机器学习第八次作业

白迎辰

2018201670

心脏病数据集第一部分

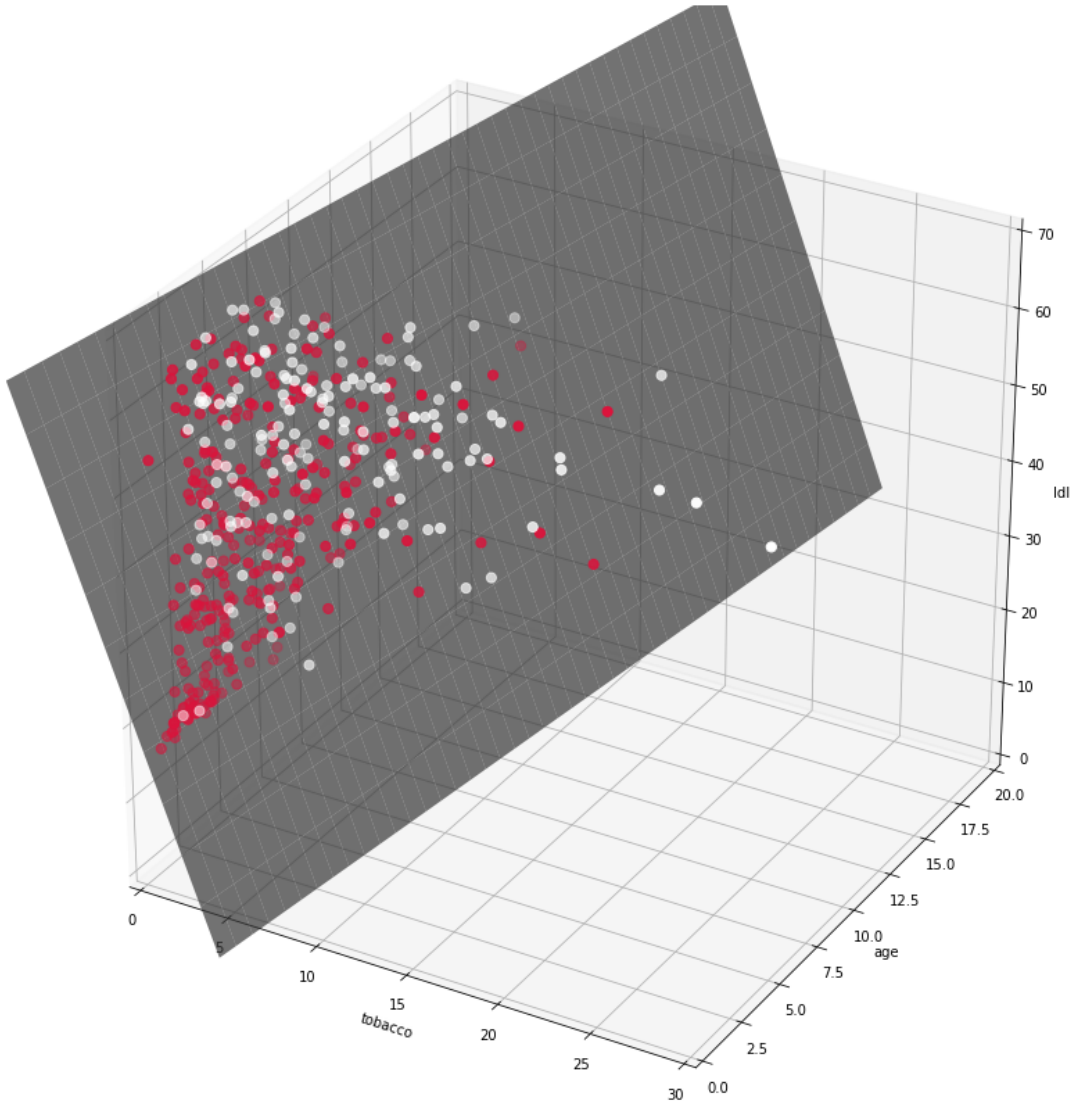
```
# 模块调用声明
from numpy import *
import pandas as pd
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d
```

```
# 读取数据
heart = pd.read_csv('C:/Users/mi/Desktop/mydata.csv')

y = heart['chd']

X = heart[['tobacco','ldl','age']]

X['intersect'] = pd.DataFrame([1]*X.shape[0])

# 查看前五行
heart.head()
```

