Relatório de aplicação de RNA em problema de classificação Binária

Bruno M. Pires[1]

[1]Departamento de Ciências da Computação
Universidade do Estado de Santa Catarina (UDESC)
Caixa Postal 15.064 – 88.035-901 – Joinville – SC – Brasil

bruno.pires@edu.udesc.br

Abstract. *This article explores the application of Artificial Neural Networks (ANNs) in binary classification tasks, specifically focusing on the implementation and training of a Multilayer Perceptron (MLP) using the Breast Cancer Wisconsin dataset. The study leverages Python along with TensorFlow for constructing the neural network architecture and scikit-learn (sklearn) for data manipulation and model evaluation.*

Resumo. *Este artigo explora a aplicação de Redes Neurais Artificiais (RNAs) em tarefas de classificação binária, concentrando-se especificamente na implementação e treinamento de um Perceptron Multicamadas (MLP) utilizando o conjunto de dados Breast Cancer Wisconsin. O estudo utiliza Python em conjunto com TensorFlow para construir a arquitetura da rede neural e scikit-learn (sklearn) para manipulação de dados e avaliação do modelo.*

1. Introdução

As Redes Neurais Artificiais (RNAs) têm se destacado como poderosas ferramentas no campo de aprendizado de máquina, especialmente em tarefas de classificação. Este trabalho propõe uma implementação e treinamento de uma Rede Neural do tipo Multilayer Perceptron (MLP) para a classificação de dados, utilizando o conjunto de dados Breast Cancer Wisconsin. A escolha deste conjunto de dados foi realizada apenas como forma de padronizar a entrega durante a disciplina de Inteligência Artificial ministrada na UDESC e foi obtido a partir do repositório da UCI Machine Learning.

Para alcançar os objetivos propostos no trabalho, foi utilizada a linguagem python juntamente com a biblioteca TensorFlow para construção da arquitetura da rede neural, e a biblioteca scikit-learn (sklearn) para facilitar a manipulação do conjunto de dados e a avaliação do modelo.

O processo de treinamento será conduzido em etapas, começando pela definição da arquitetura da rede neural, que compreenderá a quantidade de camadas ocultas, o número de neurônios em cada camada e as funções de ativação. O algoritmo de otimização escolhido para calcular os pesos será o Adam, o qual, em conjunto com o backpropagation, maximizará a eficiência do treinamento.

Em Resumo, este relatório visa proporcionar uma compreensão do processo de implementação e treinamento de RNAs para a classificação de dados, destacando a importância das escolhas arquiteturais e parâmetros no desempenho final do modelo.

2. Desenvolvimento

Como mencionado anteriormente, foi utilizada a linguagem python para desenvolvimento dos scripts, sendo o primeiro passo e mais importante carregar o dataset, existem algumas formas de se fazer isso, como baixar todo conjunto de dados e ler manualmente, mas com o uso da biblioteca ”ucimlrepo” conseguimos importar o exato dataset com apenas uma linha de código. O problema encontrado nessa abordagem é que ficamos dependentes do servidor deles estar online, coisa que não era verdade em alguns momentos, mas ainda sim foi possível realizar os testes necessários nos momentos em que estava online. Após a importação dos dados, foi necessário retirar os dados faltantes do dataset verificando se os valores eram ”NaN”, evitando dessa forma os outliers.

A segunda etapa compreende a normalização dos dados, aqui foi utilizado o método ”MinMaxScaler” da biblioteca ”scikitlearn” para padronizar os dados em X entre [-1,1], isso é útil pois ajuda a colocar todas as características em uma escala similar evitando que uma característica com valores grandes tenha um impacto desproporcional em comparação com características com valores menores. Para Y, que inicialmente assumia os valores 2 ou 4, representando respectivamente tumores benígnos e malígnos, foi transformado em 0 e 1.

A terceira etapa é responsável pela divisão dos dados em conjunto de treinamento e teste, utilizando o método ”trainTestsSplit()”, que foi recomendado na proposta do trabalho.

