é um ramo da Ciência da Computação em que se estudam problemas cujas soluções não possuem algoritmos efetivamente certeiros. Essa definição permite que essa subárea não seja restringida pelos métodos e algoritmos usados, mas sim pelos diversos casos que requerem soluções inteligentes. Outras definições afirmam que a Inteligência Computacional é o ramo da Inteligência Artificial que busca investigar e simular aspectos da cognição humana, como percepção e aprendizado (TECNOLOGIA, 2020). Devido à subjetividade do termo inteligência, torna-se cada vez mais complicado separar os conceitos de ICA e AI e as técnicas associadas a estas. Para os fins deste trabalho, levará-se em conta a definição de que a Inteligência Computacional é um ramo cujo objetivo final é a proposição de soluções e a tomada de decisão para problemas que se apresentarem. Como a ICA se inspira desde os princípios da natureza até o comportamento humano, diversas técnicas foram desenvolvidas para solucionar e investigar diversos tipos de problemas. Algumas dessas técnicas serão apresentadas neste capítulo.

2.2.2 Modelagem Preditiva e Inferência Estatística

Devido à necessidade das pessoas de saber acerca de eventos futuros para que se possam tomar as melhores decisões possíveis, vem ocorrendo um crescimento na quantidade de dados e informações recolhidas para este propósito, especialmente após a universalização do

acesso à internet. Apesar de o cérebro humano poder recolher e armazenar grandes parcelas de dados, consciente ou subconscientemente, ele é incapaz de processar todas as informações relevantes provenientes desses dados (KUHN *et al.*, 2013). Para resolver esse problema, foram desenvolvidas ferramentas que recebem essas informações, as analizam minuciosamente para encontrarem padrões relevantes para o problema e apresentam soluções estimadas o mais próximas do possível da realidade. Esse processo é conhecido como Modelagem Preditiva, enquanto que o conjunto de ferramentas utilizado para analizar e interpretar os dados e, consequentemente, gerar a predição é denominado de Aprendizado Estatístico ou Inferência Estatística.

De forma generalizada, supõe-se que, para uma dada resposta Y, existam n preditores ou variáveis associadas $(X_1, X_2, \ldots, X_n)$. O objetivo do Aprendizado Estatístico é determinar a relação entre Y e $X = X_1, X_2, \ldots, X_n$ (GARETH *et al.*, 2013), dado que essa relação possa ser escritar da forma generalizada:

$$Y = F(X) + \varepsilon \tag{2.5}$$

onde F é a relação sistemática que X provê a respeito de Y e ε o erro irredutível aleatório, de média zero.

Para fins de predição, não se tem, de antemão, uma resposta Y clara dado os preditores X disponíveis. Com isso, é necessário obter uma resposta aproximada do valor real. Dado que ε é um erro de média zero, podemos considerar a predição de Y como:

$$\hat{Y} = \hat{F}(X) \tag{2.6}$$

onde $\hat{Y}$ é a predição de Y e $\hat{F}(X)$ é a relação estimada entre X e Y. O quão próximo $\hat{Y}$ é de Y depende de dois fatores:

- Do erro reduzível associado à $\hat{F}(X)$;
- Do erro irreduzível ε;

Para fins de inferência, é necessário entender como Y muda em função dos preditores de X. Para isso, é necessário analisar, de forma exata, as seguintes discussões:

- Quais os preditores que estão associados à resposta dentre múltiplas variáveis?
- Qual a relação entre cada preditor e a resposta final, levando-se em conta possíveis dependências entre preditores?

- Qual a melhor forma de representar a relação entre a resposta e cada preditor (equações lineares, equações polinomiais mais complexos, dentre outras)?

Avaliando-se os problemas que necessitam de uma previsão coerente, pode-se dividí-los em duas categorias: problemas de Aprendizado Supervisionado, onde para cada preditor $X_i, i = 1,2\ldots n$, há dados a respeito da resposta Y, e deseja-se realizar predições com base em novos preditores de forma que a saída $\hat{Y}$ seja o mais próxima possível de Y; e problemas de Aprendizado Não Supervisionado, onde se têm informações apenas a respeito dos preditores $X_i, i = 1,2\ldots n$, demandando que a análise estatística seja feita observando-se a relação entre as variáveis e/ou a relação entre as predições obtidas (GARETH *et al.*, 2013).

Para este trabalho, o foco será dado ao estudo de modelos preditivos supervisionados, pois há informações a respeito da resposta da modelagem. As seguintes técnicas supervisioanadas serão abordadas mais detalhadamente nas próximas seções: Modelos baseado em Árvores (especificamente *Gradient Boosting Machines*) e Redes Neurais Artificiais.

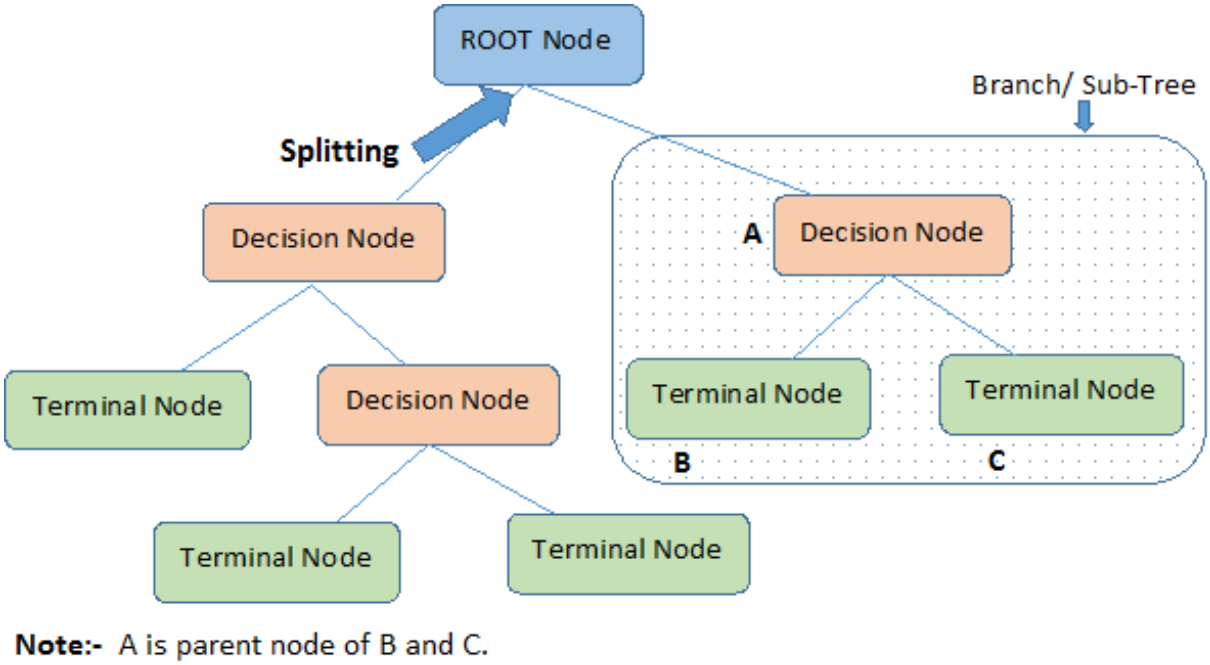
2.2.3 Modelos baseados em Árvores

Árvores, no contexto de computação, são grafos não-direcionados (cada arco que compõe o grafo é antiparalelo a outro arco) nos quais dois vértices se conectam por um único caminho. Por exemplo, o grafo com as arestas $0-1/0-5/1-2/1-3$ compõe uma árvore, pois pode-se percorrer apenas um caminho, sem a formação de um ciclo (FEOFILOFF, 2020). A partir deste princípio, pode-se criar modelos baseados em árvore para segmentar os espaços de predição de forma a permitir que caminhos únicos sejam montados, formando uma estratificação de caminhos. Esse tipo de técnica é não-linear e não possui uma equação definida para reger o funcionamento: ao invés disso, os modelos são formados por conglomerados de funções condicionais *if-else* (KUHN *et al.*, 2013), o que os torna altamente interpretáveis e mais práticos de implementar.

A estrutura de árvores pode ser definida da seguinte forma: o ponto de partida da segmentação é denomidado *root node*, onde se concentra toda a população de estudo. O processo de segmentação resulta na formação de duas ou mais pontas, denominadas de *decision nodes*, no qual ocorre o processo de decisão para um determinado valor numérico e/ou um valor categórico. Esse processo ocorre repetidamente até que se atinja um nó que não se divida mais, denominado de *terminal node* ou *leaf*. *Nodes* que originam novas decisões são denominados *parent nodes*, já os novos, *child nodes*. Os segmentos que conectam os nós são denominados de *branches*.

