山东大学 计算机科学与技术 学院

机器学习和模式识别 课程实验报告

学号：201700301042	姓名： 陈佳睿	班级： 17 智能
实验题目：支持向量机分类器		
实验学时：4 学时	实验日期： 2019.12.5	

实验目的：

学习使用 SVM 分类器来进行对数据的线性分类

使用的是 LIBSVM 的 Matlab 函数接口

硬件环境：

Macbook pro2017 A1706

8G 内存

软件环境：

MatlabR2017a

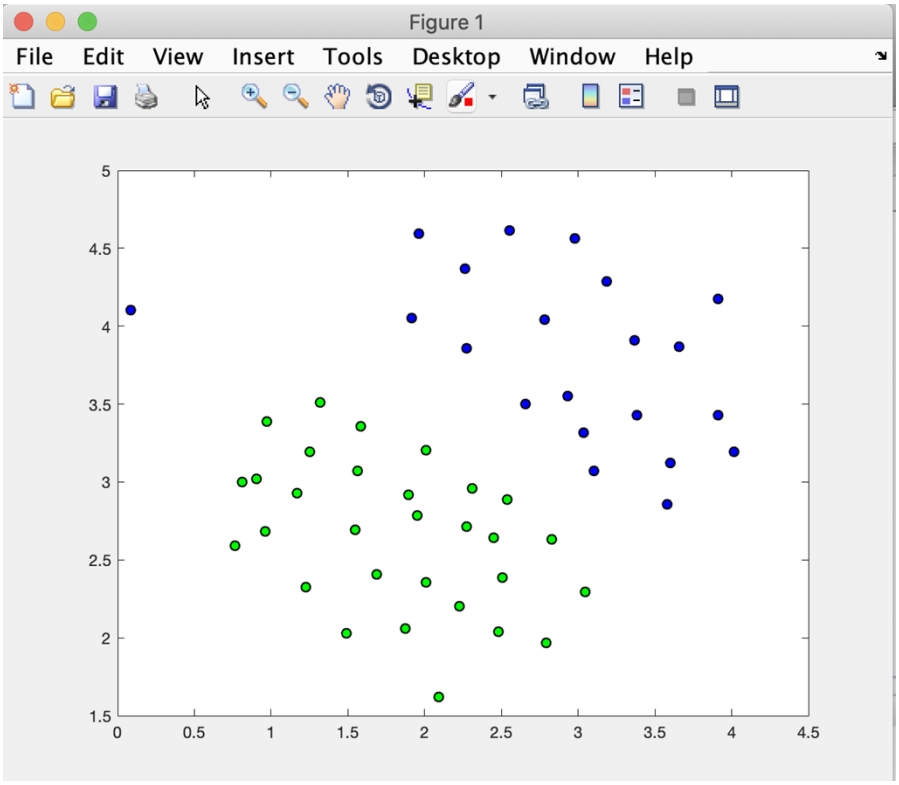
实验步骤与内容：

1. 首先使用 libsvmread 函数读入实验所提供的数据集文件 twofeature.txt

得到函数返回 trainlabels 和 trainfeatures

其中 trainlabels 包含了对于训练数据的分类 label

trainfeatures 对于每一个训练样本是一个包含两个特征的矩阵



使用 plot 函数对于数据进行绘制，可以看到对于蓝色的点有一个值偏离整体蓝色的区域较大，我们需要观察下一步他对于我们的 svm 决策边界产生的影响

2. 设置 C=1

SVM 优化问题中的参数 c 是正成本因素，会惩罚分类错误的训练示例

这里我们使用 C=1 运行分类器

使用 svmtrain 函数进行模型的训练，这样我们就得到了 w 和 b

```
model = svmtrain(y, x, sprintf('-s 0 -t 0 -c %g', C1));
```

