Classificação de dados empregando rede neural perceptron multicamadas

Data classification using multilayer perceptron neural network

Hederson Gabriel Kratsch, Hugo Valadares Siqueira

RESUMO

O presente artigo explana conceitos sobre a inteligência artificial perceptron multicamadas, empregando a linguagem de programação python. Como alvo principal, é a aplicação de redes neurais em problemas de classificação, uma rede MLP foi elaborada e aplicada em um problema no mundo real. Neste caso, abordado um conjunto de dados de classificação de câncer de mama, maligno ou benigno. Os resultados obtidos foram de 90% na precisão da rede, o que é um valor considerável, visto que problemas de desproporcionalidade no dataset foram analisados. Para implementação de uma rede neural é extremamente importante um conjunto de dados confiável e que contenha características importantes como a proporcionalidade dos resultados, neste caso verdeiro ou falso.

Palavras-chave: python, rede neural, câncer de mama, classificação.

ABSTRACT

This article explains concepts about a multilayer perceptron artificial intelligence, using a python programming language. As main target, it is an application of neural networks in classification problems, an MLP network was elaborated and applied in a problem in the real world. In this case, a dataset of classification of breast cancer, malignant or benign, was approached. The results obtained were 90% in the accuracy of the network, which is a reinforced value, since the problems of disproportionality without data were imposed. For the implementation of a neural network, it is extremely important to have a reliable dataset that contains important characteristics such as the proportionality of the results, in this case true or false.

Keywords: python, neural network, breast câncer, classification.

1 INTRODUÇÃO

Como quase toda criação cientifica possui inspiração em algo existente na natureza (braços robóticos industriais, avião, invenções da biomedicina), a concepção do neurônio artificial não foi diferente. A conformação do neurônio artificial foi concebida através da observação do neurônio biológico, em especial o humano. Tal modelo foi inspirado a partir da análise da geração e propagação de impulsos elétricos pela membrana celular dos neurônios naturais. (SILVA, 2010) A ideia básica de um neurônio artificial é receber sinais existentes em suas entradas, agregá-los de acordo com uma função pré-definida e fornecer uma resposta. O modelo mais simples de neurônio que fornece as características típicas de uma rede neural biológica, ou seja, paralelismo e alta conectividade, foi desenvolvido por McCulloch & Pitts em 1943. O neurônio artificial dos dias atuais é constituído por sete elementos básicos (GARCIA, 2020). A figura 1 mostra um exemplo de neurônio artificial (BRAGA, FERREIRA, & LUDERMIR, 2007).

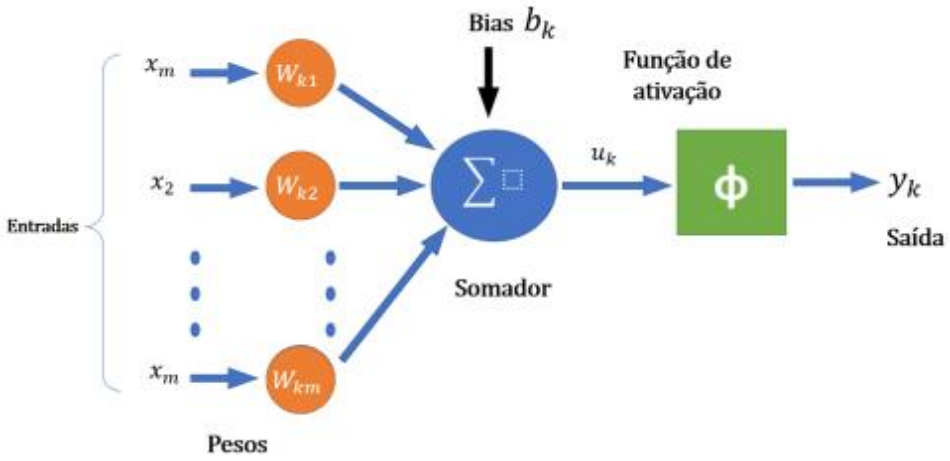

Figura 1: Exemplo de neurônio artificial
Fonte: Autoria própria.

na qual:

- **Sinais de entrada (x_1, x_2, ..., x_n):** são sinais ou informações que provem de suas entradas;
- **Pesos sinápticos (w_1, w_2 , ..., w_n):** valores de ponderação de variáveis de entrada da rede;
- **Combinador Linear (∑):** agrega todos os sinais de entrada que foram ponderados pelos respectivos pesos sinápticos;
- **Limiar de ativação (b_k):** função com valor limiar que será disparada com os dados do combinador linear com sentido a saída do neurônio;
- **Potencial de ativação (u):** resultado obtido da diferença do valor produzido entre o combinador linear e o limiar de ativação;
- **Função de ativação(φ):** será o normalizador do potencial de ativação que trará os valores obtidos neste, para valores que sejam compreendidos no intervalo desta função. além disso insere não linearidade na resposta de saída;
- **Sinal de saída (y):** valor final produzido em relação a um determinado conjunto de sinais de entrada.

2 MÉTODO (OU PROCEDIMENTOS OPERACIONAIS DA PESQUISA)

Para satisfazer as premissas estabelecidas neste estudo, uma rede neural foi construída em linguagem de programação Python. Como dataset, foi utilizado os dados de câncer de mama, realizando um link com a introdução deste trabalho (DUA & GRAFF, 2019).

É de extrema importância antes de instaurar a análise através da rede neural, conhecer os seus dados de entrada. Os dados contemplam cerca de 17 mil informações de câncer de mama. São 570 laudos com todas as características e confirmação de câncer, como raio médio, textura média, suavidade e outros. O objetivo final de rede é ler todos estes dados e decidir se estes correspondem a um câncer do tipo maligno ou trata-se de um tumor benigno. Uma rede foi elaborada com 30 neurônios de entrada, duas camadas ocultas com 16 neurônios cada e uma camada de saída do tipo booleano, que fornecerá 0 (falso) ou 1 (verdadeiro) para câncer. É

importante que o *dataset* esteja equilibrado entre as duas classes para evitar o sobretreinamento da rede (SILVA et al., 2018).

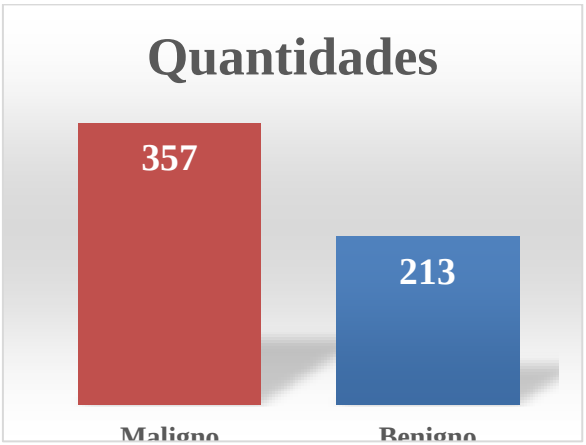

Figura 2: Gráfico de tipos de câncer do dataset.
Fonte: autoria própria.

Para implementar a rede neural foi feito uma camada de entrada com 30 neurônios, duas camadas ocultas de 16 e uma camada de saída com 1 neurônio. A explicação para a escolha das camadas de entrada é simples: tem-se 30 parâmetros de entrada. Para a camada de saída, temos um valor booleano. Para as camadas intermediárias as equações (1) e (2) foram utilizadas para definir a quantidade de neurônios:

$$Qtde\ camada\ oculta = \frac{Neuronio\ na\ camada\ de\ entrada + 1\ (bias)}{2} \quad (1)$$

$$Qtde\ camada\ oculta = \frac{30+1}{2} = 15{,}5 \quad (2)$$

Neste caso arredondamos este valor para 16 (visto que não existe meio neurônio). O bias é um valor auxiliar no bloco somador que sempre vai trabalhar com entrada constante, no caso atual, valor 1. O que muda do bias é o seu peso.

