**实验二**  **支持向量机**

1. 实验目的

了解支持向量机原理，掌握使用支持向量机进行分类

二、 实验内容

使用支持向量机进行分类

三、 实验环境

Windows环境，

安装python，

安装pythyon IDE

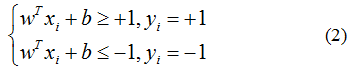
1. 实验相关理论

**1、间隔最大化和支持向量**

在样本空间中，划分超平面可通过如下线性方程来述：

**20180328153118183**

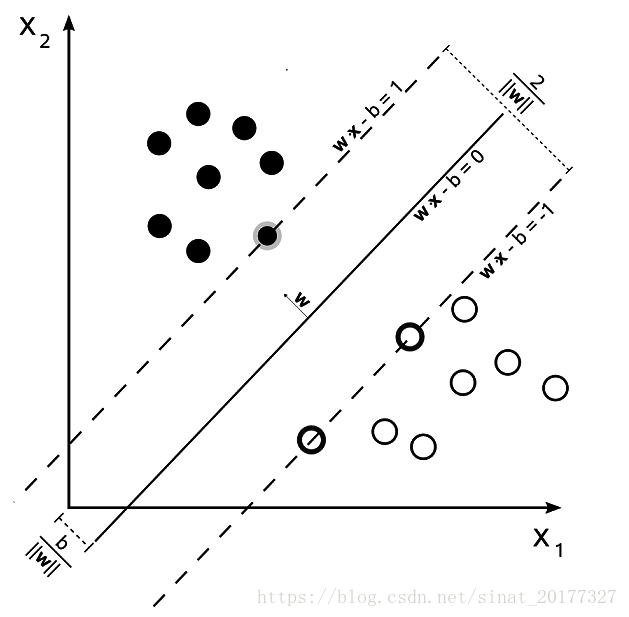
其中w为法向量，决定了超平面的方向，b为位移量，决定了超平面与原点的距离。假设超平面能将训练样本正确地分类，即对于训练样本(xi,yi)，满足以下公式：



公式（2）称为最大间隔假设，yi=+1 表示样本为正样本，yi=−1 表示样本为负样本，式子前面选择大于等于+1，小于等于-1只是为了计算方便，原则上可以是任意常数，但无论是多少，都可以通过对 w 的变换使其为 +1 和 -1 ，此时将公式（2）左右都乘以 yi，得到如下：

20180328154745293

训练集中的所有样本都应满足公式（3）。如下图所示，距离超平面最近的这几个样本点满足 yi(wTxi+b)=1，它们被称为“支持向量”。虚线称为边界，两条虚线间的距离称为间隔（margin）。



下面我们开始计算间隔，其实间隔就等于两个异类支持向量的差在 w 上的投影，即：

20180328160123465



代入公式（4）中可以得到：

20180328161131329

至此，我们求得了间隔，SVM的思想是使得间隔最大化，也就是：

20180328161900536

显然，最大化2/||w|| 相当于最小化 ||w||，为了计算方便，将公式（6）转化成如下：

2018032816192270 (2)

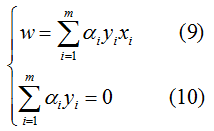
公式（7）即为支持向量机的基本型。

1. **对偶问题**

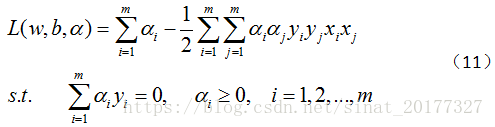
公式（7）本身是一个凸二次规划问题，可以使用现有的优化计算包来计算，但我们选择更为高效的方法。对公式（7）使用拉格朗日乘子法得到其对偶问题，该问题的拉格朗日函数可以写为：

20180328162802163

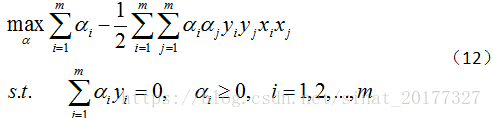
公式（8）分别对 w 和 b求偏导，令其分别为0，可以得到：



将公式（9）（10）代入公式（8），可得：



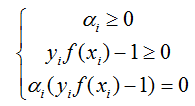
此时，原问题就转化为以下仅关于 α 的问题：



解出 α 之后，根据公式（9）可以求得 w ， 进而求得 b，可以得到模型：

20180328165654360

上述过程的KKT条件为：



我们分析一下，对于任意的训练样本 (xi,yi)，

* 若 αi=0，则其不会在公式（13）中的求和项中出现，也就是说，它不影响模型的训练；
* 若 αi>0，则 yif(xi)−1=0，也就是 yif(xi)=1，即该样本一定在边界上，是一个支持向量。

