**作业8.3**

1. **实验要求**

自己编程实现一个AdaBoost,以不剪枝决策树为基学习器，在西瓜数据集3.0α上训练一个AdaBoost集成。

1. **实验原理：**

Adaboost是以过去的学习为基础，通过加强对原来学习中出现错误概率较大的数据加强提取，改变样本的分布，从而提高其获取的概率，获得新的基学习器，通过不断这样的迭代，再将这些基学习器组合起来，进行分类。

1. **实验过程和代码：**

对于本题，直接从sklearn机器学习包里调用决策树和AdaBoostClassifier进行训练，得到划分结果，并进行表达，过程如下：

1、 读取数据并进行处理

2、 设定基学习器的数目，对数据和标签进行训练。

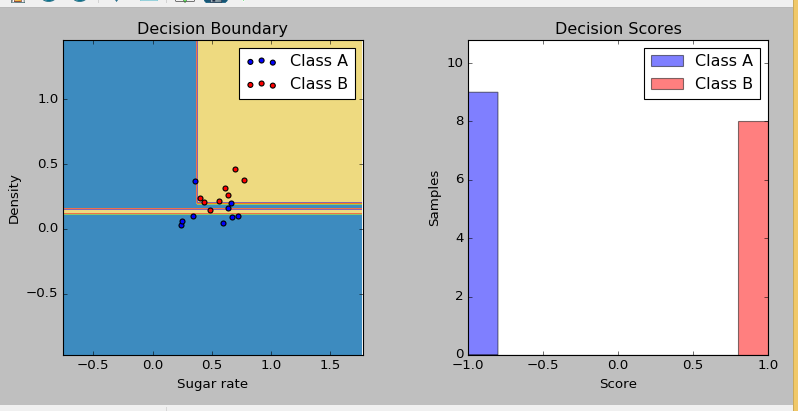
3、 将得出的决策结果画图显示出来。

代码如下：

import numpy as np  
import matplotlib.pyplot as plt  
  
from sklearn.ensemble import AdaBoostClassifier  
from sklearn.tree import DecisionTreeClassifier  
  
file1 = open('w3.csv','r')  
data = [line.strip('\n').split(',') for line in file1]  
data = np.array(data)  
#X = [[float(raw[-7]),float(raw[-6]),float(raw[-5]),float(raw[-4]),float(raw[-3]), float(raw[-2])] for raw in data[1:,1:-1]]  
  
X = [[float(raw[-3]), float(raw[-2])] for raw in data[1:]]  
y = [1 if raw[-1]=='1' else 0 for raw in data[1:]]  
X = np.array(X) #X表示两种属性  
y = np.array(y) #Y表示两种标记  
  
  
# Create and fit an AdaBoosted decision tree,不剪枝决策树  
bdt = AdaBoostClassifier(DecisionTreeClassifier(),  
 algorithm="SAMME",  
 n\_estimators=1) #调用决策树adaboost模块  
  
bdt.fit(X, y) #训练  
  
plot\_colors = "br"  
plot\_step = 0.02  
class\_names = "AB"  
  
plt.figure(figsize=(10, 5))  
  
# Plot the decision boundaries  
plt.subplot(121)  
x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1  
y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  
xx, yy = np.meshgrid(np.arange(x\_min, x\_max, plot\_step),  
 np.arange(y\_min, y\_max, plot\_step))  
  
Z = bdt.predict(np.c\_[xx.ravel(), yy.ravel()])  
Z = Z.reshape(xx.shape)  
cs = plt.contourf(xx, yy, Z, cmap=plt.cm.Paired) #画出边界  
plt.axis("tight")  
  
# Plot the training points  
for i, n, c in zip(range(2), class\_names, plot\_colors):  
 idx = np.where(y == i)  
 plt.scatter(X[idx, 0], X[idx, 1],  
 c=c, cmap=plt.cm.Paired,  
 label="Class %s" % n)  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.legend(loc='upper right')  
plt.xlabel('Sugar rate')  
plt.ylabel('Density')  
plt.title('Decision Boundary')  
  
# Plot the two-class decision scores  
twoclass\_output = bdt.decision\_function(X) #预测输出值  
plot\_range = (twoclass\_output.min(), twoclass\_output.max())  
plt.subplot(122)  
for i, n, c in zip(range(2), class\_names, plot\_colors):  
 plt.hist(twoclass\_output[y == i],  
 bins=10,  
 range=plot\_range,  
 facecolor=c,  
 label='Class %s' % n,  
 alpha=.5)  
x1, x2, y1, y2 = plt.axis()  
plt.axis((x1, x2, y1, y2 \* 1.2))  
plt.legend(loc='upper right')  
plt.ylabel('Samples')  
plt.xlabel('Score')  
plt.title('Decision Scores')  
  
plt.tight\_layout()  
plt.subplots\_adjust(wspace=0.35)  
plt.show()

1. **实验结果**

设定决策树深度为10，则结果如下：



可以看出分类得到良好的效果。