O processo de remoção de *nodes* da árvore quando necessário é denominado de *pruning*, e é comumente executado quando a árvore atinge um nível de complexidade que dificulta sua interpretabilidade ou quando os resultados do modelo não atingem os valores esperados. A Figura 4 apresenta, em detalhes, a estrutura e terminologia de um modelo baseado em árvore:

Figura 4 – Estrutura e Terminologia de um Modelo baseado em Árvore



Fonte: (VIDHYA, 2020).

O processo de criação de modelos baseados em árvore pode-se ser resumido, geralmente, em duas etapas (GARETH *et al.*, 2013):

1. O espaço de predição, que abringe todos os possíveis valores do preditor X_i, é dividido em J regiões distintas e separadas, delimitados por $R_i, i = 1,2,\ldots,J$.
2. Pra cada predição que caia em R_i, a mesma predição é feita, sendo essa a média das respostas estimadas durante o período de treino.

Para o item 1, o grande desafio é determinar qual o melhor ponto em cada processo de segmentação para a formação dos *nodes*, pois a escolha desse ponto é vital para a confiabilidade do modelo. Por isso, a divisão das regiões deve ser a mais homogênea possível (KUHN *et al.*, 2013). Uma técnica comumente utilizada para a divisão das regiões é a Segmentação Recursiva Binária, que estabelece que, em um modelo *top-down* (do topo da árvore até a base), em cada passo, é feita uma segmentação no melhor ponto no mesmo passo, ao invés de se realizar a segmentação em pontos que levariam, em passos futuros, a uma melhor árvore. Para isso, escolhe-se um

preditor X_i e um ponto de corte S de tal forma que a segmentação do espaço X em:

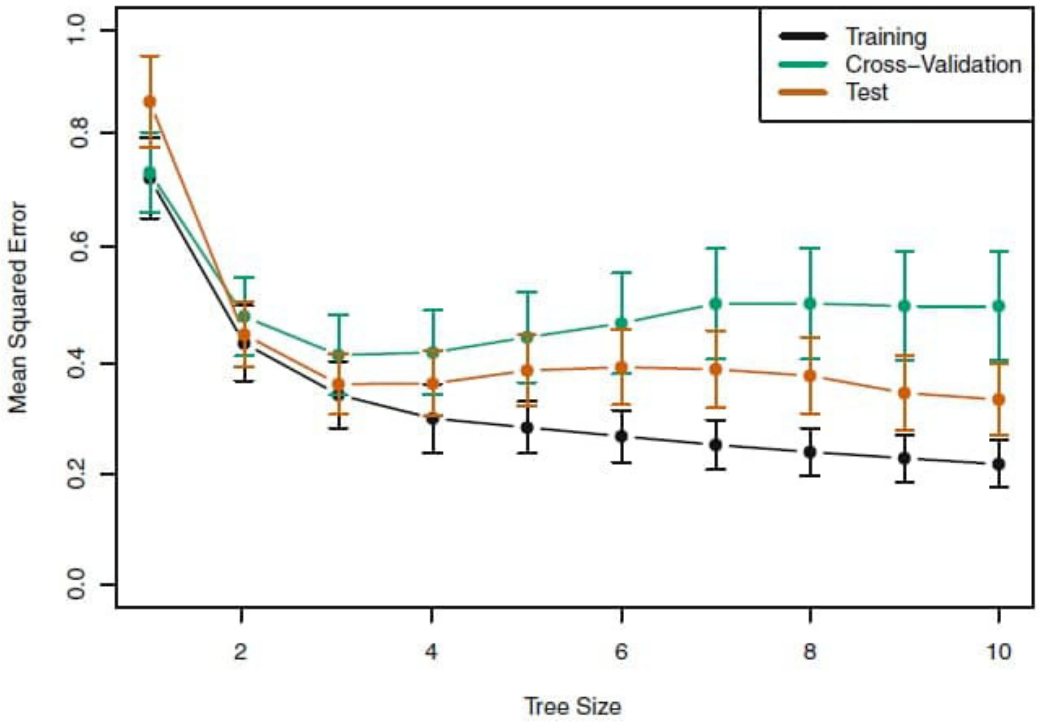
$$R_1(i,S) = \{X|X_i < S\} \quad e \quad R_2R(i,S) = \{X|X_i \geq S\} \tag{2.7}$$

permita que se encontre os valores de i e S tal que minimize a *Root Sum Square* (RSS) ou Soma dos Resíduos Quadrados:

$$RSS = \sum_{i:X_i \in R_1} (Y_i - \hat{Y_{R_1}})^2 + \sum_{i:X_i \in R_2} (Y_i - \hat{Y_{R_2}})^2 \tag{2.8}$$

onde $\hat{Y_{R_i}}$ é a média das respostas das predições de treino em R_j. A cada etapa seguinte, o processo de segmentação é feito em cada uma das novas regiões previamente criadas até que um critério de parada é atingido. A Figura 5 mostra um exemplo de determinação do ponto de corte S para um preditor X_i: após calcular todos os valores de RSS para cada preditor, verifica-se que, nas curvas de treino, *cross-validation* e teste, o valor de S que obtiveram o menor valor de RSS são 10, 3 e 10, respectivamente.

Figura 5 – Exemplo de Determinação do Ponto de Corte S para um preditor a partir da Minimização do RSS.



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

Um dos problemas advindos do processo descrito anteriormente é a possibilidade de *overfitting* devido à alta complexidade. Para evitar que o processo seja refeito com uma árvore muito pequena, opta-se por deixar crescer a árvore T_0 até um determinado tamanho e depois "podar"essa árvore até se obter uma subárvore T, sendo esse procesos denominado *Pruning*, mais especificamente, *Cost Complexity Pruning*. Ao invés de se considerar cada possível subárvore resultante, indexa-se um conjunto de subárvores a um parâmetro α. Para cada α, há uma subárvore $T \subset T_0$ tal que:

$$\sum_{m=1}^{|T|} \sum_{i:X_i \in R_m} (Y_i - \hat{Y_{R_m}})^2 + \alpha|T|, \quad |T| \rightarrow \textit{Número de leafs} \tag{2.9}$$

é minimo. α pode ser encontrado através de algoritmos de *Cross-Validation*, como *K-Fold*, e é responsável por manter o *trade-off* entre a complexidade da subárvore e sua modelagem com os dados de treino: conforme α aumenta, aumenta-se a quantidade de *leafs*, resultando em uma maior complexidade da subárvore.

