Perceptron: implementação e testes.

Eduardo Schvinn
Joinville, Brasil
Email: eschvinn@gmail.com

I. Introdução

Este projeto tem como objetivo implementar uma rede Perceptron de camada única para resolver problemas de classificação. Os dados foram gerados utilizando uma distribuição normal, e duas bases de dados distintas foram criadas para validar a eficácia do modelo. As funções desenvolvidas para o treinamento e aplicação da rede Perceptron foram testadas e validadas por meio da matriz de confusão.

II. Implementação

Inicialmente, foram criadas duas funções para gerar dois grupos linearmente separáveis e não linearmente separáveis. Em seguida, foram atribuídos rótulos para identificar a qual grupo cada ponto pertence.

Posteriormente, foram desenvolvidas mais três funções que compõem o perceptron. A primeira função ajusta os pesos com base na regra de atualização do perceptron:

$$w(n+1) = w(n) + \eta \cdot (d(n) - y(n)) \cdot x(n) \qquad (1)$$

onde $w(n)$ são os pesos na iteração n, η é a taxa de aprendizado, $d(n)$ é o valor desejado para a resposta do neuronio $y(n)$ é a saída do neurônio, e $x(n)$ é a entrada na iteração n.

Em seguida, foi implementada uma função de ativação que verifica se o retorno dos valores de entrada multiplicados pelos pesos excede um parâmetro definido como *bias* .

Posteriormente, foi desenvolvida a função que inicializa o perceptron. Esta função realiza a multiplicação dos valores de entrada pelos pesos e, em seguida, classifica os dados de entrada, adicionando-os a um vetor de acordo com sua classificação.

Além disso, foram criadas funções específicas para geração de dados, exibição dos resultados e validação das previsões feitas pelo Perceptron. Utilizou-se uma taxa de aprendizado fixa de 0,01 e um bias igual a 1 para ambos os modelos.

III. Resultados: Dados linearmente separáveis

Para entradas linearmente separáveis, similares à base de dados utilizada no treino do perceptron com 500 épocas, o modelo apresentou uma acurácia de 100% na classificação desses dados (Fig. 2). Vale ressaltar que os parâmetros para gerar tais dados foram os mesmos utilizados para o treinamento, porém, foi utilizada uma semente (*seed*) diferente para garantir que os dados não fossem completamente iguais (Fig. 1).

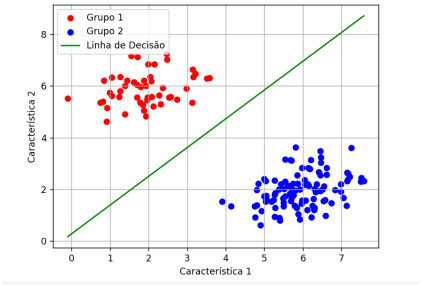

Fig. 1: Dados similares aos da base utilizada para o treino.

Para validar os resultados e calcular a acurácia, foi gerada uma matriz de confusão, conforme mostrado na (Fig. 2).

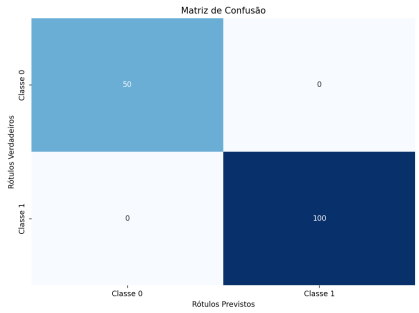

Fig. 2: Matriz de confusão.

Entretanto, a acurácia do modelo diminuiu significativamente quando foi utilizada uma base de dados diferente daquela usada para testar o modelo, caindo de 100% para 66,66% (Fig. 3). Para calcular a acurácia e validar o modelo, foi gerada uma matriz de confusão dessa classificação (Fig. 4).

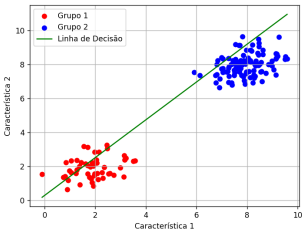

Fig. 3: Dados não similares aos da base utilizada para treino

Dessa forma, conseguimos demonstrar que o modelo é eficaz para a classificação de dados que seguem o padrão dos utilizados no treinamento do perceptron.

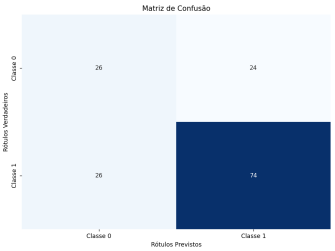

Fig. 4: Matriz de confusão dos dados não similares

IV. Resultados: Dados não linearmente separáveis

Para dados que não são linearmente separáveis foi gerada uma nova base e feito um novo treinamento do perceptron que aprensentou as seguintes caracteristicas(Fig 1):

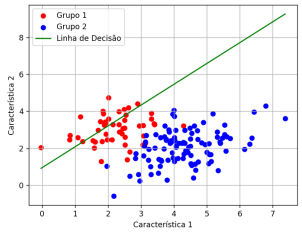

Fig. 5: Resultado do treinamento da base não linear.

Para o treinamento foram utilizadas 500 épocas e foi gerada outra base não linearmente separável semelhante a utilizada no treinamento porém com uma semente (*seed*) diferente,e então foi aplicado o modelo para fazer a previsão. Depois disso, foi gerada uma matriz de confusão para validar os resultados das previsões.

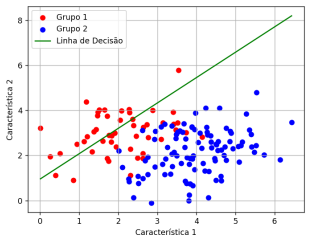

Fig. 6: Previsão dos dados não lineares.

Com os dados da matriz de confusão foi possível calcular a acurrácia do modelo que chegou a 88.66% comparando as previsões do treino com o teste.

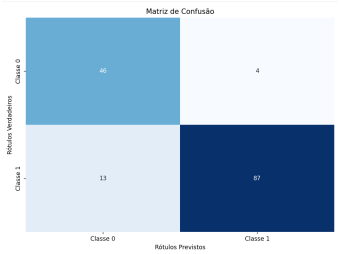

Fig. 7: Matriz de confusão de dados nao linearmente separáveis.

V. Considerações finais

O perceptron demonstrou alta eficácia quando aplicado a conjuntos de dados de teste semelhantes aos usados no treinamento. No entanto, seu desempenho foi significativamente reduzido em conjuntos que divergem um pouco dos dados de treinamento, o que era esperado devido à adaptação do modelo aos dados específicos do treino. Em cenários onde os dados não são linearmente separáveis, o perceptron ainda apresenta um desempenho satisfatório, desde que os dados possuam regiões onde não há completa sobreposição das classes.

References

[1] Simon Haykin. *Neural Networks*. Prentice Hall, 1999.
[2] Antônio P. Braga, André P. L. F. de Carvalho, Teresa B. Ludermir. *Redes Neurais Artificiais: Teoria e Aplicações*. LTC, 2000.