
北京大学

# 机器学习基础
# 课程上机试验报告

试验内容： 支持向量机

姓　　名： 王宇哲

学　　号： 1800011828

# 1 题目 6.2

## 1.1 实验题目

试使用 LIBSVM，在西瓜数据集 3.0$\alpha$ 上分别用线性核和高斯核训练一个 SVM，并比较其支持向量的差别。

西瓜数据集 3.0$\alpha$ 见《机器学习》p.89 的表 4.5。

## 1.2 实验数据

题目 6.2 使用的数据集为周志华《机器学习》p.89 所提供的西瓜数据集 3.0$\alpha$，具体内容如表 **1** 所示。

表 **1** 西瓜数据集 3.0$\alpha$

**Table 1 Watermelon dataset** 3.0$\alpha$

| 编号 | 密度 | 含糖率 | 好瓜 | 编号 | 密度 | 含糖率 | 好瓜 |
|---|---|---|---|---|---|---|---|
| 1 | 0.697 | 0.460 | 是 | 9 | 0.666 | 0.091 | 否 |
| 2 | 0.774 | 0.376 | 是 | 10 | 0.243 | 0.267 | 否 |
| 3 | 0.634 | 0.264 | 是 | 11 | 0.245 | 0.057 | 否 |
| 4 | 0.608 | 0.318 | 是 | 12 | 0.343 | 0.099 | 否 |
| 5 | 0.556 | 0.215 | 是 | 13 | 0.639 | 0.161 | 否 |
| 6 | 0.403 | 0.237 | 是 | 14 | 0.657 | 0.198 | 否 |
| 7 | 0.481 | 0.149 | 是 | 15 | 0.360 | 0.370 | 否 |
| 8 | 0.437 | 0.211 | 是 | 16 | 0.593 | 0.042 | 否 |
| | | | | 17 | 0.719 | 0.103 | 否 |

## 1.3 实验工具

实验在个人 LEVENO YOGA 710 笔记本电脑上进行，主要硬件条件为：处理器 Intel i5-7200U CPU，内存大小 8.0 GB，显卡 NVIDIA Geforce 940MX；操作系统为 Windows 10 64 位系统。

实验使用语言为 python，具体版本为 python 3.7.0，通过 Anaconda 安装了 Matplotlib、NumPy、pandas、scikit-learn 等常用库。实验代码编写和运行均在 Jupyter Notebook 上进行，具体版本为 Jupyter Notebook 6.3.0。

## 1.4 实验方法

实验采用的算法为经典的支持向量机 (SVM) 算法。具体地，分别使用线性核 (linear kernel) 和高斯核 (gaussian kernel) 进行训练，引入超参数 $C$ 作为正则化参数，以权衡对特异

点的总的松弛程度和软间隔，得到软间隔支持向量机。经典 SVM 算法的原理在课程中已经进行详细讨论，此处不再赘述。

考虑到 python sklearn 库已集成了 libsvm 库，实验使用在 libsvm 基础上扩展形成的 sklearn.svm 库进行支持向量机的训练。sklearn.svm 库支持使用核方法训练非线性支持向量机，可选的核包括线性核 (linear kernel)、多项式核 (polynomial kernel)、径向基函数核 (rbf kernel) 等。sklearn.svm 库使用参数 $C$ 进行正则化设置，$C$ 较大时对特异点的松弛幅度惩罚较大，默认 $C = 1.0$。

实验的具体操作方法如下。首先将表 **1** 数据写入 watermelon_dataset.txt，导入实验所需的必要的库，并读取西瓜数据集 $3.0\alpha$ 的数据：

```
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np

with open('watermelon_dataset.txt', 'r') as dataset:
    X=[]
    y=[]
    while True:
        lines = dataset.readline()
        if not lines:
            break
            pass
        ID, density, sugar_content, label = [float(i) for i in
            lines.split()]
        X.append([density, sugar_content])
        y.append(int(label))
        pass
    X = np.array(X)
    y = np.array(y)
    pass
```

