Práctica 4 – Entrenamiento de redes neuronales

Realizada por Mario Blanco Domínguez y Juan Tecedor Roa

* Objetivo de la práctica

En esta práctica implementaremos el código para el calculo de la función de coste de una red neuronal. Usaremos el mismo conjunto de datos de la práctica 3: “ex4data1.mat” y “ex4weights.mat”.

También usaremos dos archivos que se nos han dado para comprobar la gradiente y mostrar los datos: “displayData.py” y “checkNNGradients.py”.

* Código de la práctica

Comenzamos importando todo lo necesario y implementando la función de coste:

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  from scipy.io import loadmat  import scipy.optimize as opt  import sklearn.preprocessing  import displayData, checkNNGradients  from displayData import \*  from checkNNGradients import \*  def sigmoid(X):  return 1 / (1 + np.exp(-X))  def cost(X, Y, l, T\_1, T\_2):  A1, A2, H = forward\_propagation(X, T\_1, T\_2)  m = X.shape[0]  l1 = np.transpose(np.log(H))  l2 = np.transpose(np.log(1 - H))  ret = ((l1.T \* -Y) - ((1 - Y) \* l2.T))  ret = np.sum(ret) / m  ret += (l / (2 \* m)) \* (np.sum(np.square(T\_1[:, 1:])) + np.sum(np.square(T\_2[:, 1:])))  return ret |

Segunda parte:

|  |
| --- |
|  |

* Conclusiones