Instituto Superior de Engenharia de Lisboa

Inteligência Artificial e Sistemas Cognitivos

Mestrado Engenharia Informática e Multimédia

Inteligência Artificial e Sistemas Cognitivos

Semestre de Inverno, 2022/2023

Nome: Gonçalo Fonseca | Número: A50185

Índice

1 Introdução

O projeto descrito no presente relatório tem como objetivo apredner e explorar técnicas de machine learning (aprendizagem automática), para resolver certos problemas.
As técnicas existentes neste trabalho são as redes neuronais artificais, aprendizagem por reforço e X

Para serem aplicadas as técnicas acima referidas, as mesmas serão aplicadas em problemas referidos ao longo do relatório. A resolução desses mesmos problemas encontram-me nos ficheiro *Jupyter Notebook* presente no **repositório no Github**, e também entregue em anexo a este relatório.

2 Parte 1 - Redes Neuronais Artificais

2.1 Introdução teórica

As Redes Neurais são sistemas de equações, em que o resultado de uma equação é o valor de entrada para várias outras da rede.
Estas são um subconjunto da aprendizagem automática (*machine learning*) e estão no centro de algoritmos de aprendizagem profunda (*deep learnig*). O seu nome e estrutura são inspirados pelo cérebro humano, imitando a forma como os neurónios biológicos sinalizam uns aos outros. A figura 2 representa as similaridades entre um neurónio biológico e um neurónio nas redes neuronais.

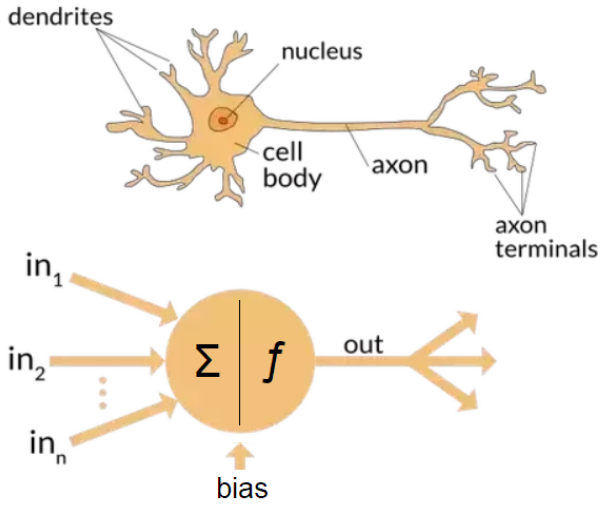

Figura 1: Um neurónio biológico e um neurónio artificial

Tal como um neurónio biológico tem dendritos para receber sinais, um uma célula para os processar, e um axónio para enviar sinais para outros neurónios, o neurónio artificial tem vários canais de entrada, uma fase de processamento, e uma saída que pode ser enviada para vários outros neurónios artificiais. E

como funciona um neurónio artificial? Este recebe entradas de alguns outros nós (neurónios), ou de uma fonte externa e calcula uma saída. Cada entrada tem um peso associado (w), que é atribuído com base na sua importância relativa a outras entradas. O neurónuo aplica uma função f (definida abaixo) à soma ponderada das suas entradas, como mostra a Figura 1 abaixo:

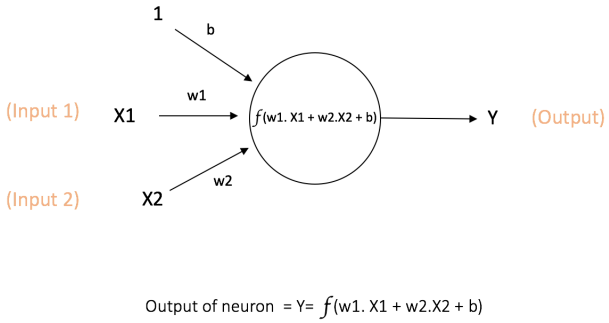

Figura 2: Neurónio único

A rede acima referida recebe entradas numéricas X1 e X2 e tem pesos w1 e w2 associados a essas entradas. Além disso, há outra entrada 1 com o peso b (chamado Bias - valor que permite deslocar a função de activação para a esquerda ou para a direita nos gráficos, que permite respostas diferentes) associado a ela. A saída Y do neurónio é calculada como se mostra na Figura 1. A função f é não linear e é chamada Função de Activação. O objectivo da função de activação é introduzir a não-linearidade na saída de um neurónio. Isto é importante porque a maioria dos dados do mundo real são não lineares e queremos que os neurónios aprendam estas representações não lineares.
Cada função de activação toma um único número e efectua sobre ele uma certa operação matemática fixa, sendo que há várias funções de activação que pode encontrar na prática. Estas irão ser referidas abaixo.

2.2 Aplicação prática

2.2.1 Aprendizagem da função lógica XOR

O XOR é um exemplo clássico de um problema de classificação. O XOR é a função que retorna 0 se duas entradas forem iguais e 1 se não forem iguais, tal como mostra a figura baixo.

Input 1	Input 2	Output
0	0	0
0	1	1
1	1	0
1	0	1

Figura 3: Inputs e Outputs na aprendizagem XOR

Se quisermos utilizar uma rede neural com uma camada de entrada (*input la*yer)

com dois inputs (e um input *bias*) e uma camada de saída (*output layer*), poderíamos fazer a classificação por uma linha de separação recta, o que para o XOR não é suficiente.