得到的 X 为西瓜数据集各样本的密度、含糖率组成的 2 维向量的集合，y 为相应的各样本的类别标记的集合，“是好瓜”标记为 +1，“不是好瓜”标记为-1。

下面使用 sklearn.svm 库进行支持向量机的训练，并使用 matplotlib 库绘制出训练结果。一般地，首先定义函数 svm_train:

```
from sklearn import svm

def svm_train(kernel='linear', C=1, label='label'):
svc = svm.SVC(C=C, kernel=kernel, gamma=10)
svc.fit(X, y)
sv = svc.support_vectors_

plt.figure()
plt.clf()
plt.scatter(X[:, 0], X[:, 1], c=y, zorder=10, cmap=plt.cm.Paired,
   edgecolor='k', s=30)
plt.scatter(sv[:, 0], sv[:, 1], s=70, facecolors='none', zorder=10,
   edgecolor='k')

plt.axis('tight')
x_min = X[:, 0].min()
x_max = X[:, 0].max()
y_min = X[:, 1].min()
y_max = X[:, 1].max()

XX, YY = np.mgrid[x_min-0.2:x_max+0.2:200j, y_min-0.4:y_max+0.4:200j]
Z = svc.decision_function(np.c_[XX.ravel(), YY.ravel()])

Z = Z.reshape(XX.shape)
plt.pcolormesh(XX, YY, Z > 0, cmap=plt.cm.Paired)
plt.contour(XX, YY, Z, colors=['k', 'k', 'k'], linestyles=['--',
   '-', '--'], levels=[-.5, 0, .5])

plt.title(label)
```

通过 svm_train 函数实现支持向量机的训练和训练结果图的绘制，函数变量 kernel 指定所用的核，$C$ 为正则化参数，label 为绘制的训练结果图的图题。训练结果图中标注了支持向量 (双线空心圈)、未作为支持向量的训练样本 (单线空心圈)、训练得到的分离超平面 (黑色实

线) 和间隔边界 (黑色虚线)，并用不同颜色标注出正例点和负例点。

最后分别用线性核和高斯核训练支持向量机，作出并保存训练结果图：

```
svm_train(kernel='linear', C=1000, label='linear')
plt.savefig('linear_result.jpg',dpi=1000, bbox_inches='tight')
svm_train(kernel='rbf', C=1000, label='gaussian')
plt.savefig('gaussian_result.jpg',dpi=1000, bbox_inches='tight')

plt.show()
```

正则化参数 $C = 1000$ 通过实验过程中的多次尝试进行选取。在尝试增大 $C$ 的过程中，$C = 1000$ 时两支持向量机的训练结果与使用的支持向量个数均基本达到稳定，因此可以认为是较合理的正则化参数，此时两 SVM 均基本达到最优训练效果。

### 1.5 实验结果

实验代码总运行时长为 2.17 s。使用线性核训练得到的支持向量机的训练结果如图 **1** 所示。

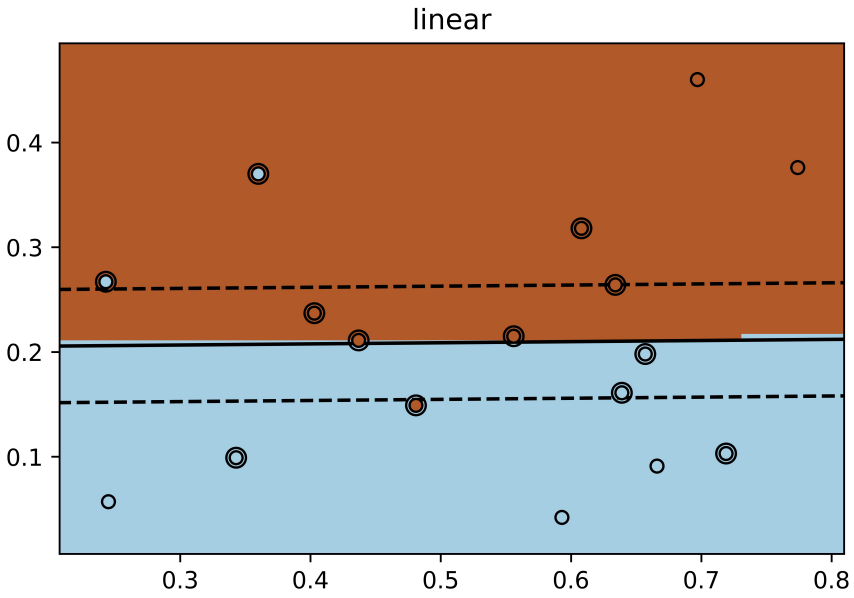


图 **1** 线性核支持向量机训练结果

**Fig. 1 Linear kernel SVM training result**

根据图 **1**，可以看出该线性核支持向量机使用了 12 个支持向量。在所有样本点中，正确分类且落在间隔边界外 8 个，正确分类但落在间隔边界内 6 个，错误分类 3 个。

