Projeto Peão – Inteligência Artificial

Júlio Cézar, Elliel Alves, Georgene Estevam, João Vitor

Curso de Ciência da Computação – Faculdade dos Guararapes (UNIFG)
Av. Governador Carlos de Lima Cavalcanti, 155 – Boa Vista, Recife – PE, 50070110

jfigueredo397@gmail.com, jfigueredo397@protonmail.com\

Abstract. *Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate inteligent human behavior. The main objective of this project is to demonstrate that it's possible to teach a machine to perform complex tasks in a way that is similar to how humans solve problems.*

Resumo. *O Aprendizado de Máquina, ou Machine Learning, é um subcampo da área de inteligência artificial que é amplamente definido como a capacidade de uma máquina de imitar o comportamento humano inteligente. O objetivo central deste projeto é demonstrar na prática que é possível ensinar uma máquina a executar tarefas complexos de maneira semelhante á maneira a um ser humano.*

1. Introdução

O aprendizado de máquina, ou Machine Learning, é um tipo de inteligência artificial que permite que os aplicativos de software se tornem mais precisos na previsão de resultados sem serem explicitamente programados para isso. O objetivo é treinar uma máquina para que performe tarefas complexas **do mesmo jeito que um humano as faria** só que com muito mais velocidade e sem se cansar.

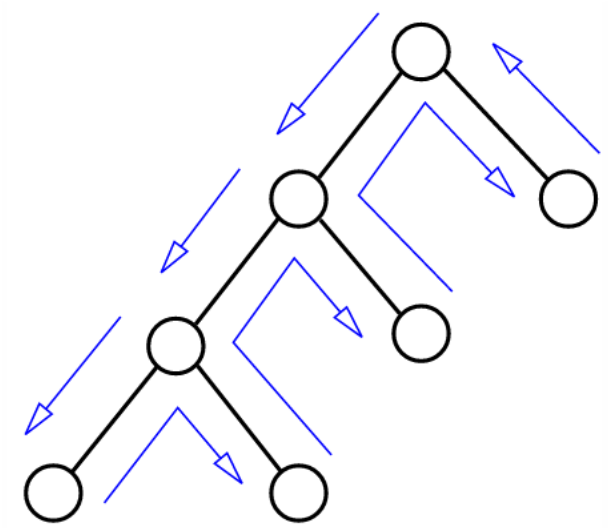
Os algoritmos de aprendizado de máquina usam dados históricos como entrada para prever novos valores de saída. Os buscadores são um caso de uso comum para aprendizado de máquina. Outros usos populares incluem detecção de fraude, filtragem de spam, detecção de ameaças de malware, automação de processos de negócios e manutenção preditiva.

Muitas das empresas líderes de hoje, como Facebook, Google e Uber, fazem do aprendizado de máquina uma parte central de suas operações. O aprendizado de máquina tornou-se um diferencial competitivo para muitas empresas.

O objetivo do “Projeto Peão” é demonstrar na prática como funciona o aprendizado de máquinas de forma simples e direta utilizando-se apenas de um algoritmo, HTML e uma linguagem de programação para criar um robô capaz de jogar Xadrez de forma tão habilidosa, senão mais habilidosa, do que um ser humano.

