Práctica 3: Regresión logística multi-clase y redes neuronales

David Godoy Ruiz
Eva Lucas Leiro

Regresión logística multi-clase:

- ### Clasificación de uno frente a todos

  Para hacer la regresión logística multi-clase utilizamos la clasificación de uno frente a todos, entrenando un clasificador por cada uno de los números que se van a intentar predecir, cada clasificador considerando 1 como correcto (el número que le corresponde) y 0 como incorrecto (cualquier otro número).

  Para predecir un valor de y calculamos la probabilidad de que sea cada número uno por uno y nos quedamos con el número con mayor probabilidad para luego compararlo con los casos de ejemplo

```
def oneVsAll(X, y, num_etiquetas, reg):
    m = np.shape(X)[0]
    X1s = np.hstack([np.ones([m, 1]), X])
    theta_opt = np.zeros([num_etiquetas, X1s.shape[1]])

    for i in range(num_etiquetas):
        theta = np.zeros(X1s.shape[1])
        if (i == 0):
            result = opt.fmin_tnc( func=coste_reg , x0=theta ,
fprime=gradiente_reg , args =(X1s,(y == 10) *1, reg))
        else:
            result = opt.fmin_tnc( func=coste_reg , x0=theta ,
fprime=gradiente_reg , args =(X1s,(y == i) *1, reg))
        theta_opt[i] = result[0]
    return theta_opt

#---------------------------------------------------------------------

def evaluacion(theta_opt, X, y):
    m = np.shape(X)[0]
    X1s = np.hstack([np.ones([m, 1]), X])
```

```
    chances = np.zeros([np.shape(theta_opt)[0], np.shape(X1s)[0]])
    for i in range(np.shape(theta_opt)[0]):
        chances[i, :] = sigmoid(np.matmul(X1s,theta_opt[i]))


    maxChance = chances.argmax(axis= 0)
    correctos = np.sum(maxChance == (y % 10))
    return correctos/m * 100

#---------------------------------------------------------------------

def regresion_multi():
    X, y = load()
    y = y.ravel()
    theta_opt = oneVsAll(X, y, 10, 0.1)
    print("Correctos: ", evaluacion(theta_opt, X, y))
```

Redes Neuronales:

Ya que la red neuronal ya está entrenada, tan solo tenemos que hacer la propagación hacia adelante en una red neuronal de tres capas. Para comparar el resultado lo hacemos igual que arriba.

```
def propagacion():
    X, y = load()
    y = y.ravel()
    theta1, theta2= loadRed()
    m = np.shape(X)[0]

    a1=np.hstack([np.ones([m, 1]), X])
    z2=np.dot(a1, theta1.T)
    a2=np.hstack([np.ones([m, 1]), sigmoid(z2)])
    z3=np.dot(a2, theta2.T)
    a3=sigmoid(z3)

    maxChance = a3.argmax(axis= 1)
    correctos = np.sum(maxChance+1 == y)
    return correctos/m * 100
```