Na quarta etapa foi definida a arquitetura da rede neural utilizando a biblioteca TensorFlow. Sendo a camada de entrada uma camada densa (Totalmente conectada) com um número de neurônios igual ao número de colunas em ”X”, e a função de ativação utilizada é a ReLU. Para camadas ocultas, inicialmente temos 3 camadas com respectivamente 40, 40 e 50 neurônios utilizando também a função de ativação ReLU. E finalmente, para a camada de saída, apenas um neurônio indicando uma tarefa de classificação binária. A função utilizada é a sigmoid após pesquisa e ver que é a função comumente utilizada pra problemas de classificação binária.

Na quinta etapa foi definido um otimizador utilizando um método da biblioteca ”keras” chamado ”compile”, sendo também recomendado na proposta do trabalho a utilização do Adam como otimizador, a função de perda passada como parâmetro para o método ”compile” foi a ’binaryCrossentropy’, que é apropriada pra problemas de classificação binária. E a métrica usada para avaliar o desenpenho foi a acurácia.

Na sexta etapa foi realizado o treinamento do modelo utilizando a função fit, passando como parâmetro os dados de treinamento e seus rótulos, a quantidade de épocas que foi recomendada na proposta do trabalho como 50 e a proporção dos dados que será usada para validação, que foi 0.2 equivalente a vinte por cento dos dados.

Na sétima e oitava etapa, avaliamos o modelo e comparamos resultados com diferentes arquiteturas.

3. 3 Camadas Ocultas e 0.01 de Taxa de Aprendizado

Uma camada oculta em uma rede neural pode ser comparada a um grupo de ”especialistas” intermediários que trabalham de maneira colaborativa para realizar uma tarefa complexa. Imagine uma equipe em uma cozinha, onde cada cozinheiro é especializado em um

aspecto específico da preparação de um prato elaborado. Da mesma forma, cada neurônio em uma camada oculta é especializado em identificar padrões específicos nos dados de entrada. À medida que os dados passam por esses especialistas intermediários, eles são transformados e refinados em representações mais abstratas e significativas, semelhante aos ingredientes sendo combinados e processados na cozinha. A interação entre esses especialistas permite que a rede neural aprenda relações complexas e representações de alto nível, assim como uma equipe culinária combina habilidades individuais para criar uma experiências únicas. A profundidade da rede, representada pelo número de camadas ocultas, é crucial, assim como o número de cozinheiros na cozinha influencia a sofisticação de um menu, mas não necessariamente a qualidade do resultado é diretamente proporcional a quantidade de camadas. Resumidamente, é a colaboração dessas camadas ocultas que capacita a rede a capturar padrões sutis nos dados e realizar tarefas de aprendizado de diferentes complexidades.

Para o problema de classificação que estamos enfrentando, inicialmente montamos uma configuração de 3 camadas ocultas, totalizando 5 camadas no total quando contadas a de entrada e saída. A quantidade de neurônios em cada camada foi escolhida de forma empírica com base em alguns padrões encontrados pela internet e foi a melhor configuração testada. A configuração final ficou como segue:

- Camada de entrada: Totalmente conectada com número de neurônios igual ao número de colunas nos dados de entrada, e função de ativação ”relu”
- 2 Camadas ocultas: Totalmente conectadas com 40 neurônios e função de ativação ”relu”
- 1 Camada oculta: Totalmente conectada com 50 neurônios e função de ativação ”relu”
- Camada de Saída: 1 neurônio com função de ativação ”sigmoid”, comum em problemas de classificação binária.

O otimizador utilizado foi o adam e a taxa de aprendizado foi de 0.01 neste caso. Quatro gráficos foram gerados para avaliar este experimento, a figura 1 mostra a acurácia do modelo no decorrer das épocas, algo similar a um gráfico de convergência, onde conseguimos ver que ambas as curvas, dos dados de treinamento e validação, se aproximam e seguem próximas um bom sinal de que o modelo está convergindo e generalizando bem.

A figura 2 apresenta o gráfico de perda para os dados de treinamento e validação, até a décima época o modelo estava aprendendo muito bem, a partir dali durante algumas épocas é possível verificar um comportamento de overfitting, onde temos a curva de treinamento diminuindo mas a curva de validação começa a subir, nesse ponto talvez o modelo estivesse se ajustando demais aos dados e perdendo capacidade de generalização, mas depois o comportamento voltou ao normal.