Atua como intercepto adicionado à equação linear. É um parâmetro adicional na rede neural, usado para ajustar a saída e a soma ponderada da entrada do neurônio. Em outras palavras, o desvio é uma constante e pode ajudar o modelo a se adaptar melhor aos dados fornecidos. Se não houver viés, o modelo será treinado em um ponto que só passa pela origem, o que é inconsistente com o mundo real. Além disso, com a introdução do bias, o modelo se torna mais flexível (GARCIA, 2020).

Para definir os melhores parâmetros possíveis ao rodar o código foi realizado uma espécie de um boost, um buscador implementado com dicionário que ficou sendo executado cerca de 6 horas. foram fornecidas 2 ou 3 opções de configuração ao algoritmo que, por sua vez, testou todos e mostrou o melhor. tais parâmetros estão sumarizados na tabela 1:

Item	Opções	Melhor parâmetro
Quantidade de neurônios das camadas ocultas.	16 e 20	16
Quantidade de épocas.	200 e 500	500
Função de ativação.	relu,tanh	relu
Dropout.	0.2,0.3	0.2
Função de perda.	binary cross entropy e hinge	binary crossentropy
Otimizador.	adam e sgd	adam

Tabela 1: opções de parâmetros.
Fonte: autoria própria.

3 RESULTADOS

Dos resultados podemos destacar:

- **Quantidade de neurônios das camadas ocultas:** de certa forma foi o esperado visto que foi realizado um cálculo para obter um bom resultado;
- **Quantidade de épocas:** Corresponde a quantidade de vezes em que a rede neural irá rodar;
- **Função de ativação:** Foram colocadas 2 opções. a tangente hiperbólica e a função Relu que inclusive foi escolhida neste caso. A função Relu é do tipo $y = x$ para valores positivos, como mostra a figura 3;

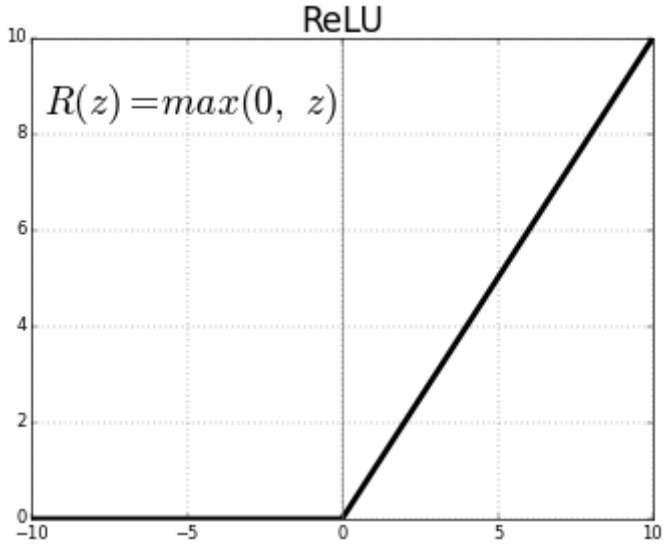

Figura 3: Função relu.
Fonte: autoria própria.

- **Drop-out:** É um algoritmo de treinamento de rede neural, baseado na eliminação aleatória de neurônios no processo de aprendizagem para evitar o *overffiting* dos dados;
- **Função de perda:** A função de perda tem o objetivo de após a fase *feedforward*, a rede atualiza os pesos dos neurônios e do bias. No caso em questão foi utilizado o *binary crossentropy*, entropia cruzada;
- **Otimizador:** Foi escolhido computacionalmente o otimizador Adam, que trabalha com estimativa de gradiente descendente.

Para analisar o comportamento da rede neural, foram plotados dois tipos de gráficos, A Precisão da rede e os parâmetros da *lossfuction*. Estes gráficos foram comparados na aprendizagem e validação e são mostrados na figura 8.