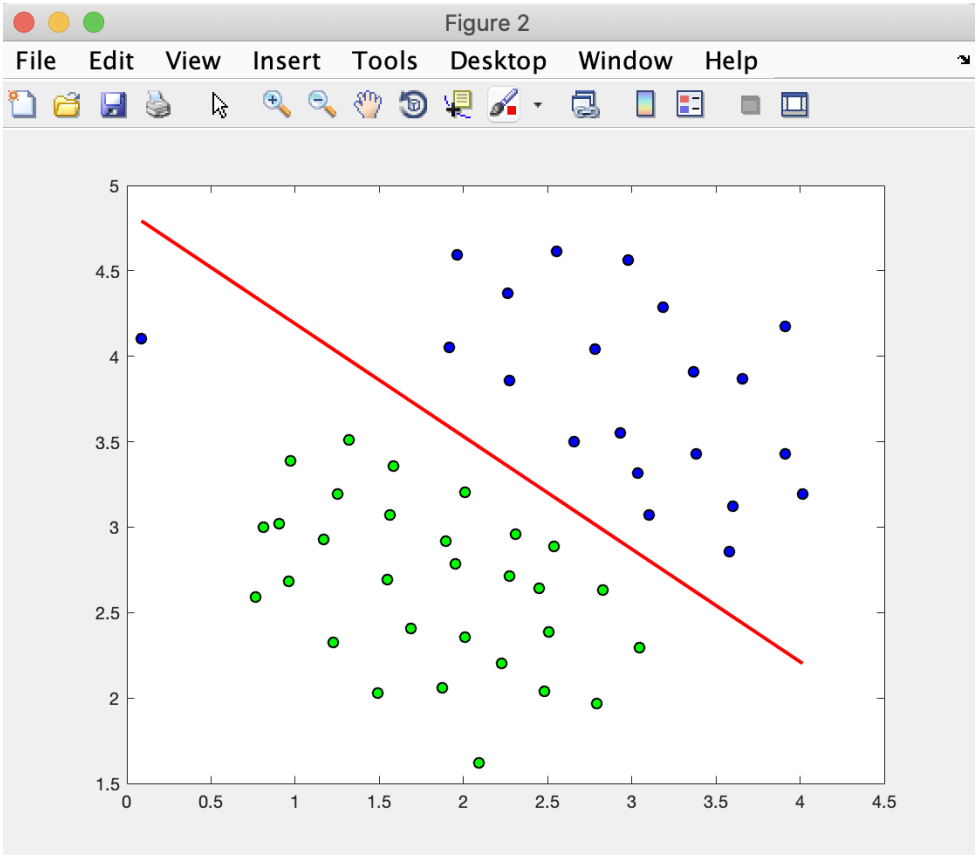
其中-s 0 参数用于分类

-t 0 参数是 linear kernel

-c 参数用来设置损失 cost

藉此，我们可以绘制出决策的边界

在 C=1 的情况下，看到异常值分类是分类错误的，但决策范围是合理的



我们得到的 w 和 b

```
*
optimization finished, #iter = 13
nu = 0.215674
obj = -7.731465, rho = 10.350343
nSV = 12, nBSV = 9
Total nSV = 12

w =

    1.4074
    2.1343


b =

  -10.3503
```

