Uso de MLP para classificar classes de doenças

Gustavo H. M. Orlandini, Luiz E. Garzon, Paulo R. Scalon, Pedro H. Berti

[1]Universidade Estadual do Oeste do Paraná - UNIOESTE

Abstract. *This article describes a class disease classification system using a Multi-layer Perceptron (MLP) neural network. The system operates based on a structured database containing a variety of symptoms. From the input symptoms, the model is capable of predicting the corresponding disease category. The process includes data preparation and processing to ensure robustness and accuracy.*

Resumo. *Este artigo descreve o sistema para classificação de classes de doenças, utilizando rede neural do tipo Perceptron Multicamadas (MLP). O sistema opera a partir de uma base de dados estruturada, contendo uma variedade de sintomas. A partir da entrada dos sintomas, o modelo é capaz de prever a categoria da doença correspondente. O processo inclui a preparação e o tratamento dos dados para garantir a robustez e a acurácia.*

1. Introdução

O presente trabalho aborda o problema da classificação de classes de doenças a partir de sintomas relatados, utilizando técnicas de rede neural artificial (RNA). O sistema, denominado Disease Prediction System, realiza prognósticos automáticos de classes de doenças utilizando uma base de dados. A base em questão contém 134 sintomas distintos e 42 doenças, permitindo ao modelo classificar a doença mais provável com base nos sintomas apresentados. O funcionamento dos datasets segue uma estrutura binária: cada sintoma é marcado com 0 para indicar que o sintoma não está presente e 1 para sinalizar sua presença. O problema de classificação enfrentado é abordado por meio de rede neural do tipo Multi-Layer Perceptron (MLP), que foi escolhida por sua capacidade de lidar principalmente com dados não-lineares.

O principal objetivo desta aplicação é oferecer uma ferramenta eficiente de prognóstico, visando reduzir o tempo necessário para diagnósticos preliminares e aumentar a acurácia ao lidar com múltiplos sintomas. Ao automatizar a análise dos sintomas, o sistema não substitui o julgamento clínico, mas serve como um suporte que pode melhorar o processo de diagnóstico prévio.

2. Metodologia, Ambiente e Arquitetura Utilizados

Para o desenvolvimento do projeto, por decisão de simplicidade e compatibilidade foi utilizado o Google Colab.

A base completa é composta pelo dataset com 314 amostras, 134 sintomas distintos com pesos de 1 a 7 para cada sintoma, esses pesos estão especificados no arquivo ‘Symptom Severity‘. A base de teste contém 43 amostras do dataset, e a base de treinamento possui 307 amostras do dataset. Devido ao número total de 45 doenças presentes no sistema, a criação de uma saída para cada uma dessas doenças resultaria em

um modelo com 45 saídas distintas para os neurônios da rede neural, o que tornaria o processo de treinamento e teste complexo e computacionalmente custoso. Para mitigar esse desafio e otimizar o desempenho do modelo, optou-se por agrupar as 45 doenças em 9 classes distintas com base em suas similaridades clínicas e características comuns, sendo elas: respiratória, infecciosa, inflamatória/autoimune, metabólica/endócrina, gastrointestinal/hepática, dermatológica, cardiovascular/circulatório, reações alérgicas e neurológica. A coluna com os nomes das doenças distintas na base de dados também foi alterada para sua respectiva classe. Essa abordagem reduziu significativamente a quantidade de saídas, estabelecendo 9 saídas correspondentes às 9 classes de doenças. Agrupar as doenças em classes simplifica o problema de classificação, deixando o modelo mais simples e diminuindo o tempo de treinamento, além de também generalizar a rede neural, já que os sintomas semelhantes são associados às classes relacionadas.

O pré-processamento dos dados envolveu a normalização e a codificação dos sintomas e diagnósticos. A normalização dos dados foi realizada com a biblioteca StandardScaler do Scikit-learn. Além disso, a técnica de Label Encoding foi aplicada para transformar os rótulos categóricos de doenças em valores numéricos. As saídas de diagnóstico foram convertidas para o formato one-hot encoding, que é útil para a saída de um modelo de classificação, onde cada diagnóstico tem sua própria ”posição” na saída. Os sintomas dos datasets carregados foram separados da coluna de prognóstico para facilitar o processamento na rede neural. Foram utilizados os datasets ”Training” com 307 amostras de treino e ”Testing” com 43 amostras de teste, para o treinamento e validação do modelo.

