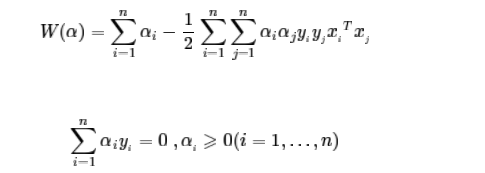
**作业6.2**

1. **实验要求**

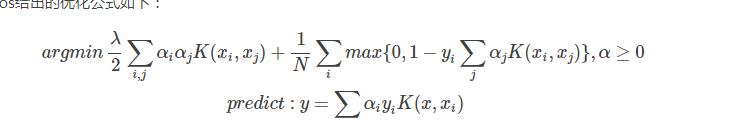
试使用LIBSVM，在西瓜数据集3.0a上分别用线性核和高斯核训练一个SVM，并比较其支持向量的差别。

1. **实验原理**

对于一般的svm，SVM优化的目标为下图中的第一个式子所示。



对于核函数，SVM优化目标如下第一个式子：



求解得出相应的a即可得到结果。

**3、实验过程**

本题有两种思路，第一种思路是使用开源机器学习库tensorflow进行求解，得出a,具体过程如下：

1. 准备好数据集、标记。
2. 根据上述公式设定好优化目标。
3. 利用tensorflow训练优化目标，得到a,进而得到w,b.
4. 画图并得到svm的边界结果。

代码如下：

**from** numpy **import** \*  
**import** tensorflow **as** tf  
**import** matplotlib.pyplot **as** plt  
data = mat([[0.697,0.460,1],  
 [0.774,0.376,1],  
 [0.634,0.264,1],  
 [0.608,0.318,1],  
 [0.556,0.215,1],  
 [0.403,0.237,1],  
 [0.481,0.149,1],  
 [0.437,0.211,1],  
 [0.666,0.091,0],  
 [0.243,0.267,0],  
 [0.245,0.057,0],  
 [0.343,0.099,0],  
 [0.639,0.161,0],  
 [0.657,0.198,0],  
 [0.360,0.370,0],  
 [0.593,0.042,0],  
 [0.719,0.103,0]])  
x=data[:,0:2].astype(float32)  
y=data[:,2].astype(float32)  
*#print(x[:,0])  
#print(shape(y))*sigma = 0.5  
kkx = square(tile(x[:,0].T,[x.shape[0],1])-tile(x[:,0],[1,x.shape[0]]))  
*#print(shape(kkx))*kkx += square(tile(x[:,1].T,[x.shape[0],1])-tile(x[:,1],[1,x.shape[0]]))  
kkx = sqrt(kkx)  
KX = exp(-sigma \* kkx )  
lam = 1./2.  
batch = x.shape[0]  
alpha = tf.Variable(tf.random\_uniform([batch,1],-1.0,1.0))  
alpha = tf.maximum(0.,alpha)  
loss = lam\*tf.reduce\_sum(tf.matmul(alpha,tf.transpose(alpha))\*KX)  
tmp = tf.matmul(KX.astype(float32), alpha)  
tmp = y\*tmp  
tmp = 1. - tmp  
tmp = tf.maximum(0.,tmp)  
tmp = 1./batch\*tf.reduce\_sum(tmp)  
loss += tmp  
optimizer = tf.train.GradientDescentOptimizer(0.2)  
train = optimizer.minimize(loss)  
init = tf.initialize\_all\_variables()  
sess = tf.Session()  
sess.run(init)  
**for** step **in** range(10000):  
 sess.run(train)  
resA = sess.run(alpha)  
*#print(resA.shape)*predict=multiply(resA,y)  
*#print(predict.shape)*predict=sum(multiply(predict,kkx),axis=0)  
predict = predict.T  
predict=tile(predict,[1,3])  
*#print(predict>0.1)*ax = array(x)  
predictSet1=ax[predict>0.1].reshape([-1,3])  
predictSet2=ax[predict<0.0].reshape([-1,3])  
fig = plt.figure()  
ax = fig.add\_subplot(211)  
ax.scatter(x=data[:,0],y=data[:,1])  
ax = fig.add\_subplot(212)  
ax.scatter(x=predictSet1[:,0],y=predictSet1[:,1])  
ax.scatter(x=predictSet2[:,0],y=predictSet2[:,1])  
fig.show()

第二种思路是使用机器学习库sklearn进行求解，具体过程如下：

1. 准备好数据集、标记。
2. 利用sklearn自带的机器学习库svm分别拟合两种核函数的数据。
3. 将两种核函数的预测结果代入自己设定的标记
4. 得到svm训练的两种核函数svm训练的边界图。

代码如下：

**from** numpy **import** \*  
**from** sklearn **import** svm  
**import** matplotlib.pyplot **as** plt  
*#data introduce*data = mat([[0.697,0.460,1],  
 [0.774,0.376,1],  
 [0.634,0.264,1],  
 [0.608,0.318,1],  
 [0.556,0.215,1],  
 [0.403,0.237,1],  
 [0.481,0.149,1],  
 [0.437,0.211,1],  
 [0.666,0.091,0],  
 [0.243,0.267,0],  
 [0.245,0.057,0],  
 [0.343,0.099,0],  
 [0.639,0.161,0],  
 [0.657,0.198,0],  
 [0.360,0.370,0],  
 [0.593,0.042,0],  
 [0.719,0.103,0]])  
x=data[:,0:2]  
y=data[:,2]  
*#linear kernal*clf1=svm.SVC(kernel=**'linear'**,C=1000)  
clf1.fit(x,y)  
*#RBF kernal*clf2=svm.SVC(kernel=**'linear'**, C=1000)  
clf2.fit(x,y)  
*#set test data*x\_min, x\_max = x[:, 0].min() - 1, x[:, 0].max() + 1  
y\_min, y\_max = x[:, 1].min() - 1, x[:, 1].max() + 1  
xx, yy = meshgrid(arange(x\_min, x\_max, 0.02),  
 arange(y\_min, y\_max, 0.02))  
t=arange(x\_min, x\_max, 0.02)  
*#draw picture*X0=data[0:8,:]  
X1=data[8:17,:]  
*#draw linear kernel picture*ax=plt.subplot(1,2,1)  
ax.scatter(X0[:,0],X0[:,1],c=**'r'**,label=**'+'**)  
ax.scatter(X1[:,0],X1[:,1],c=**'g'**,label=**'-'**)  
w = clf1.coef\_[0]  
a = -w[0] / w[1]  
y1 = a \* t- clf1.intercept\_[0] / w[1]  
plt.sca(ax)  
plt.plot(t,y1)  
plt.xlabel(**'density'**)  
plt.ylabel(**'ratio\_sugar'**)  
plt.legend()  
*#draw rbf kernel picture*ax1=plt.subplot(1,2,2)  
Z=clf2.predict(c\_[xx.ravel(),yy.ravel()]).reshape(xx.shape)  
plt.contour(xx, yy, Z)  
ax1.scatter(X0[:,0],X0[:,1],c=**'r'**,label=**'+'**)  
ax1.scatter(X1[:,0],X1[:,1],c=**'g'**,label=**'-'**)  
plt.sca(ax1)  
plt.xlabel(**'density'**)  
plt.ylabel(**'ratio\_sugar'**)  
plt.legend()  
plt.show()

**4、实验结果**

方法一本人未做出来，方法二结果如下图示：

