## 使用SVM算法确定预测股票涨跌情况

引子：对应一只股票，我们一般只希望预测它接下来的走势是涨是跌，属于分类问题。如果把股票数据集在一个平面表示出来，我们希望找到一条最合适的线将其划分为两个部分。

#### 一. SVM介绍

**1.为什么要研究线性分类？**

在讨论SVM之前首先研究，为什么要把数据集用**线性分类**分开？难道不可以用非线性分开吗？

首先，非线性分开的复杂度非常之大，而线性分开只需要一个平面。其次，非线性分开仅就二维空间而言，曲线，折线，双曲线，波浪线，以及毫无规律的各种曲线太多，没有办法进行统一处理。即使针对某一个具体问题处理得到了非线性分类结果，也无法很好的推广到其他情形，用非线性处理分类问题，耗时耗力。

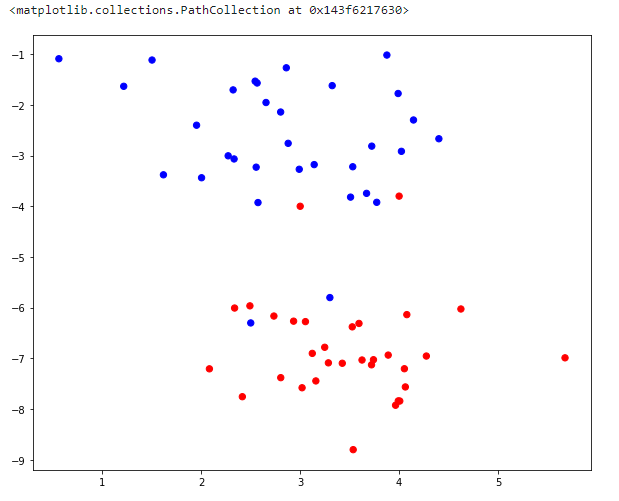
**2.SVM（Support Vector Machines）思想是什么**

2.1 硬间隔支持向量机

SVM中最关键的思想之一就是引入和定义了“间隔”这个概念。这个概念本身很简单，以二维空间为例，就是点到分类直线之间的距离。假设直线为y=wx+b，那么只要使所有正分类点到该直线的距离与所有负分类点到该直线的距离的总和达到最大，这条直线就是最优分类直线。这样，原问题就转化为一个约束优化问题，可以直接求解。这叫做硬间隔最大化，得到的SVM模型称作**硬间隔支持向量机**。

2.2 软间隔支持向量机

在实际应用中，我们得到的数据并不总是完美的线性可分的。



机器学习| 支持向量机详解(Python 语言描述) - Laugh's blog

其中可能会有个别噪声点，他们错误的被分类到了其他类中。如果将这些特异的噪点去除后，可以很容易的线性可分。但是，我们对于数据集中哪些是噪声点却是不知道的，如果以之前的方法进行求解，会无法进行线性分开。是不是就没办法了呢？假设在y=x+1直线上下分为两类，若两类中各有对方的几个噪点，在人的眼中，仍然是可以将两类分开的。这是因为在人脑中是可以容忍一定的误差的，仍然使用y=x+1直线分类，可以在最小误差的情况下进行最优的分类。同样的道理，我们在SVM中引入误差的概念，将其称作“**松弛变量**”。通过加入松弛变量，在原距离函数中需要加入新的松弛变量带来的误差，这样，最终的优化目标函数变成了两个部分组成：距离函数和松弛变量误差。这两个部分的重要程度并不是相等的，而是需要依据具体问题而定的，因此，我们加入权重参数C，将其与目标函数中的松弛变量误差相乘，这样，就可以通过调整C来对二者的系数进行调和。如果我们能够容忍噪声，那就把C调小，让他的权重降下来，从而变得不重要；反之，我们需要很严格的噪声小的模型，则将C调大一点，权重提升上去，变得更加重要。通过对参数C的调整，可以对模型进行控制。这叫做软间隔最大化，得到的SVM称作**软间隔支持向量机**。

