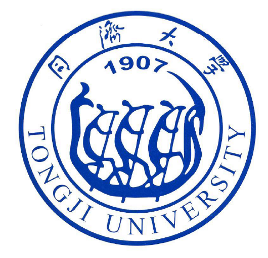
**机器学习大作业报告**

****

**-------线性分类问题**

**学 号 1751975**

**姓 名 田宇清**

**专 业 计算机科学与技术**

**授课老师 武研 李洁**

目录

[1. SVM算法原理 2](#_Toc1430)

[1.1概念 2](#_Toc32018)

[1.2最大间隔 3](#_Toc25947)

[1.3拉格朗日子乘法 3](#_Toc3701)

[1.4松弛变量 4](#_Toc17075)

[1.5核函数 5](#_Toc22278)

[1.6 KKT分析结论 5](#_Toc2010)

[2. SMO算法 5](#_Toc26917)

[2.1目标 5](#_Toc29279)

[2.2概念 5](#_Toc30228)

[2.3步骤 5](#_Toc22717)

[2.3.1选择αi和αj 6](#_Toc17302)

[2.3.2优化αi和αj 6](#_Toc30207)

[2.3.3更新w与b 7](#_Toc17305)

[2.3.4终止条件 8](#_Toc13115)

[3. 代码实现 8](#_Toc1587)

[3.1库（不直接使用sklearn库） 8](#_Toc26149)

[3.2 函数实现 8](#_Toc16498)

[3.3 使用方法 9](#_Toc29969)

[3.3.1创建新项目 9](#_Toc14037)

[3.3.2 处理数据集 9](#_Toc5138)

[3.3.3 训练 9](#_Toc4203)

[3.3.4 运行程序 9](#_Toc11299)

[4. 实验结果 9](#_Toc32733)

[4.1数据集处理 9](#_Toc17529)

[4.2训练参数设置 10](#_Toc23034)

[4.3实验结果 10](#_Toc18279)

[4.3.1 萼片长度，萼片宽度，花瓣长度，花瓣宽度 10](#_Toc11649)

[4.3.2 萼片长度，萼片宽度 10](#_Toc1926)

[4.3.3 花瓣长度，花瓣宽度 10](#_Toc20014)

[4.4结果分析 11](#_Toc20223)

[5. 后续改进方向 11](#_Toc5475)

[5.1 数据集处理 11](#_Toc21023)

[5.2 参数选择 11](#_Toc27874)

[5.3 结果展示 11](#_Toc26533)

[6. 参考文献 12](#_Toc7272)

1. **SVM算法原理**

**1.1概念**

SVM（support vector mac）：支持向量机，是一种二分类模型。分为线性和非线性两大类。找到空间中的一个能够将所有数据样本划开的超平面，并且使得本集中所有数据到这个超平面距离最短。