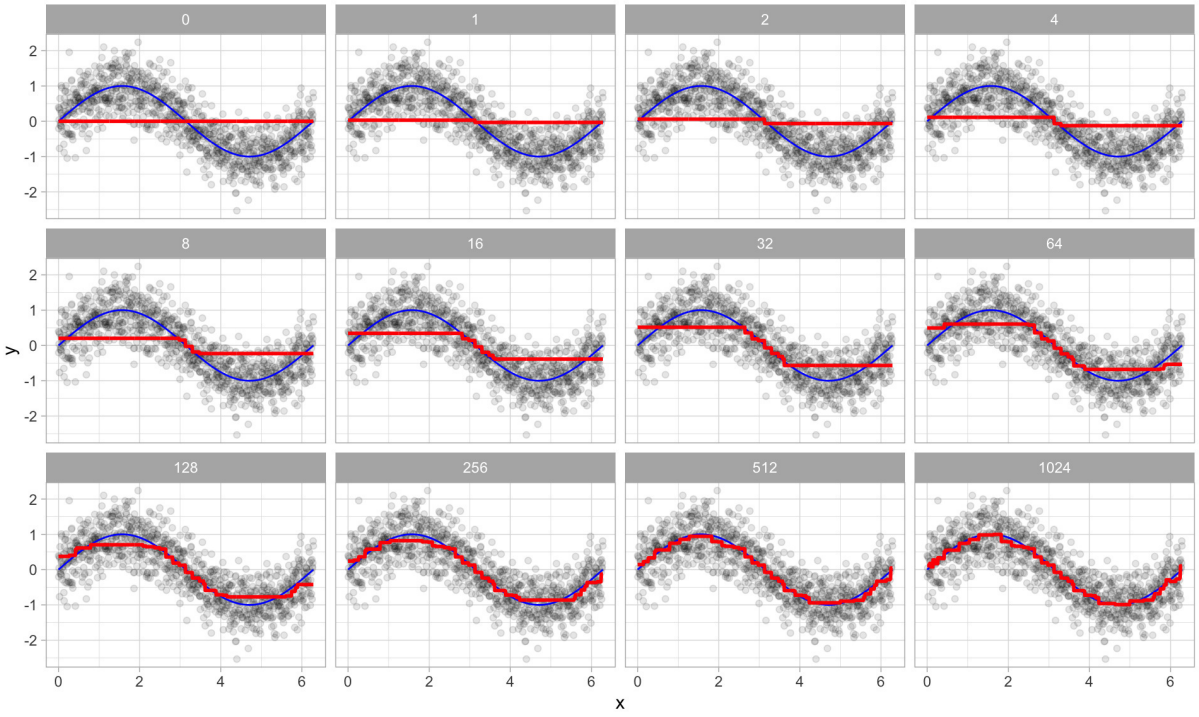
2.2.3.1 Gradient Boosting Machines

Na literatura de árvores de decisão, existem diversas técnicas que podem ser implementadas para melhorar os resultados de predição advindos desses modelos. A grande maioria consiste em recolher predições de múltiplos modelos de árvore para formar um agregado com a menor taxa de erro de previsão possível. Técnicas como *Bagging* e *Random Forest* se utilizam dessa premissa para reduzir a alta variância dos modelos de árvore tradicionais, já que a média de um conjunto de observações possui variância baixa, dado um conjunto de *n* observações com variância σ^2 (GARETH *et al.*, 2013), por isso são empregados em modelos com alta variância e viés baixo; já a técnica de *Boosting*, que será discutida nesta seção, é melhor empregada em modelos com baixa variância e alto viés (BOEHMKE; GREENWELL, 2019).

Boosting é uma técnica que consiste em agregar modelos individuais de forma sequenciada: a partir de um modelo "fraco"(uma árvore com poucas segmentações), é feita uma predição e, a partir dessa observação, monta-se uma nova árvore que foca em reduzir o erro dos segmentos com maiores taxas de erro da última árvore. A Figura 6 apresenta um exemplo de aplicação de *Boosting*: a primeira observação, advinda de uma árvore com uma unica segregação, apresenta grande quantidade de resíduos. Conforme o sequenciamente acontece, cada nova árvore atua em cima dos resíduos da árvore anterior, diminuindo o erro até que um critério

de parada seja atingido. Esse procedimento é ideal para evitar *overfitting*, já que o processso sequencial permite que o modelo agregado utiliza um modelo por vez para gerar observações e corrigir erros ao invés de se usar uma árvore grande para trabalhar em cima dos dados de uma vez só (GARETH *et al.*, 2013).

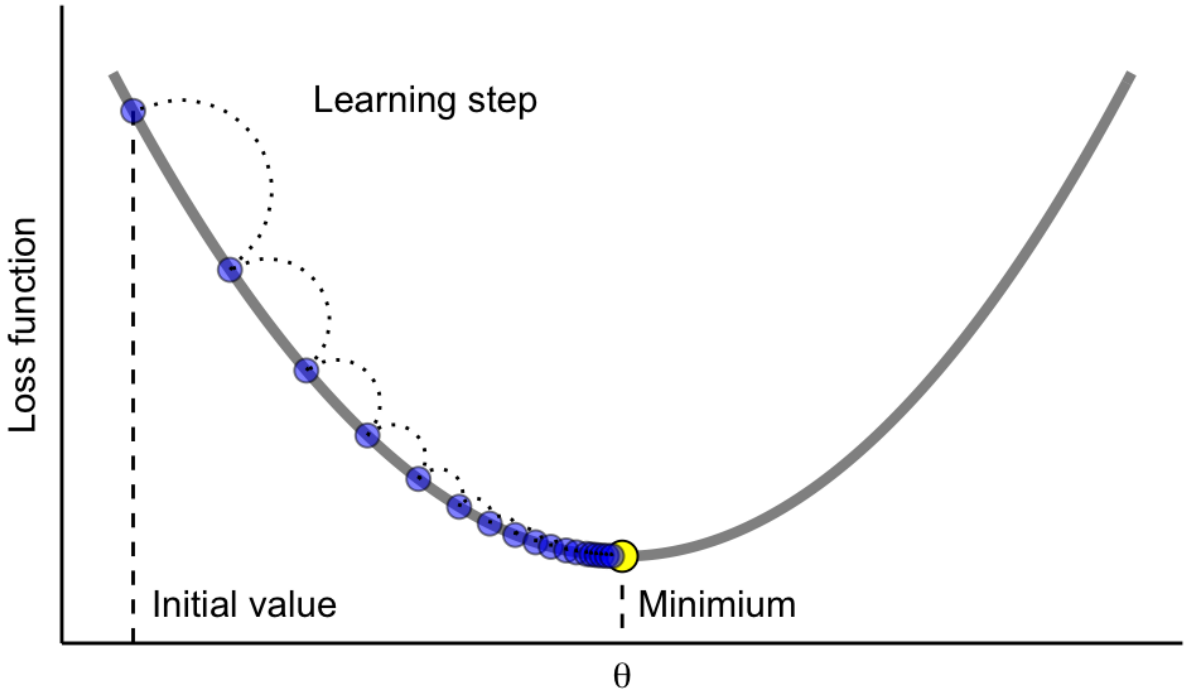
Figura 6 – Exemplo de Agregação de Árvores para o processo de Boosting.



Fonte: (BOEHMKE; GREENWELL, 2019).

O algortimo de *Gradient Boosting*, baseando-se na idéia de minimização de RSS da técnica de *Boosting*, consiste em generalizar a minimização em funções custo além da RSS, como o Erro Médio Absoluto (MAE), através de um algoritmo de *Gradient Descent*, permitindo assim que os parâmetros do modelo sejam tunados de tal forma a obter o declive máximo da curva de gradiente e, consequentemente, minimizar a função custo (BOEHMKE; GREENWELL, 2019), como apresentado na Figura 7.

Figura 7 – Processo de minimização da função custo na curva de Gradiente.



Fonte: (BOEHMKE; GREENWELL, 2019).

2.2.4 Redes Neurais Artificiais

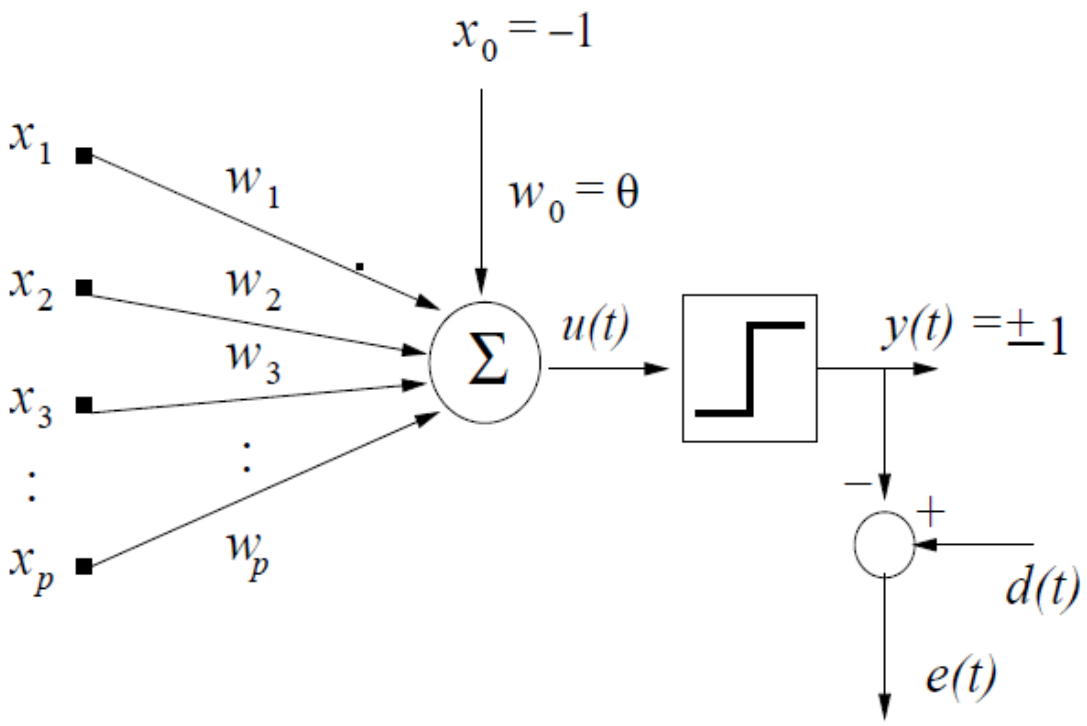
Uma das técnicas mais clássicas e reconhecidas de *Machine Learning* são as Redes Neurais Artificiais (RNA), que nada mais são do que sistemas complexos não lineares de processamento paralelo de informações (HAYKIN, 2010) que visam simular o mesmo grau de sucesso que o cérebro humano: a informação é adquirida e processada pelo ambiente através de um processo de aprendizado, sendo esta informações armazenadas nas conexões interneurais, nos denominados pesos sinápticos. Devido à sua complexa estrutura não linear, é uma técnica altamente adaptiva e facilmente aplicável no contexto de Aprendizado Supervisionado e Não Supervisionado, assim como altamente interpretável.

