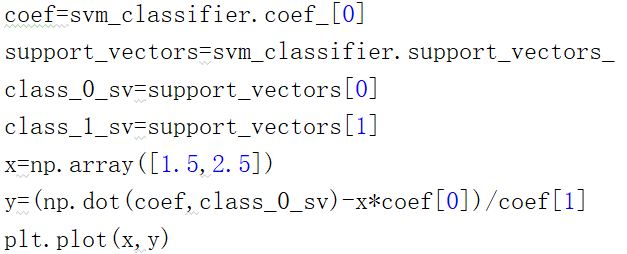
**实验 SVM分类**

**张靖祥 2017304010413**

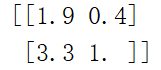
1. **编程语言：python**
2. **SVM对进行鸢尾花分类的流程图**
3. 读入数据，确定数据的格式（scv），确定数据的类型和数量
4. 用matplotlib输出各种鸢尾花在不同的特性下的分类图，观察找到最适合进行分类选择特性。
5. 用sklearn中的svm对样本数据进行分类，选择适当的核函数
6. 作图显示分类之后的图形。对待测数据进行分类测试，输出准确率。
7. **实验关键代码分析**

****

第一行，获取coef，每个特征系数（重要性），只有核函数是Linear的时候可用。在本题中是

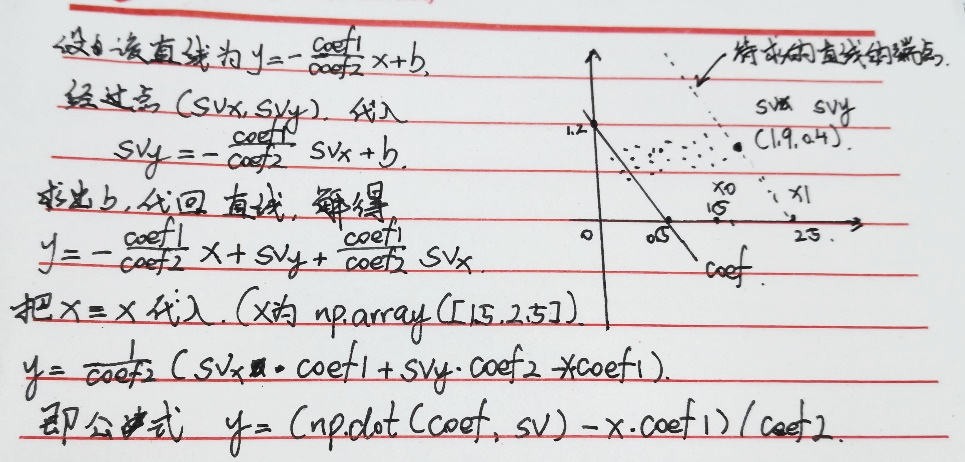


第二行，获取支持向量的索引，返回的就是那些样本成为了支持向量，输出结果为2位数组，每一对值都是一个坐标点



plot画图时画的是从x的坐标区间1.5到2.5，y的坐标区间进行画图

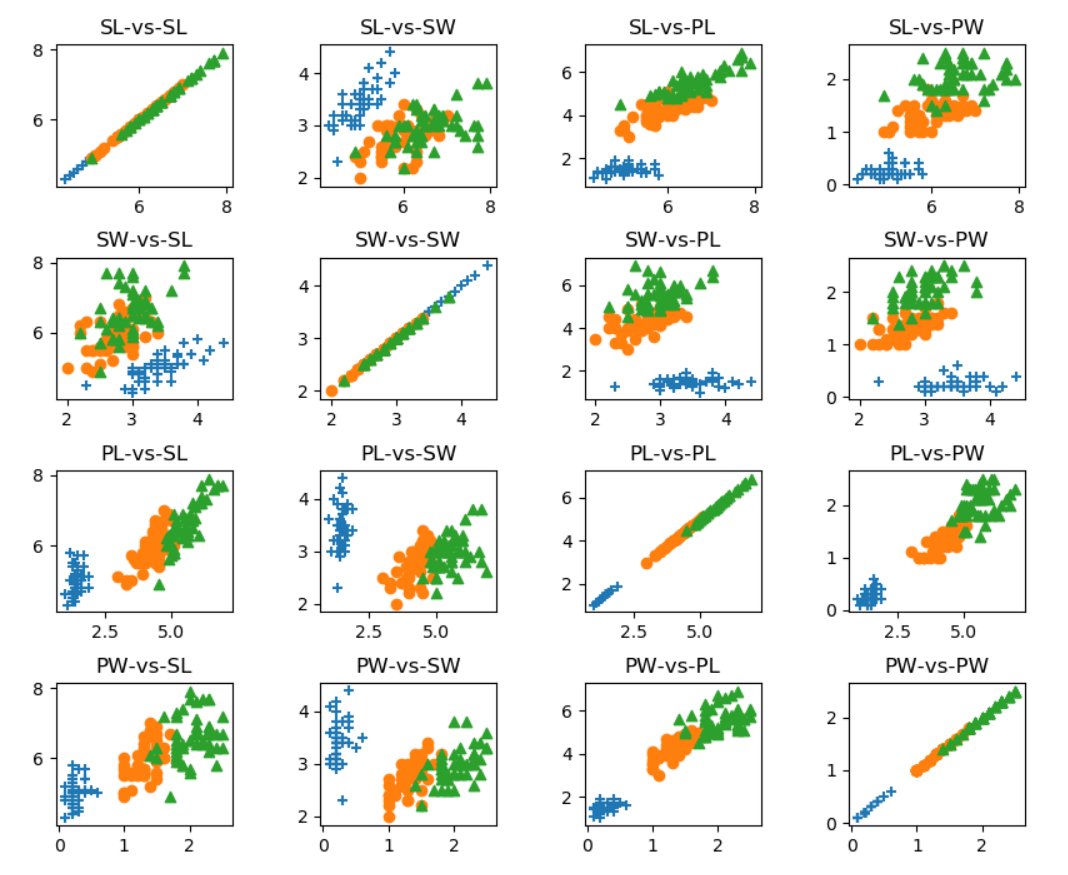
求y的含义：

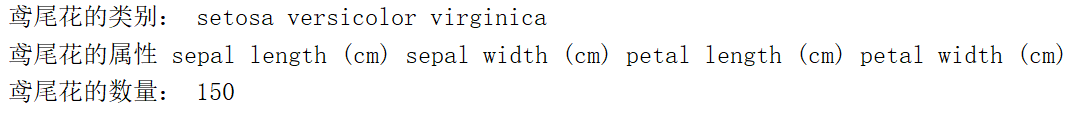
****

1. **程序截图和功能说明**

4.1 基本示例程序

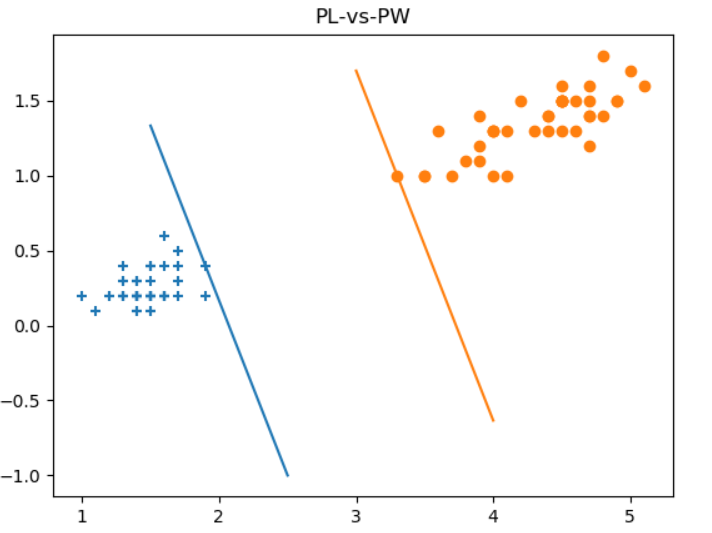
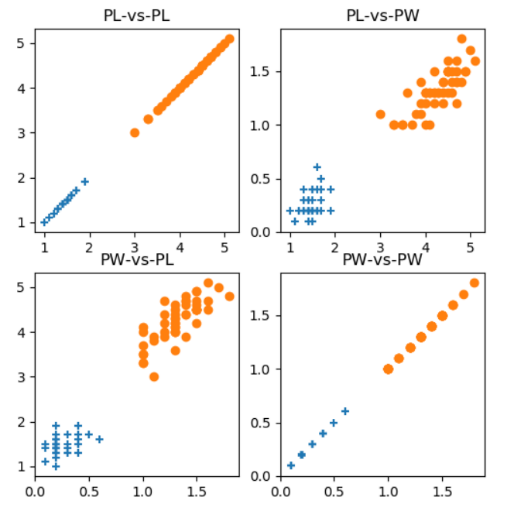
sepal length (SL) sepal width (SW) petal length (PL) petal width (PW)



