实验三 SVM

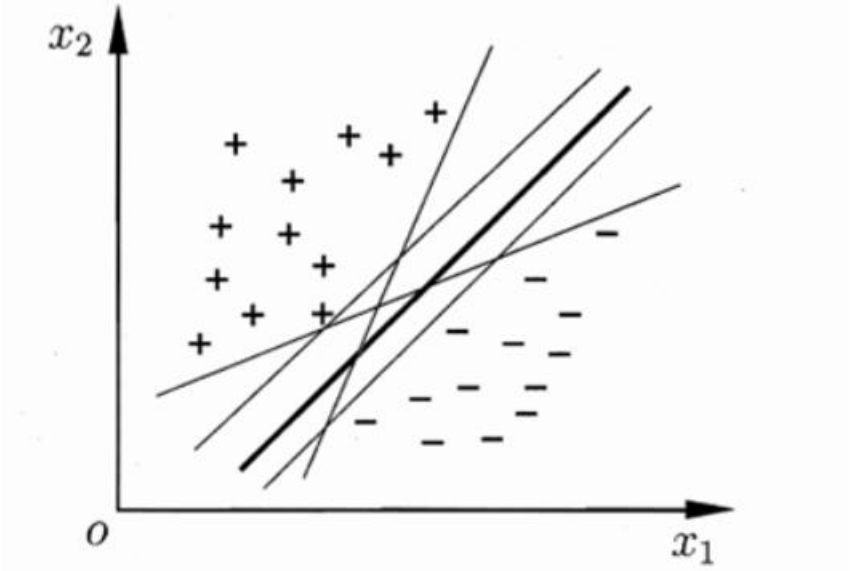
小组成员：计科 1802 张继伟（201808010829）
计科 1802 谢正宇（201808010824）
计科 1801 樊佳婷（201808010816）
计科 1801 刘怡聪（201808010813）
计科 1801 孙晶铭（201808010808）

实验完成日期：2020 年 12 月 04 日

1， 实验描述

（1） 实验原理：SVM——支持向量机

给定训练样本集 D ={ (X1 ,Y1)， (X2，Y2)…, (Xm,Ym)}, Yi ∈{−1,+1}，分类学习最基本的想法就是基于训练集 D 在样本空间中找到一个划分超平面、将不同类别的样本分开。



原则上，是去找位于两类训练样本"正中间"的划分超平面，即图 6.1 中黑色最粗的那个，因为该划分超平面对训练样本局部扰动的"容忍性“最好，鲁棒性最强，泛化能力最强。
例如，由于训练集的局限性或噪声的因素，训练集外的样本可能比图 6.1 中的训练样本更接近两个类的分隔界，这将使许多划分超平面出现错误，而黑色最粗的那个超平面受影响最小。换言之，这个划分超平面所产生的分类结果是最鲁棒的，对未见示例的泛化能力最强。
在样本空间中，划分超平面可通过如下线性方程来描述:$w^T x + b = 0$;(1)
x=(x1,x2…xd)为输入数据，维度为 d。
w=(w_1;w_2;…;w_d)为法向量，决定了超平面的方向。
b 为位移项，决定了超平面与原点之间的距离。
显然，分类超平面可被法向量 ω 和位移 b 确定，下面我们将其记为(w,b)。
样本空间中任意点 x 到超平面(w,b)的距离可写为：

$$r=\frac{|w^T x+b|}{||w||};$$ （2）

假设超平面(w,b)能将训练样本正确分类，即对于$(x_i,y_i) \in D$，

若$y_i = +1$，则有$w^T x_i + b > 0$;

若$y_i = -1$，则有$w^T x_i + b < 0$;

所以，令：

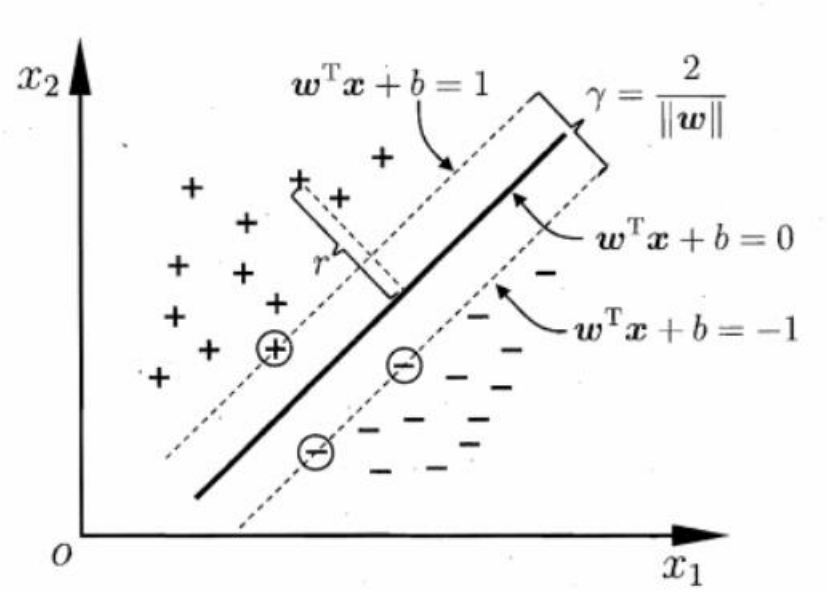
$w^T x_i + b \gg +1, y_i = +1$;