A estrutura básica de uma rede neural chama-se neurônio (assim como o neurônio biológico) e é constituído, independente do modelo adotado, de três estruturas básicas:

- Um conjunto de sinapses, que correspondem as entradas de sinais dentro do modelo neural: cada entrada apresenta um peso W_{kj}, onde k é o neurônio correspondente e j, a sinapse específica;
- Um somador dos sinais de entrada com seus respectivos pesos;
- Uma função de ativação para regular a amplitude do valor de saída do neurônio a depender do problema específico;

Um dos primeiros modelos neurais desenvolvidos foi o Perceptron de Rosenblatt, cuja estrutura é mostrada na Figura 8.

Figura 8 – Perceptron de Rosenblatt



Fonte: (WEBB, 2002).

Perceptrons realizam combinações lineares entre o sinal de entrada x_i e o peso w_i e expressam suas saídas de forma binária a partir de um *threshold*. Essa expressão é demonstrada pela Equação 2.10:

$$y(t) = \begin{cases} 0 & \text{se } \sum_j w_j x_j \leq Threshold \\ 1 & \text{se } \sum_j w_j x_j \geq Threshold \end{cases} . \tag{2.10}$$

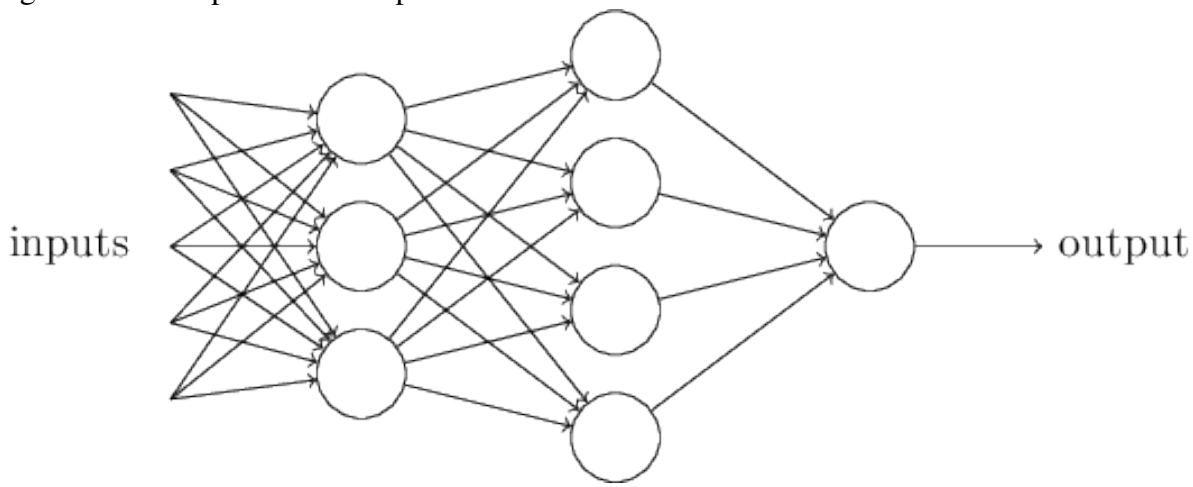
Variando-se os pesos e o *Threshold*, é possível obter diversos modelos de predição (NIELSEN, 2019).

2.2.4.1 Multilayer Perceptron

Como o perceptron corresponde a apenas uma parte básica do sistema de redes neurais, a tendência é de se construírem redes longas e complexas de perceptrons para auxiliar

na tomada de decisões mais precisas (NIELSEN, 2019). Para isso, existem os Perceptrons de Múltiplas Camadas ou *Multilayer Perceptron* (MLP), que se caracterizam como agregados de perceptrons ao longo de várias camadas de sinapses, cada uma com seus respectivos pesos, como mostrar a Figura 9.

Figura 9 – Perceptron de Múltiplas Camadas



Fonte: (NIELSEN, 2019).

Cada coluna de perceptrons está processando várias informações através dos pesos sinápticos, resultando nas colunas subsequentes processando novas informações através das saídas das colunas anteriores. Devido ao aumento da complexidade das redes neurais baseadas em MLP, é comum enunciar a saída da rede neural através da função de ativação, como enunciado na Equação 2.11:

$$y(t) = \alpha(z) \tag{2.11}$$

onde z é a combinação linear das saídas de cada perceptron + o *bias*, que nada mais é que uma inversão do *Threshold* e uma medida que avalia o quão fácil é para o perceptron atingir um determinado valor, e α é a função de ativação da rede neural, existindo vários tipos.

2.3 Inteligência Computacional Aplicada à Robótica

2.3.1 Aplicações em Manipuladores para Cirurgias Médicas

No campo de cirurgias médicas, a aplicação de manipuladores robóticos tem se tornado uma opção viável devido à grande precisão que o planejamento de trajetória desses equipamentos possuem, o que possibilita sua utilização em cirurgias mais complexas e de alto

risco: fatores como a destreza melhorada, maior espaço de trabalho, visão 3D amplificada para uso dos cirurgiões, entre outros, aumentam a procura por robôs especializados em cirurgia (HU *et al.*, 2020). Entre alguns exemplos de cirurgias em que manipuladores robóticos são utilizados, destacam-se *Minimally Invasive Sugeries* (MIS) ou procedimentos minimamente invasivos, onde o equipamento cirúrgico é utilizado para adentrar em pequenas incisões, cirurgias plásticas, neurocirurgias, dentre outras. Um exemplo recente de aplicação é a utilização de manipuladores durante a pandemia de COVID-19, para mitigar os efeitos de infecção e contaminação de pacientes e cirurgiões (ZEMMAR *et al.*, 2020)

A utilização de ICA para cirurgias operadas por manipuladores robóticos pode variar, desde a utilização de algoritmos de redes neurais para auxiliar procedimentos do tipo MIS através da melhora da acurácia (SU *et al.*, 2020) e da adaptação a erros através de controladores PID (LI *et al.*, 2017); uso de visão computacional para a melhora de imagens 3D durante cirurgias, permitindo a identificação, por exemplo, de nódulos de câncer em orgãos específicos (KHOSRAVI *et al.*, 2018); dentre outras aplicações.

Embora diversas aplicações e técnicas tenham sido mencionadas, a variedade em que as pesquisas se desenvolvem no campo de aplicações em cirurgias é muito extensa, portanto esta fundamentação teórica está longe de ser uma análise completa. Porém, os métodos de modelagem citadoss serão implementados e avaliados no contexto proposto por este trabalho, apresentado mais detalhadamente no Capítulo 3.

3 METODOLOGIA

Neste capítulo, será apresentada a Metodologia utilizada para a aplicação dos métodos de modelagem citados no Capítulo 2: primeiramente, uma análise da planta de estudo do trabalho e dos dados extraídos desta; em seguida, uma apresentação da implementação dos métodos de modelagem; e por fim, uma apresentação das métricas de avaliação que serão utilizadas para medir a performance das implementações. Todos os códigos produzidos neste trabalho foram disponibilizados em: *https://github.com/KyleStorm1812/TCC—Guilherme-Carvalho-da-Silva*.

3.1 Planta de Estudo

Para realizar as experimentações e implementações práticas com os métodos de modelagem citados no Capítulo 2, optou-se por utilizar, para fins de comodidade e dificuldade de acesso a equipamentos devido à pandemia de COVID-19, uma simulação com dados com base em um manipulador robótico. Para este trabalho, utilizou-se os dados do manipulador robótico utilizado por (ORHANLI, 2021), que tem como objetivo obter os ângulos, as velocidades angulares e o arranque mínimo de um manipulador robótico planar 2-DOF utilizando integração numérica em equações diferencias não-rígidas para se obter os parâmetros de velocidade e tempo das juntas do manipulador.

Ao final da experimentação dos dados, obtemos uma base de dados contendo as informações das velocidades e dos arranques das duas juntas do manipulador. As características da base estão contidas na Tabela 1.

Tabela 1 – Características Gerais da Base de Dados do Manipulador Robótico

Dados	Valor
Quantidade de Amostras	37
Quantidade de Variáveis	4

Fonte: O autor

Como o resultado que se espera obter são valores numéricos não limitados pela binaridade de 0 e 1, tanto os métodos de *Gradient Boosting Machine* quanto de MLP serão configurados para resolver um problema de regressão numérica.