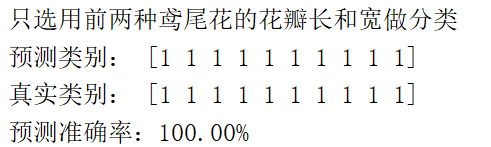


其中setosa（蓝色加号），versicolor（红色圆点），virginica（绿色三角）

实验选用前两种花的PL（花瓣宽度），PW（花瓣长度）属性进行分类，如下图所示（左），分类结束后绘制的分类器的直线（右）

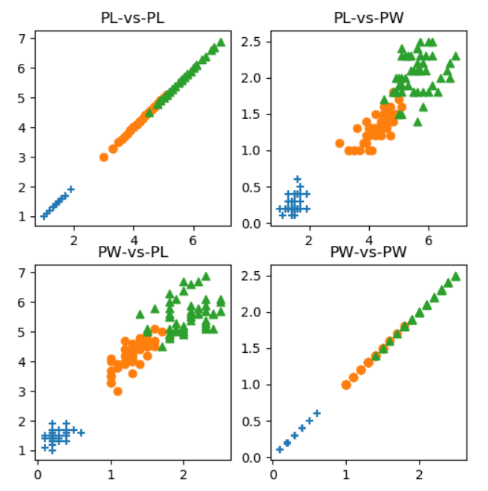


预测准确度：

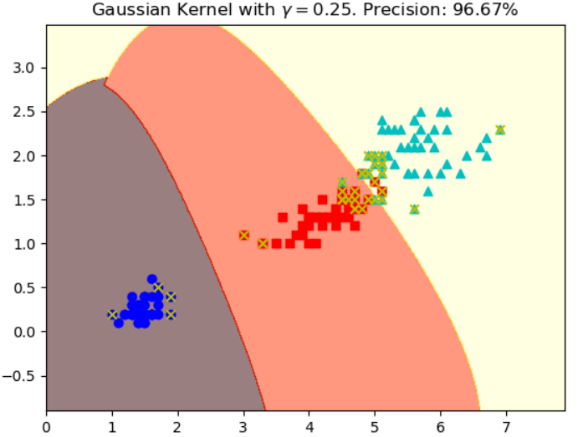
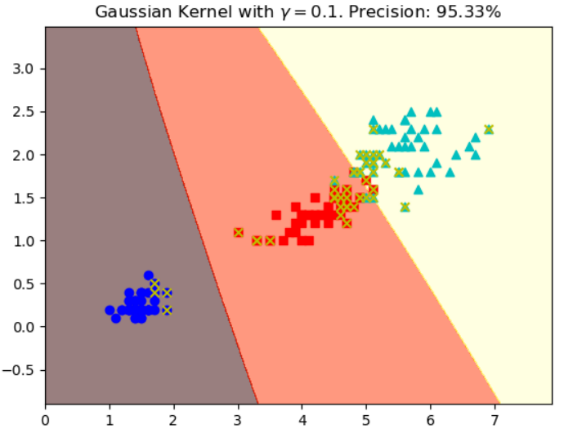


