Classificação de Flores Iris por Rede Neural MLP com Backpropagation

Eduardo de Almeida Barboza[1], Mauricio Fabiano Azevedo[1]

[1]Ciência da computação – Universidade Estadual do Centro Oeste (UNICENTRO)
CEP 85040-167 – Guarapuava – PR – Brazil

blackhuebr@gmail.com, mauricio.fabiano.azevedo@gmail.com

Abstract. *This article presents the implementation of a multilayer perceptron (MLP) neural network with the purpose of classifying the Iris flower dataset, including details on data preprocessing, the code, model architecture, and optimized training. It also discusses the impact of hyperparameters (number of layers/neurons and learning rate) based on multiple runs, highlighting the differences between each attempt and how training improved the analysis of the Iris dataset by reducing errors. The best run achieved an accuracy rate of 93.33%.*

Resumo. *Neste artigo será apresentada a implementação de uma rede neural perceptron multicamada (MLP), com a finalidade de classificar um conjunto de dados sobre a flor Iris, trazendo detalhes sobre o seu processamento, o código, a arquitetura do modelo e o treinamento otimizado. Também será mostrado o impacto dos parâmetros (número de camadas/neurônios e a taxa de aprendizado) com base em mais de uma execução, evidenciando a diferença entre cada tentativa e como o treinamento melhorou a análise sobre as Iris, diminuindo os erros. A melhor execução alcançou uma taxa de 93,33% de acerto.*

1. Introdução

As redes neurais artificiais (RNA) são um dos principais métodos de aprendizado de máquina na atualidade, permitindo que sistemas aprendam representações a partir de dados e executem desde tarefas simples até aplicações mais complexas, como classificação, regressão, detecção de padrões e resolução de problemas.

Este artigo utiliza RNA para a classificação supervisionada de três espécies de flores *Iris* (*Iris-setosa*, *Iris-versicolor* e *Iris-virginica*), a partir de um conjunto com mais de 150 amostras, descritas por quatro atributos principais: comprimento e largura de sépalas, além de comprimento e largura de pétalas.

O trabalho descreve a implementação de uma MLP com backpropagation em Python, utilizando as bibliotecas TensorFlow/Keras, Pandas e Scikit-learn, com o objetivo de demonstrar de forma didática e simples todas as etapas envolvidas: leitura e preparação dos dados, definição da arquitetura, treinamento, avaliação, aprendizado e evolução durante as épocas. Por fim, discute-se também o impacto dessas etapas no desempenho geral do estudo.

2. Fundamentação Teórica

Uma MLP basicamente consiste em várias camadas totalmente conectadas, onde cada neurônio realiza uma transformação para que os dados de entrada tenham uma ativação não linear, permitindo que a rede modele fronteiras de decisão complexas. As camadas ocultas da MLP utilizarão a função de ativação da unidade linear retificada (ReLU). A camada de saída usará o softmax para produzir uma distribuição de probabilidade sobre as três classes da íris, sendo que o aprendizado será guiado pelo gradiente descendente estocástico (SGD), e os parâmetros serão ajustados pelo backpropagation, que aplica a regra da cadeia para propagar o erro da saída para as camadas anteriores.

O modelo terá métricas de avaliação que incluem a taxa de acertos e medidas por classe, como precisão (previsão de acertos), recall (proporção de acertos entre as classes) e F1 score (média entre precisão e recall). Essas métricas permitem avaliar o desempenho global, assim como o equilíbrio de acerto entre as classes, muito relevante neste artigo, pois é comum que a Iris versicolor e a Iris virginica sejam confundidas.

3. Materiais e Métodos

Nessa sessão serão apresentados os dados e o processamento. O arquivo iris.csv contém 150 amostras com os três parâmetros e a classe do alvo. O script carregará os dados com o pandas, que basicamente é uma biblioteca usada para manipulação e análise de dados, e irá separar em X e Y, correspondendo, respectivamente, aos atributos e aos rótulos. Em seguida, os rótulos textuais são transformados em inteiros com LabelEncoder, e depois aplicados one-hot encoding com to_categorical, que são requisitos para o Softmax. Os atributos serão então normalizados com o StandardScaler, o que acelera e estabiliza o treinamento da rede. A divisão entre treino e teste é feita com train_test_split, usando 80% dos dados para treino e 20% para teste, sempre preservando a proporção de classes para validação.