2.3 非线性支持向量机

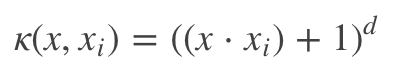
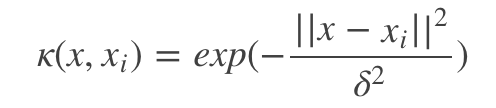
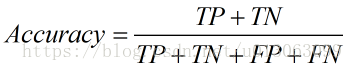
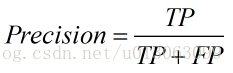
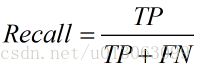
之前的硬间隔支持向量机和软间隔支持向量机都是解决线性可分数据集或近似线性可分数据集的问题的。但是如果噪点很多，甚至会造成数据变成了线性不可分的，那该怎么办？最常见的例子是在二维平面笛卡尔坐标系下，以原点(0,0)为圆心，以1为半径画圆，则圆内的点和圆外的点在二维空间中是肯定无法线性分开的。但是，学过初中几何就知道，对于圆圈内（含圆圈）的点：x^2+y^2≤1，圆圈外的则x^2+y^2＞1。我们假设第三个维度：z=x^2+y^2，那么在第三维空间中，可以通过z是否大于1来判断该点是否在圆内还是圆外。这样，在二维空间中线性不可分的数据在第三维空间很容易的线性可分了。这就是**非线性支持向量机**。

#### 二.模型搭建

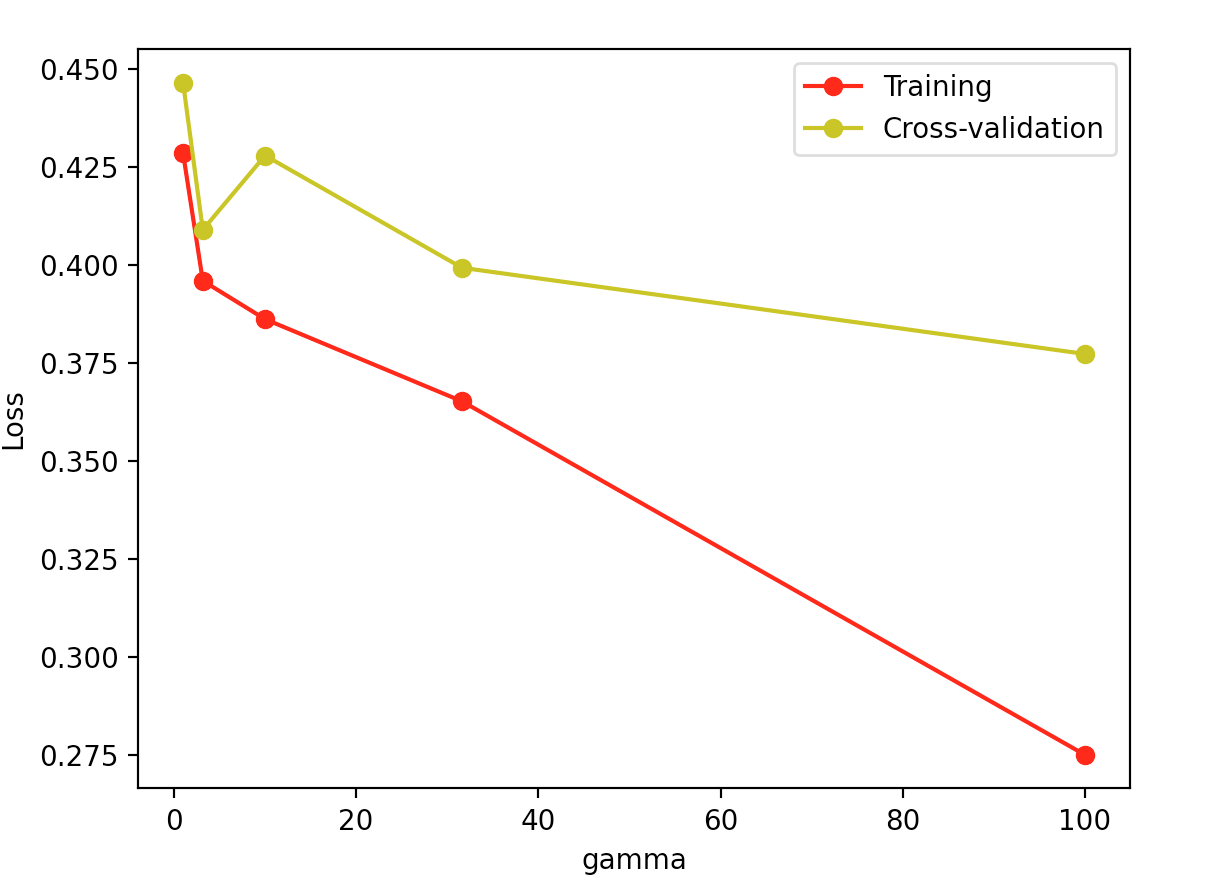
* 环境配置：采用Python3语言环境
* 用到的工具类：
  + Sklearn:机器学习模型封装库
  + matplotlib:画图
  + pandas：excel，csv文件数据读取
  + NumPy:科学计算的基础软件包

