CENTRO UNIVERSITÁRIO FEI

VICTOR BIAZON

RA: 119.115-4

RELATÓRIO III – TÓPICOS ESPECIAIS DE APRENDIZAGEM
PERCEPTRON

SÃO BERNARDO DO CAMPO

2019

Sumário:

1. Objetivo

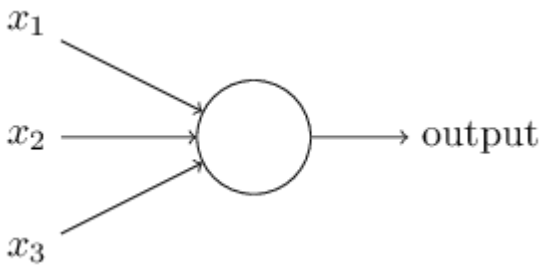
Implementar o classificador linear perceptron e testá-lo para portas logicas e o dataset Iris de Fisher.

2. Teoria

Perceptron

Segundo Goodfellow(2016) o perceptron foi proposto nas décadas de 1950 e 1960 por Frank Rosenblatt baseado nos trabalhos de Warren McCulloch e Walter Pitts. O perceptron permite uma objetiva demonstração de como funciona uma rede neural artificial, sendo este a unidade mais básica destas. Matematicamente o perceptron permite uma divisão linear de dados não podendo ser utilizado para classificação de dados não linearmente divisíveis.

O perceptron é um modelo matemático que após "aprender" com os dados de treinamento consegue através dos pesos aprendidos traduzir entradas em saídas classificando os dados.



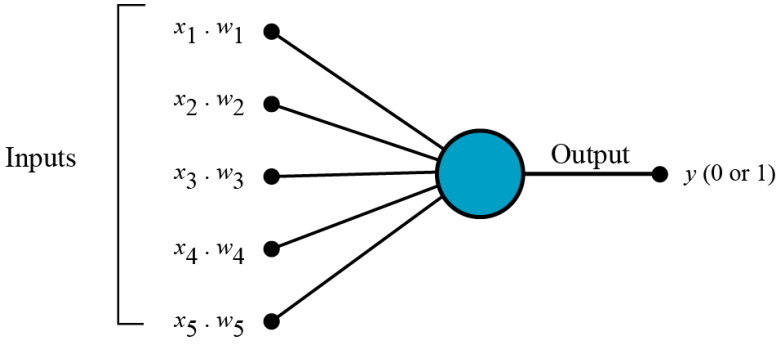
Na imagem acima é ilustrado um perceptron com três entradas e uma saída. Para realiza a classificação o perceptron admite uma função de transferência como por exemplo a formula abaixo. Também é necessário considerar um Bias, ou um "viés", que é utilizado para corrigir a posição da divisão dos dados.

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

E a formula com o Bias fica desta forma:

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

Sendo assim o modelo do perceptron com os pesos ponderados fica:



O treinamento consiste em utilizar a descida de gradiente para que o perceptron aprenda os pesos e o bias. Desta forma com o modelo treinado pode-se apresentar novos dados de entrada e receber novas saídas de acordo com o treinamento. Em resumo, um perceptron é uma rede neural de um único nível e uma rede neural é um perceptron de várias camadas.

3. Implementação

A implementação partiu do seguinte fluxo:

Para implementação foram criadas as seguintes funções:

```
def DeltaW(target, output, Xi): #calcula o Delta W sendo esta a correção aplicando os pesos
    return Ni * (target - output) * Xi


def Fx(W, X): #calcula o output, aplicando as entradas aos pesos e passando pelo função de transferencia
    soma = 0;

    for i in range(0, len(X)):
        soma += W[i + 1] * X[i]

    return 1 if soma + W[0] >= Threshold else -1


def train(x, saida): #recalcula os pesos baseados nas saidas resultantes dos pesos e relação as saidas conhecidas
    output = Fx(Ws, x);

    for i in range(0, dimensoes):
        Ws[i+1] += DeltaW(saida, output, x[i])
    Ws[0] = Ni * (saidas[i] - output)
    return

def perceptron(): #inicializa os pesos com valores bem pequenos
    for i in range(1,dimensoes + 1):
        Ws[i] = 1* Wini

    return
```

```
        def transform(entrada): #realiza o calculo da saida a partir de uma entrada
            return Fx(Ws, entrada)

        def showDivision(entradas): #plota os pontos de acordo com a classificação do perceptron
            divisionPlane = np.mgrid[4:7.5:0.07, 1:5:0.07].reshape(2,-1).T
            plt.figure()
            for i in range(0,len(divisionPlane)):
                output = transform(divisionPlane[i])
                c = 'deepskyblue' if output == 1 else 'lightcoral'
                plt.scatter(divisionPlane[i,0], divisionPlane[i,1], c = c, cmap = 'rainbow', s = 10)
            for i in range(0, linhasEntrada):
                c = 'blue' if saidas[i] == 1 else 'red'
                plt.scatter(entradas[i,0], entradas[i,1], c = c, cmap = 'rainbow', s = 20)

            plt.title('Perceptron - Data')
            plt.xlabel('X1')
            plt.ylabel('X2')
            plt.show()

            return

        #main
        #le dados do dataset
        data = pd.read_table('Iris.txt', decimal  = ",")
```