2. Algoritmo

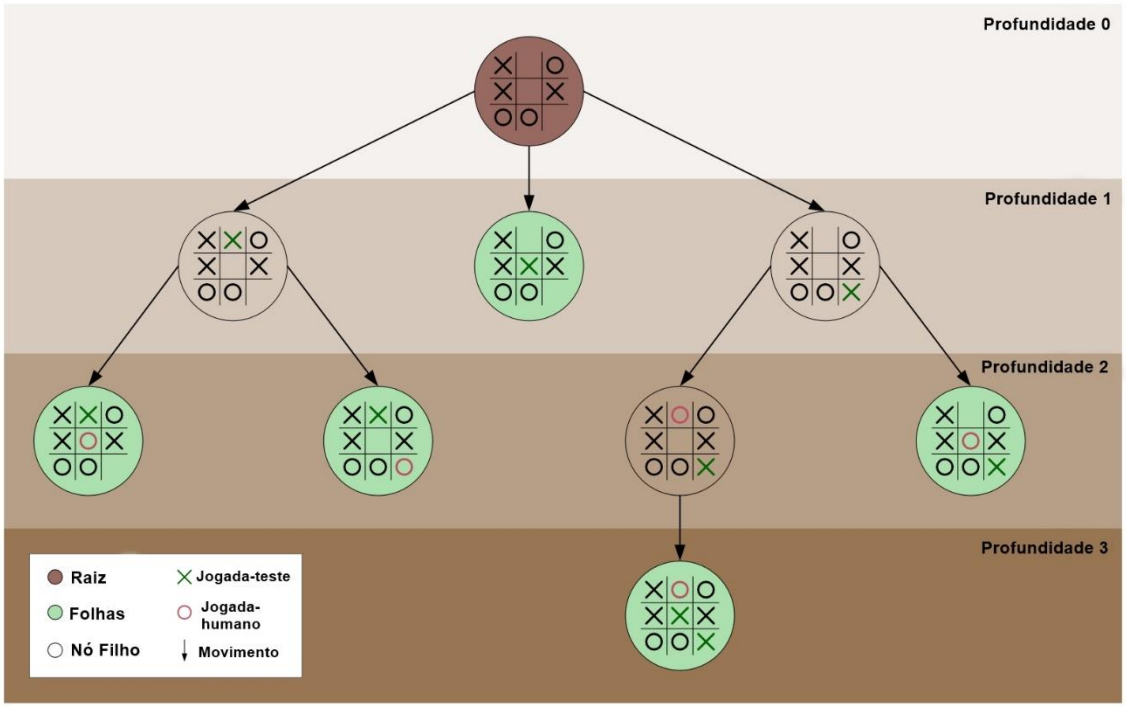
Para começar criar o nosso robô, primeiro tivemos que alimentá-lo com um algoritmo. E o algoritmo escolhido por nós foi o algoritmo “Minimax” que é um algoritmo de backtracking, ou seja, um algoritmo que busca todas as soluções possíveis e depois volta e escolhe a **melhor** solução possível eliminando as demais.



Isso foi muito importante pois nós não queremos que o robô faça qualquer jogada possível, nós queremos que ele faça a **melhor** jogada possível, da mesma forma que um jogador profissional de xadrez faria, por isso escolhemos o algoritmo de backtracking. Mas agora que já explicamos o que é um algoritmo de backtracking, vamos explicar como funciona o algoritmo de MiniMax expecificamente.

Um algoritmo minimax é um algoritmo recursivo escrito para encontrar a melhor jogada possível, que minimiza qualquer tendência de perder um jogo enquanto maximiza qualquer oportunidade de ganhar o jogo.

Graficamente, nós podemos representar o algoritmo Minimax dessa forma:



O jogador que invoca o algoritmo minimax (no nosso caso o robô) é chamado de jogador maximizador. Em outras palavras, é o jogador que quer maximizar sua porcentagem chance de ganhar o jogo. Em contraste, o oponente do jogador maximizador (no nosso caso, nós os humanos) é chamado de jogador minimizador. Assim, o jogador minimizador é aquele cujas chances de ganhar devem ser minimizadas pelo algoritmo. Em suma, um algoritmo minimax é uma função recursiva criada para ajudar o robô (o jogador maximizador) a decidir sobre a jogada que minimiza ao máximo a possibilidade de perder o jogo.

Para tomar a melhor decisão, o nosso robô precisa fazer o seguinte:

- Armazenar o estado atual (valores) do tabuleiro de xadrez em uma matriz (em uma célula vazia)

- Obter uma lista de matrizes com apenas os índices das células vazias.

- Verifique e confirmar se algum dos jogadores já ganhou o jogo.

- Se ninguém ganhou o jogo ainda, o robô deve invocar o algoritmo minimax recursivamente em cada uma das células vazias do tabuleiro.

- Retorne uma pontuação para cada movimento possível tanto para o jogador branco quanto para o jogador preto.

- De todas as pontuações retornadas, escolha a melhor (a mais alta) que garanta

minimizar as possibilidades do jogador humano vencer o jogo.

E como fazemos isso? Vamos explicar usando uma representação simples:

A primeira coisa que nós vamos precisar é de alguma função heurística que examine o estado do jogo e retorne um número. Basicamente uma função que informe ao robô que quando o valor for zero significa que ambos os jogadores estão empatados, se o valor for positivo que o humano está ganhando, e que se for negativo que ele que está ganhando.