#### 三. 模型构建

SVM常用核函数介绍：

* 线性核函数
* image-20201202002012098
* image-20201202002012098
* 线性核，主要用于线性可分的情况，我们可以看到特征空间到输入空间的维度是一样的，其参数少速度快，对于线性可分数据，其分类效果很理想，因此我们通常首先尝试用线性核函数来做分类，看看效果如何，如果不行再换别的
* 多项式核函数
* 
* image-20201202001959273
* 多项式核函数可以实现将低维的输入空间映射到高纬的特征空间，但是多项式核函数的参数多，当多项式的阶数比较高的时候，核矩阵的元素值将趋于无穷大或者无穷小，计算复杂度会大到无法计算。
* 高斯（RBF）核函数
* 
* image-20201202001943633
* 高斯径向基函数是一种局部性强的核函数，其可以将一个样本映射到一个更高维的空间内，该核函数是应用最广的一个，无论大样本还是小样本都有比较好的性能，而且其相对于多项式核函数参数要少，因此大多数情况下在不知道用什么核函数的时候，优先使用高斯核函数。
* 核函数的选择有如下原则：
  + 如果特征的数量大到和样本数量差不多，则选用LR或者线性核的SVM；
  + 如果特征的数量小，样本的数量正常，则选用SVM+高斯核函数；
  + 如果特征的数量小，而样本的数量很大，则需要手工添加一些特征从而变成第一种情况。
* 本文中特征数据量为5，样本数据量相对正常，因此SVM内核选用RBF。
* 利用概念来评价分类器的优良：
* **1.准确率（Accuracy），**所有的预测正确（正类负类）的占总的比重。
* 
* img
* **2.精确率（Precision），**正确预测为正的占全部预测为正的比例。
* 
* img
* **3.召回率（Recall）**，正确预测为正的占全部实际为正的比例。
* 
* img
* **4.F1值（H-mean值）。**F1值为算数平均数除以几何平均数，且越大越好，将Precision和Recall的上述公式带入会发现，当F1值小时，True Positive相对增加，而false相对减少，即Precision和Recall都相对增加，即F1对Precision和Recall都进行了加权。
* #X,y为数据源，将数据源分为4个数据集，X\_train,y\_train用于训练模型，  
  #X\_test，y\_test用于检测模型运行情况,训练集与测试集比例为0.7比0.3  
  X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=.3)  
  #选定内核为高斯核,参数为默认参数  
  clf = SVC(kernel='rbf')   
  #用训练数据集训练模型  
  clf.fit(X\_train,y\_train)  
  #输出准确率，召回率，F1-score  
  print(classification\_report(clf.predict(X\_test),y\_test))  
  #利用五折交叉验证输出结果  
  scores = cross\_val\_score(clf,X,y,cv=5,scoring='accuracy')
* 第一次预测结果输出：
* precision recall f1-score   
    
   0.0 0.39 0.73 0.51   
   1.0 0.80 0.48 0.60   
    
   accuracy 0.56   
   macro avg 0.60 0.61 0.56   
  weighted avg 0.67 0.56 0.57
* 我们的训练集合由于是随机分配，为了尽可能避免出现不同数据对模型的影响，我们采用**K折交叉验证**对模型准确率进行判断。
* K折交叉验证，就是将数据集**等比例划分成K份**，以其中的**一份**作为测试数据，其他的**K-1份**数据作为训练数据。然后，这样算是**一次实验**，而K折交叉验证只有**实验K次**才算完成完整的一次，也就是说交叉验证实际是**把实验重复做了K次，每次实验都是从K个部分选取一份不同的数据部分作为测试数据（保证K个部分的数据都分别做过测试数据），剩下的K-1个当作训练数据，最后把得到的K个实验结果进行平分**。
* 准确率输出结果：
* [0.59375 0.546875 0.609375 0.55555556 0.53968254]  
  平均准确率：0.569047619047619
* 可以看到利用我们RBF中默认的参数，我们的预测结果并不是很理想。需要对高斯核中的核心参数进行调校。

