TP555 - AI/ML

Lista de Exercícios #12

Redes Neurais Artificiais (Parte 2)

1. Por que a função de ativação logística foi um ingrediente chave no treinamento das primeiras redes MLPs?
2. Utilizando o exemplo activation_functions.ipynb como base, cite três funções de ativação diferentes das que vimos.Plote a função e sua derivada.
3. Suponha que você tenha uma rede MLP composta por uma camada de entrada com 10 neurônios, seguida por uma camada oculta com 50 neurônios e, finalmente, uma camada de saída com 3 neurônios. Todos os neurônios usam a função de ativação ReLU. Sendo assim, responda
    a. Qual é a dimensão da matriz de entrada X?
    b. Qual a dimensão do vetor de pesos, W_h, da camada oculta e de seu vetor de bias b_h?
    c. Qual é a dimensão do vetor de pesos, W_o, da camada de saída e de seu vetor de bias b_o?
    d. Qual é a dimensão da matriz de saída, Y **,** da rede?
    e. Escreva a equação que calcula a matriz de saída da rede, Y, como uma função de X, W_h, b_h, W_o e b_o.
4. Quantos neurônios você precisa na camada de saída de uma rede MLP se você deseja classificar emails em spam ou ham? Qual função de ativação você deve usar na camada de saída? Se você deseja classificar a base de dados MNIST (imagens de dígitos escritos à mão), quantos neurônios você precisa na camada de saída usando qual tipo de função de ativação?
5. Liste todos os hiperparâmetros que você pode ajustar em uma rede MLP? Caso você perceba que a rede MLP está ***sobreajustando***, como você pode modificar esses hiperparâmetros para tentar resolver o problema?
6. Baseando-se no exemplo MLPWithTensorFlowLowLevelAPI.ipynb, utilize objetos da classe ***Dense()*** do módulo ***tf.keras.layers*** ao invés da função ***neuron_layer()***. Após treinar o modelo, você percebeu alguma diferença na performance do seu modelo? Se sim, qual foi esta diferença?
    (**Dica**: A documentação da classe dense pode ser encontrada no seguinte link: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense)
    (**Dica**: Lembre que ***Dense*** é uma classe que precisa ser instanciada. Apenas após a criação do objeto da classe ***Dense*** é que se deve passar a entrada para esta camada de nós.)
7. Baseando-se no código do exercício anterior, implemente a técnica do ***early stopping*** de forma que modelo sempre armazene os pesos que resultem no menor erro. Além

disso, crie um contador que conte o número de épocas sem progresso, ou seja, o número de vezes em que o erro da época não diminuiu com relação ao menor erro obtido até o momento. Com base nesse contador, faça com que o treinamento se encerre caso o contador atinja 50 épocas sem progresso.

8. Baseando-se no código do exercício anterior, utilize o otimizador que implementa o algoritmo momentum ao invés do gradiente descendente e treine uma rede MLP que obtenha uma precisão maior do que 98%. Use o trecho de código abaixo.

```
optimizer = tf.train.MomentumOptimizer(learning_rate=0.01, momentum=0.9)
```