EXERCÍCIO 01

Para realizar o primeiro exercício eu criei uma classe chamada Perceptron. Dentro desta classe há um método "**input()**" e um método "**ativa()**" responsáveis respectivamente por inserir as entradas e calcular a saída do perceptron criado através da chamada **Perceptron(n_in, n_out)**.

A classe ficou assim:

```python
import numpy as np
import matplotlib.pyplot as plt

class Perceptron:
    def __init__(self, entradas=3, saida=1, bias=0.5):
        self.entradas = np.zeros(entradas)
        self.saida = np.zeros(saida)

        self.pesos = np.random.uniform(low = -1, high = +1, size = entradas)
        self.bias = np.random.uniform(low = -1, high = +1, size = 1)

        self.saida_registro = []  # Lista para armazenar os valores de saída
        self.bias_registro = []   # Lista para armazenar os valores de bias
        self.pesos_registro = []  # Lista para armazenar os valores de pesos
        self.entrada_registro = []# Lista para armazenar os valores de entradas

    def input(self, array_in):
        for i, x_i in enumerate(array_in):
            self.entradas[i] = x_i

    def ativa(self):
        self.saida = np.dot(self.entradas, self.pesos) + self.bias

        self.entrada_registro.append(self.entradas)
        self.saida_registro.append(self.saida)  # Registra o valor de saída
        self.bias_registro.append(self.bias)    # Registra o valor do bias
        self.pesos_registro.append(self.pesos.copy())  # Registra os valores de pesos (cópia)

        return self.saida, self.bias, self.pesos

    def __str__(self):
        return f'Entradas: {self.entradas}, Pesos: {self.pesos}, Saída: {self.saida}, Bias: {self.bias}'

def re_lu(valor):
    return valor > 0 or 0
# -
```

Figura 01: Classe de perceptron criada em python

Nesta classe o método ***__init__()*** é responsável por iniciar todos os valores padrões para um dado perceptron. Eu crio um vetor de entradas e um vetor de saída preenchido com zeros e também um vetor de pesos e o bias associado ao perceptron (ambos são criados sorteando valores entre -1 e 1 para suas entradas). Dentro do método também crio listas que serão utilizadas para manter registro de todos os valores de entradas, saídas, pesos e bias para o perceptron em questão.

O método ***input(array_in)*** passa uma lista como entrada e realiza a associação dos elementos da lista com as entradas do perceptron. Já o método ***ativa()*** realiza o produto interno entre o vetor de entradas e o vetor de pesos, e adiciona o bias, bem como acrescenta às listas de registro os valores obtidos na execução. É importante notar que estas listas serão úteis quando estivermos na fase de treino, de modo que será possível manter registro objetivo de como os valores de pesos e bias estão mudando ao longo do treino.

O método ***__str__()*** retorna informações úteis quando o perceptron é printado.

Para executar este código é necessário chamar a classe da seguinte maneira:

```
# -

num_entradas = 3
num_saidas = 1
perceptron = Perceptron(num_entradas, num_saidas)

entrada = np.random.uniform(low=-1, high=1, size=num_entradas)
perceptron.input(entrada)
saida, bias, pesos = perceptron.ativa()
saida_relu = re_lu(saida)

print(perceptron)
```

Figura 2: Chamada do código de um perceptron com 3 entradas e uma saída

Ao executar este código no terminal o retorno foi o seguinte:

```
PS C:\Users\User\Desktop\IA - Disciplina\EXERCÍCIOS> python .\Exercício1.py

Entradas: [ 0.3463432   0.8516938  -0.29372952]
Pesos: [-0.90024289 -0.43165309 -0.54692773]
Saída: [-0.31684213]
Bias: [0.20193831]
```

Figura 3: Saídas no prompt de comando ao executar o código.

No trecho acima é possível perceber que a operação é realizada de maneira adequada. O produto interno entre o **vetor de entradas** e o **vetor de pesos** e ao resultado é somado um **bias**.

PARA CASA

Criado este código base para um perceptron é possível alterar arbitrariamente o número de entradas, ou saídas, o trecho abaixo cria um perceptron com 10 entradas e executa 10 vezes o input neste perceptron utilizando entradas aleatórias entre -1 e 1.

```
38  # -
39
40  num_entradas = 10
41  num_saidas = 1
42  perceptron = Perceptron(num_entradas, num_saidas)
43
44  x_data = []
45  y_data = []
46
47  # Iterando sobre várias entradas
48  for i in range(10):
49      entrada = np.random.uniform(low=-1, high=1, size=num_entradas)
50      perceptron.input(entrada)               # Insere o vetor de entrada
51      saida, bias, pesos = perceptron.ativa() # Ativa o perceptron e retorna seus parâmetros
52
53      # Adicionando os dados para o plot
54      x_data.append(i+1)
55      y_data.append(perceptron.entradas[1])
56
57  plt.plot(x_data, y_data)
58  plt.xlabel('Execução #')  # Rotulo do eixo x
59  plt.ylabel('Saída')  # Rotulo do eixo y
60  plt.title('10 execuções da ativação do perceptron')  # Título do gráfico
61  plt.show()
```

Figura 4: Perceptron com 10 entradas ativado 10 vezes com entradas aleatórias.

