# ARIMA模型与SVR模型

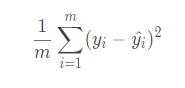
## 摘要

ARIMA模型与SVR模型各有优缺点，但由于分别对线性模型与非线性模型处理分别具有优势，他们之间存在优势互补。本文通过同例样本分别使用这两种不同模型的优缺点进行预测，探究这两种模型的预测优劣

本文将数据进行同等预处理后，分别采用两个模型的最优参数进行预测。最后将预测值与结果通过评估均方差mse的方式评估优劣。

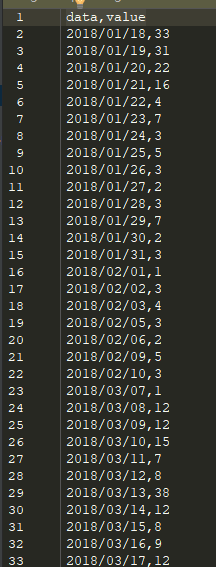
## MSE均方误差介绍

MSE用于衡量预测值和真实值之间的离差度其公式如下

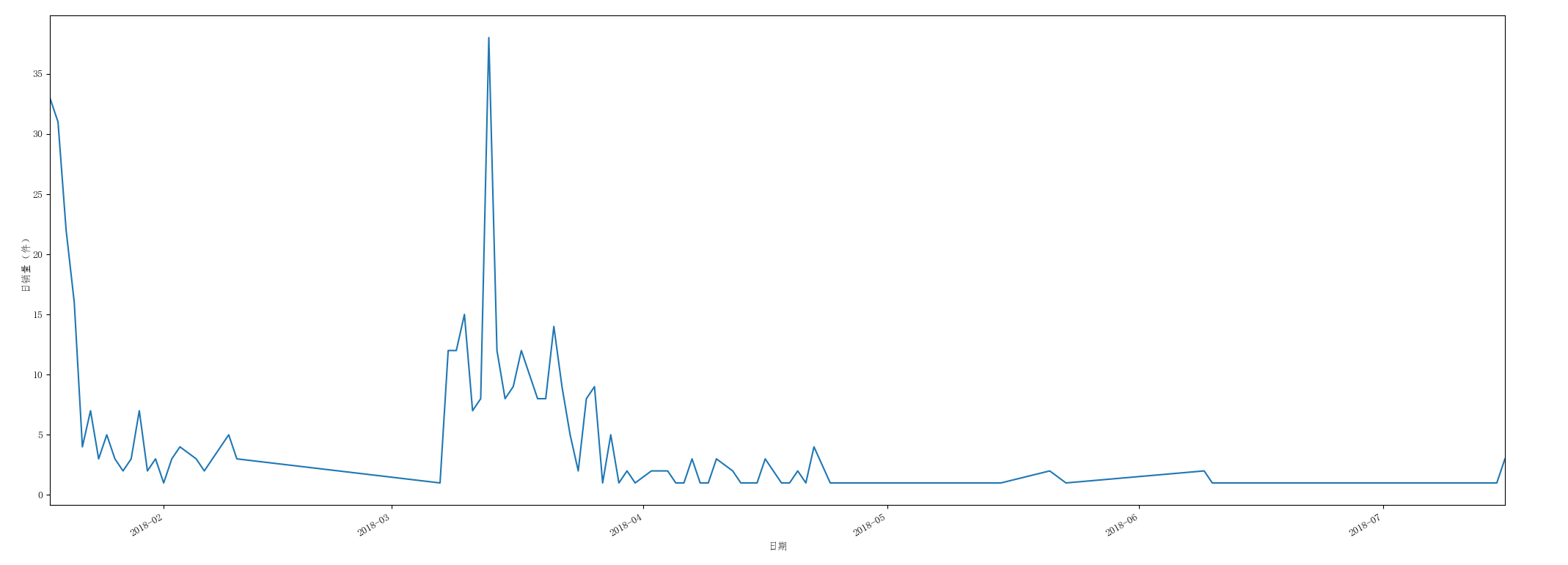


是预测值，为真实值，m为数据大小

# 样本数据



本文选举电商销量对应时间序列数据，作出数据散点图，观察得数据无明显季节性



SS73210商品2018.1.17-2019.3.12每日销量时序图

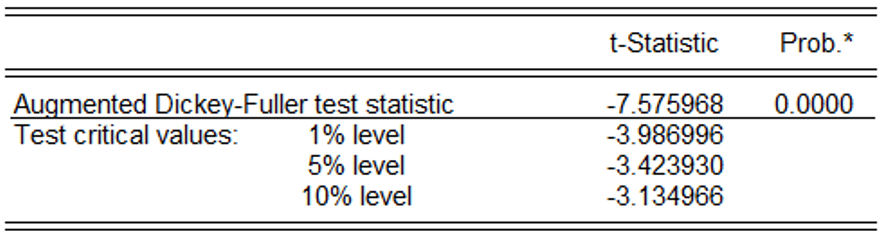
## 数据预处理

样本区间是2018年1月18-2019年3月10日期间每日的销量，记为时间序列。对数据分析，由于预测的数据需要保证在未来一段时间的精度，保留时间序列的连续性，所以需要剔除异常值。6月缺失日期太多无法保证连续性，为了保证时间的连续性和数据的可靠性，对6月舍弃处理。