A arquitetura da MLP base terá duas camadas ocultas: a primeira com 10 neurônios e função de ativação ReLU, e a segunda com 8 neurônios também com ReLU. A camada de saída possui 3 neurônios com Softmax, correspondendo às três espécies. O modelo será compilado com o otimizador SGD (taxa de aprendizado 0,01), a função de perda categorical_crossentropy e a métrica de acurácia. O treinamento ocorre por 100 épocas, com batch size de 16, registrando perda e acurácia para treino e validação.

O passo a passo do código inicia com a separação dos dados pelo pandas, onde X recebe os atributos e Y as classes. Em seguida, aplica-se o codificador de rótulos com LabelEncoder e to_categorical, convertendo as classes textuais em números inteiros e depois em vetores one-hot. Com os atributos normalizados, a base é dividida em conjuntos de treino e teste usando train_test_split, mantendo a distribuição das classes. O modelo é definido como sequencial e em camadas, com ReLU nas camadas ocultas e Softmax na saída. A compilação utiliza SGD com taxa de aprendizado 0,01 e função de perda categorical_crossentropy, registrando a taxa de acertos a cada época. O histórico do treinamento armazena perda e acurácia para treino e validação.

Após todas essas etapas, o modelo é avaliado no conjunto de teste e gera um relatório detalhado por classe. Além disso, são gerados gráficos com as curvas de perda e acurácia, permitindo uma análise visual mais clara do aprendizado.

4. Resultados

As curvas do treinamento indicam uma redução consistente da perda e o aumento da taxa de acerto, tanto no treino quanto na validação. Na Figura 1 é ilustrado o primeiro dos treinamentos, em que as curvas de aprendizado acompanham as do treino sem divergência, apenas sugerindo que seria necessário um pequeno ajuste. Nesse experimento, observou-se uma taxa de acerto de 86,67%. Após algumas execuções com ajuste dos hiperparâmetros, o resultado chegou a atingir 93,33% de taxa de acerto. Valores esses que são exatamente os esperados para o trabalho, demonstrando, além disso, como pequenas mudanças na profundidade ou na taxa de aprendizado podem impactar significativamente a geração desses resultados.

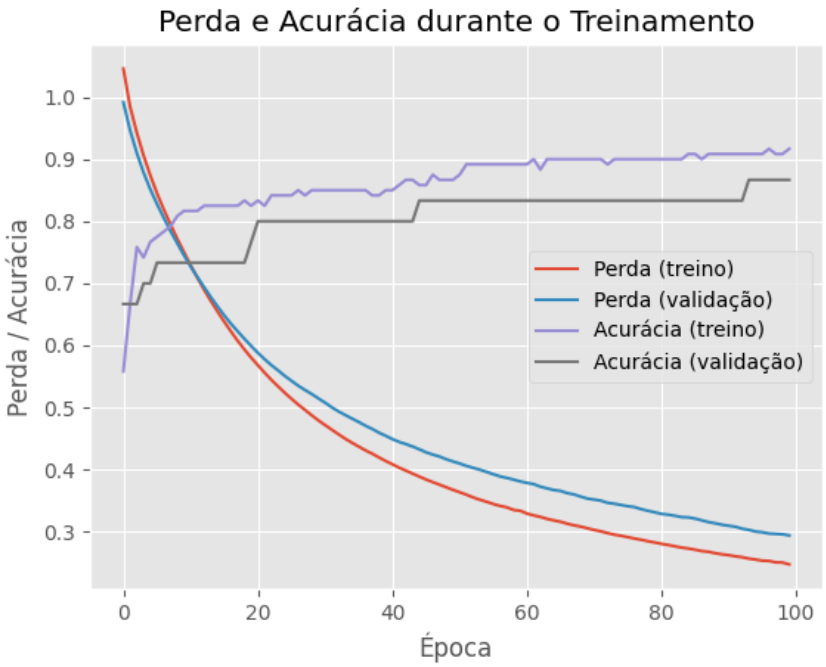

Figura 1. Perda e acurácia por época (primeira execução).

```
[INFO] Relatório de Classificação:

              precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      0.60      0.75        10
 Iris-virgínica       0.71      1.00      0.83        10

       accuracy                           0.87        30
      macro avg       0.90      0.87      0.86        30
   weighted avg       0.90      0.87      0.86        30


[INFO] Acurácia final no conjunto de teste: 86.67%
```

Figura 2. Relatório de classificação (primeira execução).