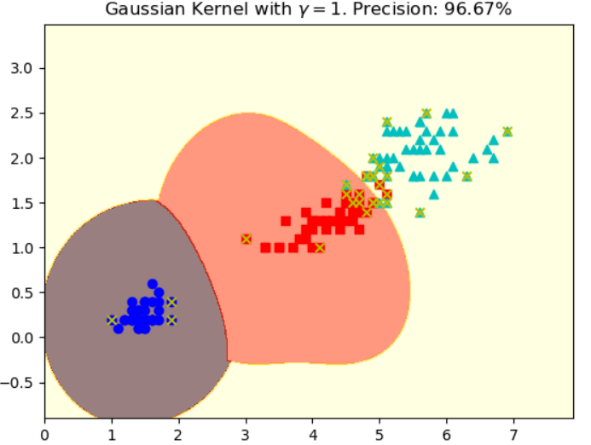
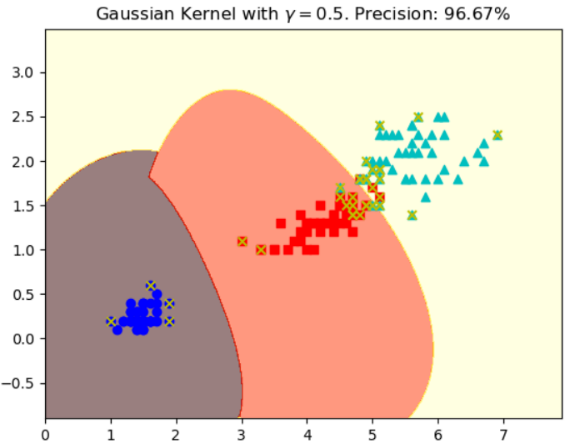
数据集介绍请使用代码：

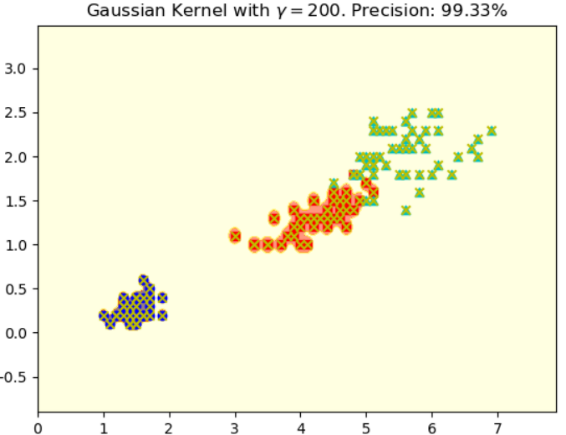
* 1. 对三种花进行分类



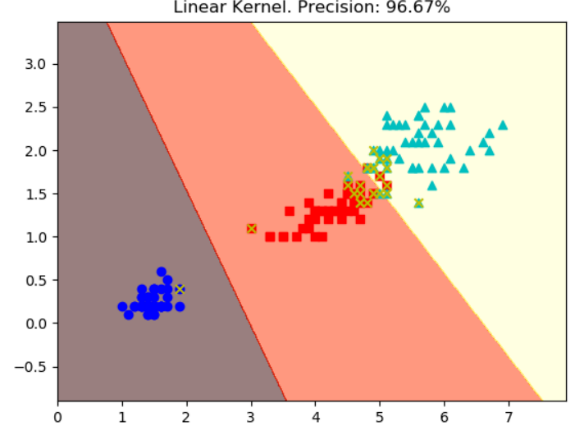
采用高斯核函数：可见，γ越大，区分的也越好，支持向量的个数也越多，但是会出现严重的过拟合



