SCC5809 - Redes Neurais e Aprendizado Profundo
Adaline
Exercício 1

Gustavo
Dhiogo
Herlisson

August 28, 2024

Abstract

Implementar e treinar o modelo Adaline para reconhecer os símbolos Y e Y invertido (letra "Y" e letra "Y" invertida). O presente projeto foi feito em Python utilizando as bibliotecas Numpy e Random.

1 Introdução

O presente projeto foi feito em Python utilizando as bibliotecas Numpy e Random. Portanto, antes de mais nada precisamos importar as bibliotecas:

```
import numpy as np
import random
```

Numpy é utilizado para manipular as matrizes e arrays. A biblioteca Random é utilizada para inicializar os pesos da matriz de pesos.

2 Entradas para Y e Y Invertido

```
# Entradas dos "Y's" normais original(entrada_1) e com ruídos ----> Saída (-1)

Conjunto de Treino para Y:

entrada_1 = np.array( [[+1, -1, -1, -1, +1],
                       [+1, -1, -1, -1, +1],
                       [-1, +1, +1, +1, -1],
                       [-1, -1, +1, -1, -1],
                       [-1, -1, +1, -1, -1]])

entrada_2 = np.array( [[+1, -1, -1, -1, +1],
                       [+1, +1, -1, -1, +1],
                       [-1, +1, +1, +1, -1],
                       [-1, -1, +1, -1, -1],
                       [-1, -1, +1, -1, -1]])

entrada_3 = np.array( [[+1, -1, -1, -1, +1],
                       [+1, -1, +1, -1, +1],
                       [-1, +1, +1, +1, -1],
```

```
                        [-1, -1, +1, -1, -1],
                        [-1, -1, +1, -1, -1]])


entrada_4 = np.array( [[+1, -1, -1, -1, +1],
                       [+1, -1, -1, +1, +1],
                       [-1, +1, +1, +1, -1],
                       [-1, -1, +1, -1, -1],
                       [-1, -1, +1, -1, -1]])

entrada_5 = np.array( [[+1, -1, -1, -1, +1],
                       [+1, -1, -1, -1, +1],
                       [-1, +1, +1, +1, -1],
                       [-1, -1, +1, +1, -1],
                       [-1, -1, +1, -1, -1]])

entrada_6 = np.array( [[+1, -1, -1, -1, +1],
                       [+1, -1, -1, -1, +1],
                       [-1, +1, +1, +1, -1],
                       [-1, -1, +1, -1, +1],
                       [-1, -1, +1, -1, -1]])

 Entradas dos "Y's" invertidos original(entrada_7) e com ruídos ----> Saída (+1)

 Conjunto de Treino para Y invertido:

entrada_7 = np.array( [[-1, -1, +1, -1, -1],
                       [-1, -1, +1, -1, -1],
                       [-1, +1, +1, +1, -1],
                       [+1, -1, -1, -1, +1],
                       [+1, -1, -1, -1, +1]])

entrada_8 = np.array( [[+1, -1, +1, -1, -1],
                       [+1, -1, +1, -1, -1],
                       [-1, +1, +1, +1, -1],
                       [+1, -1, -1, -1, +1],
                       [+1, -1, -1, -1, +1]])

entrada_9 = np.array( [[-1, -1, +1, -1, -1],
                       [-1, -1, +1, -1, +1],
                       [-1, +1, +1, +1, -1],
                       [+1, -1, -1, -1, +1],
                       [+1, -1, -1, -1, +1]])

entrada_10 = np.array( [[-1, -1, +1, -1, -1],
                        [-1, -1, +1, -1, -1],
                        [-1, +1, +1, +1, -1],
                        [+1, +1, -1, +1, +1],
                        [+1, -1, -1, -1, +1]])

entrada_11 = np.array( [[-1, -1, +1, -1, -1],
                        [-1, -1, +1, -1, -1],
                        [+1, +1, +1, +1, -1],
                        [+1, -1, -1, -1, +1],
                        [+1, -1, -1, -1, +1]])
```

```
entrada_12 = np.array( [[-1, -1, +1, -1, -1],
                        [-1, -1, +1, -1, -1],
                        [-1, +1, +1, +1, -1],
                        [+1, +1, -1, -1, +1],
                        [+1, -1, -1, +1, +1]])
```

3 Metodologia

O modelo Adaline foi construido através de classes [1] em Python com as seguintes funções presentes dentro da classe:

1 - Função de inicializar os pesos da matriz de pesos;

2 - Função de Treinamento do modelo e consequentemente o retorno é de uma matriz de pesos atualizada;

3 - Função de Teste com output do resultado da representação da matriz - se é Y (Rótulo -1) ou Y invertido (Rótulo +1).