支持向量：距离这个超平面最近的点。要求支持向量到这条直线的距离最大。

**1.2最大间隔**

超平面方程：



点到平面距离：



目标函数：



标签y（-1/+1）保证计算的距离始终为正数。目标函数表示求w与b，使得离超平面距离最近的点（支持向量），到这个超平面的距离最大。

对于：



调整w与b，使得他们等比例改变（超平面不变），保证计算支持向量的该值为1，则计算其他样本点的该值一定大于1，目标函数优化为：



约束条件：



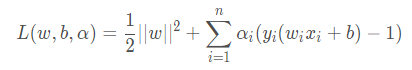
替换目标函数（？）为：



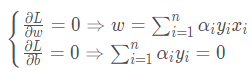
**1.3拉格朗日子乘法**

拉格朗日子乘法即求条件极值。SVM的工作即为求α。

应用拉格朗日子乘法：



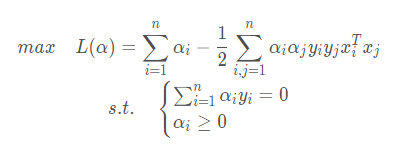
求w与b的偏导：



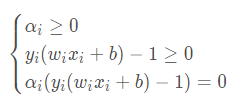
消去w与b：



最终求解对偶问题：



KKT条件：

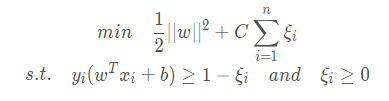


**1.4松弛变量**

增加松弛变量处理噪声，允许一些数据处于超平面错误的一侧，新的约束条件：

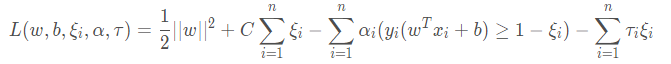


尽可能让松弛变量小，新的目标函数：

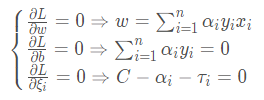


C是控制“最大化间隔”与“大部分点分类正确”的权重。

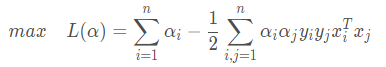
新的拉格朗日函数：



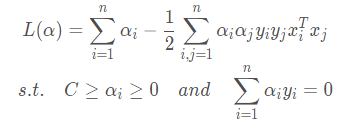
求偏导：



化简得到目标函数，和之前一样：

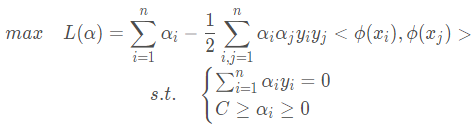


最终对偶问题：

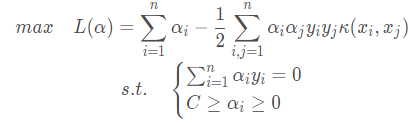


**1.5核函数**

低维线性不可分映射到高维空间：



训练和预测使用且只使用样本的内积，使用核函数优化对偶问题：



求得a的值就可以求得w与b。

**1.6 KKT分析结论**

求得αi=0，该样本落在分割线外；

求得αi=C，该样本落在分割线内；

求得0<αi<C，该样本落在分割线上，即支持向量。

1. **SMO算法**

**2.1目标**

找到一组最优的αi。

**2.2概念**

SMO：Sequential Minimal Optimization序列最小化优化算法，将大优化问题分解成多个小优化的问题。

坐标下降（上升法）：每次迭代只调整一个变量αi的值，其他变量的值在这次迭代总固定不变。

**2.3步骤**

SMO算法：

根据约束条件：



一次迭代优化两个αi和αj。二者互相表示：

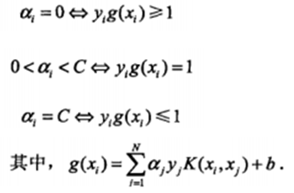


**2.3.1选择αi和αj**

启发式选择。

选择αi：违反KKT条件最严重的样本点。

外层循环所有0<αi<C样本点，检验是否满足KKT条件，选择违反KKT条件最严重的样本点。如果都满足则遍历整个训练集。整个训练过程会在这两种情况之间转换，因为遍历整个训练集后，如果有αi被优化，则需要重新考虑支持矢量。



根据之前的KKT分析，0<αi<C样本点更可能需要被调整，大部分不是支持向量的数据αi=0时就不再需要调整。

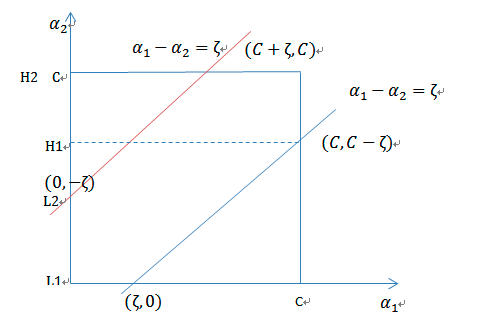
选择αj：更快接近代价函数的最大值。第二个乘子的迭代步长大致正比于[clip_image149](http://images.cnblogs.com/cnblogs_com/jerrylead/201103/201103182044414253.png)，选择第二个乘子能够最大化[clip_image149[1]](http://images.cnblogs.com/cnblogs_com/jerrylead/201103/201103182044425823.png)。即当[clip_image151](http://images.cnblogs.com/cnblogs_com/jerrylead/201103/201103182044439661.png)为正时选择负的绝对值最大的[clip_image153](http://images.cnblogs.com/cnblogs_com/jerrylead/201103/201103182044439595.png)，反之，选择正值最大的[clip_image153[1]](http://images.cnblogs.com/cnblogs_com/jerrylead/201103/201103182044443465.png)。

