PRÁCTICA 5: REGRESIÓN LINEAL REGULARIZADA: SESGO Y VARIANZA

Aprendizaje Automático y Big Data

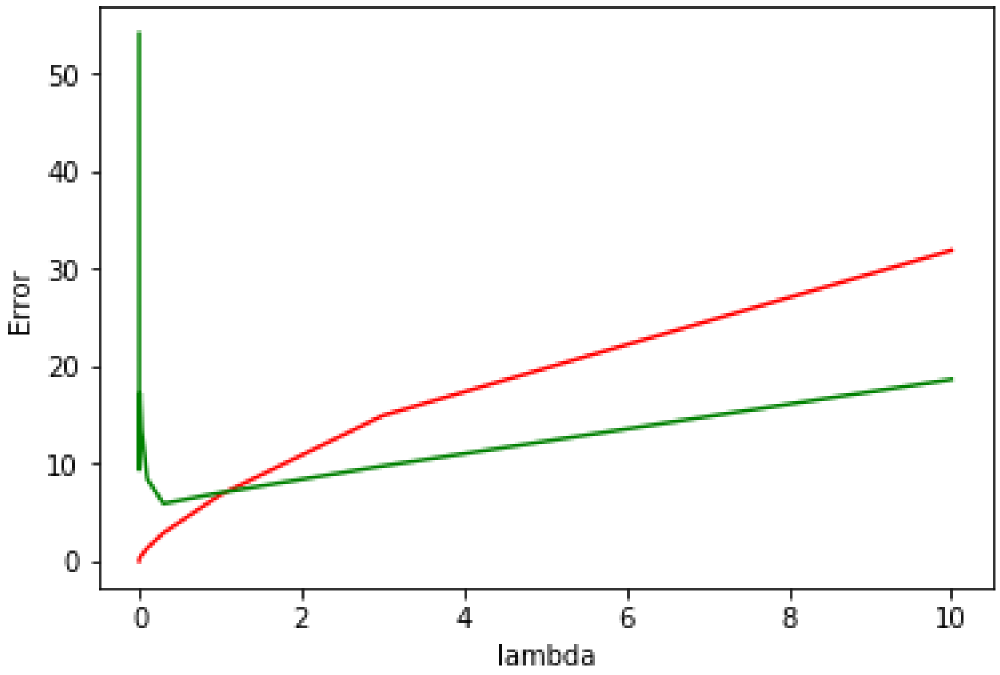
29 DE NOVIEMBRE DE 2018

FELIX VILLAR Y VÍCTOR RAMOS
Universidad Complutense de Madrid

1. ***Código***

```
import numpy as np

import scipy.io

import scipy.optimize as opt

import matplotlib.pyplot as plt


def normalizar(X):


    med = np.mean(X,axis=0)

    desv = np.std(X,axis=0)

    aux = (X-med)/desv

    return aux, med,desv


def aprendpolin(X,y,Xval,yval,p,lmdb):

    m = len(X)

    Xnorm= matrizpolin(X,p)

    Xnorm, med, desv = normalizar(Xnorm)


    Xvalnorm = matrizpolin(Xval,p)

    Xvalnorm = (Xvalnorm-med)/desv

    th_ini = np.zeros(((Xnorm.shape[1]+1),1))

    J = np.zeros((11))

    Jval= np.zeros((11))

    for i in range(1,m):

        th = opt.minimize(regreslinealreg,th_ini,args=(Xnorm[0:i],y[0:i],lmdb), jac=True).x

        J[i-1],grad = regreslinealreg(th,Xnorm[0:i],y[0:i],lmdb)

        Jval[i-1],gradval = regreslinealreg(th,Xvalnorm,yval,lmdb)


    plt.xlabel('Number of training examples')

    plt.ylabel('Error')

    plt.plot(J, c='r')

    plt.plot(Jval, c= 'b')
```

```
def errorlmdb(X,y,Xval,yval,p):
    lmdb= np.array([0,0.001,0.003,0.01,0.03,0.1,0.3,1,3,10])
    m= len(lmdb)
    Xnorm= matrizpolin(X,p)
    Xnorm, med, desv = normalizar(Xnorm)

    Xvalnorm = matrizpolin(Xval,p)
    Xvalnorm = (Xvalnorm-med)/desv
    th_ini = np.zeros(((Xnorm.shape[1]+1),1))
    J = np.zeros((m))
    Jval= np.zeros((m))
    for i in range(0,m):
        th = opt.minimize(regreslinealreg,th_ini,args=(Xnorm,y,lmdb[i]), jac=True).x
        J[i],grad = regreslinealreg(th,Xnorm,y,lmdb[i])
        Jval[i],gradval = regreslinealreg(th,Xvalnorm,yval,lmdb[i])

    plt.xlabel('lambda')
    plt.ylabel('Error')
    plt.plot(lmdb,J, c='r')
    plt.plot(lmdb,Jval, c= 'g')

def matrizpolin(X,p):
    m = len(X)
    a = np.zeros((m,p-1))
    aux = np.hstack((X,a))
    for i in range(0,p):
        aux[:,i]= X[:,0]**(i+1)
    return aux

def regresionpolinomica(X, y,p):
    matrizp= matrizpolin(X,p)
    matriznorm, media, desv = normalizar(matrizp)
    th_ini=np.zeros((matriznorm.shape[1]+1,1))
