# Trabalho 2 - Computação Inspirada Pela Natureza -

Lucca Vieira Batistão

Maio/2021

## Introdução

O propósito deste trabalho é a implementação e a experimentação com métodos de classificação baseados no sistema nervoso humano. Foram modelados dois neurônios artificiais para a classificação de dois conjuntos de dados, o *Iris Data Set* e o *Wine Data Set*, que serão descritos posteriormente. O método utilizado foi baseado no modelo do *Perceptron*, um modelo simples de aprendizado supervisionado que tenta simular matematicamente o funcionamento de um neurônio.

A implementação foi realizada na linguagem de programação *Python* 3.7.5. Foram utilizando as bibliotecas *random* para a geração de números aleatórios para a inicialização dos pesos, *pandas* para a manipualação do conjunto de dados, *copy* para manipulação do valor de vetores e matrizes, *sklearn* para a construção das matrizes de confusão e *matplotlib* para a criação dos gráficos e imagens.

Foram implementados dois algoritmos distintos, uma para cada exercício, porém os dois possuem as mesmas funções somente com algumas alterações para se adequar ao problema. As funções são:

- *AbrirDados(fileName)*: abre o arquivo que contém o conjunto de dados e normaliza as colunas das variáveis.
- *IniciaPesos()*: Cria a matriz de pesos (w) e inicia com valores aleatórios no intervalo entre -0.01 e 0.01.
- *SepararDados(df)*: Separa os conjuntos de dados em subconjuntos de treinamento (70%), validação (15%) e teste (15%).
- *SaidaDesejada(classe)*: É uma função que converte o rótulo da classe presente no conjunto de dados e converte em uma lista com os valores previstos como saída do algoritmo para cada classe.

- *Treino(maxt, taxaApren, train, validate)*: Função responsável pelo processo de treinamento do modelo. O treinamento ocorre até o número máximo de iterações (épocas), enquanto o erro acumulado for maior que 0 e se acurácia da validação não melhora a mais de 100 iterações. O valor da melhor matriz de pesos (com o melhor erro) é armazenado e é a saída dessa função.
- *Teste(w, test)*: Realiza o teste do modelo após o treinamento. Retorna a acurácia do modelo durante o treino.

# 1 º Exercício

No primeiro exercício, foi pedido que fosse usado o *Perceptron* para a classificação do conjunto de dados Iris. Esse conjunto de dados contém informações sobre três espécies da flor Iris: I. setosa, II. virginica e III. versicolor. O conjunto possuí 150 entradas, sendo 50 entradas para cada espécie. As colunas do conjunto representam as variáveis que são: comprimento da pétala, comprimento da sépala, largura da pétala e largura da sépala em cm.

A Figura 1 mostra o gráfico que ilustra a mudança do Erro acumulado ao longo das épocas com o perceptron usando uma taxa de aprendizado de 0.5 e número máximo iterações igual a 1000. O erro começa com mais de 80 e depois de 140 épocas ele fica em um valor menor que 40. Como pode ser observado, o erro não chegou a zero e após certo ponto (entre as épocas 40 e 60), o erro converge e fica alternando em valores perto de 40. Isso provavelmente se deve ao fato de que o conjunto de dados não é linearmente separável, já que se os dados fossem linearmente separáveis, o erro teria convertido para zero.

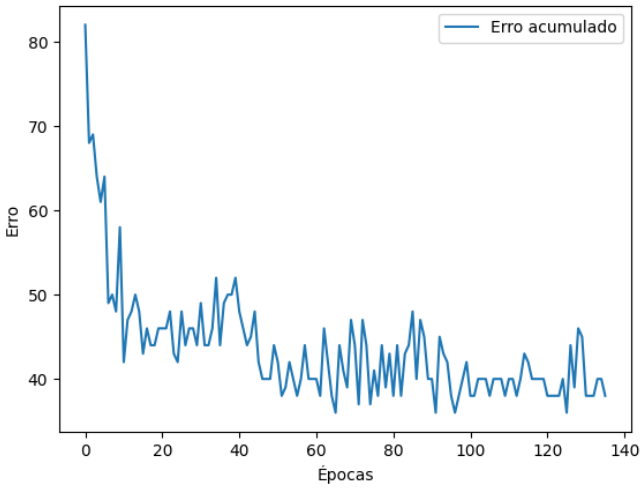
Nas Figuras 2, 3, 4 pode-se observar as matrizes de confusão do modelo, executando com as mesmas configurações descritas para a Figura 1. Na matriz de treinamento e na matriz de validação, é observado uma boa taxa de acerto nas três classes. Já na matriz de teste, vê-se que nenhuma amostra da espécie versicolor teve acertos.