Para o treinamento, foi utilizado o Método do gradiente descendente (GD), onde a atualização dos pesos é dada pela seguinte fórmula:

$$\begin{cases} \Delta w_{ij} = \eta(t_j - y_j)x_i \\ 0.1 < \alpha\eta \leq 1 \end{cases} \tag{1}$$

Onde α é o número de entradas e no presente projeto $\alpha = 12$. A variável t é o valor dos targets: -1 ou +1 a depender das entradas, x é o valor das entradas e y é output da soma dos pesos iniciais com as respectivas entradas:

$$\begin{cases} se\ y \leq 0 : y = -1 \\ se\ y > 0 : y = +1 \end{cases} \tag{2}$$

Para as entradas de 1 à 6, o rótulo da função de treinamento foi ajustado para -1, já para as entradas de 7 à 12, o rótulo foi ajustado para +1. Tais ajustes são feitos no vetor 1D-Array de nome targets (uma das entradas para a função de treinamento).

4 Saídas e Conclusão

Temos as seguintes entradas, logo abaixo, para testar no modelo após as atulizações dos pesos, geralmente o modelo passa por ciclos que resultam na melhoria dos pesos, em suma a média de ciclos foi de 2 a 3 ciclos com as 12 entradas para o treinamento.

```
# Testes pós treinamento:

entrada_13 = np.array( [[-1, -1, +1, -1, -1],
                        [-1, -1, +1, -1, -1],
                        [+1, +1, +1, +1, -1],
                        [+1, +1, -1, +1, -1],
                        [-1, +1, +1, -1, +1]]) ---> Output do teste deverá ser um
                                                    Y Invertido (Rótulo +1).
```

```
entrada_14 = np.array( [[+1, -1, +1, -1, +1],
                        [-1, +1, -1, +1, -1],
                        [-1, +1, +1, +1, -1],
                        [-1, -1, +1, -1, -1],
                        [-1, -1, +1, -1, +1]]) ---> Output do teste deverá ser um
                                                    Y (Rótulo -1).

entrada_15 = np.array( [[+1, -1, +1, -1, +1],
                        [-1, +1, +1, +1, -1],
                        [-1, +1, +1, +1, -1],
                        [-1, -1, +1, -1, -1],
                        [-1, -1, -1, -1, -1]]) ---> Output do teste deverá ser um
                                                    Y (Rótulo -1).

entrada_16 = np.array( [[-1, -1, -1, -1, -1],
                        [-1, -1, +1, -1, -1],
                        [+1, +1, +1, +1, -1],
                        [+1, +1, -1, +1, -1],
                        [+1, +1, -1, -1, +1]]) ---> Output do teste deverá ser um
                                                    Y Invertido (Rótulo +1).

entrada_17 = np.array( [[-1, -1, -1, -1, -1],
                        [-1, -1, +1, -1, -1],
                        [-1, -1, +1, -1, -1],
                        [-1, +1, -1, +1, -1],
                        [-1, -1, -1, -1, -1]]) ---> Output do teste deverá ser um
                                                    Y Invertido (Rótulo +1).

entrada_18 = np.array( [[-1, -1, -1, -1, -1],
                        [-1, +1, -1, +1, -1],
                        [-1, -1, +1, -1, -1],
                        [-1, -1, +1, -1, -1],
                        [-1, -1, -1, -1, -1]]) ---> Output do teste deverá ser um
                                                    Y (Rótulo -1).


entrada_19 = np.array( [[+1, -1, -1, +1, +1],
                        [-1, +1, +1, +1, -1],
                        [-1, -1, +1, -1, -1],
                        [-1, -1, +1, -1, -1],
                        [-1, -1, +1, -1, -1]]) ---> Output do teste deverá ser um
                                                    Y (Rótulo -1).
```

Após inicializar a função teste de dentro da classe AdalineY(), temos os seguintes resultados para as entradas:

Entrada	Output	Target	Acerto
entrada_13	Rótulo: +1 $\|\|Resultado : YInvertido$	1	sim
entrada_14	Rótulo: -1 $\|\|Resultado : Y$	-1	sim
entrada_15	Rótulo: -1 $\|\|Resultado : Y$	-1	sim
entrada_16	Rótulo: +1 $\|\|Resultado : YInvertido$	1	sim
entrada_17	Rótulo: +1 $\|\|Resultado : YInvertido$	1	sim
entrada_18	Rótulo: -1 $\|\|Resultado : Y$	-1	sim
entrada_19	Rótulo: -1 $\|\|Resultado : Y$	-1	sim

Table 1: Resultados dos testes.

5 Código

O presente código pode ser encontrado através do link:

Exercício 1 - Adaline - Classes em Python

References

[1] Sebastian Raschka. Machine Learning with Pytorch and Scikit-Learn. pages 39–50, 2022.