	sbp	tobacco	ldl	adiposity	...	obesity	alcohol	age	chd
0	160	12.00	5.73	23.11	...	25.30	97.20	52	1
1	144	0.01	4.41	28.61	...	28.87	2.06	63	1
2	118	0.08	3.48	32.28	...	29.14	3.81	46	0
3	170	7.50	6.41	38.03	...	31.99	24.26	58	1
4	134	13.60	3.50	27.78	...	25.99	57.34	49	1

```
# 函数定义
def sigmoid(X):
    return 1.0/(1 + exp(-X))

#梯度下降算法
def Gradient_Descend(X, y):
    X_train = mat(X)
    y_train = mat(y).transpose()
    row, col = X.shape
    alpha = 0.001
    times = 500
    weights = ones((col, 1))
    for i in range(times):
        y_pred = sigmoid(X_train*weights)
        error = (y_train - y_pred)
        weights = weights + alpha * X_train.transpose()*error
    return weights

#绘制决策边界
```

```
def plotBorder(X, y, weight):
    weights = weight.getA()
    dataArr = array(X)
    n = dataArr.shape[0]
    xcord1 = []; ycord1 = [];zcord1 = []
    xcord0 = []; ycord0 = [];zcord0 = []
    for i in range(n):
        if int(y[i]) == 1:
            xcord1.append(dataArr[i, 0])
            ycord1.append(dataArr[i, 1])
            zcord1.append(dataArr[i, 2])
        else:
            xcord0.append(dataArr[i, 0])
            ycord0.append(dataArr[i, 1])
            zcord0.append(dataArr[i, 2])
    fig = plt.figure(figsize=(15, 15))

    ax = fig.add_subplot(111, projection = '3d')
    ax.scatter( xcord1, ycord1, zcord1, zdir = 'z' ,s = 50, c = 'white', marker
= 'o')
    ax.scatter( xcord0, ycord0, zcord0,  zdir = 'z',s = 50, c = 'crimson',
marker = 'o')



    X = arange(-9, 3, 1)
    Y = arange(2, 50, 1)
    X,Y = meshgrid(X, Y)  # 将坐标向量变为坐标矩阵，列为x的长度，行为y的长度
    Z = (-weights[3]-weights[0]*X - weights[1]*Y)/weights[2] + weights[3]/2
    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, linewidth=0,
antialiased=True, alpha = 0.7,color = 'grey')
    ax.set_xlim(0, 30)
    ax.set_ylim(0, 20)
    ax.set_zlim(0, 70)
    ax.set_xlabel('tobacco')
    ax.set_ylabel('age')
    ax.set_zlabel('ldl')
    plt.show()

# 梯度下降获得权重
w =  Gradient_Descend(X, y)
# 绘制决策边界
plotBorder(X, y, w)
```



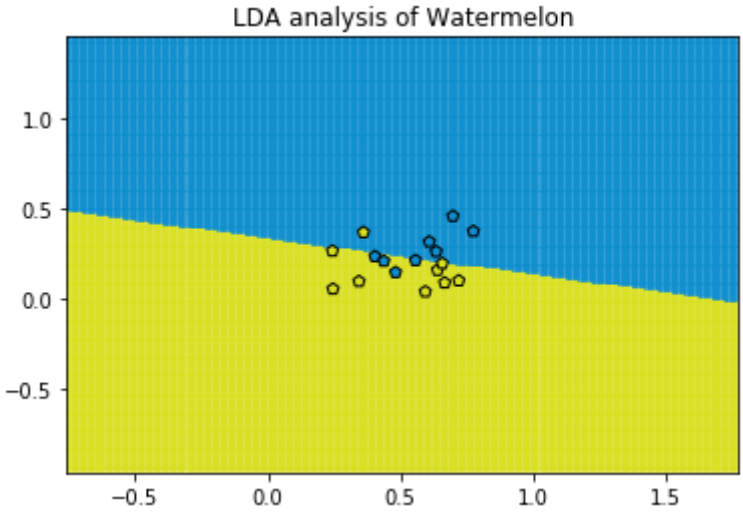
决策面前后的数据点有较为明显的颜色变化，并且两种颜色都有这种现象，说明决策面选择比较合理。

