Gustavo Graziosi Silva

Solução de cinemática de plataforma de Stewart baseada em aprendizado de máquina

Campinas

2/2020

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Gustavo Graziosi Silva - RA: 169320
Curso: Engenharia Elétrica Integral - 11

Solução de cinemática de plataforma de Stewart baseada em aprendizado de máquina

Orientador: Mateus Giesbrecht

Campinas

2/2020

Resumo

Neste trabalho foi estudado um manipulador robótico paralelo conhecido como plataforma de Stewart. A plataforma de Stewart é composta por uma base fixa ligada a uma plataforma móvel por seis atuadores lineares, chamados de pernas. Um dos desafios do uso desse manipulador em aplicações práticas está na resolução eficiente das equações matemáticas que descrevem seu movimento. No problema da cinemática direta, deseja-se encontrar a posição espacial da plataforma em função da posição de seis atuadores lineares. Este problema envolve a solução simultânea de seis equações não lineares e pode não ter solução fechada, dependendo da geometria do robô. Tradicionalmente recorre-se a métodos numéricos para a resolução dessas equações. Neste trabalho propõe-se uma abordagem utilizando um algoritmo de aprendizado de máquina conhecido como rede neural artificial. O algoritmo foi implementando utilizando a linguagem de programação *Python* e testado utilizando um modelo simulado de plataforma de Stewart.

Palavras-chave: plataforma de Stewart. aprendizado de máquina. cinemática. robótica.

Abstract

This work studies a parallel robotic manipulator known as Stewart platform. The Stewart platform consists of a fixed based and a moving platform connected by six linear actuators known as legs. A challenge in using this type of manipulator in practical applications lies in the efficient solution of its kinematic equations, i.e. the mathematical equations that describe its movement. In the forward kinematics problem it is desired to find the spatial position of the platform as a function of the displacements of each linear actuator. This problems involves the simultaneous solution of six non-linear equations and there may not be a closed form solution, depending on the geometry of the robot. Traditionally, these equations are solved using numerical methods. This work proposes a solution based on a machine learning algorithm known as artificial neural network. The algorithm was implemented using the Python programming language and tested using a simulated Stewart platform model.

Keywords: Stewart platform. machine learning. kinematics. robotics

Lista de ilustrações

Figura 1 – Um manipulador serial com seis graus de liberdade. 16
Figura 2 – Um manipulador paralelo com cinco graus de liberdade. 17
Figura 3 – Exemplo de plataforma de Stewart. 18
Figura 4 – Um exemplo de estrutura de perceptron multicamadas. 22
Figura 5 – Função de ativação sigmoidal. 24
Figura 6 – Função de ativação tangente hiperbólica. 24
Figura 7 – Função de ativação ReLU. 25
Figura 8 – Esquema para a formulação da cinemática da plataforma de Stewart. . 29
Figura 9 – Modelo da plataforma de Stewart em posição de repouso. 32
Figura 10 – Modelo da plataforma de Stewart transladado e rotacionado. 32
Figura 11 – Erro de validação a cada iteração durante o treinamento da rede neural. 36

Lista de tabelas

Tabela 1 – Componentes de manipuladores e seus graus de liberdade. 15
Tabela 2 – Limites de operação do *end-effector*. 30
Tabela 3 – Erro de validação para diferentes topologias de rede. 35
Tabela 4 – Comparação entre predições e valores esperados. 37

Sumário

1 INTRODUÇÃO 11
1.1 Justificativa 12
1.2 Objetivos 13
1.2.1 Objetivo geral 13
1.2.2 Objetivos específicos 13
1.3 Estrutura do trabalho 14

2 REVISÃO BIBLIOGRÁFICA 15
2.1 Manipuladores robóticos 15
2.1.1 Manipuladores seriais 16
2.1.2 Manipuladores paralelos 16
2.2 Cinemática de manipuladores robóticos 16
2.2.1 Cinemática direta 17
2.2.2 Cinemática inversa 17
2.3 Plataforma de Stewart 17
2.4 Aprendizado de máquina: conceitos e definições 20
2.4.1 Dados 20
2.4.2 Modelo 20
2.4.3 Aprendizado supervisionado 20
2.4.4 Regressão 20
2.5 Redes neurais artificiais 21
2.5.1 Aproximação universal 23
2.5.2 Funções de ativação 23
2.5.3 Função de custo 25
2.5.4 Treinamento da rede neural 25
2.5.5 Gradiente descendente estocástico 26
2.5.6 O algoritmo de retro-propagação 27

3 METODOLOGIA 29
3.1 Modelagem do manipulador 29
3.1.1 Base 29
3.1.2 Plataforma 30
3.1.3 Espaço de trabalho 30
3.1.4 Cinemática inversa 30
3.1.5 Cinemática direta 32
3.1.6 Implementação da rede neural 33

3.1.7 Pré-processamento dos dados de entrada e saída 34
3.1.8 Avaliação do desempenho da rede neural 34

4 RESULTADOS E DISCUSSÕES 35

5 CONCLUSÃO 39

REFERÊNCIAS 41

1 Introdução

Manipuladores robóticos têm sido amplamente utilizados em diversos campos da indústria e da ciência. Com a crescente demanda por automação de processos, aumenta-se a necessidade do estudo acerca de manipuladores robóticos. Em particular, um tipo de manipulador paralelo conhecido como plataforma de Stewart (STEWART, 1965) encontra muitas aplicações nos campos da aeronáutica, mas também em sistemas de posicionamento de precisão.

A plataforma de Stewart é um manipulador robótico paralelo composto por uma base fixa conectada a uma plataforma por meio de seis atuadores lineares, chamados de braços. A plataforma, chamada de *end-effector*, pode mover-se no espaço com seis graus de liberdade dentro dos limites de operação do manipulador. Essa liberdade de movimento, aliada à sua robustez mecânica e boa capacidade de suportar carga, torna esse manipulador muito útil em diversas aplicações.

Por essa razão, desde sua concepção em meados da década de 60, a plataforma de Stewart foi profundamente estudada. Diferentes configurações deste robô foram propostas, cada qual com sua vantagem em determinadas aplicações. Contudo, uma grande dificuldade em trabalhar com este manipulador consiste na resolução da sua cinemática, ou seja, das equações que descrevem seu movimento espacial. Estas são essenciais para seu controle e funcionamento adequado dentro de especificações de desempenho.

O problema da cinemática inversa consiste em encontrar os comprimentos das pernas do robô, conhecendo-se a pose (translação e rotação) do *end-effector*. Essa tarefa é trivial e envolve uma álgebra simples. Já no problema da cinemática direta, deseja-se encontrar a pose do *end-effector* a partir dos comprimentos das pernas. A resolução da cinemática direta é complexa e envolve a resolução de um sistema de seis equações não lineares. Dependendo da configuração do robô (de sua geometria), não existe solução analítica. Tradicionalmente utilizam-se métodos numéricos para resolver esse sistema. Os métodos numéricos dependem de uma estimativa inicial para a pose do *end-effector* e o número de iterações necessário para que se chegue a uma solução que satisfaça a precisão desejada varia de acordo com a estimativa inicial. Consequentemente, o tempo necessário para a convergência é variável, o que pode ser relevante em aplicações em tempo real. Além disso, o esforço computacional de tais métodos pode ser elevado, demandando grande recursos de hardware. Porém, se a estimativa inicial estiver dentro do espaço de trabalho do manipulador, o método certamente irá convergir para uma solução.

Por outro lado, recentemente abordagens utilizando algoritmos de aprendizado de máquina têm sido empregadas (Parikh; Lam, 2005) (YEE; LIM, 1997) (Morell; Acosta;

Toledo, 2012). Nos algoritmos de aprendizado de máquina não é necessário conhecer explicitamente a relação entre a entrada (neste caso os comprimentos das pernas) e a saída (a pose do *end-effector*). Fazendo uso de sensores, é possível medir os comprimentos das pernas em diferentes posições do robô, bem como sua pose. A partir disso, construindo um conjunto de dados suficientemente grande, pode-se treinar um modelo de aprendizado de máquina para estimar, a partir dos valores dos comprimentos das pernas, a pose correspondente. Essa abordagem elimina o equacionamento analítico e pode ser usada inclusive em situações em que não se conhece todos os detalhes da geometria do robô.

Sendo assim, este trabalho busca ampliar o estudo acerca da utilização de algoritmos de aprendizado de máquina na resolução de equações cinemáticas de manipuladores robóticos.

1.1 Justificativa

Os manipuladores robóticos têm sido muito usados na indústria, principalmente no que diz respeito a automação, devido a sua boa capacidade de repetibilidade e precisão de movimento (CRAIG, 2009). A construção, projeto e aplicação de tais manipuladores em determinado processo envolve a combinação dos conhecimentos de diversas áreas da ciência e engenharia.

