Machine Learning

January 17, 2019

1 PRACTICA 4: ENTRENAMIENTO DE REDES NEURONALES

In [3]:
```
import matplotlib.pyplot as pl
import numpy as np
from pandas.io.parsers import read_csv
import scipy.optimize as opt
#from sklearn.preprocessing import PolynomialFeatures as pf
from scipy.io import loadmat
import checkNNGradients
```

In [4]:
```
def sigmoide(z):
    return 1 / (1 + np.exp(-1*z))
```

In [5]:
```
def derivadaSigmoide(z):
    return sigmoide(z)*(1-sigmoide(z))
```

In [6]:
```
def pesosAleatorios(L_in, L_out):
    return np.random.rand(L_out, 1 + L_in) * 0.24 - 0.12
```

In [7]:
```
def backprop(params_rn, num_entradas, num_ocultas , num_etiquetas , X, y, reg):

    theta1 = np.reshape(params_rn[:num_ocultas*(num_entradas + 1)], (num_ocultas, (num_
    theta2 = np.reshape(params_rn[num_ocultas*(num_entradas + 1):], (num_etiquetas, (nu
    m = len(X)
   #Input
    ones_columns_input = np.array(np.ones(m))
    a1 = np.insert(X, 0,ones_columns_input, axis = 1)

    #hidden_layer
    z2 = np.dot(theta1, a1.transpose())
    a2 = sigmoide(z2)
    one_columns_hidden = np.array(np.ones(m))
    a2 = np.insert(a2, 0, one_columns_hidden, axis = 0)

    #Output_layer
    z3 = np.dot(theta2, a2)
    h = sigmoide(z3)
```

```
            y_converted = np.zeros(h.shape)
            for i in range (0,len(X)):
                y_converted[np.ravel(y)[i]-1][i] = 1

             #Cost
            regulation = (reg/(2*m)) * (np.sum(theta1**2) + np.sum(theta2**2))
            J = np.sum(-y_converted * np.log(h) - (1 - y_converted)*np.log(1 - h)) * (1/m)
            J_regulated = J + regulation

            # Retro-Propagation
            d3 = h - y_converted
            z2 = np.insert(z2, 0, np.ones(m), axis = 0)
            z2prima = derivadaSigmoide(z2)
            d2 = (np.dot(theta2.transpose(), d3))*z2prima

            #Gradient
            delta2 = np.dot(d3,a2.transpose())
            delta1 = np.dot(d2, a1)

            #Regularization

            D1 = (delta1[1:,:]/m + theta1*reg/m).ravel()
            D2 = (delta2/m + theta2*reg/m).ravel()
            gradient = np.r_[D1, D2]

            return J_regulated, gradient
```

In [8]:
```
def test():
    num_entradas = 400
    num_ocultas = 25
    num_etiquetas = 10
    reg = 1
    data = loadmat('ex4data1.mat')
    y = data ['y']
    X = data ['X']
    weights = loadmat( 'ex4weights.mat' )
    theta1, theta2 = weights[ 'Theta1' ], weights[ 'Theta2' ]
    coste, gradiente = backprop(np.concatenate((np.ravel(theta1), np.ravel(theta2))), r
    checkNNGradients.checkNNGradients(backprop, reg)
```

In [9]:
```
test()
```

Autores: - Montserrat Sacie Alcázar - Tomás Golomb Durán