Práctica 1: Regresión Lineal

Ana Martín Sánchez, Nicolás Pastore Burgos

21/09/2021

1 Descripción de la práctica

En esta práctica, se pedía aplicar el método de regresión logística sobre dos conjuntos de datos.

En primer lugar, se nos da un conjunto de datos que relaciona dos variables: la nota de varios candidatos en un examen de acceso a la universidad, y si fueron admitidos o no. Después, se nos presenta un conjunto de datos que relaciona el resultado de someter a varios microchips a dos controles de calidad, y si los superan o no.

Para el primer conjunto de datos, se puede observar que los datos se pueden separar linealmente, definiendo una barrera entre los que superan los exámenes de acceso y los que no. En el segundo caso, esta barrera no es lineal.

Por ello, en el primer caso podemos aplicar la regresión logística sin problemas pero, en el segundo conjunto de datos, tendremos que utilizar la versión regularizada de este mismo algoritmo.

2 Solución propuesta

2.1 Resultados obtenidos

2.1.1 Parte 1

Conseguimos implementar una función que devuelve un coste óptimo de 0.203 para los datos proporcionados, y que permite dibujar una recta como se muestra en la figura 1.

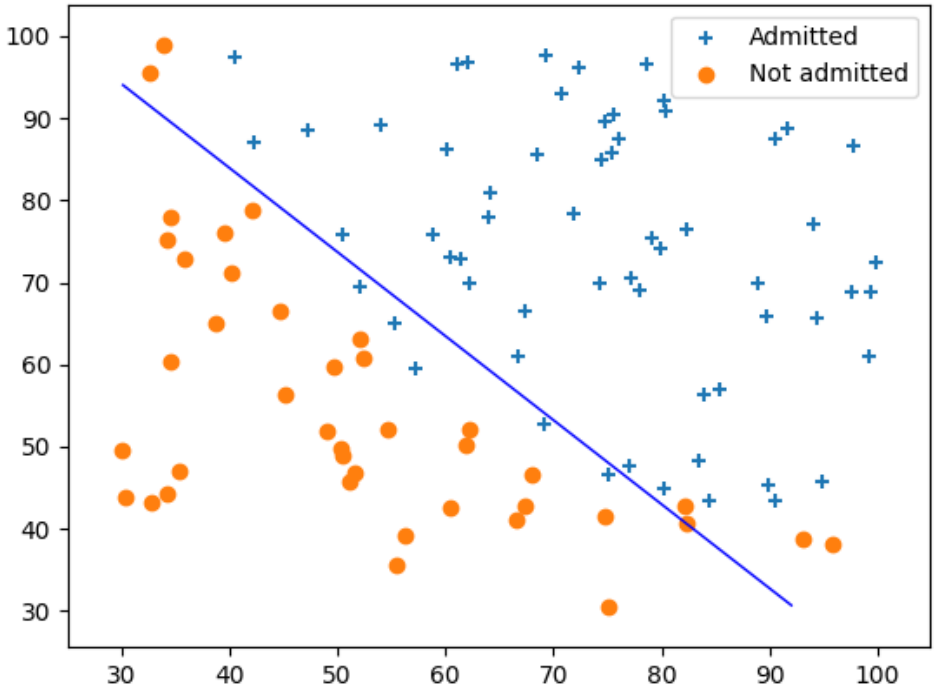

Figure 1: Gráfica que muestra la recta calculada para los datos proporcionados por el enunciado.

2.1.2 Parte 2

En esta parte hemos conseguido los reultados esperados; hemos podido implementar una función de coste y otra de descenso de gradiente tales que, para

$$\theta = 0$$

$$\lambda = 1$$

el coste inicial es de 0.693, y permite dibujar una barrera como la que se muestra en la figura 2.

También es interesante remarcar que, cambiando el valor de lambda, se regulariza el descenso, pero se sigue consiguiendo un resultado correcto.