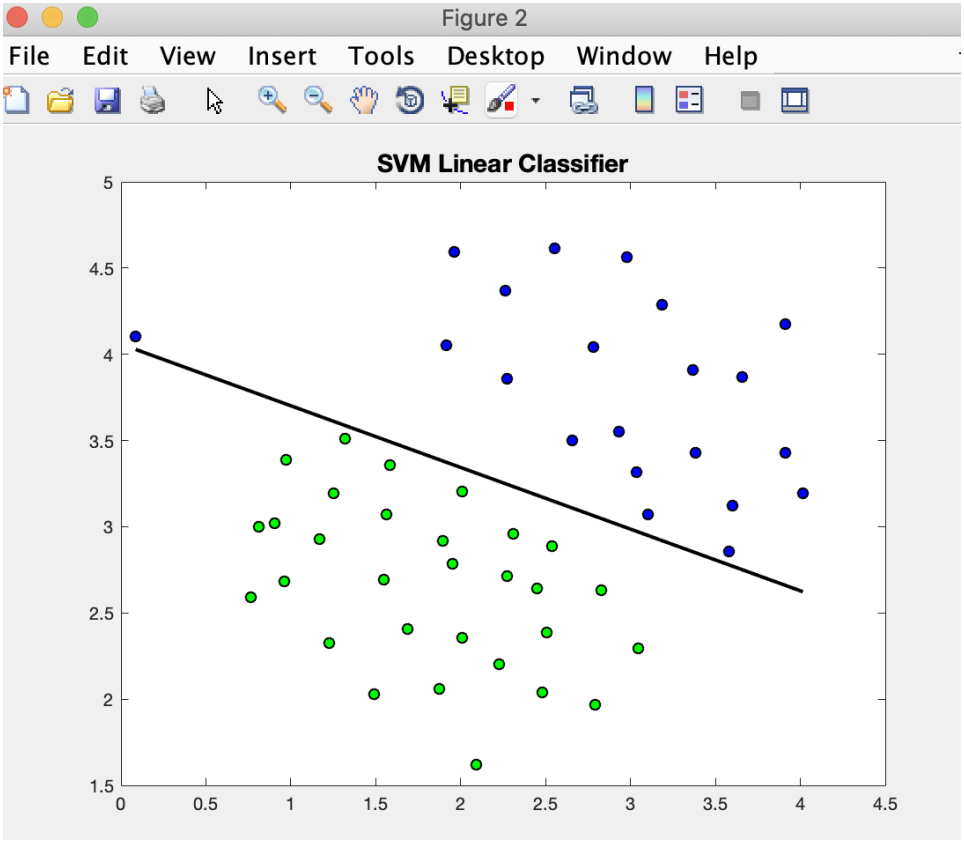
通过调整函数的参数 C，通过让它取不同的数值

分别设置

```
4 -    C1=1
5 -    C2=10
6 -    C3=50
7 -    C4=100
```

但成本因素不断变大的时候，训练模型并再次绘制决策边界

但我们选用 C4 做为函数的输入参数的时候



此时的 svm 已经可以正确地分类离群值了

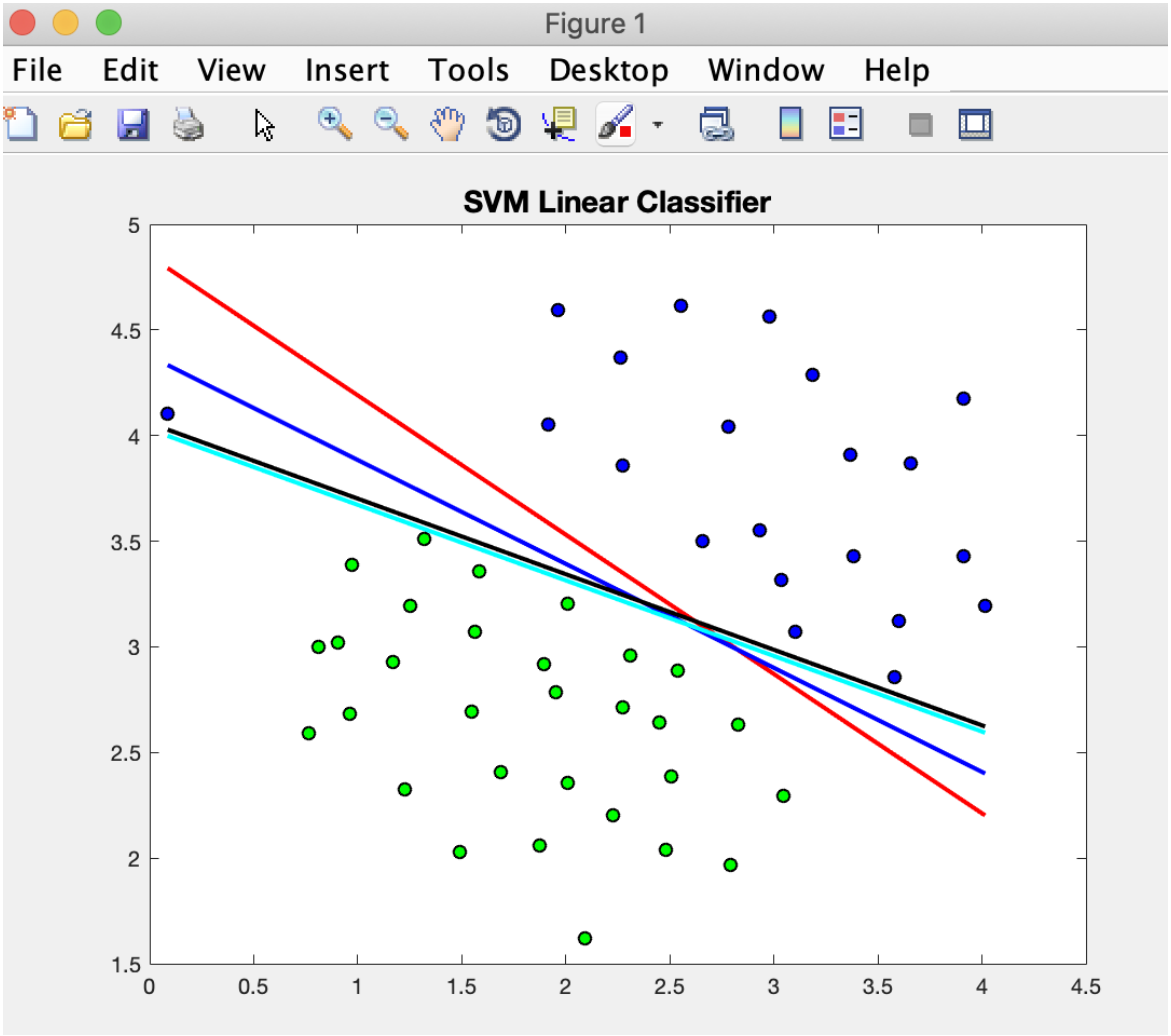
但是决策边界对于其余数据似乎不是很自然的选择

这说明，当成本代价很大，svm 算法很难避免错误分类

而折衷的办法是该算法将较少权重以产生较大的分离余量

```
........*...*
optimization finished, #iter = 582
nu = 0.037918
obj = -96.720252, rho = 53.140433
nSV = 3, nBSV = 0
Total nSV = 3

w =

    4.6826
   13.0918


b =

  -53.1404
```

上图为 C=100 时候，我们得到的参数的值



调节 C 可以调节分类面的 Margin ，C 越大，Margin 越小正确率也越高，但是在非线性的分类问题中可能是会出现过拟合，所以选择一个合适的 C 值非常重要

垃圾邮件分类任务

我们可以选择不同的训练集的规模来做比较，当训练集规模越大的时候

我们得到的预测误差也就越小

使用 email-train-50 得到的测试集结果

```
.*
optimization finished, #iter = 83
nu = 0.004760
obj = -0.118993, rho = 0.750695
nSV = 33, nBSV = 0
Total nSV = 33
Accuracy = 75.3846% (196/260) (classification)
```

使用 email-train-100 在测试集上得到的结果

```
.*
optimization finished, #iter = 123
nu = 0.004963
obj = -0.248161, rho = 0.462174
nSV = 48, nBSV = 0
Total nSV = 48
Accuracy = 88.4615% (230/260) (classification)
```

使用 email-train-400 得到的测试集结果

```
*
optimization finished, #iter = 278
nu = 0.005102
obj = -1.020290, rho = 0.177186
nSV = 109, nBSV = 0
Total nSV = 109
Accuracy = 98.0769% (255/260) (classification)
```