A figura 3 apresenta a matriz de confusão, deixando claro os resultados ”Verdadeiro Positivo”, ”Falso Positivo”, ”Falso Negativo” e ”Verdadeiro Negativo”, fornecendo uma visão mais abrangente do desempenho do modelo. Sendo 3 Falso negativo e 1 Falso positivo.

A figura 4 apresenta o gráfico da curva ROC, é possível perceber que a curva se aproxima muito canto superior esquerdo, e que a área sobre a cuva é igual a ”1.00” evidenciando um desempenho perfeito do modelo

4. 3 Camadas Ocultas e 0.1 de Taxa de Aprendizado

Para este caso, mantemos a mesma configuração de camadas e épocas do caso anterior, e variamos apenas a taxa de aprendizado como recomendado na proposta do trabalho. A taxa utilizada agora é de ”0.1”. As figuras 5, 6, 7, 8 mostram os resultados obtidos. Em comparação com o primeiro teste conseguimos diferenciar apenas na existência de uma grande oscilação nos valores nas primeiras épocas, mas no decorrer do tempo isso é resolvido e no geral o algoritmo responde bem assim como o primeiro. A área sobre a curva ROC aponta um valor de ”0.99” sendo pior que o primeiro teste mas provavelmente uma diferença que não traz nenhum grande problema, visto a proximidade com o valor perfeito ”1.00”. A matriz de confusão ficou muito parecida, com apenas 2 casos a mais onde o algoritmo falhou.

5. Conclusão e Trabalhos Futuros

O problema atacado de classificação foi bastante simples, mas serviu muito bem com o objetivo de iniciar e colocar a mão na massa em tecnologias que são realmente utilizadas na solução de problemas mais complexos. Foi possível compreender a importância da definição dos parâmetros do ambiente, por mais que muitas vezes essa definição é ajustada com base em tentativas e erros, ainda é interessante observar as mudanças nos resultados com pequenos ajustes.

Como trabalhos futuros, daria para construir uma ferramenta para melhorar a visualização dos resultados, utilizando talvez a biblioteca ”streamlit”, e também para facilitar o ajuste de parâmetros. E realizar testes em diferentes problemas de classificação, o trabalho atacou apenas um problema de classificação binário simples.