使用高斯核训练得到的支持向量机的训练结果如图 **2** 所示。

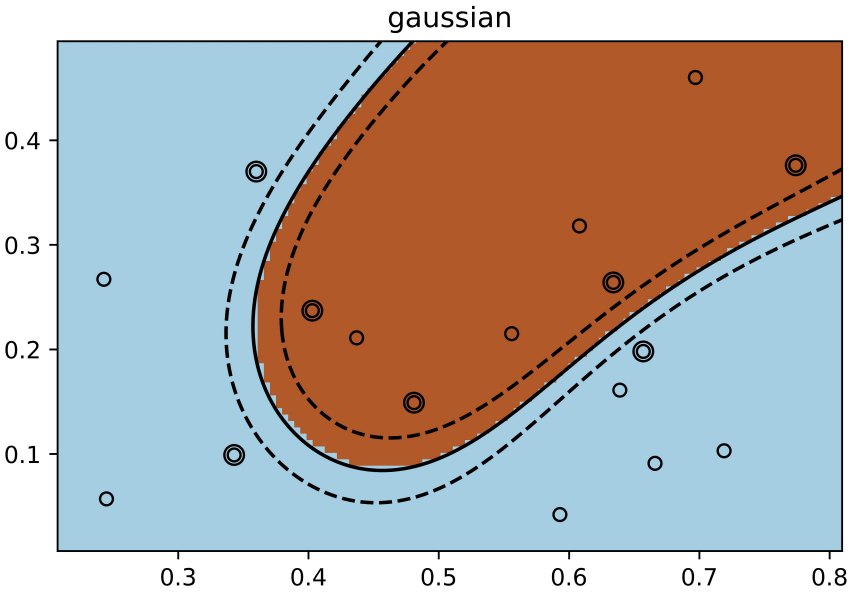


图 **2** 高斯核支持向量机训练结果

**Fig. 2 Gaussian kernel SVM training result**

根据图 **2**，可以看出该线性核支持向量机使用了 7 个支持向量，全部样本点均正确分类且落在间隔边界外。对比图 **1** 和图 **2** 可以发现，在正则化参数 $C$ 相同时，训练得到的高斯核支持向量机的间隔明显小于线性核支持向量机。

## 1.6 结果分析

根据 1.5 的实验结果，可以分析认定：对于本题所用的西瓜数据集 3.0$\alpha$，高斯核支持向量机相比线性核支持向量机的间隔更小，因此拟合更好、具有更优的表现；且高斯核支持向量机所用的支持向量数 (7 个) 小于线性核支持向量机 (12 个)，因此具有更快的分类速度和更高的效率。

# 2 题目 6.10

## 2.1 实验题目

试设计一个能显著减少 SVM 中支持向量的数目而不显著降低泛化性能的方法。

## 2.2 实验数据

题目 6.10 使用的数据集为 scikit-learn 库集成的 breast_cancer 数据集，该数据集包含了美国 Wisconsin 州记录的 569 个乳腺癌病人的恶性/良性数据 (以 1/0 表示)，以及与之对应的 30 维生理指标数据，适用于一般的二分类问题。

## 2.3 实验工具

同 1.3，不再赘述。

## 2.4 实验方法

实验采用 1-norm 支持向量机算法 (1-norm Support Vector Machine) 实现显著减少 SVM 中支持向量的数目而不显著降低泛化性能的要求。1-norm SVM 是一种稀疏算法 (Sparse Algorithm)，能够在不影响模型性能的情况下减少支持向量数目，从而有效提高 SVM 效率[1]。具体地，1-norm SVM 具有如下的形式[2]：

$$\min_{\beta_0,\beta} \sum_{i=1}^{n} \Big[1 - y_i\Big(\beta_0 + \sum_{j=1}^{q} \beta_j h_j(x_i)\Big)\Big]$$

$$\text{s.t. } ||\beta||_1 = |\beta_1| + \cdots + |\beta_q| \le s$$

其中 $\mathcal{D} = h_1(x), \cdots h_q(x)$ 是一组基函数的集合，$s$ 是调节参数。得到的模型可以表示为

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{j=1}^{q} \hat{\beta}_j h_j(x)$$