No nosso caso, essa função será simplesmenta a diferença entre o número de peças que cada jogador possui. Vamos dar um valor especifico pra cada peça, com valores diferentes para peças mais importantes como cavalos, torres, bispos, rainha e rei e para peças fúteis com os peões. Assim o algoritmo vai saber se vale apena sacrificar uma peça ou não e quais peças ele tem que proteger baseado no valor dela.

Ok, agora sabemos como avaliar o estado de um jogo. O que acontece durante o jogo é que ambos os jogadores querem vencer. Isso significa que o jogador branco (o humano) deseja alcançar estados que tenham um grande valor positivo (o que significa que ele está ganhando) e o jogador preto (o robô) deseja alcançar estados que tenham um grande valor negativo (o jogador branco está perdendo = o jogador preto está ganhando, a lógica essa).

Na profundidade 1, o robô vai fazer um movimento a partir de um determinado estado: ele pode examinar todos os movimentos possíveis, e para cada um deles observar o estado que produz e usar nossa função de avaliação para calcular seu valor. A IA então simplesmente minimiza o valor do tabuleiro - ela escolhe o movimento que produz a posição com o menor valor.

Na profundidade 2, o robô passa também considerar o que seu oponente fará em seguida. Para cada movimento possível, a IA agora pode pensar da seguinte forma: "Ok, o que acontece, se eu fizer esse movimento? Meu oponente será o próximo a mover uma peça. E como ele quer vencer, provavelmente fará o movimento que maximiza o valor do tabuleiro - o torna ruim para mim e bom para ele. Vamos examinar todas as opções que ele têm e descobrir o pior que pode acontecer."

Assim, na profundidade 2 o próximo movimento é escolhido da seguinte forma: para cada movimento possível que o robô possa fazer, ele vai examinar todos os movimentos possíveis que o humano pode fazer assumindo que o humano vai tentar aumentar o valor do tabuleiro. Assim que ele faz isso, ele vai saber na hora se vale a pena ou não fazer uma jogada, por exemplo = o robô usar uma rainha para comer um peão, o valor do tabuleiro vai diminuir em -1 o que é bom para o robô, porém no movimento seguinte o humano usa um peão para comer a rainha, aumentando o valor do tabuleiro em +8, fazendo uma conta rápida (8 – 1 = 7) podemos ver que no fim, com essa jogada, o valor do tabuleiro vai aumentar em +7, ou seja, o robô vai sair no prejuizo, não vale a pena fazer a jogada.

Agora se o robô usar um peão para comer uma rainha (-8) e na jogada seguinte o seu peão é comido por um bisco, fazendo uma conta rápida (- 8 + 1 = -7) o robô vai descobrir que ele sairá com -7, ou seja, lucrou, a jogada valeu a pena. Uma vez que o robô processa todas as possibilidades, ele escolherá aquela que vale mais a pena.

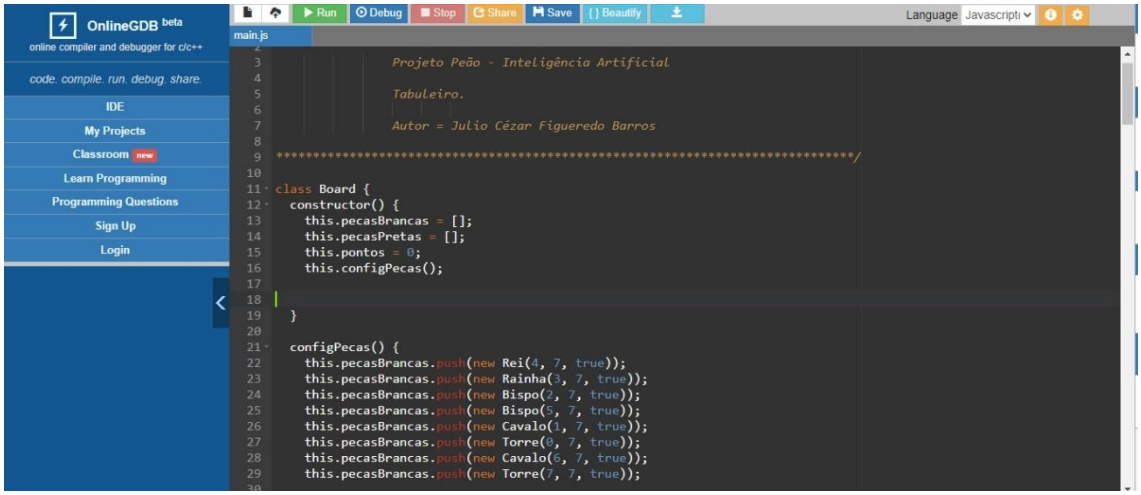
Assim funciona um algoritmo MiniMax.