## 平稳性评估及平稳化处理

散点图可以看出该序列在0附近随机波动，波动具有稳定性没有明显的趋势变动，数据为平稳时间序列。

由于散点图带有一定的主观性，需要采用统计检验方法加以判断验证，因此对序列做单位根检验(ADF)，检验统计量结果如图3所示。



ADF检验结果

可以看出检验统计量小于1%、 5%、10%显著水平下的临界值（图所示），因而序列为平稳时间序列

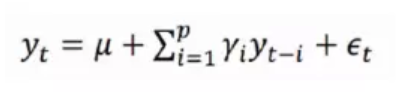
**此部分代码如下**：

'''对时间序列ADF检验'''  
train=read\_csv('../data/testData.csv', header=0, parse\_dates=[0], index\_col=0, squeeze=*True*, date\_parser=parser)  
result = ts.adfuller(train, 1)  
print(result)

# ARIMA模型的建立

## 自回归模型AR

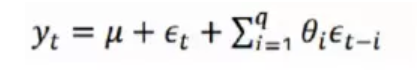
自回归模型首先需要确定一个阶数p，表示用几期的历史值来预测当前值。p阶自回归模型的公式定义为：



上式中yt是当前值,u是常数项,p是阶数 γi是自相关系数,Et是误差。

## 移动平均模型MA

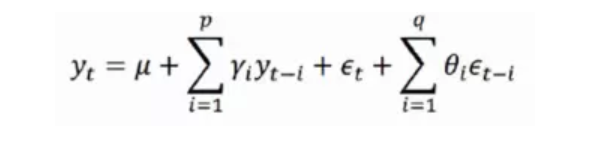
移动平均模型关注的是自回归模型中的误差项的累加 ，q阶自回归过程的公式定义如下：