Os manipuladores paralelos, em particular, são aqueles formados por uma cadeia fechada de elos. Eles apresentam boa robustez e alta razão entre carga suportada e carga própria. Entre esses, a plataforma de Stewart recebe destaque, visto que ela é um tipo de manipulador paralelo que permite o movimento nos seis graus de liberdade, três translações e três rotações. Tal característica evita a necessidade de se usar, por exemplo, uma combinação em série de diferentes manipuladores a fim de se atingir essa quantidade de graus de liberdade, o que pode propagar erros de posicionamento, entre outros fatores indesejáveis. Por essa razão, a plataforma de Stewart tem sido amplamente estudada.

A aplicação da plataforma de Stewart em situações reais envolve a resolução de sua cinemática, ou seja, das equações que descrevem seu movimento. Na cinemática inversa obtêm-se os deslocamentos dos seis atuadores lineares em função da posição da plataforma. Esta possui solução fechada e pode ser resolvida algebricamente. Já a cinemática direta, partir da qual se obtém a posição da plataforma em função dos deslocamentos dos atuadores, é complexa e pode não ter solução analítica. A cinemática direta é, então, o maior desafio na utilização deste tipo de manipulador em aplicações reais, em especial nas aplicações em tempo real, que exige o cálculo da cinemática diversas vezes em curtos intervalos de tempo.

Para a solução da cinemática direta, normalmente utilizam-se métodos numéricos (SELIG; LI, 2009) (Cardona, 2015). Os métodos numéricos têm sido cada vez mais

sofisticados, assim como as ferramentas computacionais, o que permite uma solução suficientemente rápida da cinemática direta, adequada para situações em tempo real. Contudo, o uso de algoritmos de aprendizado de máquina tem ganhado destaque na solução desse problema. Diferente da abordagem dos métodos numéricos, a aprendizado de máquina exclui a necessidade formulações matemáticas analíticas acerca do robô e sua geometria, basta dispor de um conjunto de dados suficientemente grande e representativo contendo as posições da plataforma e os deslocamentos dos atuadores lineares em vários pontos de interesse para que se construa um modelo preditivo apropriado.

Sendo assim, ampliar o estudo acerca da utilização de algoritmos de aprendizado de máquina na solução da cinemática da plataforma de Stewart e, de forma mais ampla, em robótica possui grandes implicações práticas. Este trabalho propõe uma solução para a cinemática direta da plataforma de Stewart utilizando uma rede neural artificial. O procedimento é explicado em detalhes no capítulo 3.

1.2 Objetivos

1.2.1 Objetivo geral

O objetivo deste trabalho é resolver a cinemática direta de um manipulador robótico conhecido como plataforma de Stewart utilizando um algoritmo de aprendizado de máquina aplicado a dados obtidos com a cinemática inversa.

1.2.2 Objetivos específicos

Os objetivos específicos são:

- Estudar as formas tradicionais de se resolver a cinemática direta da plataforma de Stewart
- Estudar a aplicação de algoritmos de aprendizado de máquina em robótica
- Resolver a cinemática direta de uma plataforma de Stewart utilizando um algoritmo de aprendizado de máquina
- Aplicar o algoritmo utilizando ferramentas de código aberto
- Determinar o melhor algoritmo a fim de diminuir o erro da predição
- Avaliar a performance do algoritmo comparando-o com uma solução por método numérico

1.3 Estrutura do trabalho

O desenvolvimento do trabalho é apresentado nos capítulos a seguir. No Capítulo 2 é feita uma revisão da bibliografia que foi utilizada como base para este trabalho. Nele, são explicados conceitos sobre manipuladores robóticos e seus modelos cinemáticos. Além disso, é apresentado o manipulador conhecido como plataforma de Stewart, suas características e uma revisão da literatura acerca da solução de sua cinemática. Também são apresentados conceitos de aprendizado de máquina e redes neurais artificiais.

No Capítulo 3 é apresentada a metodologia do trabalho, descrevendo o modelo da plataforma de Stewart utilizado, o equacionamento de sua cinemática e a obtenção do conjunto de dados utilizado no algoritmo de aprendizado de máquina. Também são apresentadas as ferramentas utilizadas na construção do modelo de aprendizado de máquina.

No Capítulo 4 são descritos os resultados obtidos a partir do trabalho desenvolvido e também discussões sobre os resultados. Por fim, no Capítulo 5 é apresentada uma conclusão sobre o trabalho.

2 Revisão bibliográfica

2.1 Manipuladores robóticos

Um corpo rígido no espaço possui 6 graus de liberdade, ou seja, um conjunto de seis movimentos espaciais que definem sua posição. Esses movimentos são: três translações em três eixos mutuamente ortogonais e três rotações ao redor destes mesmos eixos. Em robótica, o corpo rígido que se tem interesse em movimentar é chamado de efetuador final ou *end-effector*. As translações e rotações do *end-effector* determinam sua pose. Quando se consegue controlar vários graus de liberdade de um *end-effector* a partir de um sistema mecânico, tem-se um robô (MERLET, 2005).

Um manipulador robótico é composto por uma cadeia de elos ligados por juntas. Os elos são estruturas rígidas, enquanto as juntas permitem movimentos translacionais ou rotacionais com um determinado número de graus de liberdade. Sendo assim, a combinação de elos e juntas define o número total de graus de liberdade do efetuador final. Por exemplo, dado um sistema de coordenadas cartesiano, um manipulador de três graus de liberdade pode permitir três movimentos de translação ao longo de x, y e z ou dois movimentos de translação em x e y e um movimento de rotação em torno do eixo z, ou ainda qualquer combinação não redundante de movimentos.

Na Tabela 1 são mostrados os principais componentes de manipuladores robóticos, bem como o número de graus de liberdade de cada componente.

Tabela 1 – Componentes de manipuladores e seus graus de liberdade.

Componente	Tipo de movimento	Nº de graus de liberdade
Elo	-	-
Junta rotativa	rotacional	1
Junta prismática	linear	1
Junta esférica	rotacional	3
Junta universal	rotacional	2

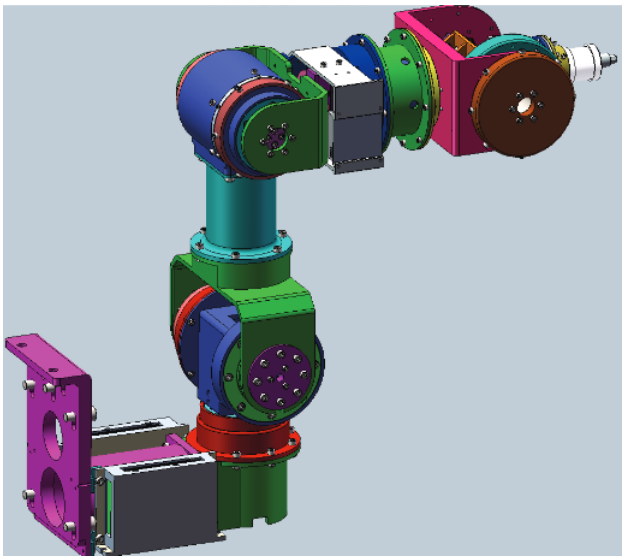
Fonte: Produzido pelo autor.

Manipuladores robóticos são amplamente utilizados na automação industrial, em aplicações aeroespaciais e em tarefas de posicionamento de precisão (CRAIG, 2009). Em geral, os manipuladores são classificados de acordo com número de graus de liberdade que possuem. Manipuladores robóticos podem ser divididos em dois grupos: manipuladores seriais e manipuladores paralelos.

2.1.1 Manipuladores seriais

Os manipuladores seriais são aqueles nos quais os elos ligam-se em série por juntas, formando uma cadeia cinemática aberta. Eles são muito utilizados na automação industrial devido a sua flexibilidade. Em geral, esse tipo de manipulador possui maior alcance de movimento. Em contrapartida, um erro na posição de um elo propaga-se ao longo de todo o robô. Sendo assim, a acurácia de um manipulador serial é em geral ruim.

Figura 1 – Um manipulador serial com seis graus de liberdade.



Fonte: Gu e Cao (2014)

2.1.2 Manipuladores paralelos

De acordo com Merlet (2005), manipuladores robóticos são aqueles cujos elos possuem um grau de conexão maior ou igual a 3, com exceção da base. Para cada elo do manipulador, o grau de conexão é definido como o número de corpos rígidos conectados ao elo por uma junta.

Sendo assim, manipuladores paralelos possuem cadeias cinemáticas fechadas. Esse tipo de manipulador possui como características maior robustez e precisão de movimento, porém um alcance de movimento geralmente menor.

2.2 Cinemática de manipuladores robóticos

A cinemática de um manipulador robótico é o estudo do seu movimento, sem se preocupar com as forças que causaram tal movimento (CRAIG, 2009). Trata-se de uma descrição matemática do movimento espacial de seus elos.

Figura 2 – Um manipulador paralelo com cinco graus de liberdade.

Fonte: Chi-Hyo Kim et al. (2007)

O problema da cinemática de manipuladores pode ser divido em: cinemática direta e cinemática inversa.

2.2.1 Cinemática direta

Na cinemática direta, deseja-se obter a posição espacial do *end-effector* conhecendo-se as posições angulares das juntas.

