Nicholas Spolti

Bruno Bolzan

Eduarda Bertiz

Henrique Xultz

Trabalho Discente Efetivo - Trabalho T1

1. Introdução

O presente trabalho aborda a análise para a atividade proposta na disciplina de Inteligência Artifical, tratando do desenvolvimento de um sistema de IA para o jogo da velha em um tabuleiro 3x3, onde o objetivo da IA, em linhas gerais, é identificar se um dos jogadores ganhou, se houve empate, ou se ainda há jogo.

Nesse contexto, um dataset contendo instâncias do tabuleiro do jogo da velha foi providenciado, ele será avaliado e, se necessário, adequado para que atenda os requisitos necessários para a realização dos testes com os algoritmos de IA.

Serão usados três algoritmos para avaliar a situação do jogo e depois eles serão comparados, esses algoritmos são: k-NN, MLP e Árvore de decisão.

2. Tratamento do Dataset

Antes de começarmos a usar nossos dados nos algoritmos, precisamos garantir a integridade deles, no caso do nosso dataset, precisamos confirmar que não existem posições do tabuleiro em branco e que todos os estados do dataset foram classificados como uma vitória ou uma derrota para o jogador usando o X.

Modificamos algumas características do dataset, para começar, mantivemos o cabeçalho do csv, pois não achamos necessário realizar alterações nele, no entanto, a coluna chamada de *class* sofreu alterações, o termo *positive* foi alterado para *X* e o *negative* para *O.*

Também adicionamos novos dados ao dataset, identificados na coluna *class* como *DRAW* e *GAME*, isso foi feito devido ao fato do dataset apenas identificar vitórias e derrotas do jogador X, sem os dados adicionados, os algoritmos ficam viciados e não conseguem identificar de maneira adequada momentos em que ainda tem jogo, que o jogo já está empatado ou empatou no final.

Para a execução dos algoritmos o dataset foi separado em um conjunto de treino e um de teste, tendo 70% e 30% dos dados, respectivamente.

3. Algoritmos

3.1. k-NN

O algoritmo k-NN é um algoritmo de aprendizado de máquina supervisionado que pode ser usado tanto para classificação quanto para regressão.

O algoritmo encontra os K vizinhos mais próximos de um novo exemplo, com base na distância entre as características desses exemplos e do novo exemplo.

Para classificação, o mais comum entre esses K vizinhos é passado ao novo exemplo. Para regressão, a média ou a mediana dos valores de saída desses K vizinhos é atribuída ao novo exemplo.

3.2. MLP

MLP é um tipo de rede neural artificial que é amplamente utilizado em tarefas de aprendizado de máquina, como classificação e regressão. É composto por várias camadas de neurônios interconectados, incluindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída.

O algoritmo funciona com algumas etapas definidas como vimos em aula, são elas a inicialização, propagação, cálculo de erro, propagação para trás, atualização de pesos e avaliação do desempenho, sendo que as etapas de propagação, cálculo de erro, a propagação para trás e a atualização de pesos são repetidas algumas vezes, definidas por parâmetros definidos pelo programador.

3.3. Árvore de Decisão

Em relação ao uso da árvore de decisão, decidimos utilizar pois é um algoritmo de aprendizado de máquina que permite construir um modelo de classificação baseado em uma estrutura de árvore, onde cada nó representa uma condição ou atributo e cada ramo representa uma possível decisão ou resultado. No nosso contexto, o algoritmo foi utilizado para viabilizar o uso da busca em grade, também conhecida como *Grid Search.*

A técnica utilizada grid search é aplicada para encontrar os melhores hiperparâmetros para o modelo de árvore de decisão, estes hiperparâmetros são valores configuráveis que afetam o comportamento do algoritmo de aprendizado de máquina, nesse caso, os nossos hiperparâmetros são a profundidade máxima e o número mínimo de amostras do nó folha.

A busca em grade examina diferentes combinações desses hiperparâmetros, treina o modelo de árvore de decisão com cada combinação e avalia o desempenho do modelo usando algum critério, como a acurácia, logo, é possível determinar quais hiperparâmetros fornecem os melhores resultados. Após a busca em grade, o modelo de decisão é treinado novamente utilizando os melhores hiperparâmetros encontrados, e em seguida, ele é utilizado para fazer previsões no conjunto de teste, ou seja, é aplicado a novos exemplo de jogadas do jogo "tic-tac-toe" não vistos durante o treinamento.

O modelo de árvore de decisão treinado é atribuído à variável chamada *arvore*, o que permite seu uso posteriormente. No caso do código em questão, ele é utilizado no endpoint */verifyTree* do serviço FastAPI para fazer as previsões com base em uma configuração de jogo fornecida. Ou seja, o modelo de árvore de decisão é aplicado em tempo real para classificar uma configuração de jogo específica e retornar a previsão de resultado.

4. Conclusão

Em conclusão, este trabalho abordou o desenvolvimento de um sistema de aprendizado de máquina para o jogo da velha, utilizando três algoritmos: k-NN, MLP e Árvore de Decisão. Os dados foram tratados para garantir sua integridade, e o dataset foi dividido em conjuntos de treino e teste.

O algoritmo k-NN encontra os vizinhos mais próximos com base na distância entre características e classifica ou atribui um valor de saída com base nos vizinhos selecionados. O MLP é uma rede neural artificial com várias camadas interconectadas, utilizado para classificação e regressão. A Árvore de Decisão é um modelo de classificação construído com base em uma estrutura de árvore, onde cada nó representa uma condição e cada ramo representa uma decisão possível.

Foi utilizada a busca em grade (Grid Search) com a Árvore de Decisão para encontrar os melhores hiperparâmetros. Esses hiperparâmetros foram ajustados para obter o melhor desempenho do modelo. O modelo treinado foi utilizado para fazer previsões no conjunto de teste e avaliar seu desempenho em dados não vistos anteriormente.

Baseado em nossa análise, decidimos utilizar o k-NN como algoritmo no front end, o MLP tem uma acurácia maior do que os outros, no entanto, ao medirmos o f1, percebemos que o k-NN supera os outros algoritmos com uma boa margem, e portanto ele foi o escolhido.

5. Referências

Documentação do scikit-learn (biblioteca Python para aprendizado de máquina): Decision Trees
Disponível em: https://scikit-learn.org/stable/modules/tree.html

Documentação do scikit-learn: GridSearchCV (utilizada para busca em grade)
Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Artigo: "A Guide to Hyperparameter Tuning in Decision Trees" (Autor: Analytics Vidhya)
Disponível em: https://www.analyticsvidhya.com/blog/2020/08/guide-advanced-hyperparameter-tuning-decision-trees/

Artigo: "Hyperparameter Tuning in Decision Tree using GridSearchCV" (Autor: Prashant Gupta)
Disponível em: https://towardsdatascience.com/hyperparameter-tuning-in-decision-tree-using-gridsearchcv-28d2aa77dd74