# Logistic regression and Neural Networks

## First part:

In this part of the practice, the objective is to apply multi-class(multinomial) logistic regression for a dataset with handwritten numbers.

The code for this part:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize as opt
from scipy.io import loadmat
from sklearn.preprocessing import PolynomialFeatures

data = loadmat('ex3data1.mat')
#se pueden consultar las claves con data.keys( )
y = data ['y']
X = data ['X']
#almacena los datos leídos en X, y

m = np.shape(X)[0]
n = np.shape(X)[1]

#Selecciona aleatoriamente 10 ejemplos y los pinta
sample = np.random.choice(X.shape[0], 10)
plt.imshow(X[sample, :].reshape(-1, 20).T)
plt.axis('off')
#plt.show()

def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def cost(theta, X, Y, lambd):
    theta = theta.reshape((len(theta), 1))
    A = sigmoid(np.matmul(X, theta))
    reg = (lambd / (2 * m)) * np.sum(theta ** 2)
    return (- 1 / (len(X))) * np.sum(Y * np.log(A) + (1 - Y) * np.log(1 - A + 1e-6)) + reg

def gradient(theta, X, Y, lambd):
    theta = theta.reshape((len(theta), 1))
    A = sigmoid(np.matmul(X, theta))
    identity = np.identity(theta.shape[0])
    identity[0][0] = 0
    return np.dot(X.T, (A - Y)) / len(Y) + (lambd / m) * np.dot(identity,
theta)

def model(x, Y, lambd):
    theta = np.zeros(X.shape[1])
    result = opt.fmin_tnc (func=cost, x0=theta, fprime=gradient, args = (X, Y,
lambd))
    theta = result[0]
    return theta

def oneVsAll(X, y, n_labels, 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
    """
    theta = []
    for i in range(n_labels):
        Y = (y % 10 == i) * 1
        theta.append(model(X, Y, reg))
    theta = np.array(theta)
    return theta


def calculate_probability(X, Y, theta):
    prediction = np.dot(X, theta.T)
    index_max = np.argmax(prediction, axis = 1)
    index_max = index_max.reshape((len(index_max), 1))
    error = np.sum(Y % 10 == index_max) / Y.shape[0]
    print(str(error * 100) + '%')


X = np.hstack([np.ones([m, 1]), X])
thetas = oneVsAll(X, y, 10, 0.1)
calculate_probability(X, y, thetas)
```

Examples classified correctly: 96.46000000000001%

Second  part:

The objective of this part is to use already pretrained Neural Network’s wights to evaluate its precision. The dataset is the same as in the first part of this practice.

The code for this part:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize as opt
from scipy.io import loadmat
from sklearn.preprocessing import PolynomialFeatures

data = loadmat('ex3data1.mat')
#se pueden consultar las claves con data.keys( )
y = data ['y']
X = data ['X']
#almacena los datos leídos en X, y

m = np.shape(X)[0]
n = np.shape(X)[1]

X = np.hstack([np.ones([m, 1]), X])

weights = loadmat('ex3weights.mat')
theta1, theta2 = weights['Theta1'], weights['Theta2']
#Theta1 es de dimensión 25x401
#Theta2 es de dimensión 10x26

def sigmoid(z):
   return 1 / (1 + np.exp(-z))

def calculate_probability(H, Y):
   index_max = np.argmax(H, axis = 1)
   index_max = index_max.reshape((len(index_max), 1))
   error = np.sum(Y - 1 == index_max) / Y.shape[0]
   print(str(error * 100) + '%')

def neuro_network(X, Y, theta1, theta2):
   Z1 = np.dot(X, theta1.T)
   A1 = sigmoid(Z1)
   A1 = np.hstack([np.ones([A1.shape[0], 1]), A1])
```

```
    Z2 = np.dot(A1, theta2.T)
    A2 = sigmoid(Z2)
    calculate_probability(A2, Y)

neuro_network(X, y, theta1, theta2)
```

Examples classified correctly: 97.52%

Conclusion:

In this practice, we used one dataset with handwritten digits and two different models to predict. The first is a multi-class logistic regression where the obtained accuracy is 96.46%, and a Neural Network with pre-trained weights where it is 97.52%. From the percentages of correctly classified examples, we can see that the NN, as it is deeper,  increases the performance by ~1.1%.

Gasan Nazer and Veronika Yankova