#### 四.调参

* **验证曲线：**validation\_curve
* 误差由偏差（bias）、方差（variance）和噪声（noise）组成。同样的数据，我们用同样的模型，但是超参数却不同，会得到不同的拟合效果。
* SVM模型有两个非常重要的参数C与gamma。其中 C是惩罚系数，即对误差的宽容度。c越高，说明越不能容忍出现误差,容易过拟合。C越小，容易欠拟合。C过大或过小，泛化能力变差
* gamma是选择RBF函数作为kernel后，该函数自带的一个参数。隐含地决定了数据映射到新的特征空间后的分布，gamma越大，支持向量越少，gamma值越小，支持向量越多。支持向量的个数影响训练与预测的速度。
* 验证曲线的横轴为某个超参数的一系列值，由此来看不同参数设置下模型准确率，而不是不同训练集大小下的准确率。
* 从验证曲线上也可以看到随着超参数设置的改变，模型可能从欠拟合到合适再到过拟合的过程，进而选择一个合适的位置，来提高模型的性能。
* 接下来用验证曲线寻找误差最小的gamma值：
* # validation\_curve 要看的是 SVC() 的超参数 gamma，  
  # gamma 的范围是取 0.1到 10, 取5个点，  
  # 评分用的是 neg\_mean\_squared\_error  
  def validationCurveGamma():  
   param\_range = np.logspace(-1, 1, 5)  
   #对SVC()中gamma追踪  
   train\_loss,test\_loss=validation\_curve(  
   SVC(),X,y,param\_name='gamma',param\_range=param\_range,  
   cv=10,scoring='neg\_mean\_squared\_error')  
   #取训练集误差的平均值  
   train\_loss\_mean = -np.mean(train\_loss,axis=1)  
   #取测试集误差的平均值  
   test\_loss\_mean = -np.mean(test\_loss,axis=1)  
   #画图  
   plt.plot(param\_range,train\_loss\_mean,'o-',color="r",label="Training")  
   plt.plot(param\_range,test\_loss\_mean,'o-',color="y",label="Cross-validation")  
   plt.xlabel("gamma")  
   plt.ylabel("Loss")  
   plt.legend(loc="best")  
   plt.show()
* 
* image-20201202014031658
* x轴为gamma的取值范围0～100，y轴为损失精度，
* 红色线条代表训练集随着gamma的数值波动，损失精度的变化
* 绿色线条代表验证集随时gamma的数值波动，损失精度的变化
* 在x轴坐标为31左右的时候，两条线段是趋于收敛的。所以我们gamma值取31。
* 通过同样的validation\_curve验证，得出C的取值为20。
* 最终我们将模型参数设置为C=20，gamma=31
* SVC(kernel='rbf', C=20,gamma=31)

#### 五.最终结果输出

* **准确率/召回率/f1**

precision recall f1-score   
  
 0.0 0.81 0.72 0.76   
 1.0 0.69 0.79 0.73   
 accuracy 0.75   
 macro avg 0.75 0.75 0.75   
weighted avg 0.76 0.75 0.75

* **五折交叉验证**

[0.859375 0.65625 0.84375 0.74603175 0.77777778]  
五折交叉验证平均分：0.7766369047619047

* **学习曲线**（learnning\_curve）

learnning\_curve 一种用来判断训练模型的一种方法，通过观察绘制出来的学习曲线图，我们可以比较直观的了解到我们的模型处于一个什么样的状态，如：过拟合（overfitting）或欠拟合（underfitting）