2.2.2 Cinemática inversa

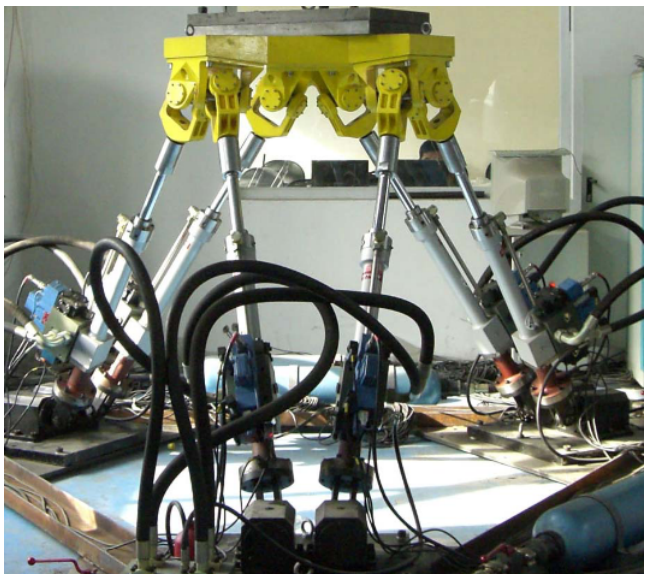
Já na cinemática inversa, conhece-se a posição do *end-effector* e deseja-se obter as posições espaciais das juntas que o levam àquela posição.

2.3 Plataforma de Stewart

A plataforma de Stewart é um tipo de manipulador paralelo de 6 graus de liberdade (6-DOF) proposto por Stewart (1965). Ela consiste de uma base, fixa, ligada a uma plataforma por meio de seis atuadores lineares. Cada atuador conecta-se à base por uma junta universal de dois graus de liberdade. Os atuadores conectam-se à plataforma por juntas esféricas de três graus de liberdade. No caso deste manipulador, o *end-effector* é a plataforma.(CRAIG, 2009)

Este tipo de manipulador foi originalmente projetado para aplicações aeronáuticas, em simuladores de voo (CRAIG, 2009). Contudo, suas características de robustez e precisão o fizeram popular em outras áreas como posicionamento de amostras em experimentos científicos. Há diferentes configurações da plataforma de Stewart. Na configuração 6-3, por exemplo, um par de nós da base conecta-se a um único nó da plataforma. Portanto, têm-se 6 nós na base e apenas 3 nós na plataforma. Diferentemente, na configuração 6-6, cada nó da base conecta-se a um nó da plataforma, de modo que se tenha 6 nós na base e na plataforma. Outras configurações também são possíveis.

Figura 3 – Exemplo de plataforma de Stewart.



Fonte: Yang et al. (2008)

Uma característica interessante da plataforma de Stewart é que, ao contrário dos manipuladores seriais, o modelo cinemático inverso é simples e de fácil solução. Já o modelo cinemático direto é complexo e dependendo da geometria do manipulador pode não haver solução fechada (CRAIG, 2009).

Tradicionalmente utilizam-se métodos numéricos para a solução da cinemática direta (SELIG; LI, 2009) (Cardona, 2015). Os métodos numéricos conseguem encontrar solução em um tempo razoável para aplicações em tempo real. Porém, esse tempo pode variar dependendo da estimativa inicial para a pose do manipulador. Se a estimativa inicial estiver próxima da pose real, o método encontrará a solução em poucas iterações e, portanto, rapidamente. Caso contrário, o método poderá demorar várias iterações até convergir para o valor da pose, considerando a tolerância desejada. O tempo de convergência variável dos métodos numéricos pode ser indesejável em certas aplicações. Apesar disso, dada uma estimativa inicial dentro do espaço de trabalho do manipulador, ou seja, dentro dos limites

possíveis de translação e rotação do *end-effector*, o método certamente irá convergir para uma solução.

Recentemente, algumas soluções para o problema da cinemática direta têm empregado técnicas de aprendizado de máquina. Morell, Acosta e Toledo (2012) propuseram uma solução usando o algoritmo *Support Vector Machine*. No entanto, o algoritmo suporta apenas uma variável de saída, enquanto o problema em questão tem seis variáveis de saída. Os autores contornaram essa limitação utilizando seis SVMs, uma para cada variável de saída. Outra abordagem para a solução da cinemática envolve o algoritmo *Multi-task Gaussian Process* (HE; GU; WANG, 2013). Além disso, outros autores utilizaram estratégias híbridas, envolvendo redes neurais artificiais combinadas com métodos numéricos, a fim de melhorar o desempenho (Parikh; Lam, 2005) (YEE; LIM, 1997) (SALEHINIA et al., 2013). Nesses trabalhos, o uso de estratégias híbridas se mostrou eficaz na redução do erro da solução obtida. Porém, essa abordagem aumenta a complexidade do algoritmo de rede neural.

Neste trabalho propõe-se uma solução do problema da cinemática direta da plataforma de Stewart utilizando puramente uma rede neural artificial. O algoritmo de rede neural é vantajoso pois exclui a necessidade de desenvolver equações analíticas. Conhecendo um conjunto de entradas (comprimentos das pernas do manipulador) e saídas (poses do *end-effector*) suficientemente grande e representativo, pode-se treinar a rede para mapear a relação entrada-saída. Sabe-se que dada uma função e um intervalo do domínio dessa função, uma rede neural pode ser usada para aproximar a função no intervalo (HORNIK, 1991a). Como o intervalo de operação do manipulador é totalmente conhecido, a rede pode ser usada para encontrar a relação entrada-saída. Além disso, a obtenção de um conjunto de dados para alimentar a rede é muito simples, basta resolver a cinemática inversa para várias poses diferentes do manipulador. Alternativamente, no caso de um manipulador real, pode-se gerar um conjunto de dados utilizando sensores para medir as posições do *end-effector* para diferentes comprimentos das pernas.

Dessa forma, o uso de uma rede neural artificial para a resolver a cinemática direta da plataforma de Stewart é apropriado. Não encontrou-se na literatura uma descrição detalhada da aplicação desse algoritmo na solução da cinemática da plataforma de Stewart. Além disso, boa parte dos trabalhos citados utilizam o *software* proprietário MATLAB. Este trabalho propõe a utilização de ferramentas de *software* gratuitas e acessíveis para solucionar o problema.

2.4 Aprendizado de máquina: conceitos e definições

2.4.1 Dados

De maneira simplificada, dados podem ser definidos como um conjunto de exemplos. Tipicamente, um exemplo consiste de uma coleção de atributos numéricos chamados *features*. O conjunto de dados pode conter também valores conhecidos como alvo da predição, dentro do contexto de aprendizado supervisionado. Assim, um conjunto de dados é um conjunto de pares *feature label*. Deseja-se, a partir de um modelo de aprendizado de máquina, predizer valores de *labels* a partir de *features* (ZHANG et al., 2020).

2.4.2 Modelo

Um modelo de aprendizado de máquina diz respeito ao conjunto de ferramentas computacionais que, a partir de valores de entrada, é capaz de predizer uma resposta. Isso é possível pois o modelo é previamente treinado, a fim de "aprender"a relação entre a entrada e a saída. A resposta é obtida a partir de sucessivas transformações nos dados de entrada.

2.4.3 Aprendizado supervisionado

Aprendizado supervisionado é uma classe de problemas em que se pretende predizer um rótulo ou *label* a partir de um dado de entrada ou *input*, dado um conjunto de dados chamado de exemplos, para o qual os *labels* são conhecidos. Tais problemas são amplamente estudados dentro da área de aprendizado de máquina (ZHANG et al., 2020).

As entradas são geralmente denotadas $\mathbf{x}$, enquanto as saídas são tipicamente denotadas y. Cada par (entrada, saída) é chamado exemplo. Um exemplo em particular pode ser denotado (x_i,y_i). Assim, um conjunto de dados é formado por n exemplos $\{x_i, y_i\}_{i=1}^{n}$. A tarefa do aprendizado supervisionado é construir um modelo f_θ que seja capaz de de mapear uma determinada entrada x_i a uma predição $f_\theta(x_i)$.

2.4.4 Regressão

Regressão é um tipo de aprendizado supervisionado no qual se deseja predizer um ou mais valores numéricos (um número real) a partir de determinados valores de entrada. Por exemplo, pode-se estimar o preço de uma casa a partir de informações como: tamanho da casa, número de quartos, número de banheiros, localização, entre outros. Reunindo-se um conjunto de dados contendo informações sobre diversas casas e o preço pelos quais elas foram vendidas, pode-se treinar um modelo de aprendizado de máquina capaz de predizer o valor da casa. Nesse exemplo, a resposta obtida pelo modelo, ou sua saída, é um

número real que representa o valor da casa em uma dada moeda. Portanto, o que define um problema de regressão é sua saída.