Também foi feito um experimento para analisar o impacto da mudança da taxa de aprendizado. O algoritmo foi executado 100 vezes com cada configuração e a média da acurácia em cada uma delas está sendo apresentada na Tabela 1.

# 2 º Exercício

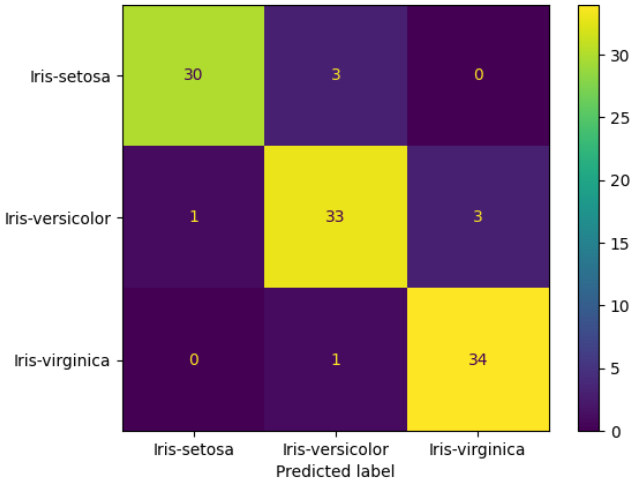
O segundo exercício pedia uma tarefa semelhante ao primeiro, porém em um outro conjunto de dados. Esse conjunto é o Wine, que contém informações sobre três tipos de

Figura 1 – Gráfico da mudança do Erro Acumulado a cada época.




Fonte: Elaborado pelo autor.

Figura 2 – Matriz de confusão do Treinamento.



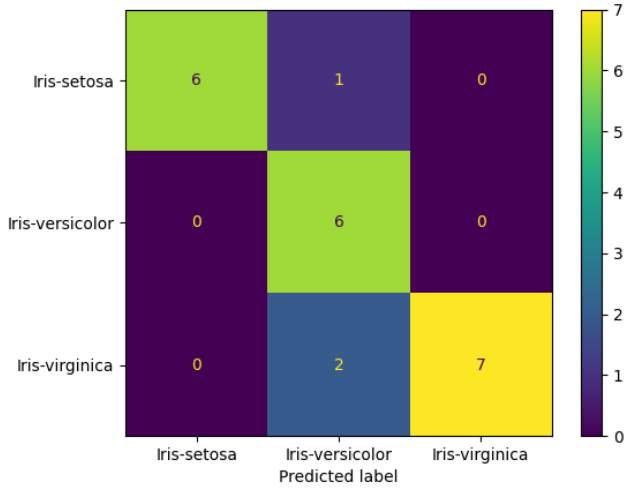

Fonte: Elaborado pelo autor.

Tabela 1 – Variação da média da acurácia no treinamento, validação e teste com diferentes taxas de aprendizado.

| | 0.01 | 0.1 | 0.5 |
|---|---|---|---|
| Treinamento | 75.2476% | 75.2523% | 75.2507% |
| Validação | 81.7272% | 81.7727% | 81.7879% |
| Teste | 65.1739% | 65.5217% | 65.7101% |