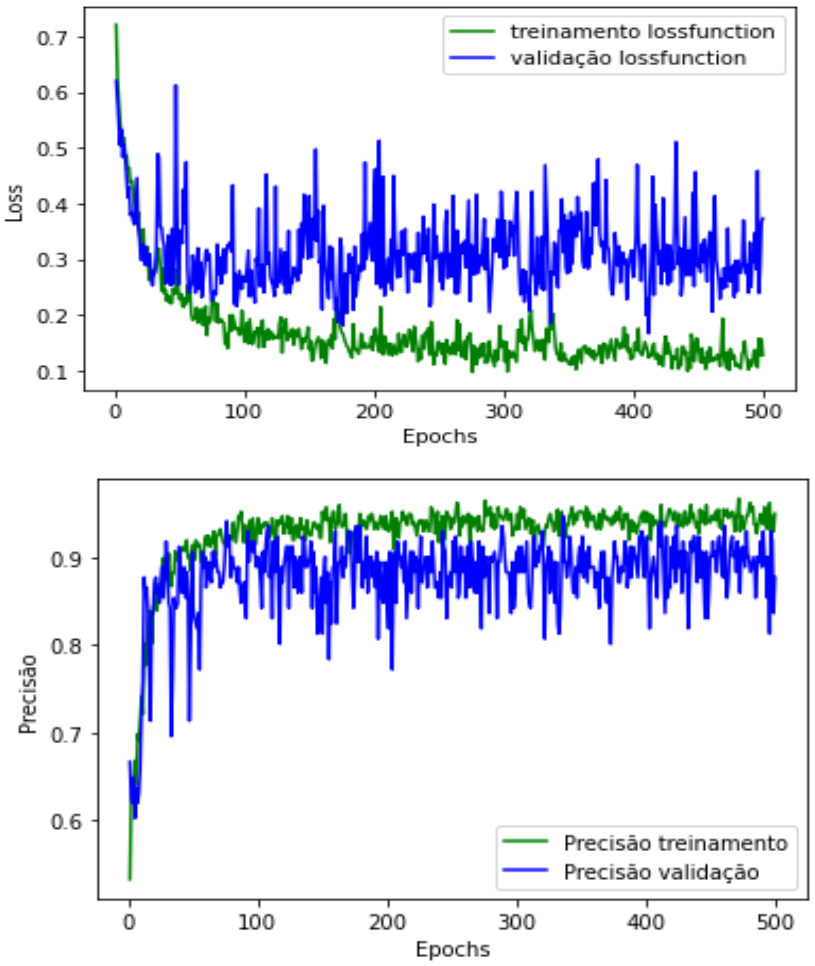

Figura 8: erro e precisão da rede durante o treinamento.
Fonte: autoria própria.

É notório que a precisão variou, mas no final a precisão da validação e treinamento se mantiveram na casa dos 80%, o que é um valor considerável, visto a desproporcionalidade do dataset. A *lossfunction* na validação também variou de forma significativa e no final da validação manteve um valor consideravelmente bom para a rede.

4 CONCLUSÃO

Nesse estudo fez-se a implementação computacional de uma rede neural do tipo perceptron multicamadas para classificação de dados de câncer de mama. foram mostrados os diversos passos para a elaboração da mesma, desde a concepção do neurônio artificial até a seleção paramétrica.

Os resultados computacionais mostram a viabilidade da proposta para problemas desse tipo. Constatou-se que analisar os dados e a confiabilidade dos mesmos influenciam nos resultados. Por isso a importância da paridade. O desbalanço de dados provoca um treinamento não conforme a rede e consequentemente instável. Como trabalhos futuros, pretende-se elaborar outras redes, além da aplicação do método em outras bases de dados.

REFERÊNCIAS

BRAGA, A., FERREIRA, A., & LUDERMIR, T., **Redes neurais artificiais: teoria e aplicações**. Rio de Janeiro, Brazil, LTC Editora, 2007.

SIQUEIRA, H. V. **Previsão de series de vazões com redes neurais artificiais e modelos lineares ajustados por algoritmos bio-inspirados**, 2009.

DA SILVA, N., **Redes neurais artificiais para engenharia e ciências aplicadas: curso prático**, vol. 1. Artliber Editora, 2010.

SILVA, N., SIQUEIRA, I., OKIDA, S., STEVAN, S. L., & SIQUEIRA, H., **Neural networks for predicting prices of sugarcane derivatives**. SUGAR TECH, 21(3), 514-523, 2018.

DUA, D. AND GRAFF, C. **UCI Machine Learning Repository** [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2019.