作业说明

SVM

使用 python 语言构建一个 SVM 的分类器，对指定数据集进行分类，必须有可视化的分析。根据指定的数据，计算对应的支持向量和超平面。

要求输出支持向量

要求画出超平面的示意图

数据生成与说明

```
import numpy as np

# 生成SVM数据
np.random.seed(0)
X = np.r_[np.random.randn(20, 2) - [2, 2],
          np.random.randn(20, 2) + [2, 2]]
Y = [-1] * 20 + [1] * 20
```

数据X中每一行代表一个二维向量作为输入

标签Y的每一行代表标签，正负样本标签分别对应1，-1

具体实现

1. 生成SVM数据，(40, 2)的X，及其对应标签Y，一半-1，一半1
2. 创建SVM模型，并喂入X，Y
3. 画出超平面
4. 打印支持向量

代码如下，每一步均有注释

```
import numpy as np
from sklearn import svm
import matplotlib.pyplot as plt

# 生成SVM数据
np.random.seed(0)

# np.r_按row来组合array，np.c_按column来组合array
# X是一个40行2列的矩阵，前20行是均值为-2的二维正态分布，后20行是均值为2的二维正态分布
X = np.r_[np.random.randn(20, 2) - [2, 2],
          np.random.randn(20, 2) + [2, 2]]
Y = [-1] * 20 + [1] * 20

# 创建SVM模型
clf = svm.SVC(kernel='linear')  # 创建线性SVM分类器
```

```
clf.fit(X, Y)  # 拟合模型



# 画出超平面
x_min = X[:, 0].min() - 2
x_max = X[:, 0].max() + 2
y_min = X[:, 1].min() - 2
y_max = X[:, 1].max() + 2

# 生成网格点矩阵，以0.02为步长
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))

print("xx:")
print(xx.shape)
print(xx)

print("yy:")
print(yy.shape)
print(yy)

# xx.ravel()将xx变成一维数组
# np.c_按column来组合array，组合后的结果是将xx和yy中的元素一一对应组合成一个二维数组
print("xx.ravel:")
print(xx.ravel().shape)
print(xx.ravel())

print("np.c_[xx.ravel(), yy.ravel()]:")
print(np.c_[xx.ravel(), yy.ravel()].shape)
print(np.c_[xx.ravel(), yy.ravel()])


Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])  # 计算每个点到超平面的距离

print("Z1:")
print(Z.shape)
print(Z)

# 这一步是为了将Z的维度转换成xx的维度
# 意思就是比如Z是一个一维数组，xx是一个二维数组，那么Z就会被转换成二维数组
Z = Z.reshape(xx.shape)

print("Z2:")
print(Z.shape)
print(Z)

# 绘制等高线，用以可视化超平面
# contour用于绘制等高线
plt.contour(xx, yy, Z, colors=['k', 'k', 'k'], linestyles=['--', '-', '--'], levels=[-1, 0, 1])

# 绘制数据点
# 手动指定颜色，例如，蓝色和红色
colors = ['b' if label == -1 else 'r' for label in Y]
```

```
plt.scatter(X[:, 0], X[:, 1], c=colors, marker='o')

# 绘制支持向量
# 圈出支持向量，即在图上圈出支持向量点
support_vectors = clf.support_vectors_
plt.scatter(support_vectors[:, 0], support_vectors[:, 1], s=100,
facecolor='None', edgecolors='k')

plt.title('SVM Classifier')  # 设置图的标题
plt.savefig('./SVM.png')    # 保存图
plt.show()  # 显示图

# 输出支持向量
print("Support Vectors:")
print(support_vectors)
```

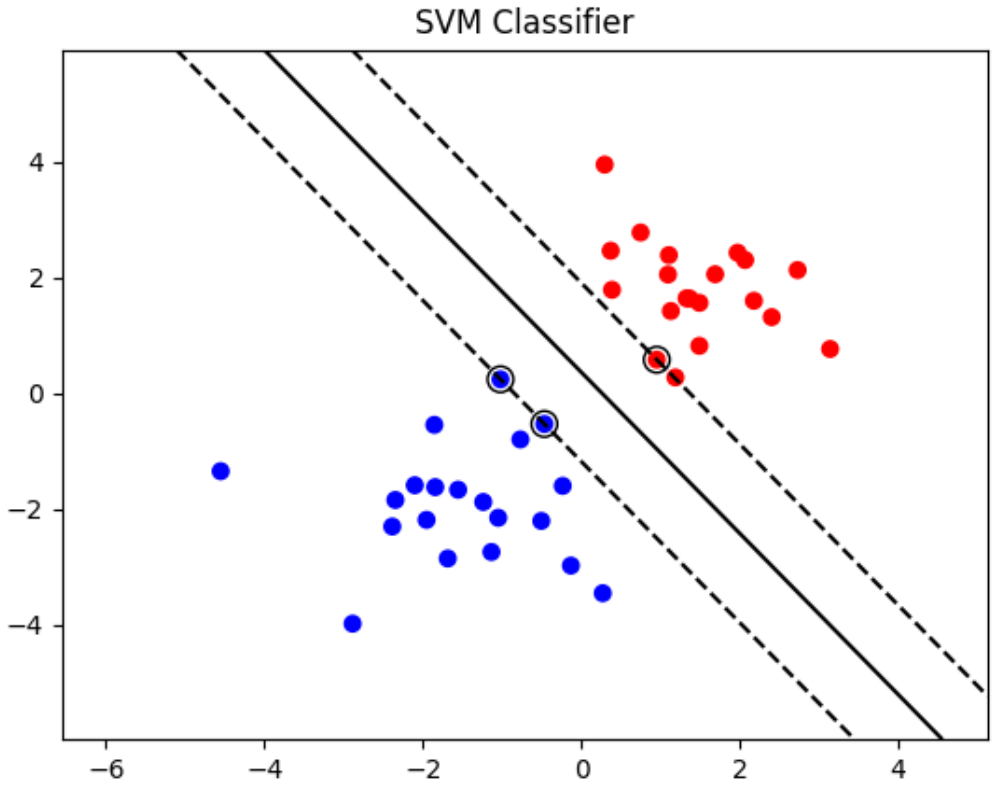
结果

支持向量：

```
Support Vectors:
[[-1.02126202  0.2408932 ]
 [-0.46722079 -0.53064123]
 [ 0.95144703  0.57998206]]

Process finished with exit code 0
```

超平面示意图：



实线是主超平面，虚线是两条次要的超平面，标明了分类边界，圈出的三个点是支持向量