Algoritmos de regressão podem ser usados para vários outros problemas como: prever o número de horas requeridas para uma cirurgia, conhecendo-se o quadro clínico do paciente; estimar a quantidade de chuva em uma cidade no futuro próximo a partir de informações meteorológicas, entre outros. Para alguns problemas, obter uma relação direta (função) entre os parâmetros conhecidos (entradas) e o resultado esperado (saída) pode ser trivial. No entanto, quando o número de parâmetros envolvidos é consideravelmente grande e de naturezas distintas. No exemplo dos preços das casas, pode-se ter um parâmetro de entrada binário que indica se a casa possui ou não uma lareira. Outros parâmetros são números inteiros, como o número de quartos ou de banheiros. Já o tamanho da casa é dado por um número real. Ou seja, os parâmetros armazenam informações de forma distinta, o que pode dificultar para um ser humano encontrar uma correlação entre eles e o preço da casa. Além disso, em problemas para os quais se dispõe de um conjunto de dados suficientemente grande os algoritmos de regressão podem obter resultados bastante precisos.

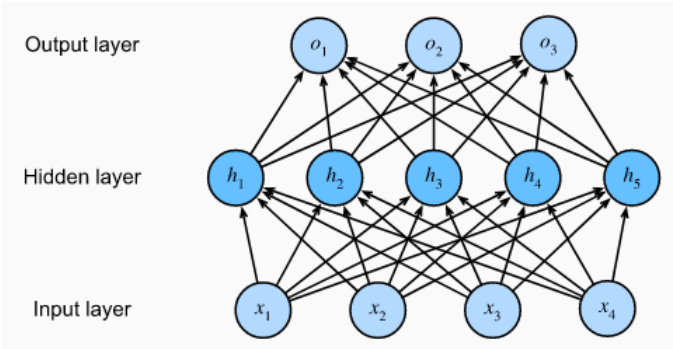
2.5 Redes neurais artificiais

Uma rede neural artificial (RNA) é um tipo de algoritmo de aprendizado de máquina. Este algoritmo utiliza combinações lineares das variáveis de entrada (neste caso, os comprimentos das seis pernas do manipulador) ponderadas por pesos ou *weights*. Tais combinações são então aplicadas a uma função não linear para gerar variáveis intermediárias conhecidas como neurônios, *neurons*, em uma analogia ao mecanismo de transmissão de informação nos sistemas nervosos de alguns seres vivos. Os neurônios também são chamados de unidades ocultas ou *hidden units*. As variáveis de entrada formam a camada de entrada da rede, enquanto as *hidden units* seguintes formam uma camada oculta ou *hidden layer*. Essa arquitetura é comumente conhecida como perceptron multicamadas, pois pode-se empilhar várias camadas em sequência até obter uma camada de saída, correspondente às variáveis de saída do modelo, conforme a Figura 4.

Na Figura 4, têm-se uma camada de entrada com 5 variáveis de entrada, uma *hidden layer* com 5 *hidden units* e uma camada de saída com 3 variáveis de saída. Observa-se, ainda, que cada unidade é densamente conectada à camada seguinte. Ou seja, uma unidade é formada pela combinação de todas as unidades da camada anterior. Essa é uma característica das redes *multilayer perceptron*. Em outras topologias, é possível haver camadas esparsas, unidades com realimentação, etc.

Um modelo básico de uma RNA pode ser descrito como uma sequência de transformações funcionais (BISHOP, 2006). Primeiro, faz-se uma sequência de M combinações

Figura 4 – Um exemplo de estrutura de perceptron multicamadas.



Fonte: (ZHANG et al., 2020)

lineares das variáveis de entrada $x_1, ..., x_D$, em que M é o número de neurônios da camada oculta:

$$a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \quad j = 1, \ldots, M \tag{2.1}$$

onde superscrito (1) indica que o parâmetro pertence à primeira camada da rede. Os termos w_{j0} são chamados de *bias* ou viés e correspondem a valores constante somados à combinação das entradas. As quantidades a_j são conhecidas como ativações. Em seguida elas são transformadas usando uma função diferenciável não linear chamada função de ativação $h(\cdot)$:

$$z_j = h(a_j^{(1)}) \tag{2.2}$$

As quantidades z_j são as saídas dos neurônios da camada oculta. Tais unidades são novamente combinadas para obter as ativações de saída (BISHOP, 2006):

$$a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \quad k = 1, \ldots, K \tag{2.3}$$

Em que K é o número total de saídas da rede. Essa transformação corresponde à segunda camada da rede. Finalmente, as ativações são transformadas usando uma função de ativação apropriada para se obter as saídas da rede y_k. A escolha da função de ativação depende do tipo de problema. Em problemas de regressão, a função de ativação da camada de saída é a função identidade, de forma que $y_k = a_k^{(2)}$ (BISHOP, 2006).

Combinando os vários estágios de cálculos acima, pode-se obter uma função única

para a saída da rede:

$$y_k(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M} w_{kj}^{(2)} h \left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) \tag{2.4}$$

Na equação 2.4 fica claro que, neste caso, o modelo da rede neural contém dois estágios de processamento. Contudo, esse modelo pode ser facilmente generalizado para incorporar mais camadas, unidades ocultas e diferentes funções de ativação (BISHOP, 2006).

2.5.1 Aproximação universal

Perceptrons multicamadas são chamados de aproximadores universais, pois são capazes de captar relações complexas entre as variáveis de entrada e saída através de seus neurônios ocultos (ZHANG et al., 2020). O chamado teorema da aproximação universal foi provado utilizando diferentes funções de ativação por alguns autores (CYBENKO, 1989) (LU et al., 2017) (HORNIK, 1991b). De acordo com o teorema, uma rede neural é capaz de aproximar qualquer função em um dado intervalo, dado que o intervalo seja totalmente conhecido e a rede possua o conjunto correto de pesos. Apesar de ser possível aproximar funções utilizando topologias de rede simples, como redes com uma única camada oculta, em algumas aplicações é preferível que a rede tenha um número maior que camadas e menor de neurônios (ZHANG et al., 2020).

2.5.2 Funções de ativação

As funções de ativação são funções não lineares aplicadas às combinações lineares das unidades da camada anterior. Essas funções são responsáveis por captar relações não lineares entre as variáveis de entrada e as de saída, fazendo com que a rede tenha uma boa capacidade de aproximação para vários tipos de função.

Cybenko (1989) provou a aproximação universal de funções a partir de redes neurais utilizando função de ativação sigmoidal. A função sigmoidal ou sigmoid é definida como:

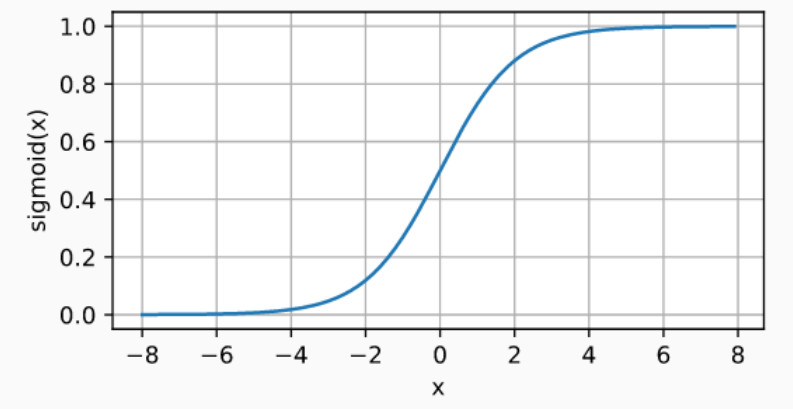
$$sigmoid(x) = \frac{1}{1 + \exp(-x)} \tag{2.5}$$

A função sigmoidal transforma uma entrada pertencente ao $\mathbb{R}$ em um valor dentro do intervalo (0,1), conforme pode-se ver na Figura 5 a seguir:

Similar à função sigmoidal é a função tangente hiperbólica, definida como:

$$tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \tag{2.6}$$

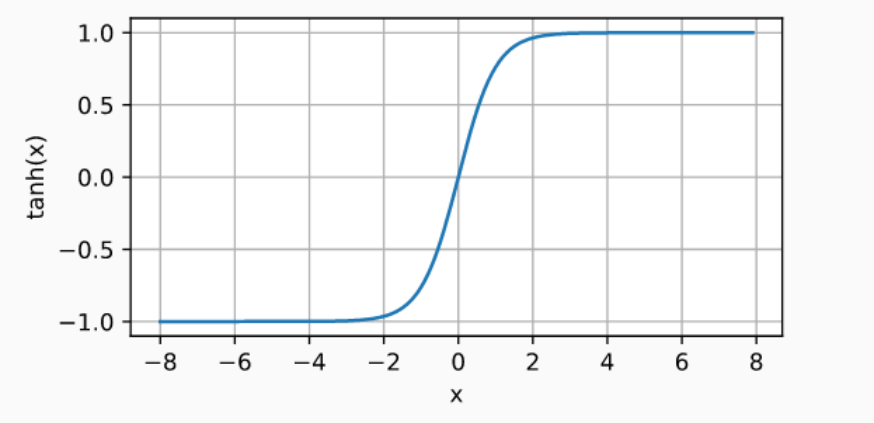
Figura 5 – Função de ativação sigmoidal.



Fonte: (ZHANG et al., 2020)

Porém, a função tangente hiperbólica transformada a entrada em uma saída dentro do intervalo (-1,1). Assim como a função sigmoidal, ela também é suave e diferenciável, mas é simétrica em relação à origem (ZHANG et al., 2020). Essa função de ativação é apresentada na Figura 6.