**2.3.2优化αi和αj**

根据约束条件：



计算αj范围L<=αj<=H



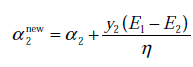
y异号：



y同号：



根据Platt的推导,：

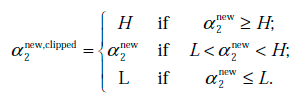


其中，E是样本标签的误差，由核函数计算得到：





根据之前计算得到的αj范围L<=αj<=H进行约束：



推导出αi：



**2.3.3更新w与b**

参考网络资料：



w的更新：



**2.3.4终止条件**

所有样本在容忍值范围内满足KKT条件即可。

1. **代码实现**

**3.1库（不直接使用sklearn库）**

Numpy：帮助完成矩阵运算。

Time：用于记录完成时间。

matplotlib.pyplot：完成数据可视化。

**3.2 函数实现**

**def** calcKernelValue(matrix\_x, sample\_x, kernelOption)

**def** calcKernelMatrix(train\_x, kernelOption)

这两个函数通过传参，将数据集train\_x通过核函数kernelOption（0为线性核，1为高斯核，高斯核需额外再传一个参数）的处理，得到了映射后的矩阵。

**def** calcError(svm, alpha\_k):

这个函数计算了上述：



**def** updateError(svm, alpha\_k)

这个函数更新了全局变量中的Ek，我们下次需要使用这个Ek时无需重新计算，直接调用即可。

**def** selectAlpha\_j(svm, alpha\_i, error\_i)

这个函数通过查阅全局变量中的errorCache，选择了提升最大的αj。

**def** innerLoop(svm, alpha\_i)

这个函数基本处理了2.3.1与2.3.2中的选择与优化部分，详细情况请查阅代码注释。

**def** trainSVM(train\_x, train\_y, C, toler, maxIter, kernelOption=(**'rbf'**, 1.0)):

通过参数控制完成训练过程。

**def** testSVM(svm, test\_x, test\_y)

SVM测试，返回准确率。

**def** showSVM(svm)

可视化展示划分结果，但只能展示二维线性可分的数据集划分。

超出二维的数据会提示无法绘制图像，非线性可分的数据集绘制的图像不正确。

**3.3 使用方法**

**3.3.1创建新项目**

创建新项目，添加python文件，SVM.py与test.py。

**3.3.2 处理数据集**

在test.py的同级目录下添加数据集文件，并修改test.py文件第8行fileIn = open(**'bloodTransfusion\_noduplicated.txt'**)

打开文件。

Test.py文件第10行lineArr = line.strip().split(**','**)

可以根据数据集文件不同的分隔符读取数据（‘,’、’ ’、’\t’），请根据所选择的数据集修改分隔符。

Test.py文件第11、12行分别读取数据与标签：

dataSet.append([float(lineArr[0]), float(lineArr[1]), float(lineArr[2]), float(lineArr[3])])  
labels.append(float(lineArr[4]))

请根据所选择的数据集的维度，修改代码。

Test.py文件第16、17、18、19行划分训练集与测试集：

train\_x = dataSet[0:501, :]  
train\_y = labels[0:501, :]  
test\_x = dataSet[501:511, :]  
test\_y = labels[501:511, :]

同样，根据所选择的数据集情况自行划分即可。

**3.3.3 训练**

Test.py文件第16、17、18、19行设置训练参数：

C = 0.6  
toler = 0.001  
maxIter = 100  
svmClassifier = SVM.trainSVM(train\_x, train\_y, C, toler, maxIter, kernelOption=(**'rbf'**, 20))

**3.3.4 运行程序**

1. **实验结果**

**4.1数据集处理**

我们从UCI machine learning官网上下载兰花数据集，这个数据基本上是线性可分的。（<https://archive.ics.uci.edu/ml/machine-learning-databases/iris/>）

处理下载好的数据，使其可被程序读取。将Iris-setosa类标记为1，其余两类标记为-1。（选取任意一类为1都可行）。并打乱数据集。

本数据集的分隔符为’ ’：



读取的数据，我们分为以下几类：

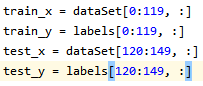
1. 四个维度的数据：sepal length萼片长度，sepal width萼片宽度，petal length花瓣长度，petal width花瓣宽度。全部读取。



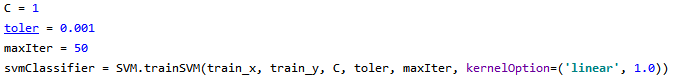
1. 任意两个维度两两组合。总计6种组合。我们选择2个组合：萼片长度，萼片宽度和花瓣长度，花瓣宽度。