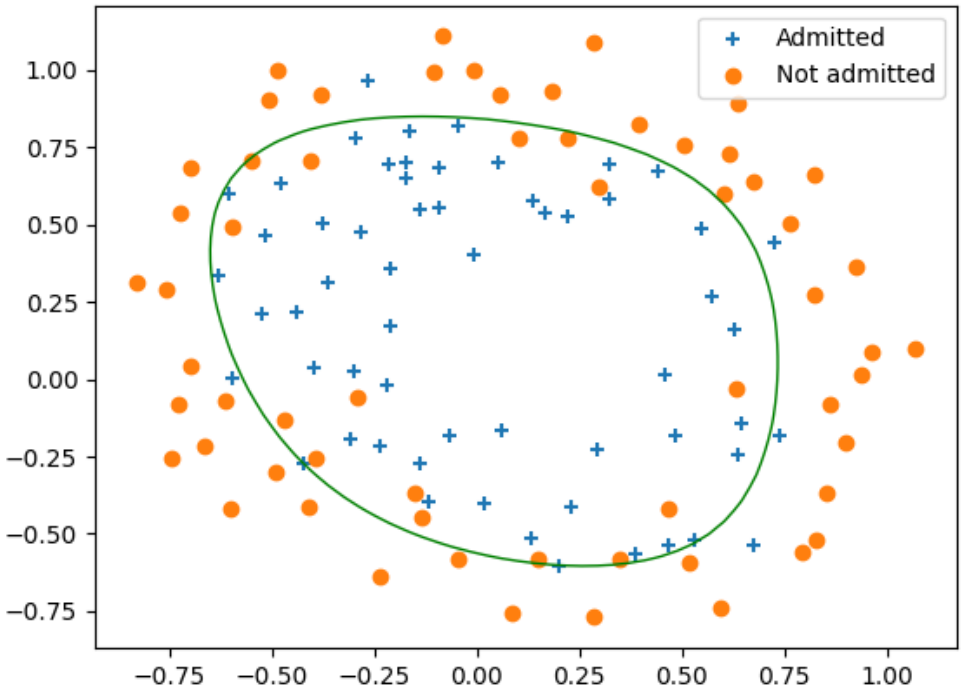

Figure 2: Gráfica que muestra la barrera que separa los microchips que pasan los tests de los que no.

2.2 Implementación

```
import numpy as np
import matplotlib.pyplot as plt
from pandas.io.parsers import read_csv
import scipy.optimize as opt
import scipy as scpy
import sklearn.preprocessing as sk

def loadCSV(fileName):
    return read_csv(fileName, header=None).to_numpy().astype(float)

def sigmoide(z):
    return 1 / (1 + np.exp(-z))

def coste(thetas, x, y):
    h = sigmoide(np.dot(x, thetas))
    return -1/len(x) *(np.dot(np.log(h), y) + np.dot(np.log(1-h), 1-y))

def gradiente(thetas, x, y):
    h = sigmoide(np.dot(x, thetas))
    return np.dot(x.T, h-y) / len(x)

def pinta_frontera_lineal(theta, x, y):
    plt.figure()

    zoom = 5

    x1min, x1max = x[:,0].min(),x[:,0].max()
    x2min, x2max = x[:,1].min(),x[:,1].max()

    xx1,xx2 = np.meshgrid(np.linspace(x1min,x1max),np.linspace(x2min,x2max))

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

    print(h)
    h = h.reshape(xx1.shape)

    # Obtiene un vector con los indices de los ejemplos positivos
    pos = np.where(y==1)
    # Dibuja los ejemplos positivos
    plt.axis([x1min-zoom, x1max+zoom, x2min-zoom, x2max+zoom])
    plt.scatter(x[pos,0], x[pos, 1], marker='+', label="Admitted")
    pos = np.where(y==0)
    plt.scatter(x[pos,0], x[pos, 1], label="Not admitted")

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

    plt.legend(loc="upper right")
    plt.show()