对应的分类规则由 $sign[\hat{f}(x)]$ 给出。相比一般的 2-norm SVM，1-norm SVM 使用 L1 范数构建惩罚项，能够减少支持向量的数目，有效提高 SVM 的分类速度。

1-norm SVM 的训练可以使用 sklearn.LinearSVC 库实现，通过指定 penalty 为 l2 或 l1，可以在一般的 2-norm SVM 和 1-norm SVM 间进行切换。实验的具体操作方法如下。首先导入实验所需的必要的库，并读取数据集 breast_cancer 的数据；随后定义主体函数 LinearSVC_train，实现对数据集的随机划分、使用划分出的训练集训练 SVM、使用测试集对 SVM 的泛化能力进行检验的功能，并计算 SVM 所用到的支持向量的个数：

```python
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np

from sklearn import datasets, svm, metrics

cancer = datasets.load_breast_cancer()
X = cancer.data
y = cancer.target

def LinearSVC_train(X, y, penalty='l2', C=1000):
n_sample = len(X)
np.random.seed()
order = np.random.permutation(n_sample)
X = X[order]
y = y[order].astype(float)

X_train = X[:int(.9 * n_sample)]
y_train = y[:int(.9 * n_sample)]
X_test = X[int(.9 * n_sample):]
y_test = y[int(.9 * n_sample):]

svc = svm.LinearSVC(C=C, penalty=penalty, dual=False)
svc.fit(X_train, y_train)
decision_function = svc.decision_function(X_train)
support_vector_indices = np.where(np.abs(decision_function) <= 1 +
   1e-15)[0]
support_vector = X_train[support_vector_indices]

y_pred = svc.predict(X_test)
accuracy = metrics.accuracy_score(y_test, y_pred)

return len(support_vector), accuracy
```

与 1.4 类似地，正则化参数 $C$ 通过实验过程中的多次尝试进行选取，$C = 1000$ 基本是较合理的正则化参数。

随后分别训练一般的 l2-norm SVM 和 l1-norm SVM，重复 100 次，输出两 SVM 所用到的支持向量个数和预测准确率的平均值：

```
for penalty in ['l2', 'l1']:
    sv_list = []
    accuracy_list = []

    for i in range(0, 100):
        sv, ac = LinearSVC_train(X, y, penalty=penalty)
        sv_list.append(sv)
        accuracy_list.append(ac)

    print('average number of support vectors ' + '(' + str(penalty) +
        '-form): ' + str("%.2f" % np.mean(sv_list)))
    print('average accuracy of linear kernel SVM ' + '(' +
        str(penalty) + '-form): ' + str("%.4f" %
        np.mean(accuracy_list)))
```

## 2.5 实验结果

实验代码总运行时长为 8.96 s。一般的 l2-norm SVM 和 l1-norm SVM 的平均支持向量数与平均预测准确率的对比如表 **2** 所示。

表 **2** **l2-norm SVM** 与 **l1-norm SVM** 对比

**Table 2 Comparison of l2-norm SVM and l1-norm SVM**

| SVM | Average Support Vector Number | Average Accuracy |
|---|---|---|
| l2-norm | 83.59 | 0.9588 |
| l1-norm | 50.35 | 0.9723 |

根据表 **2**，可以看出：在 breast_cancer 数据集上训练得到的一般的 l2-norm SVM 所使用的支持向量数的平均值 (83.59) 明显高于 l1-norm SVM(50.35)，而在测试集上的平均预测准确率 (0.9588) 略低于 l1-norm SVM(0.9723)。

### 2.6 结果分析

根据 2.5 的实验结果，可以分析认定：对于本题所用的 breast_cancer 数据集，改进后的 l1-norm SVM 相比一般的 l2-norm SVM 显著减少了所用支持向量的数目，因此大大提高了分类速度和效率；同时，l1-norm SVM 的泛化性能并未降低，在 breast_cancer 数据集上的最优表现甚至高于一般的 l2-norm SVM。

综合以上结果，可以认为 l1-norm SVM 算法很好地实现了题目中“显著减少 SVM 中支持向量数目而不显著降低泛化性能”的要求，具有一定程度的应用价值。

## 参考文献

[1] H. G. Jung and G. Kim, “Support vector number reduction: Survey and experimental evaluations,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 463–476, 2013.

[2] J. Zhu, S. Rosset, R. Tibshirani, and T. J. Hastie, “1-norm support vector machines,” in *Advances in neural information processing systems*, Citeseer, 2003.