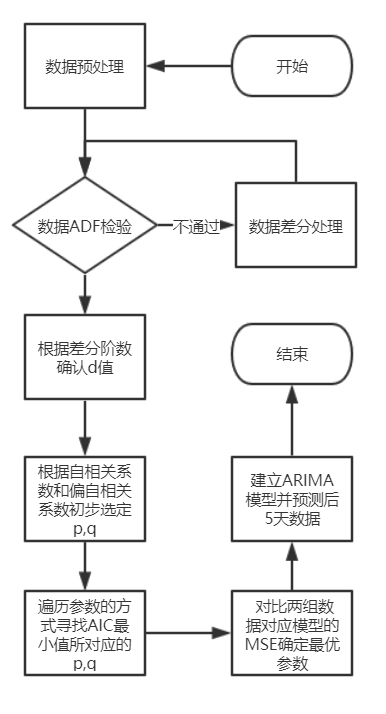
移动平均法能有效地消除预测中的随机波动。

## 自回归移动平均模型ARMA

自回归模型AR和移动平均模型MA模型相结合，我们就得到了自回归移动平均模型ARMA(p,q)，计算公式如下：



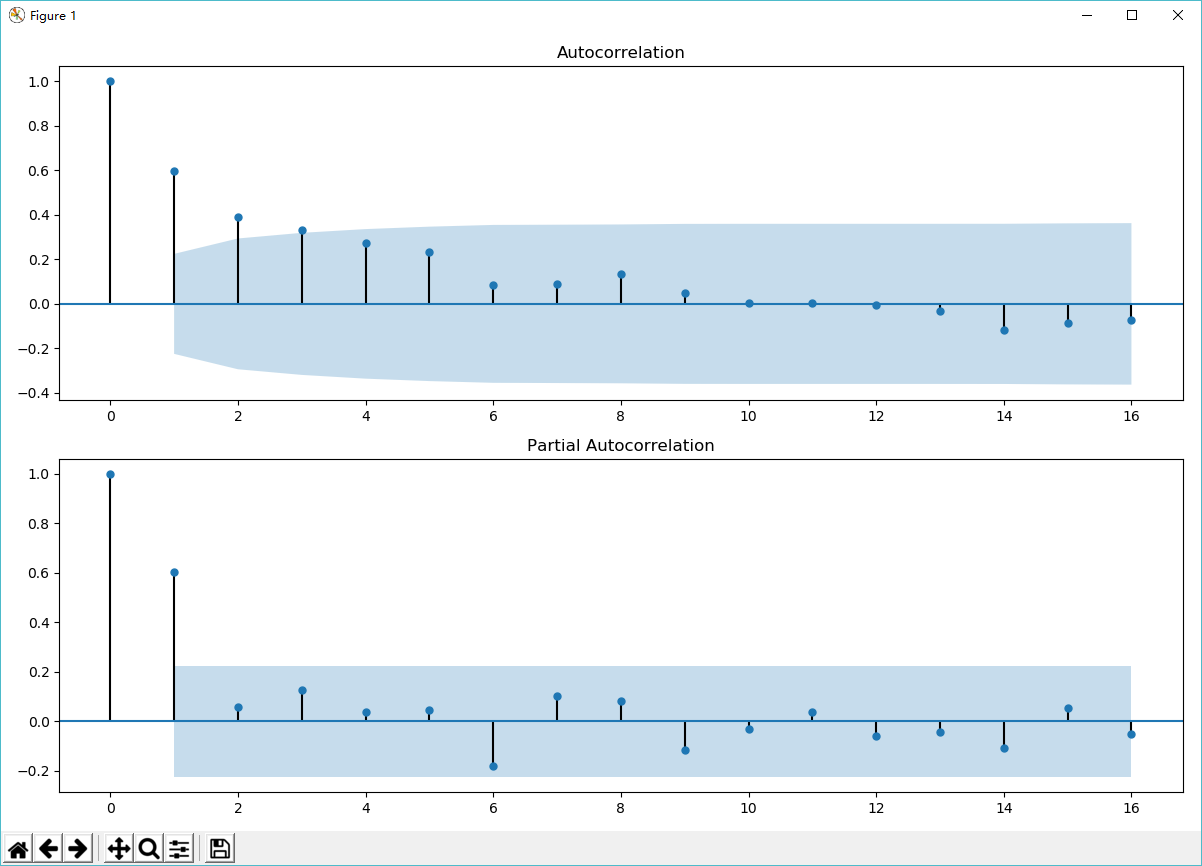
## ARIMA构建流程图



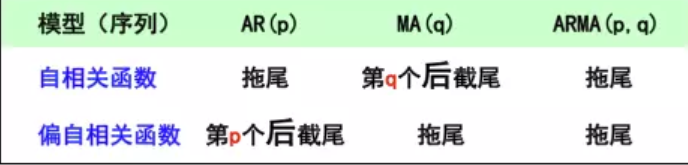
模型建立流程图

## 参数预估及检验

|  |  |  |
| --- | --- | --- |
| ARMA模型选择 | | |
| ACF | PACF | 模型 |
| 拖尾 | P阶截尾 | AR(p) |
| Q阶截尾 | 拖尾 | MA(q) |
| 拖尾 | 拖尾 | ARMA(p,q) |



时间序列的的自相关图和偏相关图



参数取值规定

画出自相关图和偏自相关图，从图中可以看出，PAC序列1、2阶偏自相关系数超出±2倍估计标准差，2阶以后偏自相关系数在±2倍估计标准差以内，并且迅速减少至0，即偏自相关函数2阶以后截尾；同理，PAC序列超出5%样本相关系数落在±2倍估计标准差以外，即自相关函数扫尾，结合表可初步确定p=1或2，q=0。而采用AIC准则遍历AIC最小值得出p=6，q=0。

综上候选模型为ARIMA(1,0,0),ARIMA(2,0,0)，ARIMA(6,0,0)。

为检验参数预估准确性，尝试拟合候选模型ARIMA(1,0,0),ARIMA(2,0,0)，ARIMA(6,0,0)，最终根据MSE准则，计算不同p,q值组合所对应的MSE值，评价模型优劣。下表为候选模型MSE值。

|  |  |
| --- | --- |
| 候选模型MSE值 | |
| ARIMA(1,0,0) | 0.085 |
| ARIMA(2,0,0) | 0.063 |
| ARIMA(6,0,0) | 0.066 |