3.1.1 Descrição dos Dados

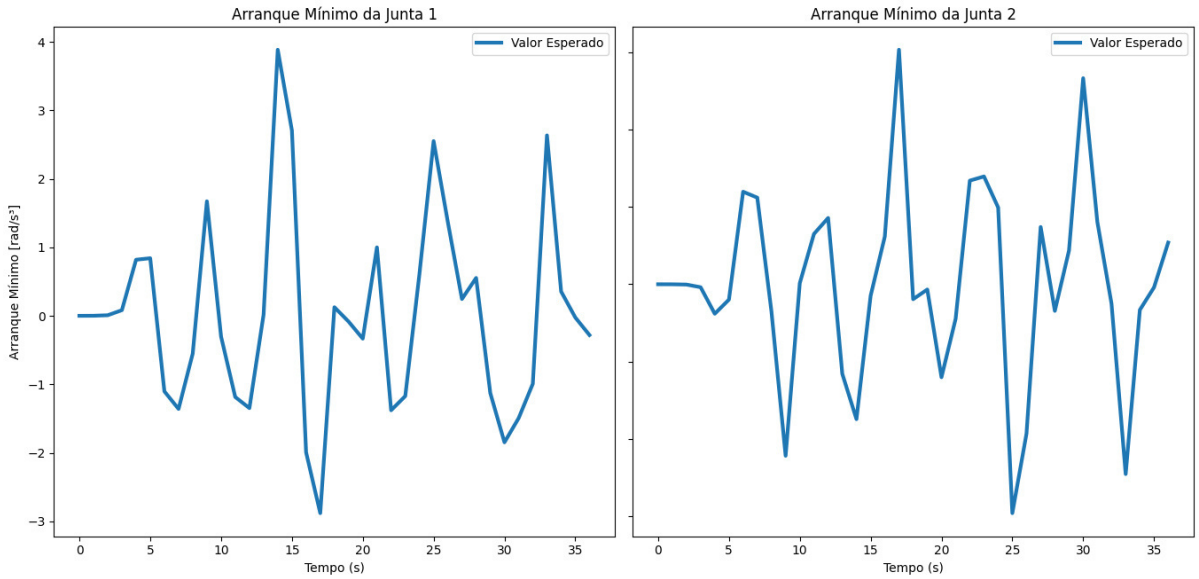
Os dados são descritos da seguinte forma:

- VEL_JUNTA_1: valor das velocidades da Junta 1 de (ORHANLI, 2021) ao longo do tempo.
- ARR_JUNTA_1: valor dos arranques da Junta 1 de (ORHANLI, 2021) ao longo do tempo.
- VEL_JUNTA_2: valor das velocidades da Junta 2 de (ORHANLI, 2021) ao longo do tempo.
- ARR_JUNTA_2: valor das arranques da Junta 2 de (ORHANLI, 2021) ao longo do tempo.

Como o objetivo deste trabalho é implementar métodos supervisionados de modelagem para se obter valores de arranque mínimos, as variáveis ARR_JUNTA_1 e ARR_JUNTA_2 serão as variáveis dependentes do modelo, enquanto VEL_JUNTA_1 e VEL_JUNTA_2 serão as variáveis independentes. Cada uma das velocidades das juntas irá influenciar na obtenção de cada arranque, portanto, para cada junta serão criados algoritmos de modelagem tomando como variáveis de entradas as velocidades de cada junta.

A figura 10 mostram o comportamento das variáveis de saída anteriormente citadas, os arranques mínimos das juntas 1 e 2, ao longo do tempo. O esperado deste trabalho é obter resultados que se aproximem o máximo possível dessas curvas ao longo do tempo.

Figura 10 – Arranques Mínimos Esperados das Juntas 1 e 2



Fonte: O autor.

3.2 Implementação

Todos os estudos e implementações deste trabalho foram feitos na linguagem de programação *Python*, utilizando, primariamente, o pacote *sklearn* (PEDREGOSA *et al.*, 2011), especializado em implementar ferramentas para análise preditiva e modelagem.

3.2.1 Gradient Boosting Machine

A implementação do algoritmo de *Gradient Boosting* foi utilizada usando a classe *GradientBoostingRegressor*, cujos hiperparâmetros de tunagem utilizados estão listados a seguir (PEDREGOSA *et al.*, 2011):

- n_estimators $\longrightarrow$ O número de estágios de *Boosting* do algoritmo.
- learning_rate $\longrightarrow$ Determina a contribuição de cada árvore para o preditor final e a velocidade com o qual o algoritmo utiliza o *Gradient Descent*.
- min_samples_split $\longrightarrow$ O número mínimo de amostras para que um nó se divida.
- min_samples_leaf $\longrightarrow$ O número mínimo de amostras para que se tornem *leafs*.
- max _depth $\longrightarrow$ O número máximo de nós na árvore.
- max_leaf_nodes $\longrightarrow$ O número máximo de *leafs* na árvore.

Para cada junta, foram listadas as configurações na Tabela 2.

Tabela 2 – Hiperparâmetros do Gradient Boosting

Hiperparâmetros	Junta 1	Junta 2
n_estimators	400	500
learning_rate	0.5	0.5
min_samples_split	6	5
min_samples_leaf	3	3
max _depth	3	4
max_leaf_nodes	10	5

Fonte: O autor

3.2.2 Multilayer Perceptron

A implementação do algoritmo de *Multilayer Perceptron* foi utilizada usando a classe *MLPRegressor*, cujos hiperparâmetros de tunagem utilizados estão listados a seguir (PEDREGOSA *et al.*, 2011):

- max_iter $\longrightarrow$ O número máximo de iterações da RNN, caso não seja atingida a convergên-

cia antes desse número.

- learning_rate = "adaptive" $\longrightarrow$ O método de aprendizado da RNN. "adaptive"indica que a atualização dos pesos a cada iteração é mantida constante enquanto a perda referente ao treino da RNN for decrescente.
- hidden_layer_sizes $\longrightarrow$ O número de neurônios em cada camada oculta da RNN. Como indicado na Tabela 3, foram adicionadas três camadas ocultas: uma com 50 neurônios, outra com 100 e mais uma com 50.

Para cada junta, foram listadas as configurações na Tabela 3.

Tabela 3 – Hiperparâmetros do Multilayer Perceptron

Hiperparâmetros	Junta 1	Junta 2
max_iter	2000	2000
learning_rate	"adaptive"	"adaptive"
hidden_layer_sizes	(50,100,50)	(50,100,50)

Fonte: O autor

3.3 Métricas de Avaliação

Para verificar se, de fato, houve convergência dos arranques mínimos preditos com os arranques mínimos reais, é preciso estabelecer métricas de avaliação para validar sua relevâncias estatísticas. Uma forma informal de avaliação é verificar graficamente se os resultados preditos estão próximos dos resultados reais. Porém, vale ressaltar que mesmo que visualmente o resultado pareça promissor, isso não necessariamente reflete que o resultado é estatísticamente relevante: é possível, inclusive, que visualmente os gráficos estejam perfeitos, mas as métricas indiquem que a modelagem não foi tão promissora quanto parece.

Para fins de Regressão Numérica, uma das métricas mais populares e eficientes é o Coeficiente de Determinação (R^2): essa métrica indica a proporção da variância dos valores preditos que pode ser explicada utilizando os valores de entrada como parâmetros. Matematicamente, essa métrica pode ser traduzida como (GARETH *et al.*, 2013):

$$R^2 = \frac{TSS - RSS}{TSS} \tag{3.1}$$

onde TSS = $\sum(Y_i - \hat{Y})^2$ é a Soma Total dos Quadrados e RSS é a Soma Residual dos Quadrados. Como TSS mede a variabilidade da saída Y antes da regressão ser realizada e RSS, a variabilidade

não descrita da performance da regressão, R^2 indica a proporção de variabilidade da saída Y conforme tenha-se informações da entrada X. Essa medida é representada uma proporção de 0 a 1: quanto mais próximo de 1, mais a variabilidade da resposta Y pode ser explicada com a entrada X. Para fins deste trabalho, serão consideradas ambas as métricas citadas como válidas para a avaliação dos métodos de modelagem. No pacote *sklearn*, a classe *r2_score* fornece o valor de R^2 conforme são fornecidos os valores reais e preditos da modelagem.

Outra métrica de Regressão Numérica popular é o Erro Quadrático Médio ou *Mean Squared Error* (MSE), que consiste na avaliação do quanto a resposta predita se aproxima da resposta esperada (GARETH *et al.*, 2013). Essa métrica é matematicamente representada como:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{3.2}$$

onde $\hat{Y}_i$ é o valor predito para a i-nésima observação da entrada X_i. Quanto menor for o valor do MSE, mais próximo do valor real a resposta predita será. No pacote *sklearn*, a classe *mean_squared_error* retorna os valores de MSE.