数据集划分：

数据集总计150组数据，选取120个数据作为训练集，30个数据作为测试集：

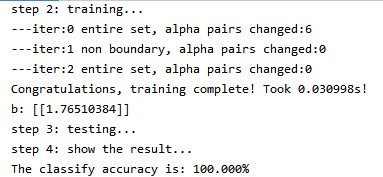


**4.2训练参数设置**

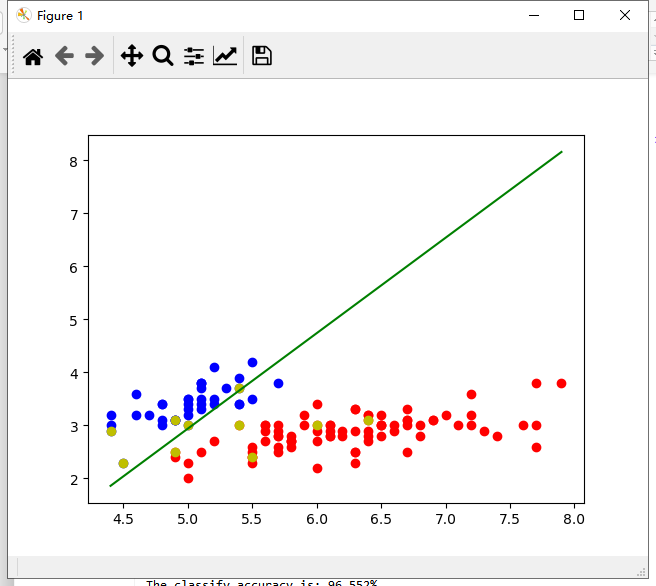
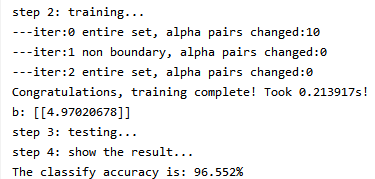


**4.3实验结果**

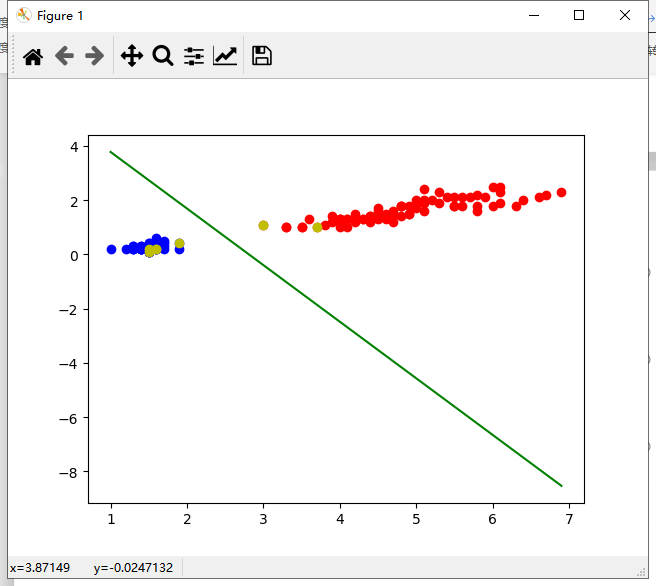
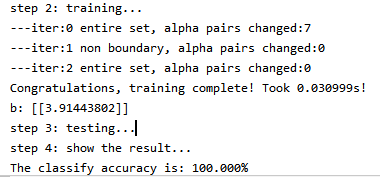
**4.3.1 萼片长度，萼片宽度，花瓣长度，花瓣宽度**



**4.3.2 萼片长度，萼片宽度**



**4.3.3 花瓣长度，花瓣宽度**



**4.4结果分析**

通过对比4.32与4.3.3中的结果，我们不难发现，通过花瓣的长度与宽度判断兰花的种类Iris-setosa，可靠性在一定程度上会优于通过萼片的长度与宽度的判断。