$w^T x_i + b \ll -1, y_i = -1$;(3)

如图所示，距离超平面最近的这几个训练样本点使(3)的等号成立，它们被称为"支持向量"，两个异类支持向量到超平面的距离之和为：

$r = \frac{2}{||w||}$;（4）

其中，γ被称为"间隔"。



欲找到具有"最大间隔"（maximum margin）的划分超平面，也就是要找到能满足式(6.3)中约束的参数 w 和 b，使得γ最大，即：

$\max \frac{2}{||w||}$;

s.t. $y_i(w^T x_i + b) \gg 1, i=1,2,\cdots,m$;(5)

显然，为了最大化间隔γ，仅需最小化‖w‖。

于是，式(5)可重写为：

$\min \frac{1}{2}||w||^2$

s.t. $y_i(w^T x_i + b) \gg 1, i=1,2,\cdots,m$;(6)

上述（6）是支持向量机的基本型。

（2） 数据集处理

Iris.data 的数据格式如下：共 5 列，前 4 列为样本特征，第 5 列为类别，分别有三种类别 Iris-setosa， Iris-versicolor， Iris-virginica。

注意：因为在分类中类别标签必须为数字量，所以应将 Iris.data 中的第 5 列的类别（字符串）转换为 num。

2，实验及结果分析

（1） 开发语言及运行环境；

Visual Studio 2019 及 anaconda

（2） 实验的具体步骤；

导入 SVM 模块

首先在使用 SVM 时，需先从 sklearn 包中导入 SVM 模块。

```
from sklearn import svm
```

读取数据集

```
#1.读取数据集
path='./Iris.data'
data=np.loadtxt(path, dtype=float, delimiter=',', converters={4:Iris_label} )
#converters={4:Iris_label}中“4”指的是第5列：将第5列的str转化为label(number)
#print(data.shape)
```

定义的转换函数

可实现将类别 Iris-setosa,Iris-versicolor,Iris-virginica 映射成 0,1,2。

```
#define converts(字典)
def Iris_label(s):
    it={b'Iris-setosa':0, b'Iris-versicolor':1, b'Iris-virginica':2 }
    return it[s]
```

划分训练样本与测试样本

```
#2.划分数据与标签
#x为数据，y为标签
x,y=np.split(data,indices_or_sections=(4,),axis=1)
x=x[:,2:4]
train_data,test_data,train_label,test_label =train_test_split(x,y, random_state=1, train_size=0.8,test_size=0.2)
#print(train_data.shape)
```

split(数据，分割位置，轴=1（水平分割） or 0（垂直分割））。

sklearn.model_selection.train_test_split 随机划分训练集与测试集。

train_test_split(train_data,train_label,test_size=数字, random_state=0)

参数解释：

train_data：所要划分的样本特征集

train_label：所要划分的样本类别

test_size：样本占比，如果是整数的话就是样本的数量

注意：

-- test_size:测试样本占比。默认情况下，该值设置为 0.25。 默认值将在版本 0.21 中更改。 只有 train_size 没有指定时，它将保持 0.25，否则它将补充指定的 train_size，例如 train_size=0.6,则 test_size 默认为 0.4。

-- train_size:训练样本占比。

random_state：是随机数的种子。随机数种子：其实就是该组随机数的编号，在需要重复试验的时候，保证得到一组一样的随机数。比如你每次都填 1，其他参数一样的情况下你得到的随

机数组是一样的。但填 0 或不填，每次都会不一样。随机数的产生取决于种子，随机数和种子之间的关系遵从以下两个规则：种子不同，产生不同的随机数；种子相同，即使实例不同也产生相同的随机数。

训练 SVM 分类器

```
#3.训练svm分类器
classifier=svm.SVC(C=2,kernel='rbf',gamma=10,decision_function_shape='ovr') # ovr:一对多策略
classifier.fit(train_data,train_label.ravel()) #ravel函数在降维时默认是行序优先
```

kernel='linear'时，为线性核，C 越大分类效果越好，但有可能会过拟合（defaul C=1）。
kernel='rbf'时（default），为高斯核，gamma 值越小，分类界面越连续；gamma 值越大，分类界面越“散”，分类效果越好，但有可能会过拟合。
decision_function_shape='ovr'时，为 one v rest（一对多），即一个类别与其他类别进行划分，decision_function_shape='ovo'时，为 one v one（一对一），即将类别两两之间进行划分，用二分类的方法模拟多分类的结果。