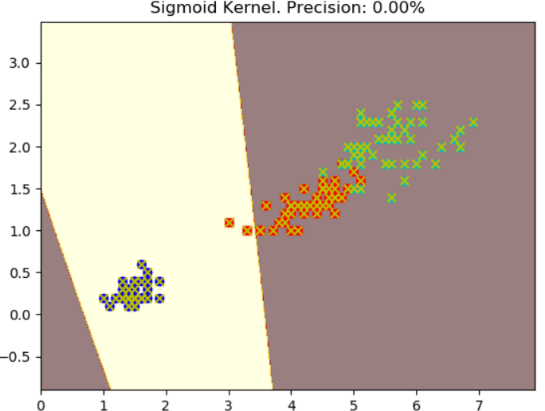




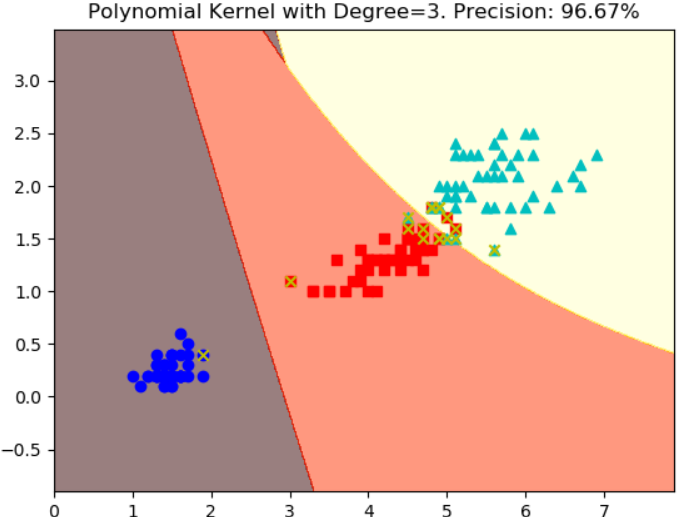
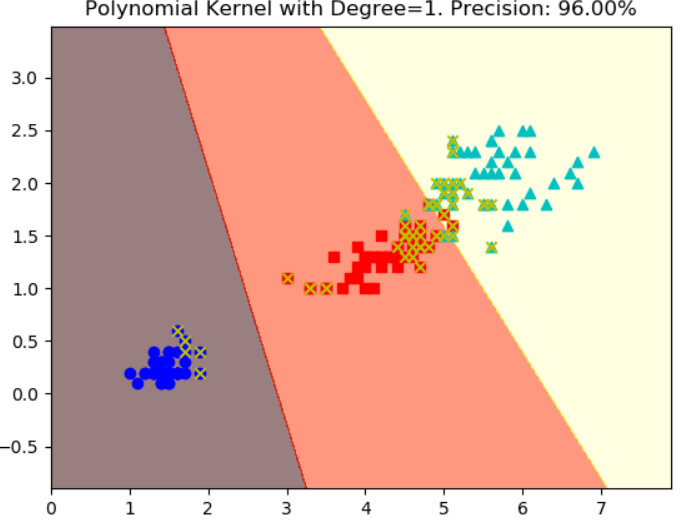
线性核函数：和高斯核函数中γ值很小的时候一样

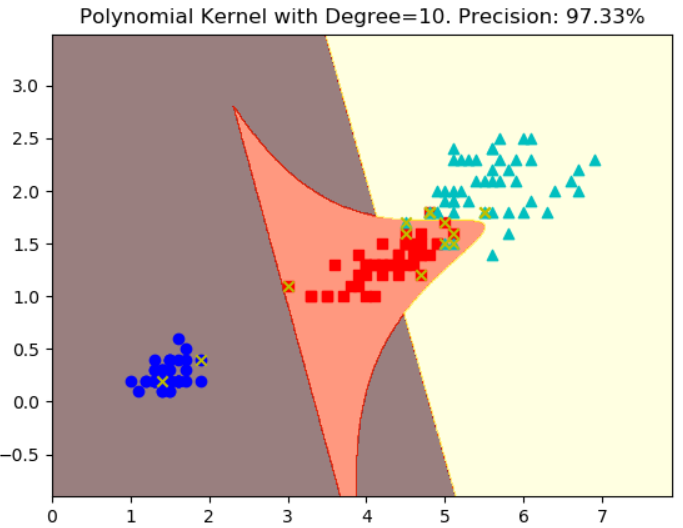
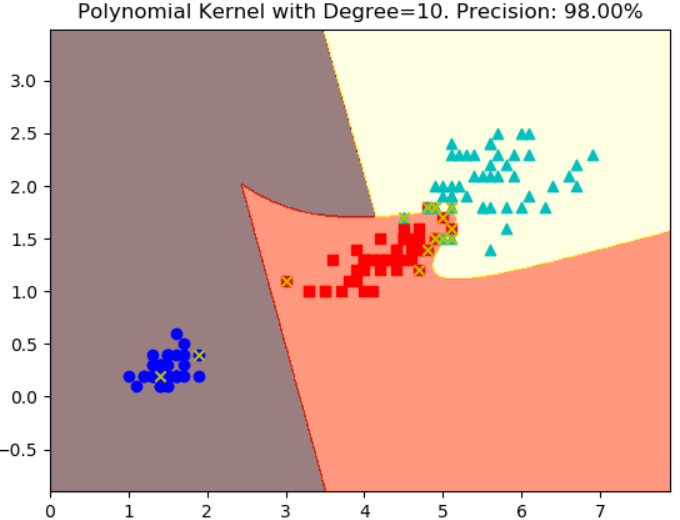