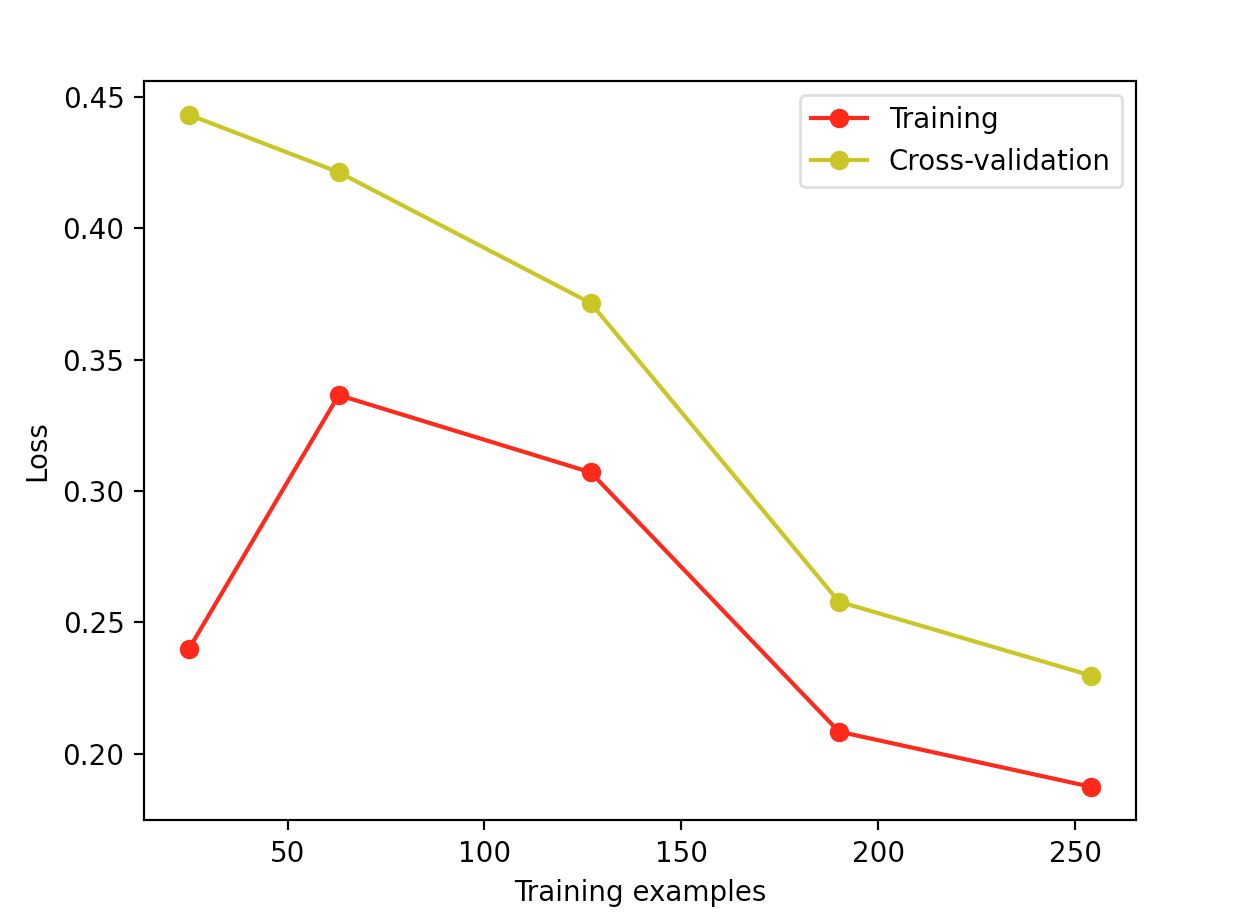


image-20201201232448575

从图中可以看出，模型数据集在0-20时，test\_loss值误差较大，随着不断训练，

训练集跟测试集的误差在不断缩小。

* 结论：使用验证曲线调整参数后，SVM使用核能够良好预测股票涨跌

## 随机森林预测

#取出数据集  
X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.3,random\_state=None)  
#使用默认函数创建随机森林  
clf = RandomForestClassifier()  
clf.fit(X\_train,y\_train)  
predictions = clf.predict(X\_test)  
#查看准确率，召回率，F1  
print(classification\_report(y\_test,predictions))  
print("AC",accuracy\_score(y\_test,predictions))  
#使用五折交叉验证  
scores = cross\_val\_score(clf,X,y,cv=5,scoring='accuracy')  
print("五折交叉验证分数",scores)  
print("五折交叉验证平均分",scores.mean())

效果出奇的好：

precision recall f1-score   
  
 0.0 0.98 1.00   
 1.0 1.00 0.98   
  
 accuracy   
 macro avg 0.99 0.99   
weighted avg 0.99 0.99   
  
AC 0.9895833333333334  
五折交叉验证 [0.984375 0.96875 1. 1. 1]  
平均分 0.990625

学习曲线：