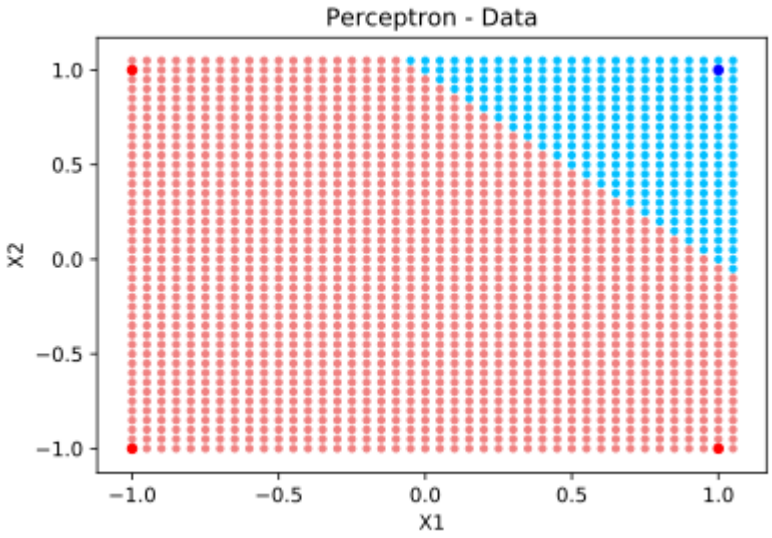
```
        x = np.asarray(data.iloc[:,:-3]) #separa dados em variaveis
independentes e a saida
        y = Encoder(np.asarray(data.iloc[:,-1]))
        x_l = []
        y_l = []
        for i in range(0,len(x)):
            if y[i] == 1 or y[i] == 2:
                x_l.append(x[i,:])
                y_l.append(y[i])

        for i in range(0, len(y_l)):
            if y_l[i] == 2:
                y_l[i]= -1

        x = np.asarray(x_l)
        y = np.asarray(y_l)
        del x_l
        del y_l

        entradas = np.copy(x)
        linhasEntrada = len(entradas)
        colunasEntrada = len(entradas[0])
        saidas = np.copy(y)

        #parametros do perceptron
        dimensoes = colunasEntrada
        Ni = 0.1
        Wini = 0.01
        Ws = np.ones((dimensoes + 1,1), float)
        Ws[0] = 1
        Threshold = 0

        perceptron();
```

```
#realiza treinamento do perceptron
for i in range(0,1000):
    for j in range(0, linhasEntrada):
        entrada = entradas[j]
        train(entrada, saidas[j])
```

4. Resultados

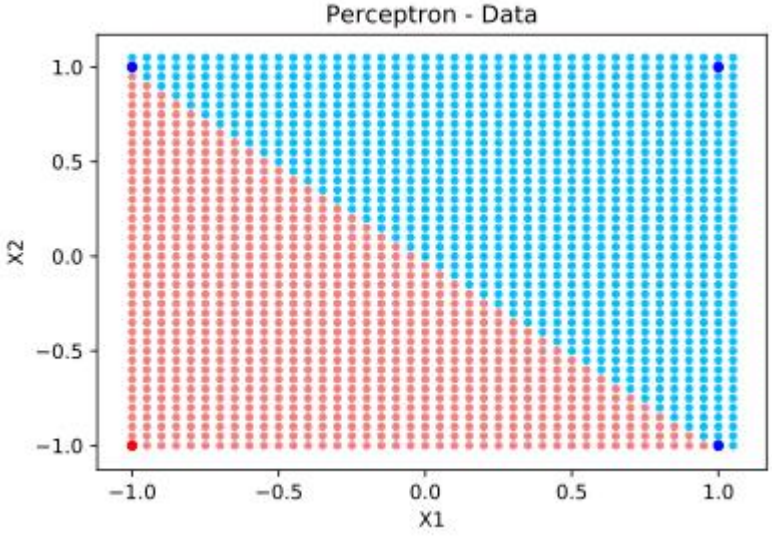
Para se testar o perceptron foi utilizado o treinamento nas funções lógicas OR, AND e XOR e no dataset IRIS de Fisher.

