**AI实验二 支持向量机**

**实验背景：**

**支持向量机是我们用于分类的一种算法。**

**每一个可能把数据集正确分开的方向都有一个最优决策面，而不同方向的最优决策面的分类间隔通常是不同的，那个具有“最大间隔”的决策面就是SVM要寻找的最优解。而这个真正的最优解对应的两侧虚线所穿过的样本点，就是SVM中的支持样本点，称为"支持向量"。**

**将所有小球从二维上升到三维空间中，找到这样一个可以将不同颜色小球分开的平面。**

**而从二维平面上看，这些球看起来像是被一条曲线分开了。**

**当一个分类问题，数据是线性可分的，也就是用一条线就可以将两种小球分开的时候，我们只要将直线的位置放在让小球距离直线的距离最大化的位置即可，寻找这个最大间隔的过程，就叫做最优化。**

**但是，现实往往是很残酷的，一般的数据是线性不可分的，也就是找不到一条直线将两种小球很好的分类。这个时候，我们就需要像大侠一样，将小球拍起，用一张纸代替直线将小球进行分类。想要让数据飞起，我们需要的东西就是核函数(kernel)，用于切分小球的纸，就是超平面。**

**实验代码：**

**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():  
 *"""  
 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(**r'C:\Users\apple\Documents\LXL\AI\featuredata.xls'**)  
 **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():  
 *"""  
 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  
 """* colnameindex=0  
 dataMat = [] *# 定义数据列表* labelMat = [] *# 定义标签列表* path = **'C:\\Users\\apple\\Documents\\LXL\\AI'** training\_sample = **'featuredata.xls'** filename = path + training\_sample *# 形成特征数据的完整路径* data = open\_excel() *# 打开文件获取数据* table = data.sheet\_by\_name(**'sheet1'**) *# 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 = **"C:\\Users\\apple\\Documents\\LXL\\AI"** training\_sample = **'featuredata.xls'** *# 特征数据文件* trainingSet,trainingLabels = loadDataSet() *# 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()

实验结果：