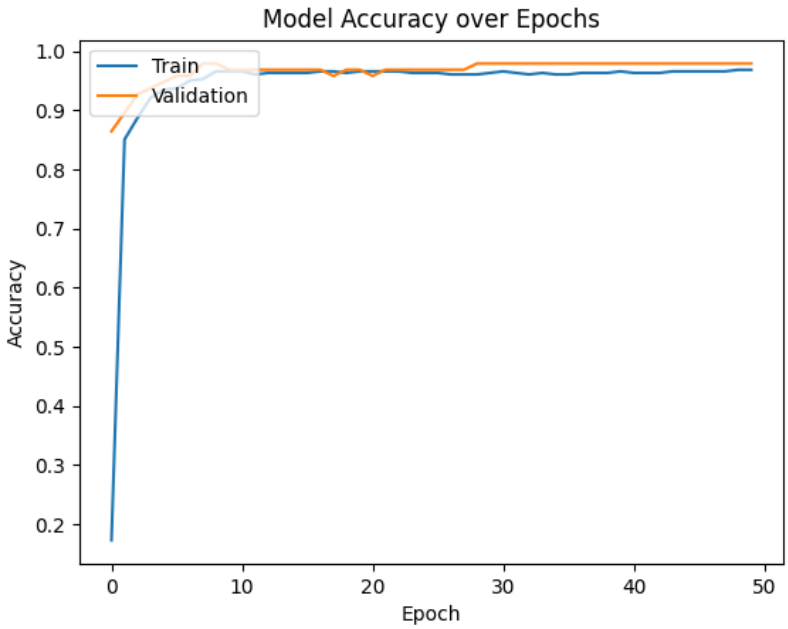

Figure 1. Gráfico de Acurácia para primeiro teste

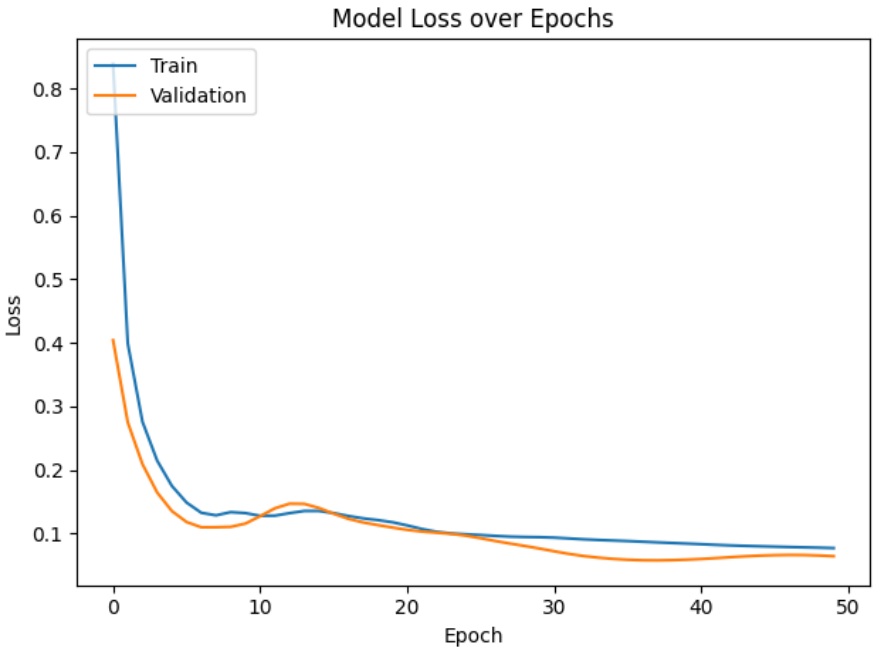

Figure 2. Gráfico de Perda para primeiro teste

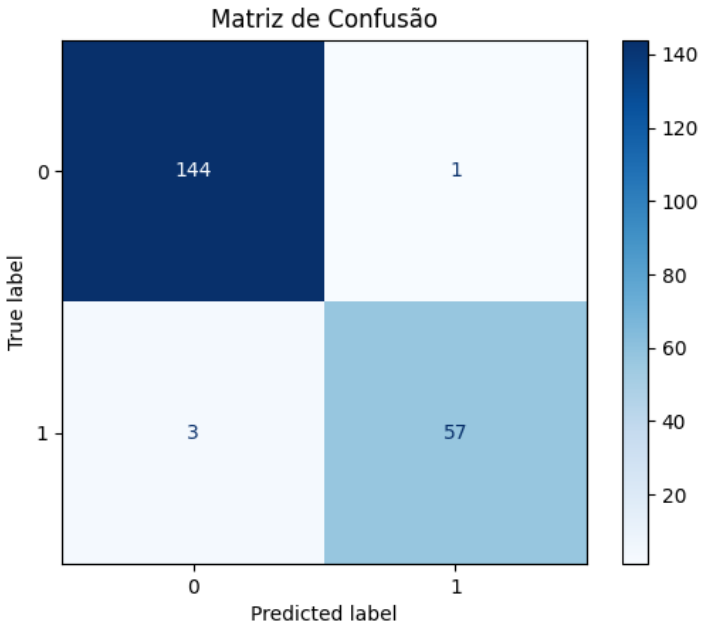