计算分类准确率

```
#4.计算svc分类器的准确率
#print("训练集：",classifier.score(train_data,train_label))
#print("测试集：",classifier.score(test_data,test_label))
#也可直接调用accuracy_score方法计算准确率
from sklearn.metrics import accuracy_score
tra_label=classifier.predict(train_data) #训练集的预测标签
tes_label=classifier.predict(test_data) #测试集的预测标签
print("训练集：", accuracy_score(train_label,tra_label) )
print("测试集：", accuracy_score(test_label,tes_label) )
```

结果：

```
训练集： 0.975
测试集： 0.9666666666666667
```

查看决策函数

```
#查看决策函数
#print('train_decision_function:\n',classifier.decision_function(train_data)) # (90,3)
#print('predict_result:\n',classifier.predict(train_data))
```

绘制图形

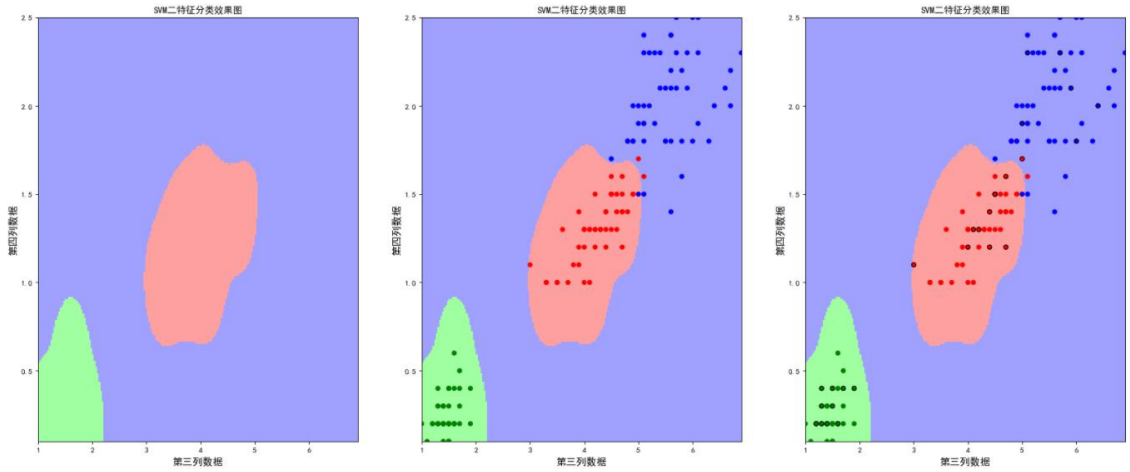
根据第三列和第四列数据绘制分类图形（二维平面图形）
确定坐标轴范围、字体、背景颜色

```
def showTitle():
    plt.xlabel('第三列数据', fontsize=13)
    plt.ylabel('第四列数据', fontsize=13)
    plt.xlim(x1_min,x1_max)
    plt.ylim(x2_min,x2_max)
    plt.title('SVM二特征分类效果图')
```

```
#5.绘制图形
#确定坐标轴范围
x1_min, x1_max=x[:,0].min(), x[:,0].max() #第0维特征的范围
x2_min, x2_max=x[:,1].min(), x[:,1].max() #第1维特征的范围
x1,x2=np.mgrid[x1_min:x1_max:200j, x2_min:x2_max:200j ] #生成网络采样点
grid_test=np.stack((x1.flat,x2.flat) ,axis=1) #测试点
#指定默认字体
matplotlib.rcParams['font.sans-serif']=['SimHei']
#设置颜色
cm_light=matplotlib.colors.ListedColormap(['#A0FFA0', '#FFA0A0', '#A0A0FF'])
cm_dark=matplotlib.colors.ListedColormap(['g','r','b'] )

grid_hat = classifier.predict(grid_test)        # 预测分类值
grid_hat = grid_hat.reshape(x1.shape)  # 使之与输入的形状相同
```

（3） 根据实验数据集，按实验要求给出相应的结果（截图）；



（4） 对实验结果进行简要分析。

通过图形表示来看，整体训练的效果良好，无论是训练集还是测试集，训练结果的准确率都比较高。

3， 实验心得

计科 1802 张继伟（201808010829）：

支持向量机（support vector machines, SVM）是一种二分类模型，它的基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使它有别于感知机；SVM 还包括核技巧，这使它成为实质上的非线性分类器。SVM 的的学习策略就是间隔最大化，可形式化为一个求解凸二次规划的问题，也等价于正则化的合页损失函数的最小化问题。SVM 的的学习算法就是求解凸二次规划的最优化算法。通过本次实验，我基本明白了支持向量机的基本原理，通过自我动手对机器学习充满强烈兴趣，虽然实验中遇到许多困难，但是通过博客查询等网络手段还是可以有效解决问题。