Figura 6 – Função de ativação tangente hiperbólica.



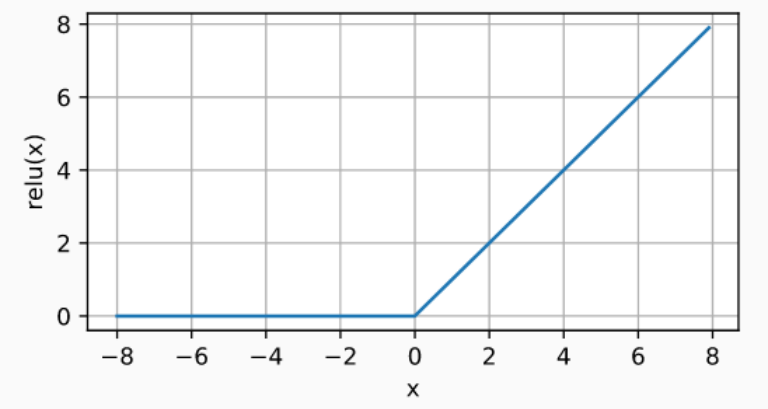
Fonte: (ZHANG et al., 2020)

Recentemente, a função de ativação preferida nas redes neurais tem sido a função ReLU (*rectified linear unit*), pela sua simplicidade de implementação e boa performance em diferentes tarefas de predição (ZHANG et al., 2020). A função ReLU é definida como:

$$ReLU(x) = max(x, 0) \tag{2.7}$$

Ou seja, a função ReLU mantém os valores de entrada que são positivos enquanto zera todas as entradas negativas, conforme indicado na Figura 7.

Figura 7 – Função de ativação ReLU.



Fonte: (ZHANG et al., 2020)

2.5.3 Função de custo

Para se determinar se a predição dada pelo modelo de regressão está próxima do valor esperado (e o quão próxima ela está) é necessária uma métrica. Tal métrica é dada pela função de custo, que determina o quão distante a predição está do valor esperado. Em geral, o custo é dado por número real não negativo, de forma que quanto mais próximo de zero for o custo, melhor a predição do modelo.

Para problemas de regressão, como é o caso do problema da cinemática direta de uma plataforma de Stewart, uma função de custo apropriada é dada pela soma dos quadrados dos erros de predição entre as entradas $\mathbf{x}$ e as saídas $\mathbf{y}$ (BISHOP, 2006). Dado um conjunto de *features* de entrada $\mathbf{X} = \mathbf{x}_1, ..., \mathbf{x}_N$, a função é definida como:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n\|^2 \tag{2.8}$$

2.5.4 Treinamento da rede neural

Ao se treinar uma rede neural, deseja-se obter os valores dos parâmetros da rede, ou seja, dos pesos e vieses, que minimizam o valor da função de custo. Sendo assim, a tarefa se resume a encontrar os valores dos parâmetros tal que o gradiente da função de custo se anule:

$$\nabla E(\mathbf{w}) = 0 \tag{2.9}$$

Os valores de $\mathbf{w}$ que satisfizerem a equação 2.9 são chamados de pontos estacionários. Eles podem ser pontos de mínimo, de máximo, ou ponto de sela. Logicamente deseja-se encontrar um ponto de mínimo, que minimize o valor da função de custo. Porém, em

geral as funções de erro apresentam uma dependência bastante não linear com relação aos pesos e vieses. Consequentemente, existem vários pontos estacionários no espaço dos pesos (BISHOP, 2006). Um ponto de mínimo que corresponde aos valores dos pesos e vieses tais que a função de custo assume o menor valor possível é chamado de mínimo global. Qualquer outro ponto de mínimo é chamado de mínimo local. Para redes neurais, muitas vezes é suficiente procurar por alguns pontos de mínimo locais e comparar os valores da função de custo correspondente a cada um desses pontos até encontrar uma solução que satisfaça a um critério de exatidão da predição (BISHOP, 2006).

A maioria das técnicas para encontrar pontos de mínimo envolve estimar um valor inicial para o vetor de pesos $\mathbf{w}_0$ e então mover-se no espaço dos pesos de forma iterativa:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta\mathbf{w}^{(\tau)} \tag{2.10}$$

Diferentes algoritmos são utilizados para determinar o incremento $\Delta\mathbf{w}^{(\tau)}$.

2.5.5 Gradiente descendente estocástico

Uma técnica para treinar a rede neural baseia-se em atualizar o vetor de pesos tomando um incremento na direção negativa à do gradiente da função de custo:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E(\mathbf{w}^{\tau}) \tag{2.11}$$

onde o parâmetro $\eta > 0$ é conhecido como taxa de aprendizado (*learning rate*). Após a atualização do vetor $\mathbf{w}$, o gradiente é novamente avaliado e o processo é repetido. Tal algoritmo é conhecido como gradiente descendente. Como a função de custo é definida pra um conjunto de treino, a cada iteração é necessário que todo o conjunto de treino seja processado para que se possa avaliar ∇E. Técnicas que utilizam todo o conjunto de treinamento são chamadas de processamento em lote ou *batch* (BISHOP, 2006). No entanto, quando o conjunto de dados é muito grande esse processo pode ser computacionalmente custoso e lento, já que a função custo representa a média dos erros associados a cada exemplo do conjunto de dados. Uma abordagem mais prática consiste em selecionar de forma aleatória um subconjunto do conjunto de exemplos, chamado *minibatch*. A cada iteração seleciona-se um *minibatch* de tamanho fixo, calcula-se o gradiente da função custo com respeito aos parâmetros do modelo e, por fim, atualiza-se os valores dos parâmetros usando a taxa de aprendizado (ZHANG et al., 2020).

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_n\mathbf{w}^{(\tau)} \tag{2.12}$$

Essa técnica é conhecida como gradiente descendente estocástico. Uma vantagem deste tipo de algoritmo com relação às técnicas em *batch* é que o primeiro é capaz de lidar de forma muito melhor dados redundantes (BISHOP, 2006).

2.5.6 O algoritmo de retro-propagação

O cálculo do gradiente da função de custo com relação aos parâmetros da rede neural pode demandar muito esforço computacional e possui complexidade de ordem $O(W^3)$ (BISHOP, 2006). Um algortimo conhecido retro-propagação mostrou-se bastante eficiente nesse cálculo (RUMELHART; HINTON; WILLIAMS, 1986). Usando informações sobre o gradiente ele é capaz de reduzir a complexidade para uma ordem $O(W^2)$ (BISHOP, 2006).

Em suma, o algortimo consiste em percorrer a rede neural na ordem reversa, ou seja, da camada de saída para a camada de entrada, aplicando a regra da cadeia do cálculo diferencial para calcular o gradiente da função de custo com respeito a cada um dos pesos da rede. A eficiência desse algortimo deve-se ao fato de que ele elimina a necessidade do cômputo direto das derivadas parciais da função de custo com respeito aos pesos. Conhecendo-se a função de custo e percorrendo a rede na ordem reversa os cálculos dessas derivadas podem ser feitos utilizando operações computacionalmente pouco custosas como somas e produtos.

Para aplicar o algoritmo, a função de custo deve satisfazer duas condições. Primeiro, deve ser possível escrevê-la como uma média de funções de custo avaliadas para cada elemento do conjunto de dados de entrada. A segunda condição é que deve ser possível representar a função de custo como uma função das saídas da rede neural (NIELSEN, 2015). Uma função de custo como a definida em 2.8 satisfaz ambas as confições.

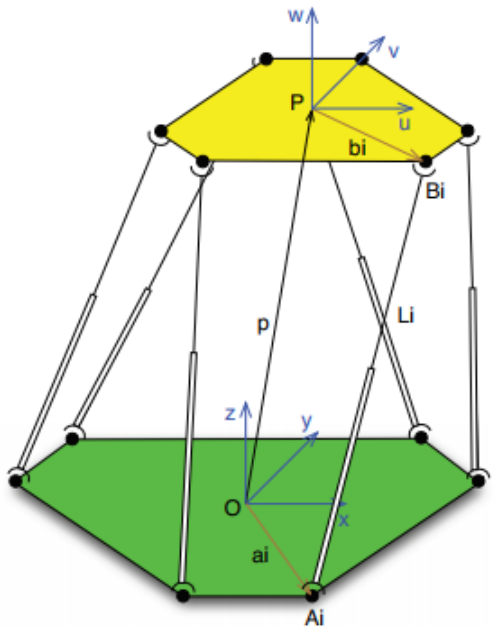
3 Metodologia

Neste capítulo são descritos o modelo projetado da plataforma de Stewart em estudo, os procedimentos para o equacionamento de sua cinemática, a aplicação de uma rede neural artificial para a solução do problema da cinemática direta e a avaliação do resultado.

3.1 Modelagem do manipulador

Conforme descrito anteriormente, a plataforma de Stewart consiste de um manipulador paralelo formado por uma base fixa ligada a uma plataforma através de seis atuadores lineares. Primeiro, as dimensões do robô serão definidas em termos das posições angulares das juntas da base e da plataforma, bem como a partir dos raios das circunferências da base e da plataforma que passam pelas juntas.