A arquitetura da rede neural foi definida com base nas características dos dados. Foi implementada uma camada oculta com 72 neurônios, utilizando a função de ativação ReLU (Rectified Linear Unit). A camada de saída contém 9 neurônios, representando as classes de doenças, e utiliza a função de ativação softmax, adequada para problemas de classificação multiclasse, já que gera probabilidades associadas a cada classe.

O treinamento foi realizado utilizando o otimizador Adam, pela sua eficiência e baixa demanda por ajustes de hiperparâmetros, com a taxa de aprendizado definida em 0.001. A função de perda escolhida foi a categorical crossentropy, para problemas de classificação multiclasse, enquanto a métrica monitorada durante o treinamento foi a acurácia. E, para evitar o sobreajuste (overfitting), foi implementada a técnica de early stopping, interrompendo o treinamento caso não haja melhora no desempenho no conjunto de validação após cinco épocas consecutivas. Os melhores pesos foram restaurados automaticamente ao final do treinamento, com base na melhor performance obtida durante a validação.

O modelo foi treinado por um máximo de 15 épocas, com um tamanho de lote de 4 amostras por iteração. O treinamento foi conduzido com os dados ponderados e normalizados, e o desempenho foi avaliado com base na matriz de confusão, que permitiu mensurar a precisão da classificação para cada uma das classes de doenças.

3. Resultados e Conclusões

O modelo de Rede Neural foi treinado com uma base de dados composta por 314 amostras. Durante o treinamento, a rede apresentou uma boa taxa de acerto, variando entre 99% e 100%, mesmo após várias execuções. Essa alta taxa de acerto se deve bas-

tante ao agrupamento das 45 doenças em 9 classes diferentes, levando em consideração a semelhança dos sintomas. A rede neural foi configurada com a função de ativação ReLU na camada oculta, que evita o problema do gradiente desaparecendo. Na camada de saída, foi utilizada a função softmax, pois gera probabilidades associadas a cada uma das classes. O otimizador selecionado foi o Adam, por sua eficiência em ajustar parâmetros de forma adaptativa e baixa necessidade de ajuste de hiperparâmetros, o que facilita a estabilidade e convergência.

Para avaliar o desempenho do modelo, foi utilizada uma matriz de confusão. Houve um pequeno índice de confusão entre as classes 2 e 6, pois apresentaram sintomas muito semelhantes. Em todos os testes, o maior erro observado foi de apenas duas classificações incorretas. Além disso, com um número reduzido de neurônios na camada oculta definido pela fórmula '(n+s)/2' (134 + 9)/2 , o modelo atingiu uma taxa de acerto de 100% em até 15 épocas, com a configuração de batch size entre 4 e 10 amostras.

O sistema mostrou-se eficaz para a classificação de doenças predefinidas a partir dos sintomas apresentados. Agrupar as doenças em classes diminuiu a complexidade do problema permitindo um bom treinamento com uma alta taxa de acerto.

Referências

ANBUAZHAGAN, Saarathi. A Complete Guide to train Multi-Layered Perceptron Neural Networks. 2021. Disponível em: https://paarthasaarathi.medium.com/a-complete-guide-to-train-multi-layered-perceptron-neural-networks-3fd8145f9498. Acesso em: 26 out. 2024.

HEATON, J. Introduction to Neural Networks with Java. 2. ed. St. Louis: Heaton Research, Inc, 2008.

JAISWAL, Sejal. Multilayer Perceptrons in Machine Learning: A Comprehensive Guide. 2024. Disponível em: https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning. Acesso em: 26 out. 2024.

SENA, Marcus. Building a Perceptron from Scratch: A Step-by-Step Guide with Python. 2023. Disponível em: https://python.plainenglish.io/building-a-perceptron-from-scratch-a-step-by-step-guide-with-python-6b8722807b2e. Acesso em: 26 out. 2024.

VAN OTTEN, Neri. Multilayer Perceptron Explained And How To Train & Optimise MLPs. 2024. Disponível em: https://spotintelligence.com/2024/02/20/multilayer-perceptron-mlp. Acesso em: 26 out. 2024.