西瓜数据集

```
# 引入新的模块
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from matplotlib.colors import ListedColormap

# 读取数据
watermelon = pd.read_csv('C:/Users/mi/Desktop/watermelon.csv')
# 选择特征
X = array(watermelon[['密度','含糖率']])
Y = array((watermelon['好瓜'] == '是').astype(int))
# 查看数据
watermelon.head()
```

	编号	密度	含糖率	好瓜
0	1	0.697	0.460	是
1	2	0.774	0.376	是
2	3	0.634	0.264	是
3	4	0.608	0.318	是
4	5	0.556	0.215	是

```
#建立LDA模型
classifier = LDA()
classifier.fit(X, Y)

# 确定取值范围
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1

#生成网格矩阵
xx, yy = meshgrid(arange(x_min, x_max, 0.01), arange(y_min, y_max, 0.01))

#拟合LDA模型
Z = classifier.predict(c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

#绘制分类结果
cm_bright = ListedColormap(['#D9E021', '#0D8ECF'])
plt.pcolormesh(xx, yy, Z, cmap=cm_bright, alpha=0.6)

plt.scatter(X[:, 0], X[:, 1],  c=Y, cmap=cm_bright, marker = 'p', edgecolors =
'black')

plt.title('LDA analysis of Watermelon')
plt.axis('tight')
plt.show()
```




```
# 输出降维平面参数
print('intercept: %f' %(classifier.intercept_))
print('coefficient %f %f' %(classifier.coef_[0][0], classifier.coef_[0][1]))
```

intercept: -3.692197
coefficient： 2.197647 11.080734

心脏病数据集第二部分

```
# 分割数据集
healthy = heart[heart['chd'] == 0]
sick = heart[heart['chd'] == 1]

healthy_X = healthy[['tobacco', 'ldl', 'age']]
healthy_y = healthy['chd']

sick_X = sick[['tobacco', 'ldl', 'age']]
sick_y = sick['chd']
```

```
#计算特征均值
healthy_X_mean = array(mean(healthy_X))
sick_X_mean = array(mean(sick_X))

healthy_X = mat(healthy_X)
sick_X = mat(sick_X)

#计算类间散度矩阵
Sw0 = (healthy_X - healthy_X_mean).transpose()*((healthy_X - healthy_X_mean))
Sw1 = (sick_X - sick_X_mean).transpose()*((sick_X - sick_X_mean))

w = ((Sw0 + Sw1).I)*mat(healthy_X_mean - sick_X_mean).T

#将w 归一化
w_standard = w/sqrt(w.T*w)
```

[-0.407701, -0.89122, -0.198764]

$$b^* = -\frac{\max w^T X_1 + \min w^T X_0}{2}$$

```
# 计算截距项的系数估计
b = -(max(sick_X*w_standard)+ min(healthy_X*w_standard))*(1/2)
```

15.98428696

```
#计算新的超平面截距
w_1 = mat([0.61, -0.45, 0.65]).T
b_1 = -(max(sick_X*w_1)+ min(healthy_X*w_1))*(1/2)
#-31.3755
```

```
#比较两个超平面划分数据的能力，根据均值与wT判断health_X分布在界面下方，将标记改为-1
y_neo = y.replace(0, -1)

# 计算预测结果与类别标签同号数据所占比例
result1 = array(mat(X)*w_standard + b).T*(array(mat(y_neo)))
result11 = list(result1[0])
count1 = 0

for i in range(len(result11)):
    if result11[i] > 0:
        count1 += 1
```

```
score1 = count1/len(result11)
#0.29653679653679654

result2 = array(mat(X)*w_1 + b_1).T*(array(mat(y_neo)))
result22 = list(result2[0])
count2 = 0

for i in range(len(result22)):
    if result22[i] > 0:
        count2 += 1

score2 = count2/len(result22)
#0.6688311688311688
```

score1: 0.29653679653679654

score2: 0.6688311688311688

因此后一种权重决定的决策面分类效果更好