机器学习第六章作业

任齐轩
重庆大学-计卓 2 班-20204154

日期：October 17, 2022

1 第一题

证明. 假设点 x 在超平面 (ω, b) 上的投影为 x'，则有 $\omega x' + b = 0$.

$$d = |\vec{xx'}| = |\frac{\omega}{||\omega||}||\vec{xx'}| \tag{1}$$

由于 $\vec{xx'}$ 与超平面法向量平行，所以 $\cos\theta = 1$ ，即

$$d = \frac{\omega}{||\omega||} \cdot (x - x') = \frac{|\omega \cdot (x - x')|}{||\omega||} = \frac{\omega \cdot x - \omega \cdot x'}{||\omega||} \tag{2}$$

由于 x' 在超平面上，即有 $\omega x' + b = 0$ 所以 $\omega \cdot x' = -b$，即

$$d = \frac{\omega \cdot x + b}{||\omega||} \tag{3}$$

因此，式 (6.2) 成立。 □

2 第二题

(代码见附录) 如图所示，为线性核和高斯核的 SVM，可以看出由于西瓜数据集不是线性可分的，用线性核的 SVM 无法做到 100% 的准确率；而高斯核的 SVM 由于通过将二维数据集升到高维空间，使这些数据在高维空间中线性可分，所以可以做到 100% 的准确率。但同时，这也会导致过拟合的问题。

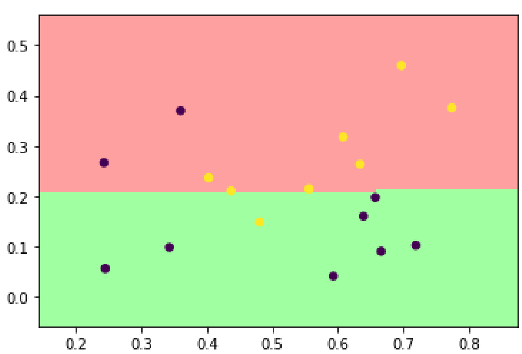

图 1: Accuracy = 82.3529% (14/17)

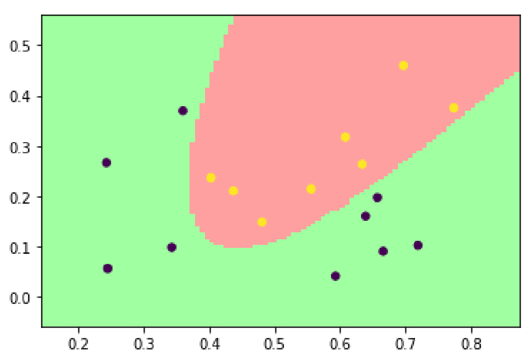

图 2: Accuracy = 100% (17/17)

3 第四题

当 LDA 所求得投影方向 ω 与线性支持向量机所求得投影方向 ω' 相垂直时，二者等价

4 第五题

若将隐层神经元数设置为训练样本数，且每个训练样本对应一个神经元中心，则以高斯径向基函数为激活函数的 RBF 网络恰与高斯核 SVM 的预测函数相同。（书本 P145）

5 第六题

因为 SVM 的结果只与支持向量有关，其余大部分数据都对结果不会造成影响。若支持向量出现噪声，则将直接影响最终都结果。而 LDA 等其他方法的结果与所有数据都有关，因此受噪声影响较小，对噪声不敏感。

A 第二题代码

```
from libsvm.svmutil import *
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np


def load_dataset():
    file = open(r'melon3_dataset.txt')
    dataMat = []
    labelMat = []
    for line in file.readlines():
        lineArr = line.strip().split()
        dataMat.append([float(lineArr[0]), float(lineArr[1])])
        labelMat.append(int(lineArr[2]))
    return dataMat, labelMat


def Lst_To_Scale(dataMat, labelMat):
    with open("./melon.scale", 'w') as f:
        for i in range(len(labelMat)):
        data_class = labelMat[i]
        atr_0, atr_1 = dataMat[i][0], dataMat[i][1]
        line = str(data_class) + " 1:" + str(atr_0) + " 2:" + str(atr_1)
        f.writelines(line + "\n")


dataMat, labelMat = load_dataset()
Lst_To_Scale(dataMat, labelMat)
train_label, train_value = svm_read_problem("./melon.scale")

# 线性核
model = svm_train(train_label, train_value, '-t 0 -c 1000')
p_label, p_acc, p_val = svm_predict(train_label, train_value, model)

# 高斯核
model = svm_train(train_label, train_value, '-t 2 -c 10000')
p_label, p_acc, p_val = svm_predict(train_label, train_value, model)

# 数据可视化
x1 = [mapi[1] for mapi in train_value]
x2 = [mapi[2] for mapi in train_value]
x = np.c_[x1,x2]

np_x = np.asarray(x)
np_y = np.asarray(train_label)
```

```python
N, M = 100, 100

x1_min, x2_min = np_x.min(axis=0)
x1_max, x2_max = np_x.max(axis=0)

x1_min -= 0.1
x2_min -= 0.1
x1_max += 0.1
x2_max += 0.1

t1 = np.linspace(x1_min, x1_max, N)
t2 = np.linspace(x2_min, x2_max, M)

grid_x, grid_y = np.meshgrid(t1,t2)

grid = np.stack([grid_x.flat, grid_y.flat], axis=1)
y_fake = np.zeros((N*M,))
y_predict, _, _ = svm_predict(y_fake, grid, model)

cm_light = mpl.colors.ListedColormap(['#A0FFA0', '#FFA0A0'])
plt.pcolormesh(grid_x, grid_y, np.array(y_predict).reshape(grid_x.shape), cmap=
    cm_light)
plt.scatter(x[:,0], x[:,1], s=30, c=train_label, marker='o')

plt.show()
```