```
#PARTE 1 PRACTICA 3

from ML_UtilsModule import Data_Management
from scipy.optimize import fmin_tnc as tnc
from matplotlib import pyplot as plt
import numpy as np

learning_rate = 0.1


def J(theta, X, Y):
    m = np.shape(X)[0]
    n = np.shape(X)[1]
    theta = np.reshape(theta, (1, n))
    var1 = np.dot(np.transpose((np.log(g(np.dot(X, np.transpose(theta)))))), Y)
    var2 = np.dot(np.transpose((np.log(1 - g(np.dot(X, np.transpose(theta)))))), (1 - Y))
    var3 = (learning_rate/(2*m)) * np.sum(theta[1:]**2)
    return (((-1/m)*(var1 + var2)) + var3)

def gradient(theta, X, Y):
    m = np.shape(X)[0]
    n = np.shape(X)[1]
    theta = np.reshape(theta, (1, n))
    var1 = np.transpose(X)
    var2  = g(np.dot(X, np.transpose(theta)))-Y

    theta = np.c_[[0], theta[:, 1:]]
    var3 = (learning_rate/m) * theta
    return ((1/m) * np.dot(var1, var2)) + np.transpose(var3)

def g(z):
    """
    1/ 1 + e ^ (-0^T * x)
    """
    return 1/(1 + np.exp(-z))

def oneVSAll(X, y, num_etiquetas, reg):
    """
    oneVsAll entrena varios clasificadores por regresión logística con término
    de regularización ’reg’ y devuelve el resultado en una matriz, donde
    la fila i-ésima corresponde al clasificador de la etiqueta i-ésima
    """
    clasificadores = np.empty((num_etiquetas, np.shape(X)[1]))
    theta = np.random.standard_normal((1, np.shape(X)[1]))
    mask = np.empty((num_etiquetas, np.shape(y)[0]), dtype=bool)

    for i in range(num_etiquetas):
        mask[i, :] = (y[:, 0]% num_etiquetas == i)
        clasificadores[i] = tnc(func=J, x0=theta, fprime=gradient, args=(X, np.reshape(mask[i], (np.shape(X)[0
], 1))))[0]
    return clasificadores

def checkLearned(X, y, clasificadores):
    result = checkNumber(X, clasificadores)

    maxIndexV = np.argmax(result, axis = 1);

    checker = ((y[:,0]%np.shape(clasificadores)[0]) == maxIndexV)
    count = np.size(np.where(checker == True))
    fin = count/np.shape(y)[0] * 100

    return fin

def checkNumber(X, clasificadores):
    result = np.zeros((np.shape(X)[0],np.shape(clasificadores)[0]))

    result[:] = g(np.dot(X, np.transpose(clasificadores[:]))) #result[0] = todo lo que piensa de cada numero r
especto a si es 0
    return result

X, y = Data_Management.load_mat("ex3data1.mat")
clasificadores = oneVSAll(X, y, 10, 2)
```

```
print ("Precision: " + str(checkLearned(X, y, clasificadores)) + " %")

Data_Management.draw_random_examples(X)
plt.show()
```

```
    9   92  7.847772330914368E-02   1.28437446E-06
   10  101  7.828153190340245E-02   1.88280103E-06
   11  109  7.810717424768297E-02   2.62945187E-06
   12  117  7.789570474974815E-02   5.59104821E-07
   13  132  7.781958606785146E-02   3.52946174E-07
tnc: fscale = 873.339
   14  140  7.781131352898749E-02   6.71852685E-08
   15  150  7.780671662509688E-02   2.96793834E-08
   15  199  7.780671662509688E-02   2.96793834E-08
tnc: Linear search failed
Precision: 95.86 %
```

Ilustración 1: Precisión a la hora de distinguir números escritos a mano.

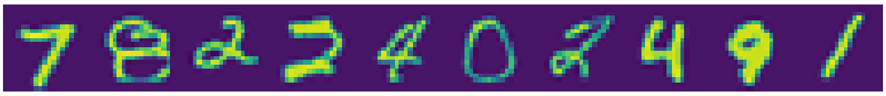

Ilustración 2: Números representados y elegidos aleatoriamente.

```python
#PARTE 2 PRACTICA 3

from ML_UtilsModule import Data_Management
from scipy.io import loadmat
from scipy.optimize import fmin_tnc as tnc
from matplotlib import pyplot as plt
import numpy as np


def g(z):
    """
    1/ 1 + e ^ (-0^T * x)
    """
    return 1/(1 + np.exp(-z))


def propagation(X, theta1, theta2):
    hiddenLayer = g(np.dot(X, np.transpose(theta1)))
    hiddenLayer = Data_Management.add_column_left_of_matrix(hiddenLayer)

    outputLayer = g(np.dot(hiddenLayer, np.transpose(theta2)))

    return outputLayer


def checkLearned(y, outputLayer):

    maxIndexV = np.argmax(outputLayer, axis = 1) + 1

    checker = (y[:,0] == maxIndexV)
    count = np.size(np.where(checker == True))
    fin = count/np.shape(y)[0] * 100
    return fin



weights = loadmat('ex3weights.mat')
theta1, theta2 = weights['Theta1'], weights['Theta2']

X, y = Data_Management.load_mat("ex3data1.mat")

X = Data_Management.add_column_left_of_matrix(X) #añadida columna de 1s

outputLayer = propagation(X, theta1, theta2)

print("Precision de la red neuronal: " + str(checkLearned(y, outputLayer)) + " %")
```

```
Python - practica3_2.py:22 ✔

Precision de la red neuronal: 97.52 %
[Finished in 1.455s]
```

Ilustración 3: Precisión de la red neuronal ya entrenada.