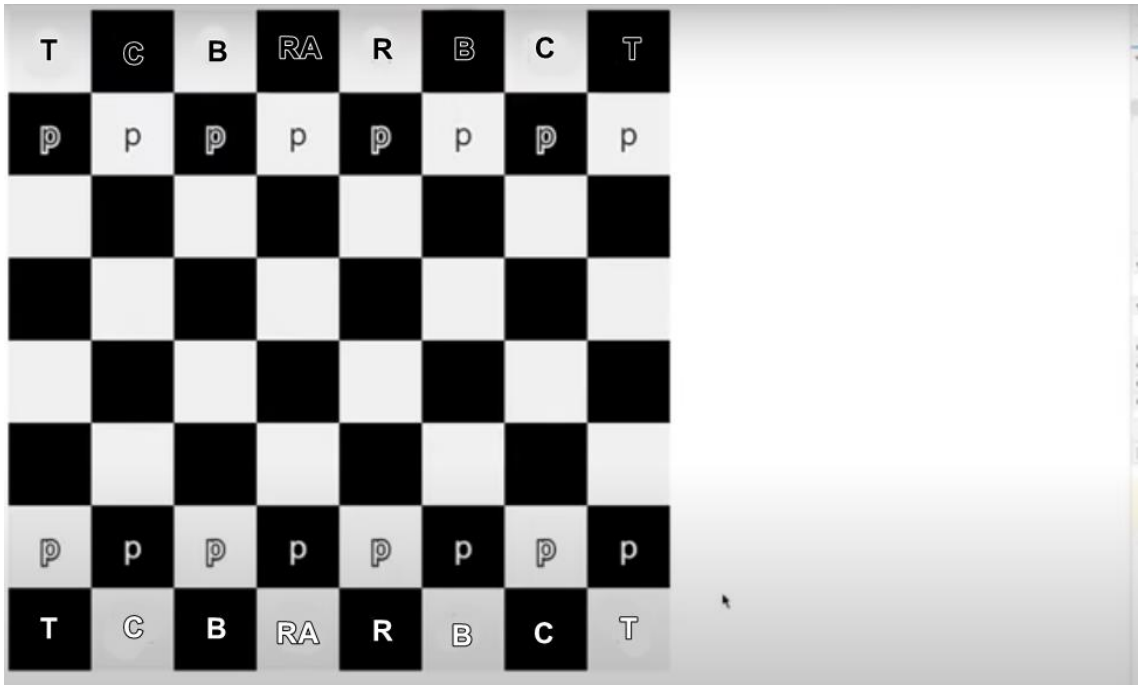
3. Código (Front-End e Back End).