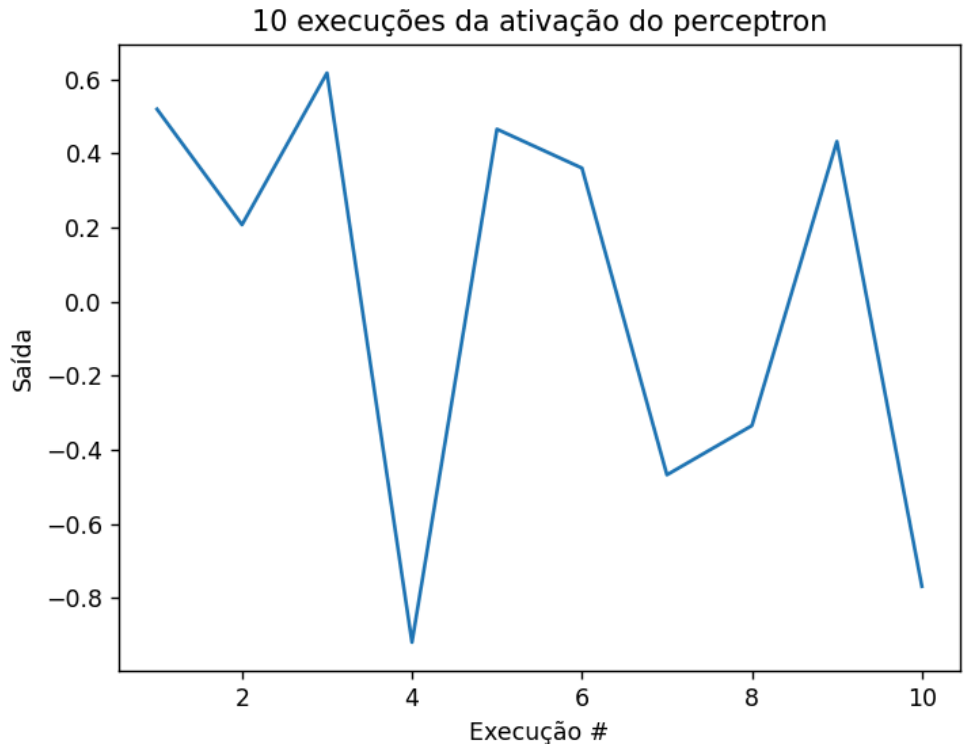

Figura 5: Registro de saída para cada execução

COMPLEMENTAR

Para visualizar as saídas do perceptrons criado é possível executar um código complementar:

```
# -

num_entradas = 2
num_saidas = 1
perceptron = Perceptron(num_entradas, num_saidas)

# Lista para armazenar os dados para o plot 3D
x_data = []
y_data = []
z_data = []

# Iterando sobre várias entradas
for i in range(100):
    entrada = np.random.uniform(low=-1, high=1, size=num_entradas)
    perceptron.input(entrada)               # Insere o vetor de entrada
    saida, bias, pesos = perceptron.ativa() # Ativa o perceptron e retorna seus parâmetros
    saida_relu = re_lu(saida)

    # Adicionando os dados para o plot
    x_data.append(perceptron.entradas[0])
    y_data.append(perceptron.entradas[1])
    z_data.append(perceptron.saida)

# Plot 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x_data, y_data, z_data, c='r', marker='o')
ax.set_xlabel('Entrada 1')
ax.set_ylabel('Entrada 2')
ax.set_zlabel('Saída')
ax.set_title('Saída do Perceptron em 3D')
plt.show()
```

Figura : Código para plotar saídas de 100 execuções em um perceptron com bias e pesos fixados

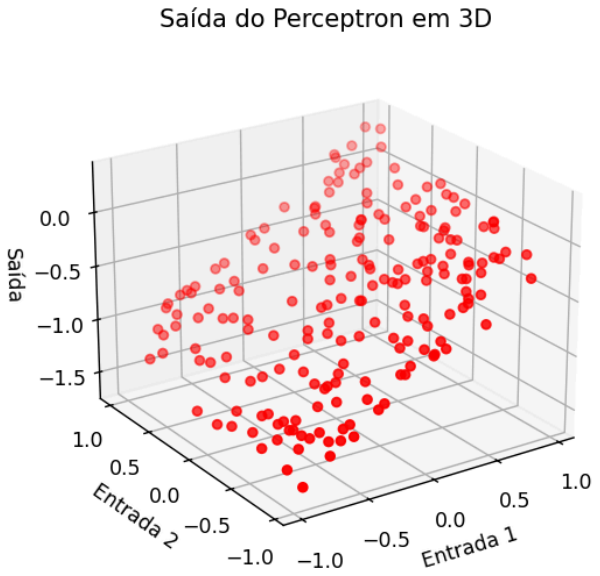

Figura : Plot obtido ao executar 200 vezes o código. O eixo vertical é a saída para um perceptron de duas entradas