这里显示出了支持向量机的重要特征：当训练完成后，大部分样本都不需要保留，最终模型只与支持向量有关。

五、 实验步骤

（1）收集数据

（2）准备数据：数值型数据

（3）分析数据：有助于可视化分隔超平面

（4）训练算法：SVM的大部分时间都源自训练，该过程主要实现两个参数的调优

（5）测试算法

（6）使用算法：几乎所有分类问题都可以使用SVM，SVM本身就是一个二类分类器

六、 实验报告要求

按时完成实验要求，提交完整代码及实验报告，进行实验总结。

# -\*- coding: utf-8 -\*-  
*"""  
Created on Sun Oct 14 13:52:47 2018  
  
@author: Administrator  
"""*import numpy as np #常用包  
import xlrd #读excel使用的包  
# from sklearn import preprocessing #进行标准化数据时，需要引入这个包  
from sklearn import svm #调用支持向量机  
from sklearn.model\_selection import train\_test\_split #将数据分开  
# from sklearn.metrics import accuracy\_score , f1\_score#引入正确率  
from sklearn.metrics import classification\_report #结果评估  
  
  
def open\_excel(file):  
 *"""  
 打开excel文件获取数据* ***:param*** *file: 文件所在的位置* ***:return****: 文件数据  
 """* try:  
 data = xlrd.open\_workbook(file)  
 return data  
 except Exception as e:  
 print(str(e))  
  
  
def split\_feature(row):  
 *"""  
 将该行特征处理后放入列表中* ***:param*** *row:一行特征数据* ***:return****: 返回数据列表  
 """* app = [] #定义列表  
 for i in range(16):  
 app = app + [row[i]]  
 return app  
  
  
def loadDataSet(path, training\_sample, colnameindex=0, by\_name=u'sheet1'):  
 *"""  
 加载数据* ***:param*** *path: 数据文件存放路径* ***:param*** *training\_sample: 数据文件名* ***:param*** *colnameindex: 文件列名下标* ***:param*** *by\_name: 表名* ***:return****: 数据集和类别标签  
 """* dataMat = [] # 定义数据列表  
 labelMat = [] # 定义标签列表  
 filename = path + training\_sample # 形成特征数据的完整路径  
 data = open\_excel(filename) # 打开文件获取数据  
 table = data.sheet\_by\_name(by\_name) # 获得数据表  
 nrows = table.nrows # 得到表数据总行数  
 for rownum in range(1, nrows): # 也就是从Excel第二行开始，第一行表头不算  
 row = table.row\_values(rownum) # 取一行数据  
 if row:  
 app = split\_feature(row) # 将特征值转化为列表  
 dataMat.append(app)  
 labelMat.append(float(row[16])) # 获取类别标签  
 return dataMat, labelMat  
  
  
def main():  
 *"""  
 主函数* ***:return****: null  
 """* path = "D:\\"  
 training\_sample = 'featuredata.xls' # 特征数据文件  
 trainingSet, trainingLabels = loadDataSet(path, training\_sample) # 取特征数据和标签数据  
 x = np.array(trainingSet) # 将数据部分列表（list）格式转化为数组(array)格式  
 y = np.array(trainingLabels) # 将标签部分的列表（list）格式转化为数组格式（array）  
 '''  
 将数据分为训练数据和测试数据两部分  
 train\_data 训练数据 train\_label 训练数据标签   
 test\_data 测试数据 test\_label 测试数据标签  
 '''  
 train\_data, test\_data, train\_label, test\_label = train\_test\_split(x, y, random\_state=1, test\_size=0.3)  
  
 """  
 svm.SVC API说明  
 功能：使用SVM分类器进行模型构建  
 参数说明：  
 C:误差项的惩罚系数，默认为1.0；一般为大于0的一个数字，C越大表示在训练过程中对总误差的关注度越高，也就是说当C越大的时候  
 ，对于训练集的表现会越好，但是有可能引发过度拟合的问题。  
 kernel:指定SVM内部函数的类型，可选值：linear、poly、rbf、sigmoid、precomputed（基本不用，有前提要求，  
 要求特征属性数目和样本数目一样）;，默认是rbf;  
 degree:当使用多项式函数作为svm内部的函数的时候，给定多项式的项数，默认为3  
 gamma:当SVM内部使用poly、rbf、sigmoid的时候，核函数的系数值，当默认值为auto的时候，实际系数为1/n\_features.  
 coef0:当核函数为poly或者sigmoid的时候，给定的独立系数，默认为0  
 probability：是否启用概率估计，默认不启动，不太建议启动。  
 probability：是否启用概率估计，默认不启动，不太建议启动。  
 shrinking：是否开启收缩启发式计算，默认为true  
 tol：模型构建收敛参数，当模型的误差变化率小于该值的时候，结束模型构建的过程，默认值：1e-3  
 cache\_size:在模型构建过程中，缓存数据的最大内存大小，默认为空，单位MB  
 class\_weight:给定各个类别的权重，默认为空  
 max\_iter:最大迭代次数，默认-1表示不限制  
 decision\_function\_shape:决策函数，可选值：ovo和ovr,默认为：None:推荐使用ovr;(1.7以上版本才有)  
 """  
 clf = svm.SVC(C=0.8, kernel='rbf', gamma=1, decision\_function\_shape='ovr')  
 """  
 gamma值越大，训练集的拟合就越好，但是会造成过拟合，导致测试集拟合变差  
 gamma值越小，模型的泛华能力越好，训练集和测试集的拟合相近，但是会导致训练集出现欠拟合问题，  
 从而，准确率变低，导致测试集准确率也变低。  
 """  
   
 clf.fit(train\_data, train\_label.ravel()) # 利用训练数据训练模型  
 hat\_test\_label = clf.predict(test\_data) # 对x\_test数据进行预测  
 # precision:精准率，recall：召回率  
 print(classification\_report(test\_label, hat\_test\_label))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 """  
 程序入口  
 """  
 main()