

Práctica 2: regresión lOGística

Aprendizaje Automático y Big Data



25 de octubre de 2018

Felix Villar Y víctor ramos

Universidad Complutense de Madrid

1. ***Regresión logística***

***Código:***

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):

a=sigmoide(np.matmul(entradas, th))

b=np.matmul(np.log(a).T,salidas)+np.matmul(np.log(1-a).T,(1-salidas))

return b/(len(entradas)\*-1)

def gradiente(th,entradas,salidas):

a=sigmoide(np.matmul(entradas,th))-salidas

return (np.matmul(entradas.T,a))/len(entradas)

def pinta\_frontera\_recta(X, Y, theta):

plt.figure()

pos = np.where(Y==1)

neg = np.where(Y==0)

plt.scatter(X[pos,0],X[pos,1],marker ='+',c='k')

plt.scatter(X[neg,0],X[neg,1],marker ='o',c='y')

x1\_min, x1\_max = X[:, 0].min(), X[:, 0].max()

x2\_min, x2\_max = X[:, 1].min(), X[:, 1].max()

xx1, xx2 = np.meshgrid(np.linspace(x1\_min, x1\_max),np.linspace(x2\_min, x2\_max))

h = sigmoide(np.c\_[np.ones((xx1.ravel().shape[0], 1)), xx1.ravel(), xx2.ravel()].dot(theta))

h = h.reshape(xx1.shape)

plt.contour(xx1, xx2, h, [0.5], linewidths=1, colors='b')

plt.savefig("frontera.jpg")

plt.close()

def regresion(data):

x =np.genfromtxt(data, delimiter = ',')

numparametros = x[0].size-1

unos = np.ones(int(x.size/(numparametros+1)))

ent = x[:,:-1]

entradas = np.concatenate((np.atleast\_2d(unos).T,ent),axis=1)

y = x[:,-1]

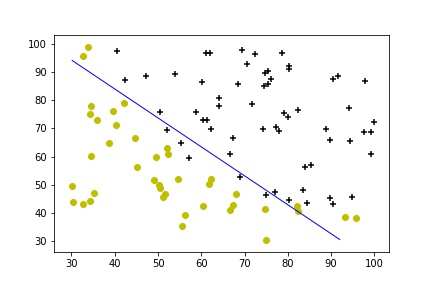
th = np.zeros(numparametros+1)

result = opt.fmin\_tnc(coste, th, gradiente, args=(entradas, y))

th = result[0]

pinta\_frontera\_recta(ent,y,th)

regresion('ex2data1.csv')



1. ***Regresión logística regularizada***

***Código:***

import numpy as np

import matplotlib.pyplot as plt

import scipy.optimize as opt

from sklearn.preprocessing import PolynomialFeatures

def sigmoide(x):

return 1/(1+ np.exp(np.negative(x)))

def coste(th,entradas, salidas, alpha):

a = sigmoide(np.matmul(entradas, th))

b = (np.matmul(np.log(a).T,salidas) + np.matmul(np.log(1-a).T,(1-salidas)))/(len(entradas)\*-1)

c = (alpha/2\*len(entradas))\*np.sum(np.square(th))

d = b+c

sol = d

return sol

def gradiente(th, entradas, salidas, alpha):

a= np.matmul(entradas.T,(sigmoide(np.matmul(entradas,th))-salidas))

b=alpha\*th

b[0]= 0

sol = (a+b)/len(entradas)

th= sol

return sol

def plot\_decisionboundary(x,y,theta,poly):

neg = np.where(y==0)

pos = np.where(y==1)

plt.figure()

x1\_min,x1\_max = x[:,0].min(), x[:,0].max()

x2\_min,x2\_max = x[:,1].min(), x[:,1].max()

xx1,xx2= np.meshgrid(np.linspace(x1\_min,x1\_max),np.linspace(x2\_min,x2\_max))

h = sigmoide(poly.fit\_transform(np.c\_[xx1.ravel(),xx2.ravel()]).dot(theta))

h = h.reshape(xx1.shape)

plt.scatter(x[pos,0],x[pos,1],marker ='+',c='k')

plt.scatter(x[neg,0],x[neg,1],marker ='o',c='y')

plt.contour(xx1,xx2,h,[0.5],linewidths=1,colors='g')

plt.savefig("boundary.jpg")

plt.close()

def regresion(data):

x =np.genfromtxt(data, delimiter = ',')

poly = PolynomialFeatures(6)

ent = x[:,:-1]

entradas = poly.fit\_transform(x[:,:-1])

th = np.zeros(entradas.shape[1])

alpha = 1

#entradas = np.concatenate((np.atleast\_2d(unos).T,entradas),axis=1)

salidas = x[:, -1]

res = opt.fmin\_tnc(coste,th,fprime=gradiente,args=(entradas,salidas,alpha))

plot\_decisionboundary(ent,salidas,res[0],poly)

regresion('ex2data2.csv')