Uma última forma em que se podem avaliar se a modelagem ocorreu adequadamente é verificar se há algum problema na distribuição das amostras de entrada. Teste não-paramétricos são adequados para analisar a qualidade dos dados de entrada, especialmente para conjuntos de dados com distribuições assimétricas como a deste trabalho, já que não é possível assumir a normalidade dos dados: um desses testes, o Teste de Kolmogorov-Smirnov, quantifica a distância entre uma amostra e uma distribuição de probabilidade acumulada (KS1) ou entre duas amostras (KS2). A partir disso, ele consegue determinar, a partir da máxima distância, se houveram mudanças nas distribuição dos dados ao longo dos testes (PRINS, 2013). Para os fins deste trabalho, será utilizada a métrica de KS1 para avaliar as distribuições entre treino e validação. Uma função criada pelo próprio autor foi utilizada para calcular essa métrica.

4 RESULTADOS

Nesta seção, são apresentados os resultados da implementação de cada método de modelagem para obtenção dos arranques mínimos das duas juntas, assim como as avaliações de perfomance comforme as métricas de avaliação citadas no Capítulo 3.

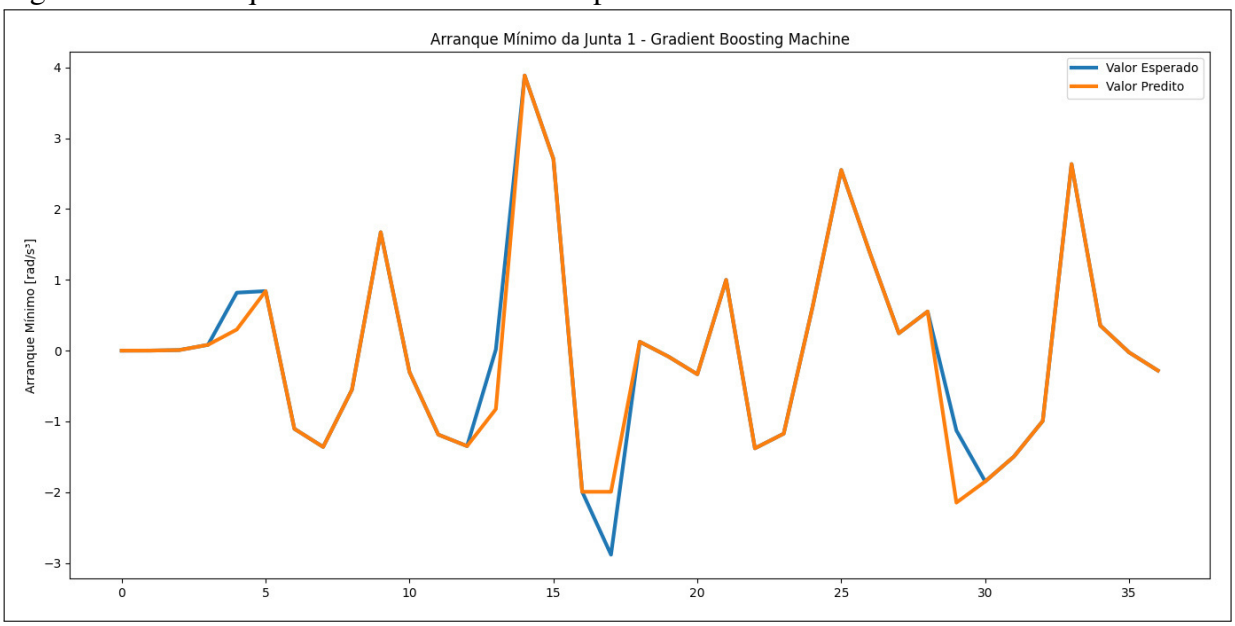
4.1 Resultados do Gradient Boosting Machine

A partir das configurações de performance citadas no Capítulo 3, foram encontrados os seguintes resultados para cada modelo de *Gradient Boosting* utilizado para cada junta:

4.1.1 Junta 1

A Figura 11 apresenta o arranque mínimo modelado pelo *Gradient Boosting* na junta 1, em comparação com o arranque real que a simulação obteve.

Figura 11 – Arranque Mínimo da Junta 1 - Esperado e Predito

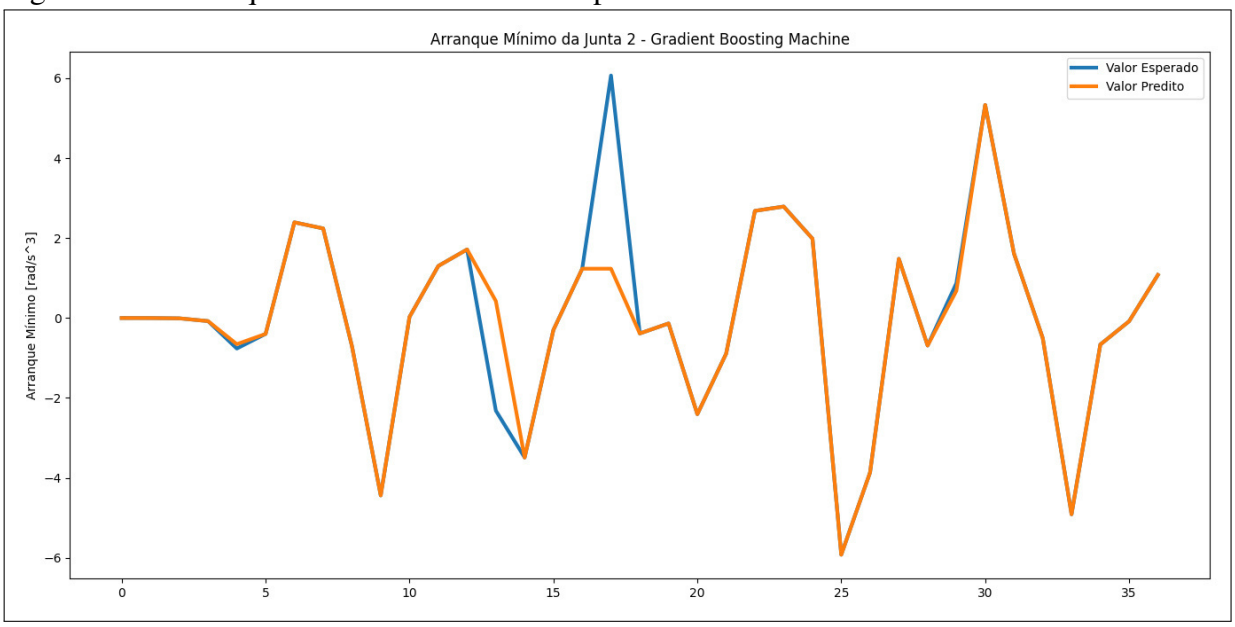


Fonte: O autor.

4.1.2 Junta 2

A Figura 12 apresenta o arranque mínimo modelado pelo *Gradient Boosting* na junta 1, em comparação com o arranque real que a simulação obteve.

Figura 12 – Arranque Mínimo da Junta 2 - Esperado e Predito



Fonte: O autor.

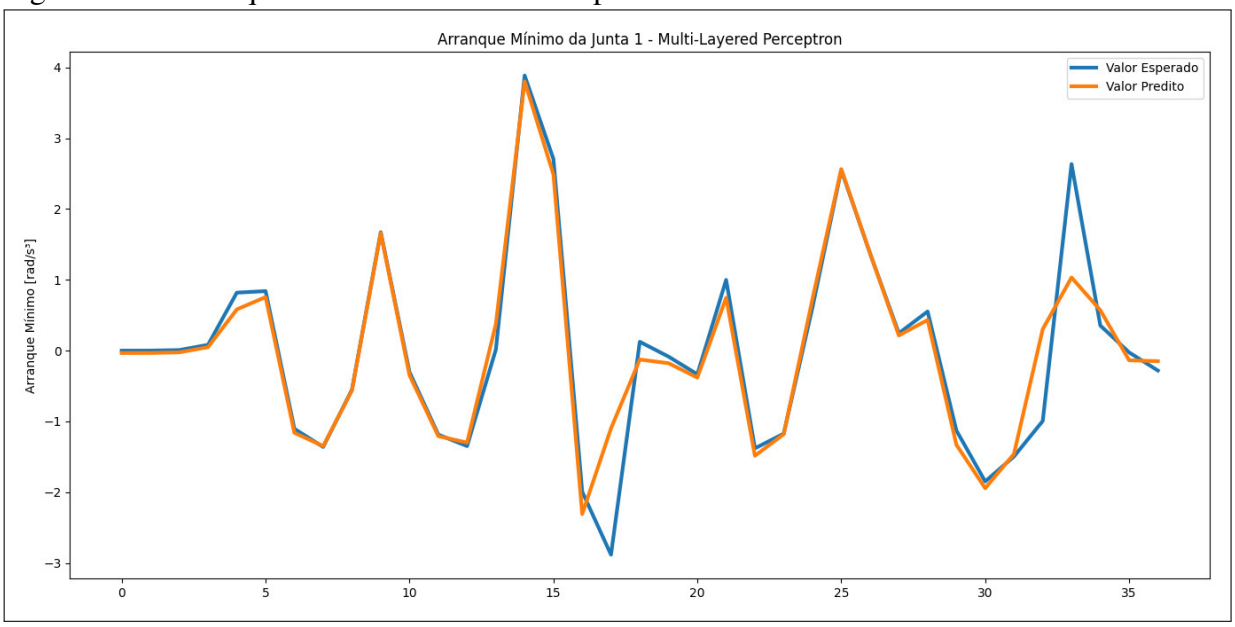
4.2 Resultados do Multilayer Perceptron

A partir das configurações de performance citadas no Capítulo 3, foram encontrados os seguintes resultados para cada modelo de *Multilayer Perceptron* utilizado para cada junta:

4.2.1 Junta 1

A Figura 13 apresenta o arranque mínimo modelado pelo *Multilayer Perceptron* na junta 1, em comparação com o arranque real que a simulação obteve.

Figura 13 – Arranque Mínimo da Junta 1 - Esperado e Predito

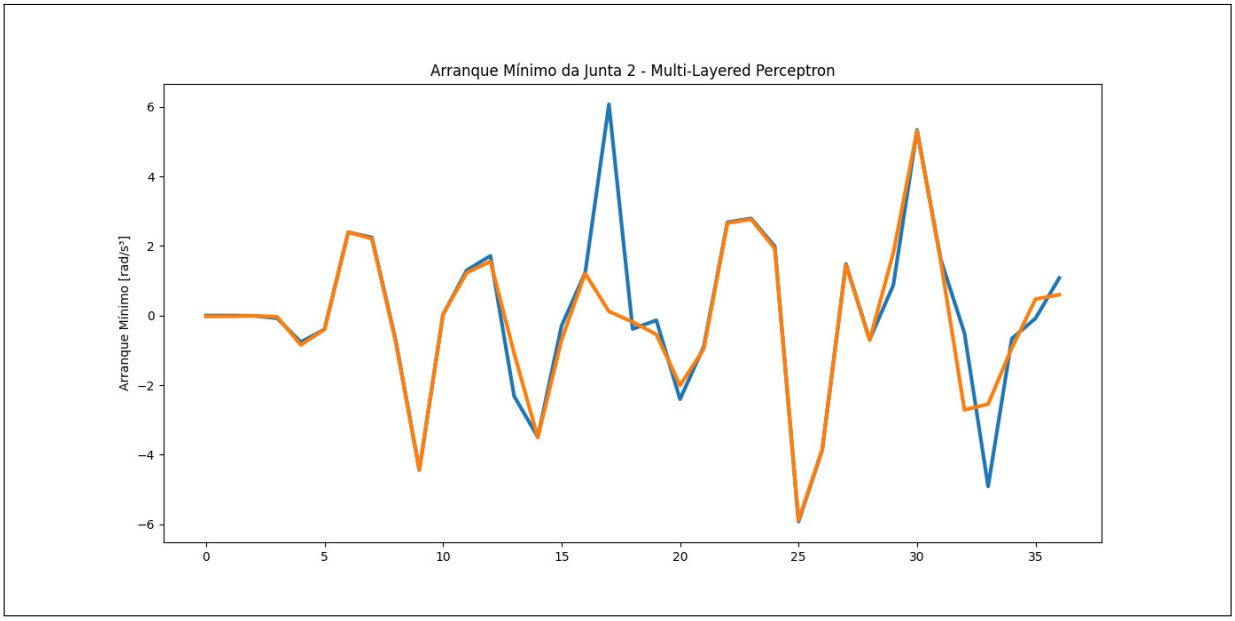


Fonte: O autor.

4.2.2 Junta 2

A Figura 14 apresenta o arranque mínimo modelado pelo *Multilayer Perceptron* na junta 1, em comparação com o arranque real que a simulação obteve.

Figura 14 – Arranque Mínimo da Junta 2 - Esperado e Predito



Fonte: O autor.

4.3 Discussão dos Resultados e Avaliação de Performance

Avaliando-se os resultados da saída dos métodos de modelagem nas Figuras 11 a 14, observa-se que o *Gradient Boosting Machine* obteve curvas mais suaves visualmente do que o *Multilayer Perceptron*, especialmente na junta 1. O mesmo fenômeno observa-se na junta 2: vale ressaltar que no eixo X de ambos os gráficos, os dois métodos tiveram dificuldades em conseguir prever corretamente o pico que ocorre no segundo 18, o que indica que ambos os métodos não foram suficientemente robustos para se adaptarem a mudança brusca de resultados, optando por traçar trajetórias mais suaves, o que pode ser, em termos de implantação prática, algo mais desejável. A Figura 15 apresenta um comparativo entre os resultados obtidos de cada método e os valores esperados em cada junta.

Figura 15 – Conjunto de Resultados de Arranque Mínimo das Juntas - Esperado e Predito

Fonte: O autor.

Avaliando-se as distribuições das amostras de entrada, calcularam-se os valores de KS1 encontrados na Tabela 4. Como o KS1 avalia a máxima distância entre as amostras da distribuição, o indicativo de que o valor da métrica foram baixos indica que, entre a população de treino e a população de validação, não houveram mudanças significativas em suas características, o que é ideal para fins de modelagem. Para ter uma medida de referência, optou-se por considerar faixas de KS1 abaixo de 10 como ideais, portanto, ambas as variáveis de entrada apresentam baixa variedade em suas distribuições.

Com relação ao Coeficiente de Determinação, a Tabela 5 apresenta os valores de

Tabela 4 – Valores de $KS1$ para as variáveis de entrada do Manipulador Robótico

Variáveis	$KS1$
Velocidade da Junta 1	6.797707
Velocidade da Junta 2	4.258804

Fonte: O autor.

R^2 obtidos para cada junta. Observa-se que, em coerência com a avaliação visual, ambos os métodos apresentam alta proporção de variabiliade na saída, já que estão muito próximos de 1: porém, observa-se que o *Gradient Boosting Machine* obteve os resultados mais satisfatórios para o intuito deste trabalho. Este é um resultado coerente com o propósito do algoritmo, que é extrair o máximo de informações de cada iteração e minimizar o erro o máximo possível: o *Multilayer Perceptron* apresenta resultados próximos, mas é computacionalmente mais custoso e mais complexo, o que dificulta justificar seu uso em relação ao *Gradient Boosting*.

Tabela 5 – Valores de R^2 para cada junta do Manipulador Robótico

Método de Modelagem	R^2 - Junta 1	R^2 - Junta 2
Gradient Boosting Machine	0.9621824317787345	0.8645842780601787
Multilayer Perceptron	0.9317809809281942	0.7833019580490528

Fonte: O autor.

Por fim, com relação ao Erro Quadrático Médio, a Tabela 6 apresenta os valores de MSE obtidos para cada junta. Observa-se que, em conformidade com o fenômeno observado na Tabela 5, o algortimo de *Gradient Boosting* apresentou melhores resultados em comparação ao *Multilayer Perceptron*: é perceptível nesses resultados que o Gradient Boosting é o método mais próximo de zerar o MSE e, portanto, é o que a resposta predita mais se aproxima do valor esperado. O *Multilayer Perceptron*, apesar de eficiente, apresenta resultados mais distantes do que o observado no *Gradient Boosting*, o que torna ainda mais justificável a preferência por este algoritmo.

Tabela 6 – Valores de MSE para cada junta do Manipulador Robótico

Método de Modelagem	MSE - Junta 1	MSE - Junta 2
Gradient Boosting Machine	0.0757902025792758	0.8359245288793052
Multilayer Perceptron	0.18005240899619762	1.1485128346655071

Fonte: O autor.

5 CONCLUSÕES E TRABALHOS FUTUROS

A expansão das aplicações da força de trabalho dos robôs nas últimas décadas para áreas fora da indústria, como o setor agrícola e o de serviços, foi essencial para o desenvolvimento de pesquisas não só para a Robótica em si, mas para áreas correlatas como a Inteligência Computacional Aplicada. A utilização de Redes Neurais, algoritmos baseados em árvores e outros métodos de implementação de Inteligência Computacional vem proporcionando uma revolução na automação de processos repetitivos e de difícil destreza por parte dos humanos. No campo hospitalar, por exemplo, a utilização de robôs para cirurgias médicas, desde as mais simples até as consideradas impossíveis para cirurgiões, e a consequente procura pelo desenvolvimento da Inteligência Computacional Aplicada em cima deles vêm melhorando a qualidade do setor de serviços hospitalares, o que por consequência ajuda a melhorar a qualidade de vida dos pacientes e a qualidade do trabalho dos médicos responsáveis.