计科 1802 谢正宇（201808010824）：

本次实验是一个支持向量机（SVM）的实验。老师上课的听的稀里糊涂，在做了这个实验之后感觉明白了许多。有了 svm 这一部分 python 有可以直接用的第三方库，所以我们重点设置合适的参数。首先是核函数的选取。当 kernel='linear'时，为线性核，C 越大分类效果越好，但有可能会过拟合（defaul C=1）。而当 kernel='rbf'时（default），为高斯核，gamma 值越小，分类界面越连续；gamma 值越大，分类界面越“散”，分类效果越好，但有可能会过拟合。我们最终选取的是高斯核。其次关于决策函数对应关系问题，当 decision_function_shape='ovr'时，为 one v rest（一对多），即一个类别与其他类别进行划分，而当 decision_function_shape='ovo'时，为 one v one（一对一），即将类别两两之间进行划分，用二分类的方法模拟多分类的结果。最后数据可视化部分也是用的 python 的 numpy 库。总的来说，本次实验让我更深入的理解了核函数和支持向量机。

计科 1801 樊佳婷（201808010816）：

SVM 是一个非常优雅的算法，具有完善的数学理论。SVM（Support Vector Machines）——支持向量机是在所有知名的数据挖掘算法中最健壮，最准确的方法之一，它属于二分类算法，可以支持线性和非线性的分类。本次实验学习策略便是 SVM 间隔最大化，最终可转化为一个凸二次规划问题的求解。核函数的选取，kernel='linear'时，为线性核，C 越大分类效果越好，但有可能会过拟合（defaul C=1）。而当 kernel='rbf'时（default），为高斯核，gamma 值越小，分类界面越连续；gamma 值越大，分类界面越“散”，分类效果越好，但有可能会过拟合。对训练数据和测试数据的标记，在训练和预测的时候，数据的组成部分里是一直都有标记的，对每一组数据进行标记，而不是想当然的对某一行或者某一列进行标记，标记矩阵的维数要和输入的训练数据和测试数据的维数相同，要不然就不能对所有的样本进行标记。

计科 1801 刘怡聪（201808010813）：

本次机器学习实验我负责代码的编写，因为 Python 中的 sklearn 库也集成了 SVM 算法，所以在 Python 中一样可以使用支持向量机做分类。熟悉了利用 SVM 进行数据分类的代码编写过程，从导入数据，编写转换函数，训练 SVM 模型和根据测试准确率进行调参，并通过可视化展示来分析代码可改进的地方。SVM 的核心是寻找对训练样本局部扰动的"容忍性“最好，鲁棒性最强，泛化能力最强的划分超平面，所以我们修改 SVC 函数的参数，最终选择高斯核，一对多策略进行训练，最后得到一个比较高的准确率。在支持向量机的算法中，也涉及到了对偶问题、拉格朗日乘子法、核函数、软间隔和正则化等，也学习到了很多知识。

计科 1801 孙晶铭（201808010808）：

SVM 学习:

SVM 本身是一种典型的二分类器，

常用的有三种方法:

1.一对多

也就是“一对其余”的方式，就是每次仍然解一个两类分类的问题。这样对于 n 个样本会得到 n 个分类器。但是这种方式可能会出现分类重叠现象或者不可分类现象，而且由于“其余”的数据集过大，这样其实就人为造成了“数据偏斜”的问题

2.一对一

每次选择一个类作为正样本，负样本只用选其余的一个类，这样就避免了数据偏斜的问题。很明显可以看出这种方法训练出的分类个数是 k*(k-1)/2，虽然分类器的个数比上面多了，但是训练阶段所用的总时间却比“一类对其余”方法少很多。

这种方法可能使多个分类器指向同一个类别，所以可以采用“投票”的方式确定哪个类别：哪

个分类器获得的票数多就是哪个分类器。
这种方式也会有分类重叠的现象，但是不会有不可分类的情况，因为不可能所有类别的票数都是 0。
但是也很容易发现这种方法是分类器的数目呈平方级上升。
3.DAG SVM
假设有 1、2、3、4、5 五个类，那么可以按照如下方式训练分类器（这是一个有向无环图，因此这种方法也叫做 DAG SVM）
这种方式减少了分类器的数量，分类速度飞快，而且也没有分类重叠和不可分类现象。
但是假如一开始的分类器回答错误，那么后面的分类器没有办法纠正，错误会一直向下累积。
为了减少这种错误累积，根节点的选取至关重要。

4， 程序文件名的清单

lab3.ipynb
machineLearing1.py
iris.data