Sigmod核函数：准确率百分之0，可能是参数没调好



采用多边形核函数





其中最后一幅图，单独设置了γ等于0.1

1. **心得体会**

通过SVM支持向量机的实验，我初识了人工智能的算法基础

1. **源代码**

引用详见：https://blog.csdn.net/Douhh\_sisy/article/details/80622694

示例程序1：

**import** sklearn.datasets **as** datas  
**from** sklearn **import** svm  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
  
iris\_data = datas.load\_iris()  
print(**"鸢尾花数据集介绍"**)  
print(iris\_data[**"DESCR"**])  
print(**"鸢尾花的类别："**, \*iris\_data[**"target\_names"**])  
print(**"鸢尾花的属性"**, \*iris\_data[**"feature\_names"**])  
  
num\_samples, num\_features = iris\_data[**"data"**].shape  
print(**"鸢尾花的数量："**, num\_samples)  
num\_classs = 2  
feature\_names = [**"SL"**, **"SW"**, **"PL"**, **"PW"**]  
fig, ax = plt.subplots(num\_features, num\_features, figsize=(10, 10))  
markers = **"+o^"  
for** i **in** range(num\_features):  
 **for** j **in** range(num\_features):  
 **for** k **in** range(num\_classs):  
 ax[i][j].scatter(iris\_data[**"data"**][(k \* 50):(k \* 50 + 50), i],  
 iris\_data[**"data"**][(k \* 50):(k \* 50 + 50), j],  
 marker=markers[k])  
 ax[i][j].set\_title(feature\_names[i] + **"-vs-"** + feature\_names[j])  
plt.subplots\_adjust(wspace=0.5, hspace=0.5)  
plt.show()  
  
num\_classs = 2  
print(**"只选用前两种鸢尾花的花瓣长和宽做分类"**)  
fig, ax = plt.subplots(2, 2, figsize=(6, 6))  
**for** i **in** range(2):  
 **for** j **in** range(2):  
 **for** k **in** range(num\_classs):  
 ax[i][j].scatter(iris\_data[**"data"**][(k \* 50):(k \* 50 + 50), i + 2],  
 iris\_data[**"data"**][(k \* 50):(k \* 50 + 50), j + 2],  
 marker=markers[k])  
 ax[i][j].set\_title(feature\_names[i + 2] + **"-vs-"** + feature\_names[j + 2])  
plt.show()  
  
num\_classs = 2  
xTrain = iris\_data[**"data"**][0:(50 \* num\_classs - 10), 2:4]  
yTrain = iris\_data[**"target"**][0:(50 \* num\_classs - 10)]  
xPredict = iris\_data[**"data"**][(50 \* num\_classs - 10):(50 \* num\_classs), 2:4]  
yPredict = iris\_data[**"target"**][(50 \* num\_classs - 10):50 \* num\_classs]  
  
svm\_classifier = svm.SVC(kernel=**"linear"**)  
*# It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable.*svm\_classifier.fit(xTrain, yTrain)  
predict = svm\_classifier.predict(xPredict)  
print(**"预测类别："**, predict)  
print(**"真实类别："**, yPredict)  
correct = len(np.argwhere(predict == yPredict)) / len(yPredict) \* 100  
print(**"预测准确率：{:.2f}%"**.format(correct))  
  
plt.scatter(iris\_data[**"data"**][0:50, 2], iris\_data[**"data"**][0:50, 3], marker=**"+"**)  
plt.scatter(iris\_data[**"data"**][50:90, 2], iris\_data[**"data"**][50:90, 3], marker=**"o"**)  
coef = svm\_classifier.coef\_[0]  
support\_vectors = svm\_classifier.support\_vectors\_  
print(support\_vectors)  
class\_0\_sv = support\_vectors[0]  
class\_1\_sv = support\_vectors[1]  
x = np.array([1.5, 2.5])  
y = (np.dot(coef, class\_0\_sv) - x \* coef[0]) / coef[1]  
plt.plot(x, y)  
print(coef)  
print(y)  
x = np.array([3, 4])  
y = (np.dot(coef, class\_1\_sv) - x \* coef[0]) / coef[1]  
plt.plot(x, y)  
plt.title(**"PL-vs-PW"**)  
plt.show()