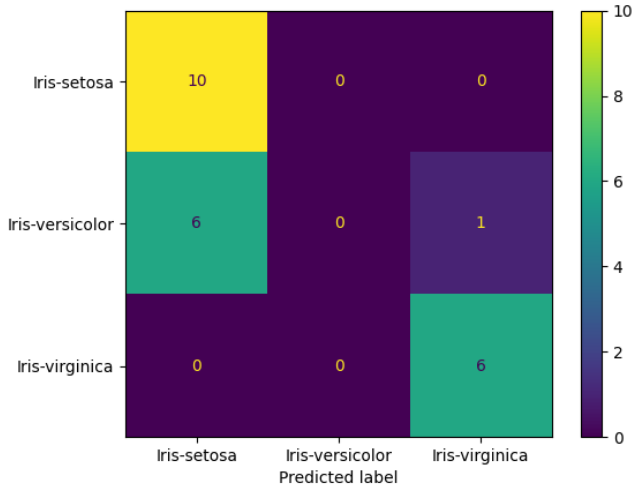
Fonte: Elaborada pelo autor.

Figura 3 – Matriz de confusão da Validação.




Fonte: Elaborado pelo autor.

Figura 4 – Matriz de confusão do Teste.

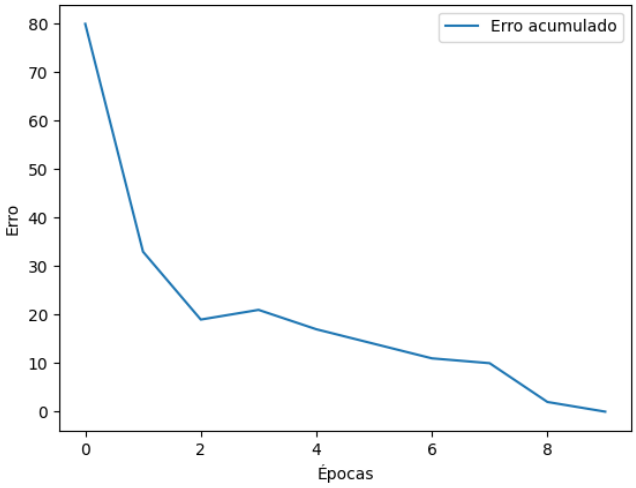



Fonte: Elaborado pelo autor.

vinhos. Esse conjunto contém uma maior quantidade de variáveis, sendo elas: Taxa de Álcool, de Ácido Málico, de Cinzas, de Alcalinidade das Cinzas, de Magnésio, de Fenóis Totais, de Flavonóides, de Fenóis não flavonóides, de Proantocianidinas, de Intesidade de Cor, de Matiz, de OD280/OD315, de Prolina. Também possuí um maior número de entradas que o conjunto de dados Iris, possuindo o total de 178, sendo 59 da primeira classe, 71 da segunda e 48 da terceira.

Na Figura 5, pode-se observar a redução do Erro Acumulado ao longo das épocas da execução do modelo no conjunto de dados Wine. Nesse caso, os dados são linearmente separáveis pois há a redução do erro para 0 em menos de 10 épocas. A taxa de aprendizado usada para esse experimento foi 0.5 e o máximo de interações era 1000.

Figura 5 – Gráfico da mudança do Erro Acumulado a cada época.




Fonte: Elaborado pelo autor.

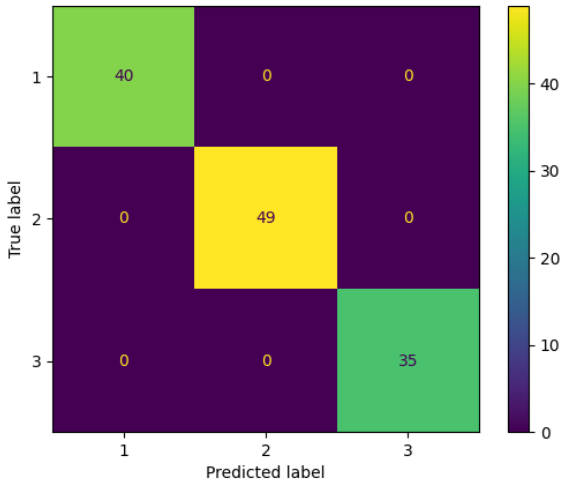
Nas Figuras 6, 7 e 8 são mostradas as matrizes de confusão do treinamento, validação e teste. Através delas, somente ocorreu um erro na validação, mostrando a eficiência do modelo quando o conjunto é linearmente separável.