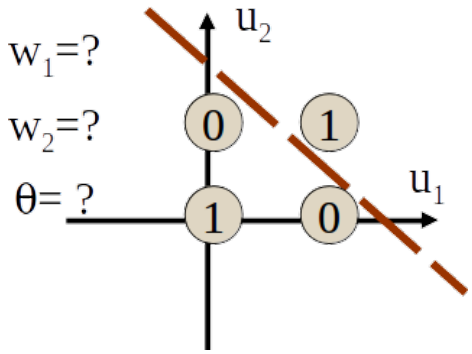

Figura 4: Inputs e Outputs na aprendizagem XOR

Como se pode observar na figura acima, não conseguimos desenhar apenas 1 linha para separar todos os "1"de todos os "0". E por isso, na rede neuronal na parte prática, terá que ser usada mais uma camada de neurónios. A isto se chama de **Multi Layered Perceptron** (ou **MLP**). Por isso, o plano inicial a ser usado é o da figura abaixo.

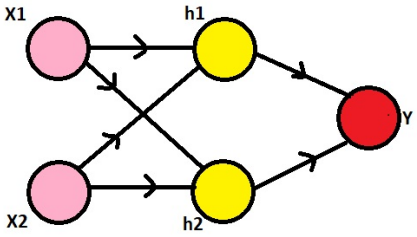

Figura 5: Inputs e Outputs na aprendizagem XOR

2.2.1.1 Implementação

Para implementar os problemas deste projeto que envolvam redes neuronais, usou-se a biblioteca **SciKit-Learn** para a linguagem Python, usando a classe **MLPClassifier**. E quais os parâmetros a serem usados nesta classe? Algumas classes serão implementadas logo de ínicio.

- ***Valores X***:
  - Significado: Conjunto de dados a ser usados para treinar a rede (*training data*);

- Escolha: [[-1, -1], [-1, 1], [1, -1], [1, 1]]

- ***Valores Y***:
  - Significado: Conjunto de dados a ser usados para testar a rede (*target data*);
  - Escolha: [-1, 1, 1, -1]

- ***solver***:
  - Significado: Otimizador que atualiza os pesos;
  - Escolha: **SGD (Stochastic Gradient Descent)**

- ***Função de ativação***:
  - Escolha: "tanh", visto que os dados estão com codificação bipolar [-1,1], devido à forma do gráfico da tangente. Abaixo os vários exemplos de funções de ativação.

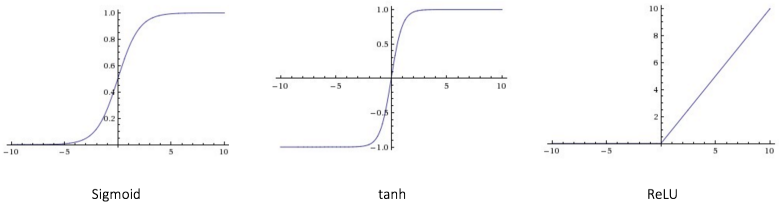

Figura 6: Diferentes funções de activação

- ***learning_rate***:
  - Significado: A taxa de aprendizagem é um hiperparâmetro que controla quanto alterar o modelo em resposta ao erro estimado cada vez que os pesos do modelo são actualizados [2];
  - Escolha: Como pedido, são aplicados os seguintes valores: 0.05, 0.25, 0.5, 1 e 2.

- ***max_iter***:
  - Significado: Número máximo de iterações. O solver itera até à convergência (determinada por 'tol') ou este número de iterações [3];
  - Escolha: Foi escolhido o valor de 10000 para garantir que o solver consegue chegar à convergência.

- ***tol***:

  - Significado: é a tolerância para os critérios de paragem. Isto diz ao modelo para parar de procurar um mínimo (ou máximo), uma vez alcançada alguma tolerância [4];
  - Escolha: Foi escolhido o valor de 10000 para garantir que o solver consegue chegar à convergência.

- ***hidden_layer_sizes***:
  - Significado: Número de neurónios das hidden layers (entre a layer de input e layer de output);
  - Escolha: Foi escolhido o valor de 10000 para garantir que o solver consegue chegar à convergência.

2.2.1.1.1 Efeito da taxa de aprendizagem

2.2.2 Aprendizagem de padrões de imagem

2.2.3 Aplicação e resolução de um problema livre

3 Parte 2 - Aprendizagem pro reforço

3.1 Introdução teórica

3.2 Aplicação prática

4 Parte 3 -

4.1 Introdução teórica

4.2 Aplicação prática

Referências

[1] Documentação de apoio à unidade curricular Inteligência Artificial e Sistemas Cognitivos
Luís Morgado, ISEL-DEETC.

[2] BROWNLEE, Jason - Understand the Impact of Learning Rate on Neural Network PerformanceMachine Learning Mastery, 24 jan. 2019. [Consult. 29 out. 2022]. Disponível em https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

[3] sklearn.neural_network.MLPClassifier - [Em linha] [Consult. 29 out. 2022]. Disponível em https://scikit-learn/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[4] ILANMAN - Answer to ≪What exactly is tol (tolerance) used as stopping criteria in sklearn models?≫Cross Validated, 9 jan. 2017. [Consult. 29 out. 2022]. Disponível em https://stats.stackexchange.com/a/255380