Figure 3. Matriz de Confusão para primeiro teste

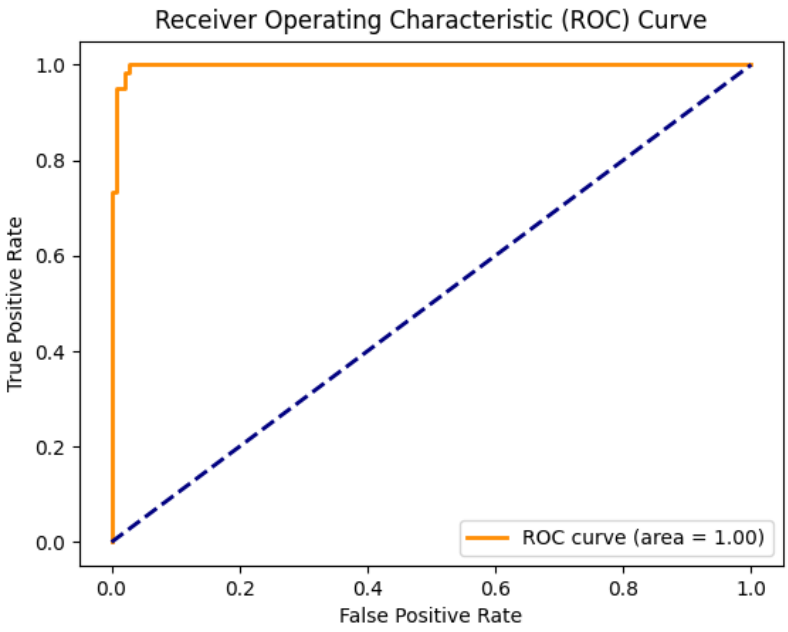

Figure 4. Gráfico ROC para primeiro teste

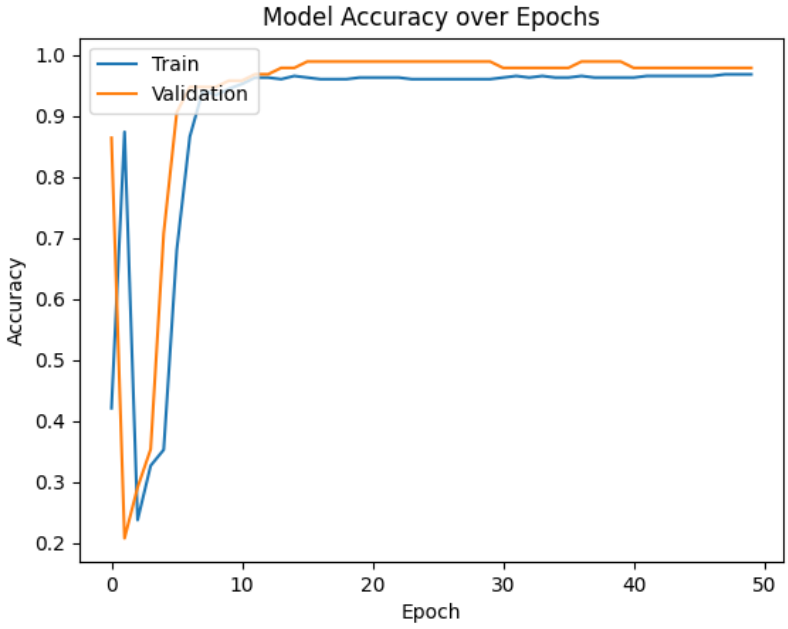

Figure 5. Gráfico de Acurácia para segundo teste

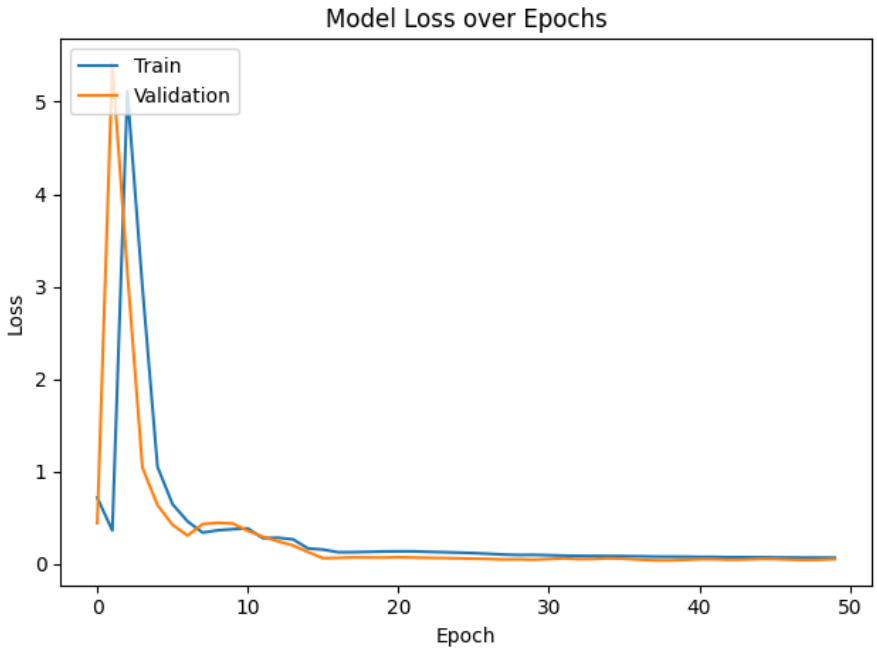