Na Tabela 2, é ilustrado os resultados da execução do modelo 100 vezes com diferentes taxas de aprendizado. Assim como no primeiro exercício, a mudança na taxa de aprendizado não pareceu impactar a acurácia d método em nenhuma capacidade.

# 3 º Exercício

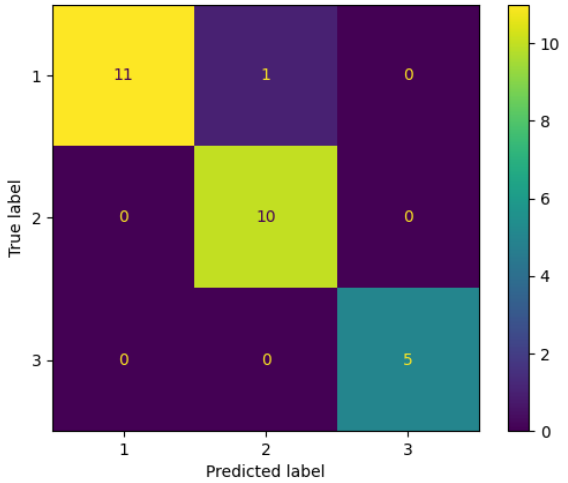
Para a última atividade, foram utilizados o *Keras* e *Tensorflow* para realizar experimentos com o conjunto de dados *Pima Indians onset of diabetes*. O conjunto contém dados em formato

Figura 6 – Matriz de confusão do Treinamento.




Fonte: Elaborado pelo autor.

Figura 7 – Matriz de confusão da Validação.



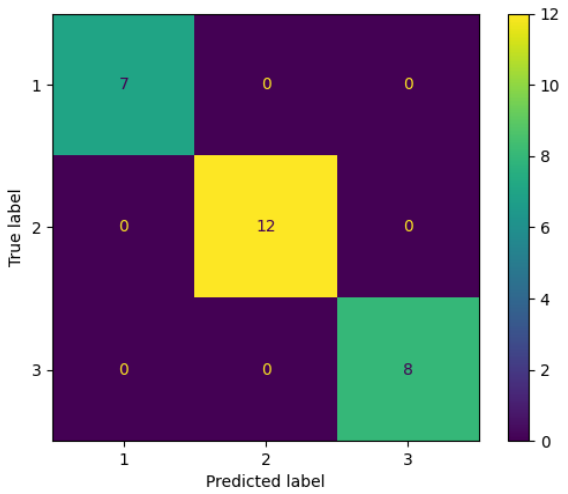

Fonte: Elaborado pelo autor.

Tabela 2 – Variação da média da acurácia no treinamento, validação e teste com diferentes taxas de aprendizado.

| | 0.01 | 0.1 | 0.5 |
|---|---|---|---|
| Treinamento | 100% | 100% | 100% |
| Validação | 88.1851% | 88.4321% | 88.5555% |
| Teste | 89.3703% | 89.6667% | 89.3333% |

Fonte: Elaborada pelo autor.

Figura 8 – Matriz de confusão do Teste.




Fonte: Elaborado pelo autor.

CSV sobre registros médicos de pacientes e se eles tiveram um início de diabetes ou não nos últimos cinco anos. Portanto se trata de um problema de classificação de binária. O conjunto de dados se encontra no link: Pima Indians onset of diabetes.

As variáveis do conjunto são: número de gravidezes, concentração de glucose no plasma, pressão sanguínea diastólica, espessura da dobra da pele do tríceps, insulina sérica de 2 horas, Índice de Massa Corporal (IMC), função de pedigree de diabetes e idade em anos.

A rede criada possuí quatro camadas: uma de entrada com oito neurônios, duas escondidas com doze e oito neurônios e uma de saída. A função de ativação das camadas invisíveis é a ReLU (*Rectified Linear Unit*) e a da camada de saída é a sigmoide.

Para o treinamento do modelo, foram usadas 150 épocas e um lote de 10 amostras antes de atualizar os pesos. O conjunto de dados não foi dividido e todo ele foi usado para treinamento. Ao executar o treino 100 vezes a acurácia média de 87.40%.