1. **后续改进方向**
   1. **数据集处理**

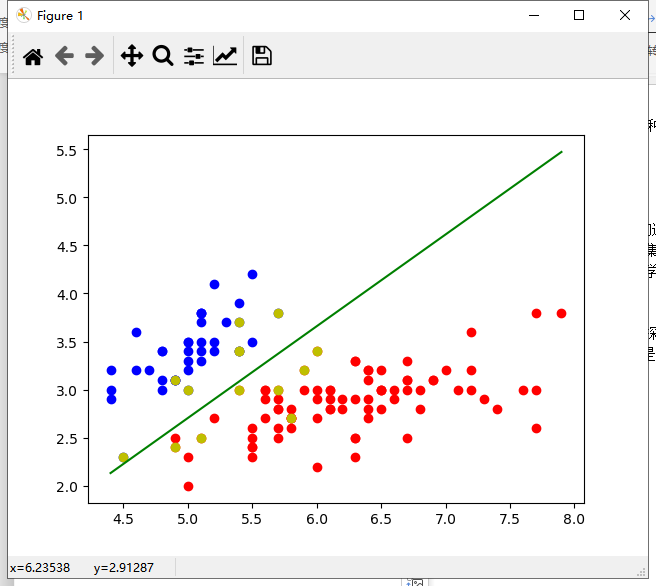
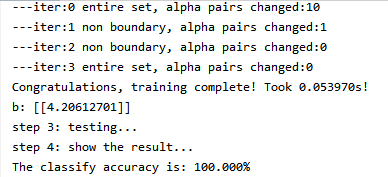
首先，在本程序中，需要人工调试数据集的部分较多，包括数据集的导入，分隔符的选择，数据维度的处理，数据集的划分，程序显得并不是那么智能。特别指出，程序中数据集的划分，不但需要人工调试，且划分并不科学，之后程序的改进可以考虑对数据集有更科学的划分。

* 1. **参数选择**

本程序同样需要选择合适的参数进行训练，但这些参数怎样才算“合适”是一个值得探讨的问题。如上述4.3.2中的实验结果，在4.2的参数选择下，准确度只有96.552%,，但是稍微改动一下设置的参数：



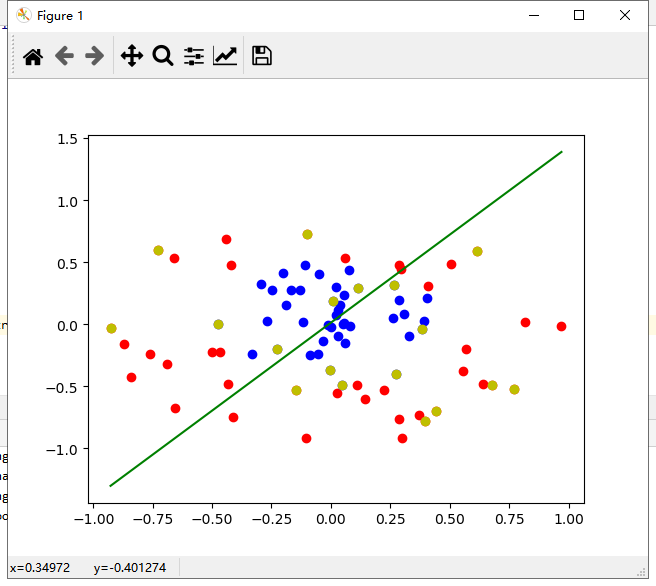
我们就可以得到新的实验结果：



对于松弛变量、违反KKT条件的容忍度、核函数及其参数的选择，我们需要查阅更多的文献，经过多次实验可能才能确定一个合适的值。

* 1. **结果展示**

对于二维线性可分数据，我们可以通过画图的方式将分类结果展示出来，对于更高维度的数据，我们一般不考虑通过作图的方式展示结果。但是对于二维线性不可分的数据，我们理论上应当做出展示，但本程序的作图是采取取两点作直线的方法，因此我们可以看一个实验结果：



我们可以看出，支持向量的选择可能是正确的，但因为作图的问题，显得实验结果有些不伦不类，如何更准确的作图，也是之后程序的改进方向。

1. **参考文献**

[1]机器学习之手把手实现，第 1 部分 支持向量机的原理和实现：<https://www.ibm.com/developerworks/cn/analytics/library/machine-learning-hands-on1-svn/index.html>

[2]SVM算法原理以及Python实现：<https://blog.csdn.net/a1013552415/article/details/103176447?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-2.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-2.nonecase>

[3]Machine Learning in Action ,Peter Harrington

[4]机器学习算法与Python实践之（四）支持向量机（SVM）实现：<https://blog.csdn.net/zouxy09/article/details/17292011>

[5][机器学习实战之SVM](https://www.cnblogs.com/zy230530/p/6901277.html)：<https://www.cnblogs.com/zy230530/p/6901277.html>