使用 email-train-all 得到的测试集结果

```
*
optimization finished, #iter = 392
nu = 0.003571
obj = -1.249807, rho = 0.063035
nSV = 135, nBSV = 0
Total nSV = 135
Accuracy = 98.4615% (256/260) (classification)
```

结论分析与体会：

这次实验没有让我们直接自己实现 svm 算法的细节，确实也感觉如果自己实现 svm 的话，难度会相比调用现有的函数库指数级别上升，通过两个特殊的例子加深了对于 svm 的理解。

附录：程序源代码

1.

```
clear all;clc
[y,x] = libsvmread('/Users/chenjiarui/Desktop/ex7Data/twofeature.txt');

C1=1
C2=10
C3=50
C4=100

model = svmtrain(y, x, sprintf('-s 0 -t 0 -c %g', C1));
w = model.SVs' * model.sv_coef
b = -model.rho
if (model.Label(1) == -1)
    w = -w; b = -b;
end


figure
pos=find(y==1)
neg=find(y==-1)
```

```
plot(x(pos,1),x(pos,2),'ko','MarkerFaceColor','b');
hold on;
plot(x(neg,1),x(neg,2),'ko','MarkerFaceColor','g');


% Plot the decision boundary
plot_x = linspace(min(x(:,1)), max(x(:,1)), 30);
plot_y = (-1/w(2))*(w(1)*plot_x + b);
plot(plot_x, plot_y, 'r-', 'LineWidth', 2)

% Plot the decision boundary2
model = svmtrain(y, x, sprintf('-s 0 -t 0 -c %g', C2));
w = model.SVs' * model.sv_coef
b = -model.rho
if (model.Label(1) == -1)
    w = -w; b = -b;
end
plot_x = linspace(min(x(:,1)), max(x(:,1)), 30);
plot_y = (-1/w(2))*(w(1)*plot_x + b);
plot(plot_x, plot_y, 'b-', 'LineWidth', 2)
% Plot the decision boundary3
model = svmtrain(y, x, sprintf('-s 0 -t 0 -c %g', C3));
w = model.SVs' * model.sv_coef
b = -model.rho
if (model.Label(1) == -1)
    w = -w; b = -b;
end
plot_x = linspace(min(x(:,1)), max(x(:,1)), 30);
plot_y = (-1/w(2))*(w(1)*plot_x + b);
plot(plot_x, plot_y, 'c-', 'LineWidth', 2)
% Plot the decision boundary4
model = svmtrain(y, x, sprintf('-s 0 -t 0 -c %g', C4));
w = model.SVs' * model.sv_coef
b = -model.rho
if (model.Label(1) == -1)
    w = -w; b = -b;
end
plot_x = linspace(min(x(:,1)), max(x(:,1)), 30);
```

```
plot_y = (-1/w(2))*(w(1)*plot_x + b);
plot(plot_x, plot_y, 'k-', 'LineWidth', 2)
title(sprintf('SVM Linear Classifier'), 'FontSize', 14)
```

2. spam_email

```
% SVM Email text classification
clear all; close all; clc
% Load training features and labels
[train_y, train_x] = libsvmread('/Users/chenjiarui/Desktop/ex7Data/email_train-all.txt');

% Train the model and get the primal variables w, b from the model
% Libsvm options
% -t 0 : linear kernel
% Leave other options as their defaults
model = svmtrain(train_y, train_x, '-t 0');
w = model.SVs' * model.sv_coef;
b = -model.rho;
if (model.Label(1) == -1)
    w = -w; b = -b;
end

% Load testing features and labels
[test_y, test_x] = libsvmread('/Users/chenjiarui/Desktop/ex7Data/email_test.txt');

[predicted_label, accuracy, decision_values] = svmpredict(test_y, test_x, model);
% After running svmpredict, the accuracy should be printed to the matlab
% console
```