```

```
1049 def parte1():
1050     data = loadCSV("Data/ex2data1.csv")
1051 
1052     x = data[:, :-1]
1053     y = data[:, -1]
1054 
1055     m = np.shape(x)[0]
1056     n = np.shape(x)[1]
1057 
1058     xNew = np.hstack([np.ones([m, 1]), x])
1059 
1060     thetas = np.zeros(n+1)
1061     # print(coste(thetas, xNew, y))
1062     # print(gradiente(thetas, xNew, y))
1063 
1064     result = opt.fmin_tnc(func=coste, x0=thetas, fprime=gradiente, args=(
         xNew,y))
1065     theta_opt = result[0]
1066     print(coste(theta_opt,xNew,y))
1067 
1068     # x1min = np.min(x[:,0])
1069     # x1max = np.max(x[:,0])
1070     # x2min = -(theta_opt[0] + (x1min * theta_opt[1])) / theta_opt[2]
1071     # x2max = -(theta_opt[0] + (x1max * theta_opt[1])) / theta_opt[2]
1072 
1073     # plt.figure()
1074     # # Obtiene un vector con los indices de los ejemplos positivos
1075     # pos = np.where(y==1)
1076     # # Dibuja los ejemplos positivos
1077     # plt.scatter(x[pos,0], x[pos, 1], marker='+', label="Admitted")
1078     # pos = np.where(y==0)
1079     # plt.scatter(x[pos,0], x[pos, 1], label="Not admitted")
1080     # plt.plot([x1min,x2min], [x1max, x2max])
1081     # plt.legend(loc="upper right")
1082     # plt.show()
1083 
1084     pinta_frontera_lineal(theta_opt, x, y)
1085 
1086 """
1087     ==========================================================
1088     ==========================================================
1089     ==========================================================
1090     ==========================================================
1091 """
1092 def costeReg(thetas, x, y, l):
1093     h = sigmoide(np.dot(x, thetas))
1094     return -((np.dot(np.log(h), y) + np.dot(np.log(1 - h), 1 - y)) / len(x)
         ) + (l/(2*len(x))) * l * np.sum(thetas[1:] ** 2)
1095 
1096 def gradienteReg(thetas, x, y, l):
1097     h = sigmoide(np.dot(x, thetas))
1098     return np.dot(x.T, h-y) / len(y) + (thetas * l) / len(y)
1099 
```

```
1100 def pinta_frontera_poli(thetas, x, y, poly):
1101     plt.figure()
1102 
1103     zoom = 0.1
1104 
1105     x1min, x1max = x[:, 0].min(), x[:, 0].max()
1106     x2min, x2max = x[:, 1].min(), x[:, 1].max()
1107     xx1, xx2 = np.meshgrid(np.linspace(x1min, x1max), np.linspace(x2min,
         x2max))
1108 
1109     h = sigmoide(poly.fit_transform(np.c_[xx1.ravel(), xx2.ravel()]).dot(
         thetas))
1110 
1111     h = h.reshape(xx1.shape)
1112 
1113     plt.contour(xx1, xx2, h, [0.5], linewidths=1, colors='g')
1114 
1115     # Obtiene un vector con los indices de los ejemplos positivos
1116     pos = np.where(y==1)
1117     # Dibuja los ejemplos positivos
1118     plt.scatter(x[pos,0], x[pos, 1], marker='+', label="Admitted")
1119     pos = np.where(y==0)
1120     plt.scatter(x[pos,0], x[pos, 1], label="Not admitted")
1121     plt.legend(loc="upper right")
1122     plt.axis([x1min-zoom, x1max+zoom, x2min-zoom, x2max+zoom])
1123     plt.show()
1124 
1125 def parte2():
1126     data = loadCSV("Data\ex2data2.csv")
1127     x = data[:, :-1]
1128     y = data[:, -1]
1129 
1130     l = 1.0
1131 
1132     polynomial = sk.PolynomialFeatures(6)
1133     xNew = polynomial.fit_transform(x)
1134 
1135     n = np.shape(xNew)[1]
1136     thetas = np.zeros(n)
1137 
1138     print(costeReg(thetas, xNew, y, l))
1139     print(gradienteReg(thetas, xNew, y, l))
1140 
1141     result = opt.fmin_tnc(func=costeReg, x0=thetas, fprime=gradienteReg,
         args=(xNew,y,l))
1142     theta_opt = result[0]
1143     print(coste(theta_opt,xNew,y))
1144 
1145     pinta_frontera_poli(theta_opt, x, y, polynomial)
1146 
1147 if __name__ == "__main__":
1148     parte1()
1149     parte2()
```

main.py