    sol = opt.minimize(regreslinealreg,th_ini,args=(matriznorm,y,0), jac=True)
    th =sol.x
    res = np.sum(matriznorm*th[1:],axis=1)+th[0]
    plt.scatter(X,y,c='r')
    plt.plot(X,res)


def aprendizaje(X,y,Xval, yval):
    m = len(X)
    th_ini = np.zeros((2,1))
    J = np.zeros((11))
    Jval= np.zeros((11))
    for i in range(1,m):
        th = opt.minimize(regreslinealreg,th_ini,args=(X[0:i],y[0:i],0), jac=True).x
        J[i-1],grad = regreslinealreg(th,X[0:i],y[0:i],0)
        Jval[i-1],gradval = regreslinealreg(th,Xval,yval,0)

    plt.xlabel('Number of training examples')
    plt.ylabel('Error')
    plt.plot(J, c='r')
    plt.plot(Jval, c= 'b')

def pintar(X,y,th):
    plt.scatter(X,y,c='r')
    plt.xlabel('Water level')
    plt.ylabel('Water out')

    plt.plot(X,X*th[1]+th[0])
    plt.show()

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

def regreslinealreg(th,X,y,lmdba):
    m = len(X)
    aux = np.hstack((np.ones((X.shape[0],1)),X))
    a = np.matmul(aux,th)
    b = np.sum(np.square(a - y.T))/(2*m)
    c= (lmdba/(2*m))*np.sum(np.square(th[1:]))
    coste = b+c

    b= np.matmul(aux.T,(a-y.T).T)/m
    c=(lmdba/m)*th[1:]
    b = np.reshape(b,(b.shape[0]))
    b[1:]+= c
    b = np.reshape(b,b.shape[0])
    return coste,b

def calcerror(X,y,Xtest,ytest,p):
    Xnorm= matrizpolin(X,p)
    Xnorm, med, desv = normalizar(Xnorm)

    Xtestnorm = matrizpolin(Xtest,p)
    Xtestnorm = (Xtestnorm-med)/desv
    th_ini = np.zeros(((Xnorm.shape[1]+1),1))
    th = opt.minimize(regreslinealreg,th_ini,args=(Xnorm,y,3), jac=True).x
    J,grad = regreslinealreg(th,Xtestnorm,ytest,0)
    return J
def main():
    data = scipy.io.loadmat('ex5data1.mat')
    y = data['y']
    X = data['X']
    Xval = data['Xval']
    yval = data['yval']
    Xtest = data['Xtest']
    ytest = data['ytest']
    #th = np.array((1,1))
    #sol = opt.minimize(regreslinealreg,th,args=(X,y,0), jac=True)
    #print(result)
    #print(sol)
    #pintar(X,y,sol.x)
    #aprendizaje(X,y,Xval,yval)
    #regresionpolinomica(X,y,8)

    #aprendpolin(X,y,Xval,yval,8,0)
    errorlmdb(X,y,Xval,yval,8)
    res = calcerror(X,y,Xtest,ytest,8)
    print(res)
main()
```


Error
lambda