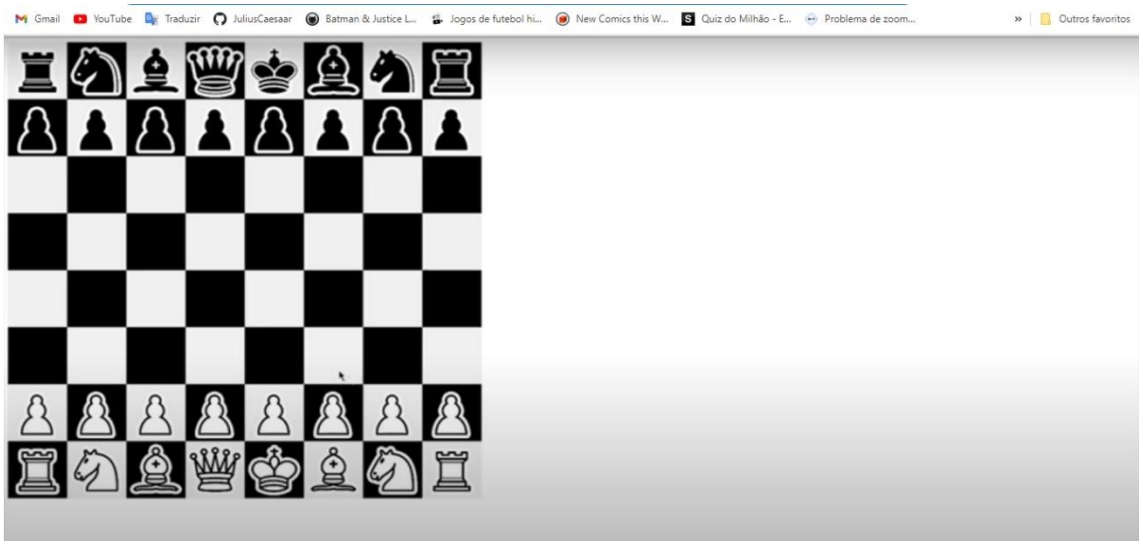
A linguagem escolhida foi Javascript pelo nosso domínio com ela, pela alta compatibilidade com algoritmos de backtracking e também a facilidade de se fazer um belo front end com ela.

Então já começamos botando a mão na massa criando o tabuleiro, as peças e seus locais, onde elas podiam se mover e também as condições de vitória (usamos algumas letras como placeholders enquanto não tinhamos as imagens das peças)

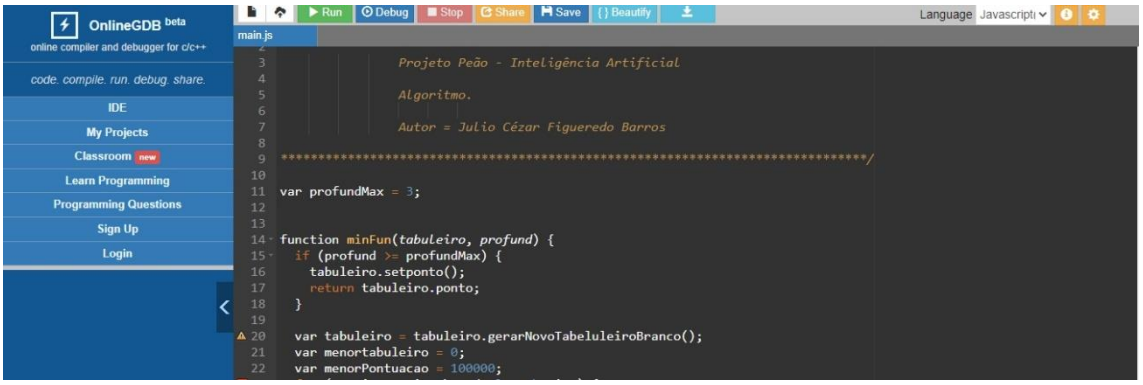
Não entraremos em muitos detalhes sobre o código pois nos estenderiamos muito e o código não é o ponto principal do projeto e sim os resultados. Porém, vale a penas destacar algumas coisas:


OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
Run
Debug
Stop
Share
Save
{ } Beautify
Language
main.js
Projeto Peão - Inteligência Artificial
Tabuleiro.
Autor = Julio Cézar Figueredo Barros
class Board {
constructor() {
this.pecasBrancas = [];
this.pecasPretas = [];
this.pontos = 0;
this.configPecas();
}
configPecas() {
this.pecasBrancas.push(new Rei(4, 7, true));
this.pecasBrancas.push(new Rainha(3, 7, true));
this.pecasBrancas.push(new Bispo(2, 7, true));
this.pecasBrancas.push(new Bispo(5, 7, true));
this.pecasBrancas.push(new Cavalo(1, 7, true));
this.pecasBrancas.push(new Torre(0, 7, true));
this.pecasBrancas.push(new Cavalo(6, 7, true));
this.pecasBrancas.push(new Torre(7, 7, true));





Trocamos os placeholders por peças verdadeiras de Xadrez que achamos no Google Imagens. Elas foram colocadas na pasta “Assets” do nosso programa referenciadas no código.