Figura 8 – Esquema para a formulação da cinemática da plataforma de Stewart.



Fonte: Cardona (2015)

3.1.1 Base

A base é definida por uma circunferência de raio R_b e por um conjunto de seis ângulos θ_{bi} $(i = 1, ..., 6)$, que determinam as posições angulares as juntas conectadas a

uma das extremidades das pernas do manipulador. Neste modelo, foi considerado um raio $R_b = 300$ mm e um conjunto $\theta_{bi} = [40, 60, 120, 140, 260, 280]$, em que os ângulos são dados em graus.

3.1.2 Plataforma

De forma similar à base, a plataforma é definida pelo raio R_p e um conjunto θ_{pi}, $(i = 1, ..., 6)$. Deve-se ressaltar, porém, que as coordenadas das juntas da plataforma são escrita no sistema de coordenadas O_p da plataforma. Ou seja, a componente vertical é nula. Considerou-se um raio $R_p = 200$ e $\theta_{bi} = [80, 100, 200, 220, 320, 340]$, com ângulos dados em graus.

3.1.3 Espaço de trabalho

Além das posições das juntas, também definiram-se os limites de operação de cada eixo do manipulador, ou seja, os intervalos contendo as posições do espaço que o *end-effector* é capaz de alcançar dentro das limitações do manipulador. Em um manipulador real, tais limitações são dadas pela geometria e pela construção mecânica do robô. O lugar geométrico de todas as posições espaciais alcançáveis pelo *end-effector*, incluindo as rotações em torno de cada um dos três eixos coordenados, constitui o espaço de trabalho do manipulador.

Tabela 2 – Limites de operação do *end-effector*.

	x [mm]	y [mm]	z [mm]	α [°]	β [°]	γ [°]
Intervalo de operação	[-25,25]	[-25,25]	[275,300]	[-10,10]	[-10,10]	[-15,15]

Fonte: Produzido pelo autor.

3.1.4 Cinemática inversa

Na cinemática inversa, deseja-se encontrar, a partir da pose (translação e rotação) do *end-effector*, os comprimentos de cada perna do manipulador. O *end-effector* é a própria plataforma mas, na formulação da cinemática, considera-se que sua pose é dada pela pose do seu centróide. Aqui, entende-se por centróide o centro da circunferência que passa pelas juntas da plataforma.

Além disso, na formulação da cinemática é conveniente definir dois sistemas de coordenadas. Um sistema O_b fixo no centro da circunferência da base e outro sistema O_p fixo no centróide da plataforma. Dessa forma, as posições das juntas da base ficam dadas pelos vetores $\vec{b_i}$ $(i = 1, ..., 6)$, escritos no sistema de coordenadas da base. Similarmente, as posições das juntas da plataforma são dadas pelos vetores $\vec{a_i}$ $(i = 1, ..., 6)$, escritos no sistema de coordenadas da plataforma.

Seja $\vec{p} = [x, y, z]$ o vetor posição do *end-effector*. Ou seja, o vetor que vai da origem do sistema de coordenadas O_b até o centróide da plataforma. Esse vetor determina o movimento de translação da plataforma. Na condição de repouso, ou seja, quando a plataforma não apresenta movimento de rotação ou translação e todas as pernas estão na posição nominal, o vetor posição é dado por $\vec{p} = [0, 0, h]$, onde h é a altura da plataforma em relação à base na condição de repouso.

A rotação do *end-effector* é dada por três rotações sucessivas ao longo dos três eixos do sistema de coordenadas O_p da plataforma. e é definida em termos dos ângulos de Euler α, β e γ. Na Equação 3.1 é mostrado o produto das três matrizes de rotação, cada qual em torno de um dos eixos coordenados.

$$R_z(\gamma)R_y(\beta)R_x(\alpha) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{3.1}$$

Dada uma translação e uma rotação do *end-effector*, é possível escrever a equação que fornece os vetores $\vec{s_i}$ de cada perna do manipulador. Ou seja, os vetores que ligam as juntas da base às juntas da plataforma. Esta é a equação da cinemática inversa da plataforma de Stewart.

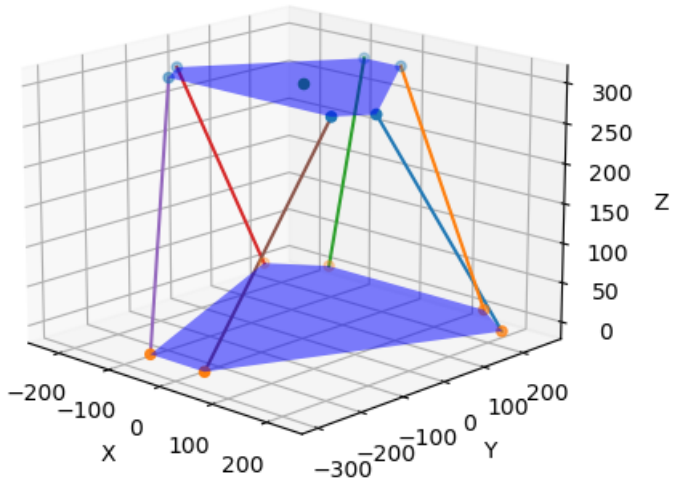
$$\vec{s_i} = \vec{p} + R \cdot \vec{a_i} - \vec{b_i} \tag{3.2}$$

Na Equação 3.2, $\vec{s_i}$ corresponde ao vetor que liga a i-ésima junta da base à i-ésima junta da plataforma, ou seja, o vetor que representa a i-ésima perna do manipulador. O vetor $\vec{b_i}$ corresponde ao vetor que liga o centróide da base à i-ésima junta da base, expresso no sistema de coordenadas O_b da base. Similarmente, o vetor $\vec{a_i}$ corresponde ao vetor que liga o contróide da plataforma à i-ésima junta da plataforma, expresso no sistema de coordenadas O_p da plataforma. O vetor $\vec{p}$ é o vetor que parte da origem do sistema de coordenadas da base e vai até a posição espacial do centróide da plataforma, ou seja, é o vetor posição do centróide da plataforma, considerando as translações. A matriz R é a matriz de rotação, dada pelo produto em 3.1. A partir da equação 3.2, os comprimentos das pernas podem ser obtidos a partir da norma dos vetores $\vec{s_i}$

$$l_i = \|\vec{s_i}\| \tag{3.3}$$

A Figura 9 a seguir apresenta o modelo da plataforma de Stewart, simulado usando a linguagem de programação *Python*, na situação de repouso, ou seja, sem nenhuma translação ou rotação do *end-effector*.

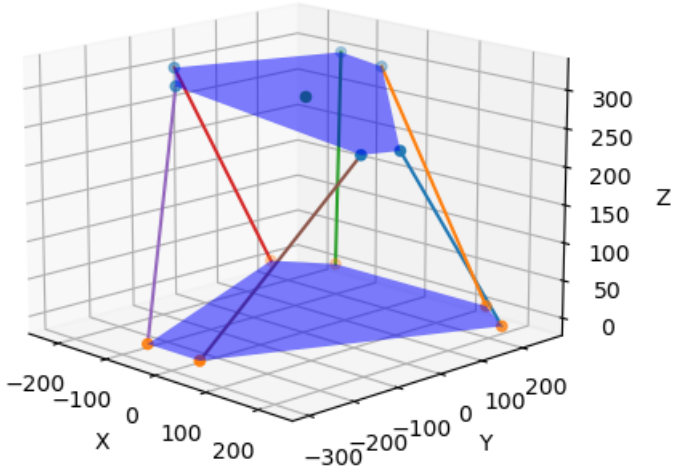
Figura 9 – Modelo da plataforma de Stewart em posição de repouso.



Fonte: Produzido pelo autor

Aplicando uma translação tal que o *end-effector* mova-se, por exemplo, até a posição $\mathbf{p} = (8.7, -5.4, 292.3)$ e rotações $\alpha = 5.1°$, $\beta = 9.6°$e $\gamma = 14.8°$, o modelo fica de acordo com a Figura 10.

Figura 10 – Modelo da plataforma de Stewart transladado e rotacionado.



Fonte: Produzido pelo autor

3.1.5 Cinemática direta

Na cinemática direta, são conhecidos os comprimentos das seis pernas do manipulador e deseja-se obter a posição do *end-effector* correspondente. Este problema envolve

a resolução de um sistema de seis equações não lineares nas seis variáveis $x, y, z, \alpha, \beta, \gamma$, conforme a equação a seguir:

$$l_i^2 = (\vec{p} + R\vec{a_i} - \vec{b_i})^T(\vec{p} + R\vec{a_i} - \vec{b_i}), \quad i = 1, ..., 6 \tag{3.4}$$

Primeiro, gerou-se um conjunto de dados correspondendo a várias poses do *end-effector*. Ou seja, cada amostra do conjunto de dados corresponde a uma posição $[x, y, z, \alpha, \beta, \gamma]$ dentro do espaço de trabalho do manipulador. Cada foi coordenada foi gerada de maneira aleatória dentro do seu intervalo de operação usando uma distribuição de probabilidade uniforme nesse intervalo. No total foram gerados 100000 dados. Esses dados serão as *labels* do algoritmo da rede neural.