Figure 6. Gráfico de Perda para segundo teste

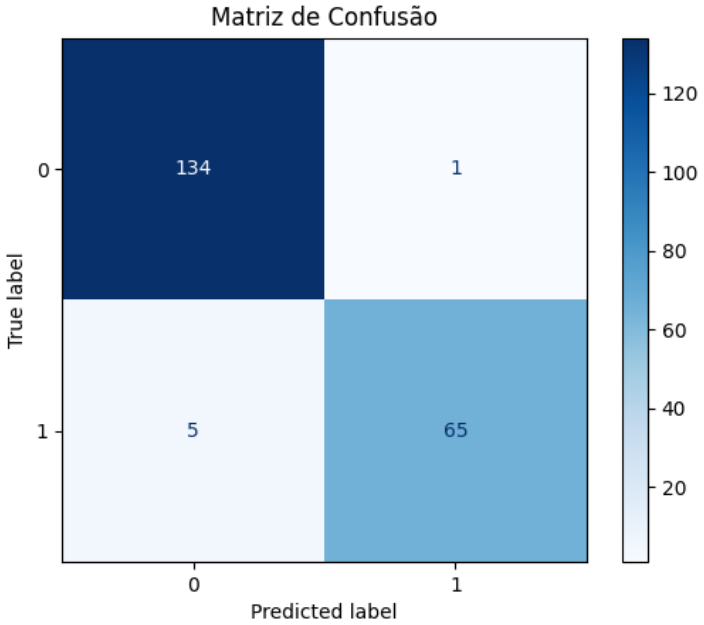

Figure 7. Matriz de Confusão para segundo teste

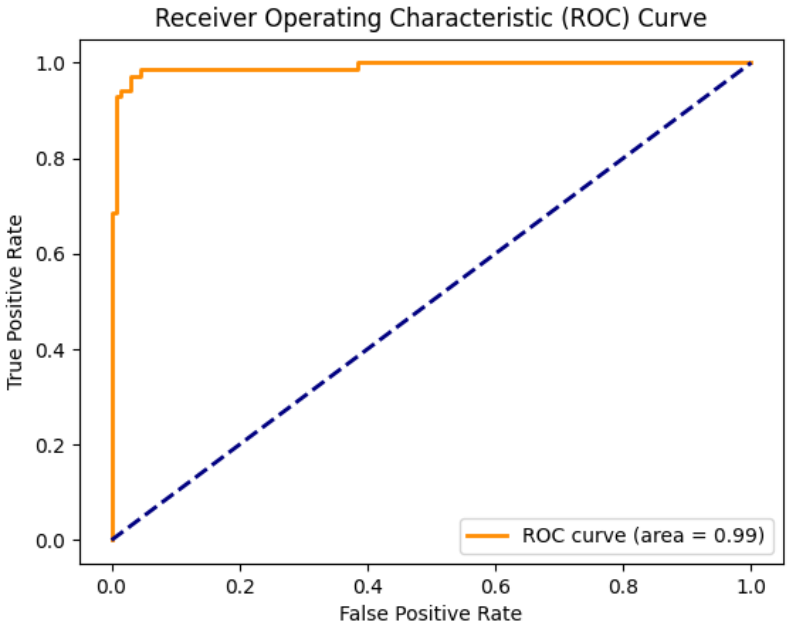

Figure 8. Gráfico ROC para segundo teste