程序2：

**import** sklearn.datasets **as** datas  
**from** sklearn **import** svm  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
  
**def** plot\_hyperplane(clf, X, y,  
 h=0.02,  
 draw\_sv=**True**,  
 title=**'hyperplan'**):  
 *# create a mesh to plot in* x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1 *#找到X中最大和最小值* y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  
 xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), *#形成一个网格，从最大值到最小值之间，步长为h的网格* np.arange(y\_min, y\_max, h))  
 plt.title(title)  
 plt.xlim(xx.min(), xx.max())  
 plt.ylim(yy.min(), yy.max())  
 *#plt.xticks(()) #这个可以取消坐标  
 #plt.yticks(())* Z = clf.predict(np.c\_[xx.ravel(), yy.ravel()])*#叠加两个矩阵,生成一个n\*2的列表，n为xx矩阵的长\*宽，2为叠加的结果* Z = Z.reshape(xx.shape) *#将预测结果生成等高线图* plt.contourf(xx, yy, Z, cmap=**'hot'**, alpha=0.5)  
 *#等高线图，xx为X轴数据，yy为Y轴数据，Z为高度数据* markers = [**'o'**, **'s'**, **'^'**]  
 colors = [**'b'**, **'r'**, **'c'**]  
 labels = np.unique(y) *#去除重复元素  
 #y是一个真实结果的矩阵，y==0生成一个矩阵，结果矩阵中所有等于0的位置都变为True，其余都为False  
 #X[y==0] X里面放一个结果矩阵，每个为True对应位置全都取出来，其余全都不要* **for** label **in** labels:  
 plt.scatter(X[y==label][:, 0],  
 X[y==label][:, 1],  
 c=colors[label],  
 marker=markers[label])  
 **if** draw\_sv:  
 sv = clf.support\_vectors\_ *#画出支持向量* plt.scatter(sv[:, 0], sv[:, 1], c=**'y'**, marker=**'x'**)  
 plt.show()  
  
iris\_data = datas.load\_iris()  
num\_samples, num\_features = iris\_data[**"data"**].shape  
num\_classs = 3  
feature\_names = [**"SL"**, **"SW"**, **"PL"**, **"PW"**]  
markers = **"+o^"**print(**"只选用前两种鸢尾花的花瓣长和宽做分类"**)  
fig, ax = plt.subplots(2, 2, figsize=(6, 6))  
**for** i **in** range(2):  
 **for** j **in** range(2):  
 **for** k **in** range(num\_classs):  
 ax[i][j].scatter(iris\_data[**"data"**][(k \* 50):(k \* 50 + 50), i + 2],  
 iris\_data[**"data"**][(k \* 50):(k \* 50 + 50), j + 2],  
 marker=markers[k])  
 ax[i][j].set\_title(feature\_names[i + 2] + **"-vs-"** + feature\_names[j + 2])  
plt.show()  
  
xTrain = iris\_data[**"data"**][:, 2:4]  
print(xTrain)  
yTrain = iris\_data[**"target"**]  
svm\_classifier = svm.SVC(kernel=**"poly"**,degree=10,gamma=0.1)  
*# It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable.*svm\_classifier.fit(xTrain, yTrain)  
predict = svm\_classifier.predict(xTrain)  
print(**"预测类别："**, predict)  
print(**"真实类别："**, yTrain)  
correct = len(np.argwhere(predict == yTrain)) / len(yTrain) \* 100  
print(**"预测准确率：{:.2f}%"**.format(correct))  
plot\_hyperplane(svm\_classifier, xTrain, yTrain, h=0.01,  
 title=**'Polynomial Kernel with Degree=10. Precision: {:.2f}%'**.format(correct))