Em seguida, para cada *label* gerada, calculou-se, usando a equação 3.2, os valores correspondentes dos comprimentos de cada perna do manipulador. Sendo assim, obteve-se um conjunto, também de tamanho 100000, no qual cada dado corresponde aos seis comprimentos das pernas. Esses são as *features* ou as entradas do algoritmo de rede neural.

O conjunto de dados foi dividido em um conjunto de treino e um conjunto de teste. O conjunto de treino contém 80% do total, ou seja, 80000 amostras (*feature*, *label*). O restante, 20000 amostras, compõe o conjunto de teste.

3.1.6 Implementação da rede neural

Para a implementação da rede neural, utilizou-se o *framework scikit-learn* (PEDREGOSA et al., 2011). O *scikit-learn* é um API (*Application Programming Interface*) que reúne ferramentas e funções para análise de dados e aprendizado de máquina para a linguagem de programação *Python*. Utilizou-se o modelo MLPRegressor, o qual implementa uma rede neural do tipo MLP para problemas de regressão.

Foram testadas diferentes topologias de rede, alterando o número de camadas ocultas e o número de neurônios nas camadas ocultas. Também foram testadas diferentes funções de ativação, como as funções tanh, sigmóide e ReLU. Além disso, diferentes algoritmos de otimização como o algoritmo SGD, Adam e LBFGS foram utilizados, e o desempenho da rede para cada caso foi observado.

Outros parâmetros como a taxa de aprendizagem e número de iterações também foram alterados a fim de se obter o melhor resultado possível.

Além disso, uma fração do conjunto de testes foi usado para validação da rede. Em todos os casos, 20% do conjunto de testes foi usado para validação, o que corresponde a 16000 amostras.

3.1.7 Pré-processamento dos dados de entrada e saída

Como os dados de saída estão em intervalos diferentes, é necessário transformá-los para que estejam dentro de uma mesma faixa de valores. Dessa forma, valores em escalas diferentes não têm influência sobre os pesos da rede neural. Além disso, o pré-processamento para melhorar o desempenho computacional da rede. Segundo a documentação do *sci-kit learn* (PEDREGOSA et al., 2011):

> *Multi-layer Perceptron is sensitive to feature scaling, so it is highly recommended to scale your data. For example, scale each attribute on the input vector X to [0, 1] or [-1, +1], or standardize it to have mean 0 and variance 1. Note that you must apply the same scaling to the test set for meaningful results. You can use StandardScaler for standardization.*

Os algoritmos de aprendizado de máquina funcionam melhor se os dados estiverem em um intervalo próximo de zero, como no intervalo [0,1] ou [-1,1]. Optou-se por transformar os dados para que pertencessem ao intervalo [0,1]. Assim, no vetor de entrada, calculou-se a média e o desvio padrão de cada coluna, correspondendo aos comprimentos de uma das pernas do manipulador. Subtraindo de elemento da coluna a média e dividindo o resultado pelo desvio padrão obtém-se um conjunto pertencente ao intervalo desejado. O mesmo foi feito para cada uma das variáveis de saída $[x, y, z, \alpha, \beta, \gamma]$ no vetor de *labels*.

3.1.8 Avaliação do desempenho da rede neural

Para avaliar o desempenho da rede neural, o erro quadrático médio entre a predição e o valor esperado foi calculado utilizando o conjunto de teste. Além disso, o algoritmo de rede neural foi comparado com uma solução método numérico, comparando o o erro entre o valor obtido e o esperado e o tempo de execução. O valor esperado foi obtido a partir da solução da cinemática inversa para uma dada pose do manipulador.

Para a implementação do método numérico foi utilizada a biblioteca de computação científica *SciPy* (VIRTANEN et al., 2020). Nela existe a função *fsolve*, a qual implementa um método numérico para encontrar raízes de funções. Segundo a documentação do *SciPy*, o parâmetro *xtol* da função *fsolve* define o valor do erro relativo entre duas iterações consecutivas do algoritmo para o qual o algoritmo deve ser encerrado. Usando esse parâmetro estabeleceu-ce um erro desejado para a solução e avaliou-se o tempo requerido para encontrar a solução a partir de uma estimativa inicial para a solução igual à condição de repouso do manipulador.

4 Resultados e discussões

A rede foi treinada sobre o mesmo conjunto de dados de treino utilizando diferentes topologias, ou seja, número de camadas ocultas e números de neurônios nas camadas. O número de iterações ou *epochs* utilizado para todos os testes foi 300. O número de *minibatches* para o algoritmo de otimização foi definido como 200 e foi utilizada uma taxa de aprendizagem constante $\eta = 0.001$.

A métrica de desempenho utilizada foi o erro quadrático médio, avaliado sobre o conjunto de teste. De acordo com a documentação do *sklearn*, Se $\hat{\mathbf{y}}_\mathbf{i}$ for a predição do valor de saída para a i-ésima entrada, e $\mathbf{y_i}$ for o valor real da saída, então o erro quadrático médio (*mean squared error* ou *MSE*) sobre n amostras é calculado da seguinte forma:

$$MSE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n_{amostras}} \sum_{i=0}^{n_{amostras}-1} (\mathbf{y_i} - \hat{\mathbf{y}}_\mathbf{i})^2 \tag{4.1}$$

Os melhores resultados obtidos estão representados na Tabela 3. É importante ressaltar que o valor do erro quadrático médio pode variar levemente para uma mesma topologia de rede e para um mesmo conjunto de dados, dada a natureza estocástica do algoritmo de otimização utilizado no treinamento da rede.

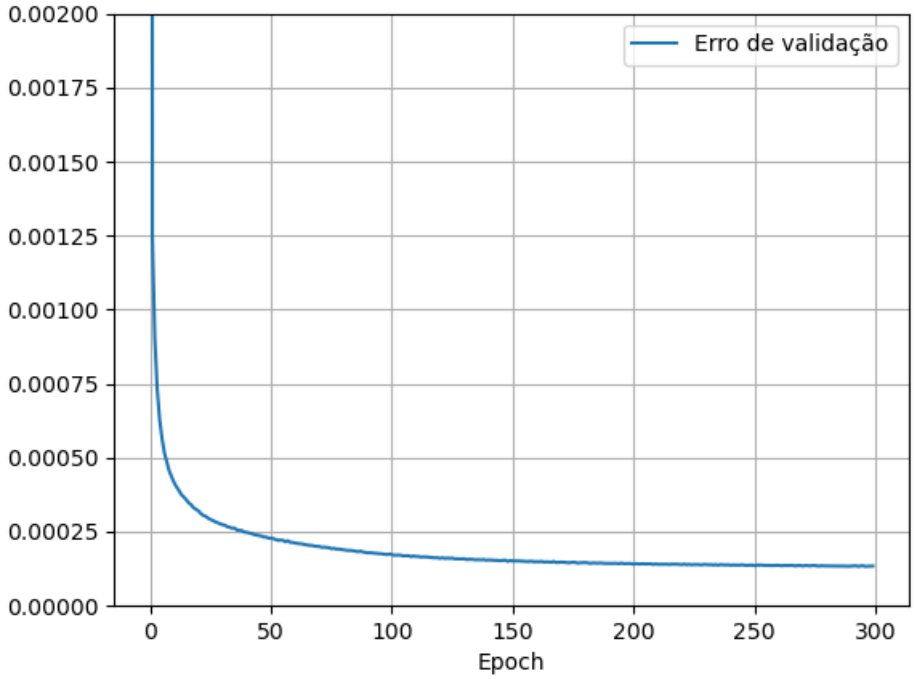
Tabela 3 – Erro de validação para diferentes topologias de rede.

Nº de camadas ocultas	Nº de neurônios por camada	Função de ativação	Erro quadrático médio
5	12, 14, 15, 23, 18	ReLU	0.0498669
3	13, 18, 14	ReLU	0.0726869
3	100, 100	ReLU	0.0019983
3	100, 100	Sigmóide	0.0036778
2	50, 50	Tangente hiperbólica	0.0024999
3	100, 100, 50	ReLU	0.0087441

Fonte: Produzido pelo autor.

O melhor resultado foi obtido utilizando uma rede neural com duas camadas ocultas, cada qual com 100 *hidden units* e função de ativação ReLU. As funções de ativação das camadas ocultas são a função ReLU. O algoritmo de otimização para o treinamento da rede utilizado foi o algoritmo Adam (KINGMA; BA, 2017), uma variação do gradiente descendente estocástico. Para essa situação obteve-se a curva que mostra o erro entre a saída da rede e a resposta esperada correspondente no conjunto de validação, chamado de erro de validação, para cada iteração do algoritmo de otimização.

Figura 11 – Erro de validação a cada iteração durante o treinamento da rede neural.



Fonte: Produzido pelo autor

Observa-se que o erro de validação diminui a cada iteração. O erro diminui abruptamente durante as primeiras iterações e tende a estabilizar-se conforme aumenta o número de iterações. É importante notar que o valor do erro durante o treinamento é afetado pelo processamento realizado nos dados do conjunto de treino. Como os dados foram transformados para que estivessem dentro do intervalo [0,1], essa transformação é refletida no valor do erro, ou seja, no eixo vertical do gráfico.