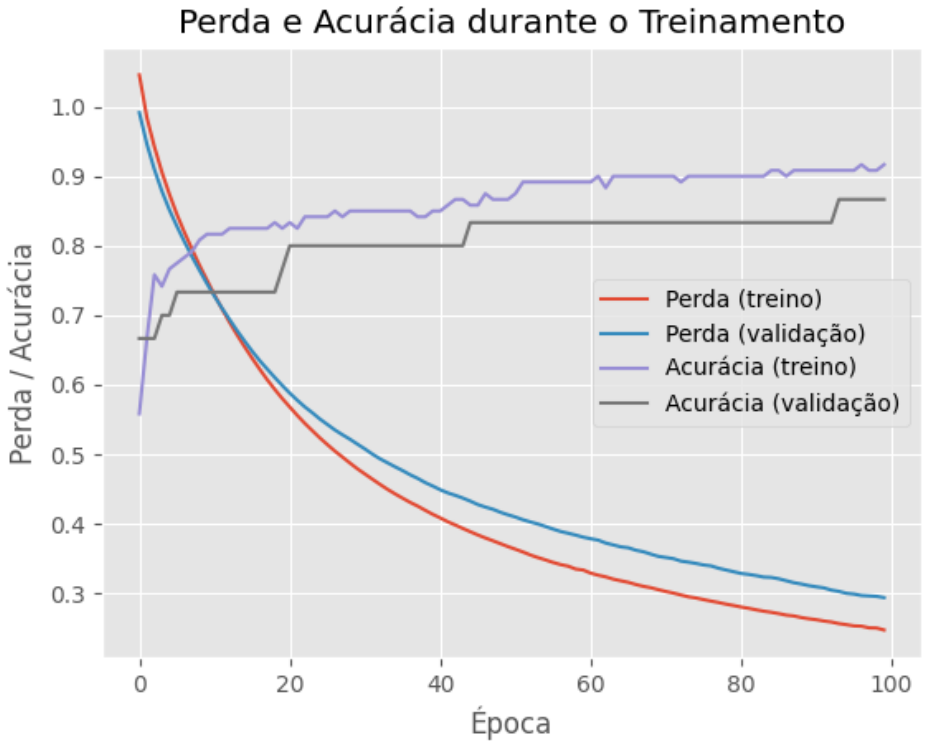

Figura 3. Perda e acurácia por época (última execução).

```
[INFO] Relatório de Classificação:

                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      0.80      0.89        10
 Iris-virgínica       0.83      1.00      0.91        10

       accuracy                           0.93        30
      macro avg       0.94      0.93      0.93        30
   weighted avg       0.94      0.93      0.93        30


[INFO] Acurácia final no conjunto de teste: 93.33%
```

Figura 4. Relatório de classificação (última execução).

5. Conclusão

Este artigo demonstrou a implementação completa de uma MLP para classificar o conjunto de dados das flores Iris, detalhando a preparação, a arquitetura e o treinamento com SGD. As execuções reportadas que variaram entre 86,67% e 93,33% de taxa de acerto no conjunto de teste. O estudo evidencia que, mesmo com redes pequenas, é possível obter ótimos resultados, desde que se apliquem de uma boa maneira o pré-

processamento e ajuste cuidadoso de hiperparâmetros. Trabalhos futuros podem incluir busca sistemática (grid/random search) dos hiperparâmetros, comparação com outros otimizadores (Adam, RMSProp) e inclusão de regularização (dropout ou L2).

Referências

Sousa, J. R. (2020) “Python e predição de dados usando redes neurais multicamadas”, *Brazilian Journal of Development*, disponível em: https://ojs.brazilianjournals.com.br/ojs/index.php/BRJD/article/view/14311.

Vicentini, E. D. (2021) “Introdução às redes neurais para regressões não-lineares”, *Química Nova*, disponível em: https://www.scielo.br/j/qn/a/gDX7ZGVnPPdBPfjvLpdqC4d/?lang=pt

Arshid, M. (2025) “Iris Flower Dataset”, disponível em: https://www.kaggle.com/datasets/arshid/iris-flower-dataset

Sociedade Brasileira de Computação (SBC) (2025) “Publicações Institucionais”, disponível em: https://www.sbc.org.br/documentosinstitucionais/#publicacoes