Para a porta OR de duas entradas o perceptron criou o seguinte classificador:

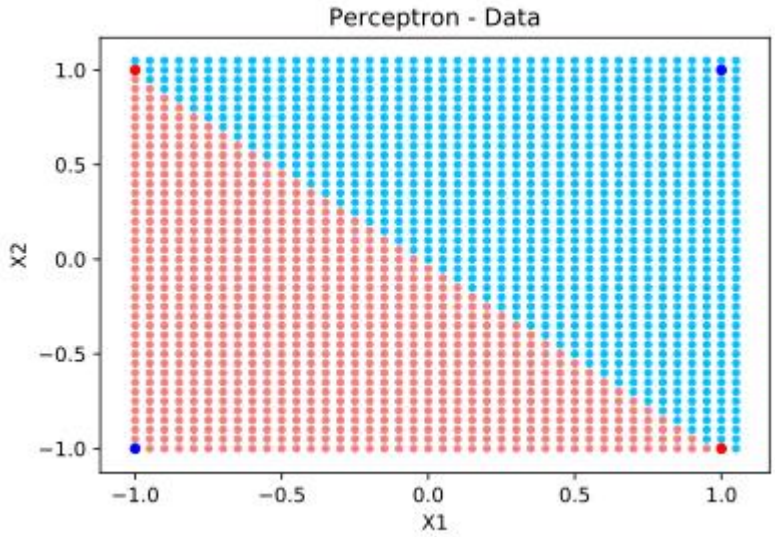


Os pontos azuis representam saídas 1, e os vermelhos -1.

Já para a porta AND de duas entradas o perceptron criou o seguinte classificador:



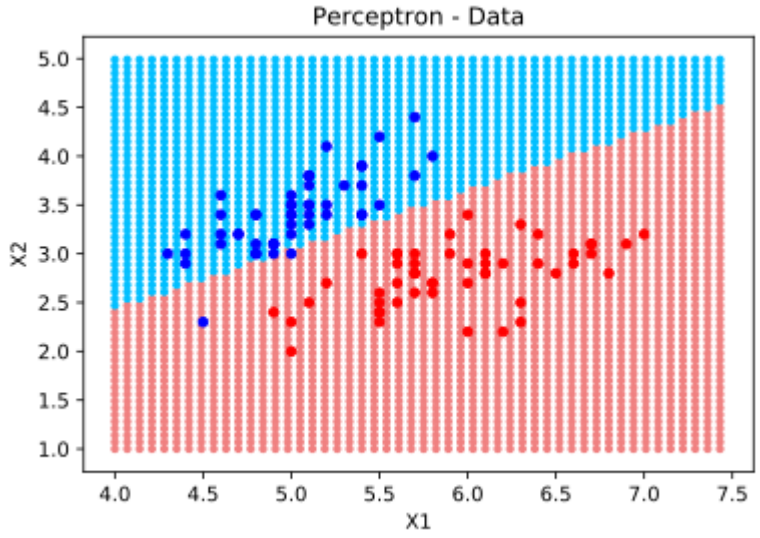
Para o XOR, não foi possível realizar a separação linear dos dados, sendo retornado o seguinte classificador:



O classificador não consegue aprender a divisão dos dados devido a estes não serem linearmente separáveis.

Para o dataset de Iris de Fisher foram realizados dois treinamentos, primeiramente com as primeiras duas dimensões e depois com todas as quatro.

O resultado para as primeiras duas foi:



Como pode-se notar houve um ponto que caiu na região de classificação errada, este se deve ao fato do mesmo se comportar como um outlier neste dataset para duas dimensões.

A matriz de confusão dos dados classificou 2 erros, sendo 50 pontos classificados corretamente como Iris Versicolor e 48 como Iris Setosa. 2 pontos foram erroneamente classificados como versicolor conforme tabela abaixo:

50	0
2	48

Para quatro dimensões devido a impossibilidade de mostrar as quatro dimensões ao mesmo tempo, para realizar a análise foi calculada a matriz de confusão entre os dados de saída conhecidos e os previstos pelo classificador resultando nos seguintes dados:

50	0
0	50

Não sendo classificado nenhum dado de forma errônea;

.

5. Conclusão

Com estes experimentos pôde-se comprovar a eficácia da classificação linear do perceptron ao ser treinado baseado nos dados de entrada com a saída esperada. Por se tratar de treinamento supervisionado a saída esperada apresentada a ele tem grande influência em como o aprendizado é realizado, sendo que o perceptron em sua implementação básica tem grande sensibilidade a outliers. Pode-se notar com o dataset do Iris que o número de dimensões a serem analisadas tem grande influência na forma como o classificador aprende a distribuição dos dados. Sendo que ao se retirar dimensões no primeiro estudo feito, o classificador teve erros e ao considerar todas as dimensões estes não ocorreram.

Referências bibliográficas

[1] Goodfellow, Ian et al , 2016, Deep Learning, MIT Press, disponível em: < http://www.deeplearningbook.org > Acesso em: 23/11/2019 23:08