Neste trabalho, foram avaliadas, com sucesso, as performances de métodos de modelagem para a determinação de um problema recorrente na aplicação de manipuladores robóticos, que é a determinação dos arranques mínimos das juntas para que o robô não entre em ressonância. Analisando-se graficamente, observou-se que os valores preditos pelos métodos se aproximam com bastante precisão dos valores: não só isso, mas a qualidade dos dados avaliada pelo Teste de Kolmogorov-Smirnov garantiu que os dados de treino e validação não sofreram mudanças significativas em suas distribuições, e as métricas de R^2 e MSE garantiram que a proporção de variabilidade da saída e o grau de aproximação da resposta obtida com a resposta esperada, respectivamente, fossem adequadas para as avaliações, resultando em ambas as performances serem estatisticamente corretas e viáveis.

Para trabalhos futuros, pode-se adaptar os métodos utilizados neste projeto em bases de dados mais complexos, como manipuladores robóticos com mais DOFs, assim como avaliar novos métodos de modelagem nessas bases. Também pode-se estudar depois avaliações de Interpretabilidade dos métodos e como as variáveis influenciam nas respostas obtidas.

REFERÊNCIAS

BOEHMKE, B.; GREENWELL, B. M. **Hands-on machine learning with R**. [S.l.]: CRC Press, 2019.

CRAIG, J. J. **Introduction to robotics: mechanics and control, 3/E**. [S.l.]: Pearson Education India, 2009.

FEOFILOFF, P. **Algoritmos para Grafos via Sedgewick**. 2020. Disponível em: <v>.

FREEMAN, P. Minimum jerk trajectory planning for trajectory constrained redundant robots. 2012.

GARETH, J.; DANIELA, W.; TREVOR, H.; ROBERT, T. **An introduction to statistical learning: with applications in R**. [S.l.]: Spinger, 2013.

HAYKIN, S. **Neural networks and learning machines, 3/E**. [S.l.]: Pearson Education India, 2010.

HU, Y.; SU, H.; CHEN, G.; FERRIGNO, G.; MOMI, E. D.; KNOLL, A. Hierarchical optimization control of redundant manipulator for robot-assisted minimally invasive surgery. In: IEEE. **2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.], 2020. p. 2929–2934.

KELLY, R.; DAVILA, V. S.; PEREZ, J. A. L. **Control of robot manipulators in joint space**. [S.l.]: Springer Science & Business Media, 2006.

KHOSRAVI, P.; KAZEMI, E.; IMIELINSKI, M.; ELEMENTO, O.; HAJIRASOULIHA, I. Deep convolutional neural networks enable discrimination of heterogeneous digital pathology images. **EBioMedicine**, Elsevier, v. 27, p. 317–328, 2018.

KUHN, M.; JOHNSON, K. *et al.* **Applied predictive modeling**. [S.l.]: Springer, 2013. v. 26.

KYRIAKOPOULOS, K. J.; SARIDIS, G. N. Minimum jerk path generation. In: IEEE. **Proceedings. 1988 IEEE International Conference on Robotics and Automation**. [S.l.], 1988. p. 364–369.

KYRIAKOPOULOS, K. J.; SARIDIS, G. N. Minimum jerk trajectory planning for robotic manipulators. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. **Cooperative Intelligent Robotics in Space**. [S.l.], 1991. v. 1387, p. 159–164.

KYRIAKOPOULOS, K. J.; SARIDIS, G. N. *et al.* Minimum jerk for trajectory planning and control. **Robotica**, v. 12, n. 2, p. 109–113, 1994.

LI, Y.; LI, S.; CABALLERO, D.; MIYASAKA, M.; LEWIS, A.; HANNAFORD, B. Improving control precision and motion adaptiveness for surgical robot with recurrent neural network. In: IEEE. **2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.], 2017. p. 3538–3543.

LU, S.; ZHAO, J.; JIANG, L.; LIU, H. Solving the time-jerk optimal trajectory planning problem of a robot using augmented lagrange constrained particle swarm optimization. **Mathematical Problems in Engineering**, Hindawi, v. 2017, 2017.

MARR, B. **What is Industry 4.0? Here's A Super Easy Explanation For Anyone**. Forbes Magazine, 2019. Disponível em: <https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/#3dc628ad9788>.

NEVES, F. **Agricultura de precisão erobótica**. 2020. Disponível em: <https://venturus.org.br/agricultura-de-precisao-e\protect\begingroup\immediate\write\@unused\def\MessageBreak˙\let\protect\edefYoumayprovideadefinitionwith\MessageBreak\DeclareUnicodeCharacter\errhelp\let\def\MessageBreak˙(inputenc)\def\errmessagePackageinputencError: UnicodecharacterâĂŕ(U+202F)\MessageBreaknotsetupforusewithLaTeX.˙˙Seetheinputencpackagedocumentationforexplanation.˙TypeH<return>forimmediatehelp\endgrouprobotica/>.

NIELSEN, M. A. **Neural networks and deep learning**. [S.l.]: Determination press San Francisco, CA, 2019. v. 25.

OLIVEIRA, P. W.; BARRETO, G. A. *et al.* A general framework for optimal tuning of pid-like controllers for minimum jerk robotic trajectories. **Journal of Intelligent & Robotic Systems**, Springer, p. 1–20, 2020.

ORHANLI, T. **Forward Dynamics of Planar 2-DOF Robot Manipulator**. 2021. Disponível em: <https://www.mathworks.com/matlabcentral/fileexchange/69756-forward-dynamics-of-planar-2-dof-robot-manipulator/>.

OWEN-HILL, A. **5 Ways Robotics Are Used in Medicine and Healthcare**. 2021. Disponível em: <https://blog.robotiq.com/5-ways-cobots-are-used-in-medicine-and-healthcare#:~:text=Surgeryisthemostcommonly,,sanitation,andprescriptiondispensing.>

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PRINS, J. Nist/sematech e-handbook of statistical methods, chapter 3. **NIST/SEMATECH e-Handbook of Statistical Methods**, 2013.

QURESHI, M. O.; SYED, R. S. The impact of robotics on employment and motivation of employees in the service sector, with special reference to health care. **Safety and health at work**, Elsevier, v. 5, n. 4, p. 198–202, 2014.

RUSSEL, S.; NORVIG, P. *et al.* **Artificial intelligence: a modern approach**. [S.l.]: Pearson Education Limited, 2013.

SICILIANO, B.; SCIAVICCO, L.; VILLANI, L.; ORIOLO, G. **Robotics: modelling, planning and control**. [S.l.]: Springer Science & Business Media, 2010.

SIMPLÍCIO, P. V. G.; LIMA, B. R. Manipuladores robóticos industriais. **Caderno de Graduação-Ciências Exatas e Tecnológicas-UNIT-SERGIPE**, v. 3, n. 3, p. 85, 2016.

SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. *et al.* **Robot modeling and control**. [S.l.: s.n.], 2006.

SU, H.; HU, Y.; KARIMI, H. R.; KNOLL, A.; FERRIGNO, G.; MOMI, E. D. Improved recurrent neural network-based manipulator control with remote center of motion constraints: Experimental results. **Neural Networks**, Elsevier, v. 131, p. 291–299, 2020.

TECNOLOGIA, P. p. A. P. **Inteligência computacional: veja quais são suas vantagens!** 2020. Disponível em: <https://www.meupositivo.com.br/panoramapositivo/inteligencia-computacional/>.

TURING, I. B. A. Computing machinery and intelligence-am turing. **Mind**, v. 59, n. 236, p. 433, 1950.

VIDHYA, A. **Tree Based Algorithms: Implementation In Python R**. 2020. Disponível em: <https://www.analyticsvidhya.com/blog/2016/04/tree-based-algorithms-complete-tutorial-scratch-in-python/>.

WEBB, A. Statistical pattern recognition, john wiley & sons. **New York, USA**, 2002.

WLODZISLAW, D. What is computational intelligence and what could it become. **Challenges for Computational Intelligence**, p. 1–11, 2007.

ZEMMAR, A.; LOZANO, A. M.; NELSON, B. J. The rise of robots in surgical environments during covid-19. **Nature Machine Intelligence**, Nature Publishing Group, v. 2, n. 10, p. 566–572, 2020.