Inserimos o algoritmo neste momento e o bot estava enfim finalizado. Agora era hora de ligá-lo e começar as sessões de treinamento.

4. Treinamento e Resultados.

Quando se fala em treinar um robô usando aprendizado de máquina, existem quatro abordagens básicas: aprendizado supervisionado, aprendizado não supervisionado, aprendizado semi-supervisionado e aprendizado por reforço. O tipo de algoritmo que os cientistas de dados escolhem usar depende do tipo de dados que desejam prever.

- Aprendizado supervisionado: nesse tipo de aprendizado de máquina, os cientistas de dados fornecem algoritmos com dados de treinamento rotulados e definem as variáveis que desejam que o algoritmo avalie as correlações. Tanto a entrada quanto a saída do algoritmo são especificadas.

- **Aprendizado não supervisionado**: esse tipo de aprendizado de máquina envolve algoritmos que treinam em dados não rotulados. O algoritmo varre os conjuntos de dados procurando por qualquer conexão significativa. Os dados sobre os quais os algoritmos treinam, bem como as previsões ou recomendações que eles produzem, são predeterminados.

- **Aprendizado semi-supervisionado**: essa abordagem de aprendizado de máquina envolve uma mistura dos dois tipos anteriores. Os cientistas de dados podem alimentar um algoritmo principalmente com dados de treinamento rotulados, mas o robô é livre para explorar os dados por conta própria e desenvolver sua própria compreensão do conjunto de dados.

Aprendizado por reforço: os cientistas de dados normalmente usam o aprendizado por reforço para ensinar uma máquina a concluir um processo de várias etapas para o qual existem regras claramente definidas. Os cientistas de dados programam um algoritmo para concluir uma tarefa e fornecem dicas positivas ou negativas à medida que descobrem como concluir uma tarefa. Mas, na maioria das vezes, o algoritmo decide por conta própria quais etapas seguir ao longo do caminho.

No nosso robô, utilizaremos o aprendizado por reforço para treiná-lo. Por quê? Simplesmente pois o aprendizado por reforço dá ao robô um objetivo distinto e um conjunto prescrito de regras para atingir esse objetivo. Com esse tipo de treinamento, nós podemos programar o robô para buscar recompensas positivas - que recebe quando executa uma ação que é benéfica para o objetivo final - e evita punições - que recebe quando executa uma ação que o afasta de seu objetivo final. Exatamente o que estamos procurando.

Então nós passamos a duelar contra esse robô nas horas vagas para treiná-lo, pelo menos umas 30 vezes por dia para cada membro do grupo. Inicialmente ele só conseguia prever um movimento a frente, mas depois de trezentos duelos ele já conseguia prever quatro e até cinco movimentos a frente.

O experimento prático foi considerado um sucesso quando o robô repetiu uma das minhas (Júlio) estratégias que costume usar em jogos de Xadrez como podemos ver na imagem abaixo:

Com isso nós nos demos por satisfeitos, nossa tese estava provada na prática então nós imediatamente partimos para as considerações finais.

5. Considerações Finais.

Aprendizado de máquina está aqui para ficar. Os computadores podem aprender, memorizar e gerar resultados precisos com aprendizado de máquina. Ele permitiu que as empresas tomassem decisões informadas, críticas, para simplificar suas operações de negócios. Essas decisões baseadas em dados ajudam as empresas em setores verticais da indústria, desde manufatura, varejo, saúde, energia e serviços financeiros, a otimizar suas operações atuais enquanto buscam novos métodos para aliviar sua carga de trabalho geral. Funciona e pode ser feito até por estudantes como nós.

À medida que os algoritmos de computador se tornam cada vez mais inteligentes, podemos antecipar uma trajetória ascendente do aprendizado de máquina em 2023 e além.

6. Referências Bibliográficas.

Emerson Alecrim (2018) “Machine learning: O que é e o porquê é importante?” https://tecnoblog.net/responde/machine-learning-ia-o-que-e/

SAS (2017) “O que é e qual é a sua importância?” https://www.sas.com/pt_br/insights/analytics/machine-learning.html

CETAX (2022) “Machine Learning: O que é? Conceito e definição.” https://cetax.com.br/machine-learning/

AMAZON (2019) “O que é Machine Learning?” https://aws.amazon.com/pt/what-is/machine-learning/