O tempo necessário para predizer uma resposta para a pose do manipulador utilizando o algoritmo de rede neural foi de 1 ms. Já o tempo para obter uma resposta utilizando o método numérico oscilou entre 5 e 10 ms. Porém, utilizando o método numérico o erro obtido na saída foi consideravelmente menor. Com o método numérico, o erro atingiu algumas dezenas ou centenas de bilionésimos de milímetros ou de graus, a depender da variável de saída.

Na Tabela 4 são mostrados os valores de poses do modelo da plataforma de Stewart, ou seja, o valor esperado de saída da rede neural, a corresponde predição obtida a partir da rede e o erro percentual entre o valor predito e o valor esperado para alguns *features* do conjunto de dados de teste. Nota-se que para a maioria dos valores o erro é inferior a um por cento.

Tabela 4 – Comparação entre predições e valores esperados.

		x [mm]	y [mm]	z [mm]	α [°]	β [°]	γ [°]
1	Valor esperado	-17.116683	4.005165	284.522296	-6.598679	8.035339	-11.946290
	Predição	-17.11243	4.059649	284.551095	-6.576546	8.044873	-11.981759
	Erro da predição	0.024847%	1.360343%	0.010122%	0.335415%	0.118650%	0.296904%
2	Valor esperado	4.111462	7.950491	297.457921	-9.545561	-2.265360	-3.641513
	Predição	4.125220	7.936206	297.480985	-9.544854	-2.241454	-3.684615
	Erro da predição	0.334625%	0.179674%	0.007754%	0.007407%	1.055280%	1.183628%
3	Valor esperado	1.894769	14.861688	290.501415	8.218153	9.156130	-13.074001
	Predição	1.857874	14.846128	290.47716	8.212121	9.151156	-13.089039
	Erro da predição	1.947203%	0.104699%	0.008349%	0.073398%	0.054324%	0.115022%
4	Valor esperado	8.869089	12.425898	276.276323	1.556353	4.546340	7.127303
	Predição	8.757118	12.470270	276.257125	1.551121	4.553523	7.107825
	Erro da predição	1.262486%	0.357093%	0.006949%	0.336171%	0.157995%	0.273287%
5	Valor esperado	-24.313152	9.550549	281.380267	3.122472	-3.538265	-12.857402
	Predição	-24.387429	9.563318	281.412474	3.113528	-3.541330	-12.842726
	Erro da predição	0.305501%	0.133699%	0.011446%	0.286440%	0.086624%	0.114144%
6	Valor esperado	14.609070	23.121464	297.688077	-2.124032	-9.376162	-7.544639
	Predição	14.542167	23.016998	297.669565	-2.108446	-9.387240	-7.537425
	Erro da predição	0.457955%	0.451814%	0.006219%	0.733793%	0.118151%	0.095618%

Fonte: Produzido pelo autor.

5 Conclusão

Neste trabalho foi proposto um método para a resolução da cinemática direta de uma plataforma de Stewart utilizando aprendizado de máquina. O algoritmo utilizado foi uma rede neural artificial. A motivação por trás desse estudo é ampliar a utilização desse tipo de abordagem nos problemas de cinemática da plataforma de Stewart, a fim de obter desempenhos iguais ou superiores aos métodos numéricos tradicionalmente utilizados.

O melhor resultado obtido corresponde a um erro quadrático médio de 0.0019983. Segundo esse resultado, a utilização de uma rede neural artificial para a solução da cinemática direta da plataforma de Stewart não apresenta desempenho superior ao de um método numérico, a partir do qual se obtém erros diversas ordens de grandeza menores.

Contudo, o método proposto pode ser utilizado em um aplicação em que a precisão requerida seja baixa ou para simular uma geometria do manipulador, para fins de prototipagem por exemplo. Além disso, a velocidade com que se obtém uma estimativa para a pose da plataforma a partir é grande o suficiente para aplicações em tempo real. Deve-se notar, ainda, que a qualidade da solução é afetada pela dimensão do espaço de trabalho do manipulador. No caso de uma plataforma de Stewart que opera em uma região menor do espaço, ou seja, para a qual os intervalos de translações e rotações são pequenos, o erro da predição tende a ser menor. Isso ocorre porque o conjunto de dados de treino e de testes está limitado a um intervalo menor, fazendo com que o algoritmo de rede neural tenha mais facilidade em aproximar a relação entre entrada e saída no intervalo.

Referências

BISHOP, C. M. *Pattern recognition and machine learning*. [S.l.]: springer, 2006. Citado 6 vezes nas páginas 21, 22, 23, 25, 26 e 27.

Cardona, M. A new approach for the forward kinematics of general stewart-gough platforms. In: *2015 IEEE Thirty Fifth Central American and Panama Convention (CONCAPAN XXXV)*. [S.l.: s.n.], 2015. p. 1–6. Citado 3 vezes nas páginas 12, 18 e 29.

Chi-Hyo Kim et al. Topological design of the 5-dof parallel-wrist manipulator with a constraining mechanism. In: *2007 International Conference on Control, Automation and Systems*. [S.l.: s.n.], 2007. p. 2278–2282. Citado na página 17.

CRAIG, J. J. *Introduction to robotics: mechanics and control, 3/E*. [S.l.]: Pearson Education India, 2009. Citado 5 vezes nas páginas 12, 15, 16, 17 e 18.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, Springer, v. 2, n. 4, p. 303–314, 1989. Citado na página 23.

Gu, K.; Cao, Q. Control system design of 6-dofs serial manipulator based on real-time ethernet. In: *2014 IEEE International Conference on Information and Automation (ICIA)*. [S.l.: s.n.], 2014. p. 118–120. Citado na página 16.

HE, J.; GU, H.; WANG, Z. Solving the forward kinematics problem of six-dof stewart platform using multi-task gaussian process. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, v. 227, p. 161–169, 01 2013. Citado na página 19.

HORNIK, K. Approximation capabilities of multilayer feedforward networks. *Neural networks*, Elsevier, v. 4, n. 2, p. 251–257, 1991. Citado na página 19.

HORNIK, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, v. 4, n. 2, p. 251 – 257, 1991. ISSN 0893-6080. Disponível em: <http://www.sciencedirect.com/science/article/pii/089360809190009T>. Citado na página 23.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017. Citado na página 35.

LU, Z. et al. The expressive power of neural networks: A view from the width. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2017. p. 6231–6239. Citado na página 23.

MERLET, J.-P. *Parallel robots*. [S.l.]: Springer Science & Business Media, 2005. v. 128. Citado 2 vezes nas páginas 15 e 16.

Morell, A.; Acosta, L.; Toledo, J. An artificial intelligence approach to forward kinematics of stewart platforms. In: *2012 20th Mediterranean Conference on Control Automation (MED)*. [S.l.: s.n.], 2012. p. 433–438. Citado 2 vezes nas páginas 12 e 19.

NIELSEN, M. A. *Neural networks and deep learning*. [S.l.]: Determination press San Francisco, CA, 2015. v. 2018. Citado na página 27.

Parikh, P. J.; Lam, S. S. Y. A hybrid strategy to solve the forward kinematics problem in parallel manipulators. *IEEE Transactions on Robotics*, v. 21, n. 1, p. 18–25, 2005. Citado 2 vezes nas páginas 11 e 19.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado 2 vezes nas páginas 33 e 34.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986. Citado na página 27.

SALEHINIA, Y. et al. Solving forward kinematics problem of stewart robot using soft computing. In: . [S.l.: s.n.], 2013. Citado na página 19.

SELIG, J. M.; LI, H. A geometric newton-raphson method for gough-stewart platforms. In: KECSKEMÉTHY, A.; MÜLLER, A. (Ed.). *Computational Kinematics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 183–190. ISBN 978-3-642-01947-0. Citado 2 vezes nas páginas 12 e 18.

STEWART, D. A platform with six degrees of freedom. *Proceedings of the institution of mechanical engineers*, SAGE Publications Sage UK: London, England, v. 180, n. 1, p. 371–386, 1965. Citado 2 vezes nas páginas 11 e 17.

VIRTANEN, P. et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, v. 17, p. 261–272, 2020. Citado na página 34.

Yang, C. et al. Modeling and simulation of 6-dof parallel manipulator based on pid control with gravity compensation in simulink/adams. In: *2008 International Workshop on Modelling, Simulation and Optimization*. [S.l.: s.n.], 2008. p. 391–395. Citado na página 18.

YEE, C. seng; LIM, K. bin. Forward kinematics solution of stewart platform using neural networks. *Neurocomputing*, v. 16, n. 4, p. 333 – 349, 1997. ISSN 0925-2312. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925231297000489>. Citado 2 vezes nas páginas 11 e 19.

ZHANG, A. et al. *Dive into Deep Learning*. [S.l.: s.n.], 2020. <https://d2l.ai>. Citado 6 vezes nas páginas 20, 22, 23, 24, 25 e 26.