PRÁCTICA 3: REGRESIÓN LOGÍSTICA MULTICLASE Y RED NEURONAL

Aprendizaje Automático y Big Data

8 DE NOVIEMBRE DE 2018

FELIX VILLAR Y VÍCTOR RAMOS
Universidad Complutense de Madrid

1. Regresión logística multiclase

Código:

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

def sigmoide(x):
    return 1/(1+ np.exp(np.negative(x)))

def coste(th,entradas, salidas,reg):
    #Formula vectorizada
    a = sigmoide(np.matmul(entradas, th))
    b = np.matmul(np.log(a).T,salidas)
    c = np.matmul(np.log(1-a).T,(1-salidas))
    d= (b+c)/(len(entradas)*-1)
    #Termino de regularizacion
    e = (reg/2*len(entradas))*np.sum(np.square(th))
    return d +e

def gradiente(th,entradas,salidas,reg):
    #Formula vectorizada
    y = np.reshape(salidas, salidas.shape[0])
    a=sigmoide(np.matmul(entradas,th))-y
    b = (np.matmul(entradas.T,a))/len(entradas)
    #Termino de regularizacion
    c=reg*th/len(entradas)
    c[0]= 0
    d = b + c
    return d

def oneVsAll(X, y, num_etiquetas, reg):
    X = np.concatenate((np.atleast_2d(np.ones(X.shape[0])).T,X),axis=1)
    entrenador = np.zeros((num_etiquetas,X.shape[1]))
    for i in range(0, num_etiquetas):
        if(i==0):
            z = 10
        else:
            z = i
        entrenador[i]=
opt.fmin_tnc(coste,entrenador[i],fprime=gradiente,args=(X,(y==z)*1,reg))[0]
    result = np.matmul(entrenador,X.T)
    maximo = np.argmax(result, axis = 0)
    maximo[maximo == 0] = 10
    comparacion=(maximo==y[:,0])*1
    bienPredecidos= np.count_nonzero(comparacion)
    #Calculo del porcentaje
    porcentaje=(bienPredecidos/len(comparacion))*100
    print(porcentaje)


data = loadmat('ex3data1.mat')
y= data['y']
x= data['X']

oneVsAll(x,y,10,0.1)
```

2. Red Neuronal

Código:

```
import scipy.io
import numpy as np


#FUNCION DE ACTIVACION -> FUNCION SIGMOIDE
def sigmoide(x):
    return 1/(1+ np.exp(np.negative(x)))

def propagacionHaciaDelante(theta1,theta2,X):
    X=np.transpose(X)
    z2=np.matmul(theta1,X)
    a2=sigmoide(z2)
    a2=np.transpose(a2)
    a2=np.concatenate((np.atleast_2d(np.ones(len(a2),int)).T,a2),axis=1)
    z3=np.matmul(theta2,np.transpose(a2))
    a3=sigmoide(z3)
    return a3

def porcentajeRedNeuronal(h,y):
    #Busca el maximo de la matriz h
    maximo = np.argmax(h, axis = 0)
    #1 si está bien y 0 si está mal
    salida = np.reshape(y,5000)
    maximo = maximo +1
    comparacion=(maximo==salida)
    #Cuenta el numero de unos que tiene comparacion
    comparacion = comparacion*1
    bienPredecidos= np.count_nonzero(comparacion==1)
    #Calculo del porcentaje
    porcentaje=(bienPredecidos/len(comparacion))*100
    return porcentaje


def redNeuronal():
    data = scipy.io.loadmat('ex3data1.mat')
    y = data['y']
    X = data ['X']
    weights = scipy.io.loadmat('ex3weights.mat')
    theta1, theta2 = weights['Theta1'], weights['Theta2']
    aux=np.ones(X.shape[0],dtype=int)
    X = np.concatenate((np.atleast_2d(aux).T,X),axis=1)
    h=propagacionHaciaDelante(theta1,theta2,X)
    porcentaje=porcentajeRedNeuronal(h,y)
    print(porcentaje)

    #¿Como añado aux a X?


redNeuronal()
```

3. Comentarios

- La regresión logística multiclase tiene un porcentaje alrededor de 75% cuando el enunciado dice 95% pero con este porcentaje tan suficientemente grande se puede demostrar el buen funcionamiento de la práctica.
- La red neuronal concuerda con el enunciado (95%).