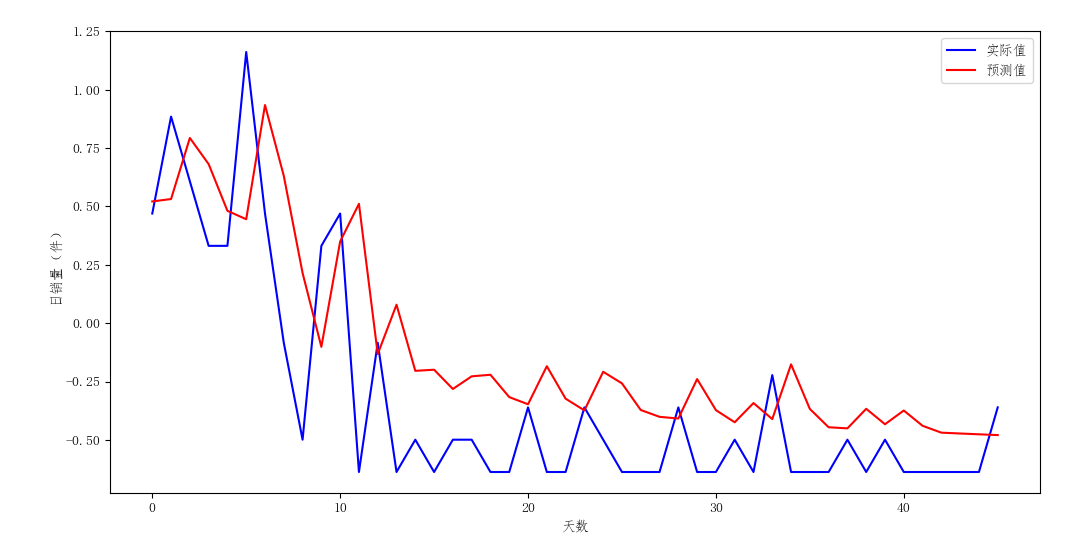
由表可看出，当p=1，q=0时，MSE值为0.085；当p=2，q=0时，MSE值为0.063；当p=6，q=0时，MSE值为0.066。当p=2时，MSE值较小，所以确立模型为ARIMA(2,0,0 )。

**相关代码如下：**

'''通过AIC准则寻找最优参'''  
*def* findC(*series*):  
 temp = 1000000  
 ansp = 0  
 ansq = 0  
 ansd = 0  
 *for* p *in* range(0, 8):  
 *for* q *in* range(0, 8):  
 # if p+q!=0:  
 *try*:  
 testModel = ARIMA(*series*, order=(p, 0, q))  
 testModel\_fit = testModel.fit(disp=0)  
 aic = testModel\_fit.aic  
 *if* aic < temp:  
 temp = aic  
 ansp = p  
 ansq = q  
 ansd = 0  
 *except*:  
 *continue  
 return* ansp,ansd,ansq

'''比较三个候选参数'''  
series = read\_csv('../data/testData.csv', header=0, parse\_dates=[0], index\_col=0, squeeze=*True*, date\_parser=parser)  
X=preprocessing.scale(series.values)  
mse = buildArima.evaluate\_arima\_model(X,(1,0,0))  
print("p=1,d=0,q=0 mse= %.3f" %mse)  
mse = buildArima.evaluate\_arima\_model(X,(2,0,0))  
print("p=2,d=0,q=0 mse= %.3f" %mse)  
mse = buildArima.evaluate\_arima\_model(X,(6,0,0))  
print("p=6,d=0,q=0 mse= %.3f" %mse)

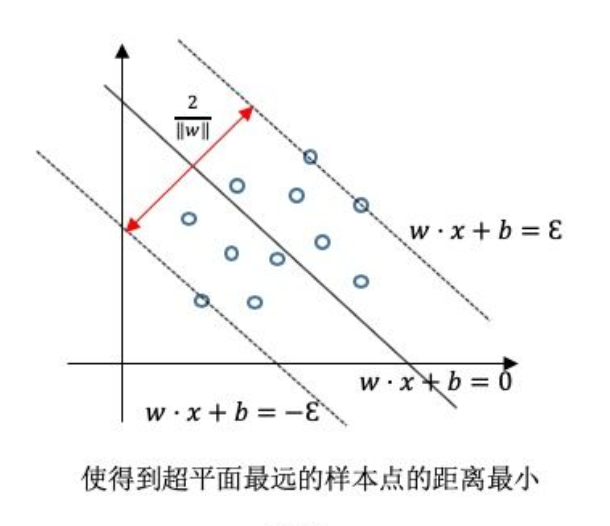
## 模型检验

选取出最优参数之后，将模型拟合并采用测试样本前30条数据预测后40条数据

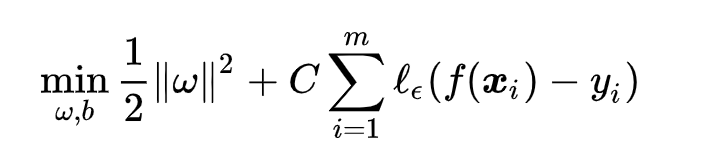
ARIMA拟合图

拟合结果MSE = 0.139

# SVR模型建立

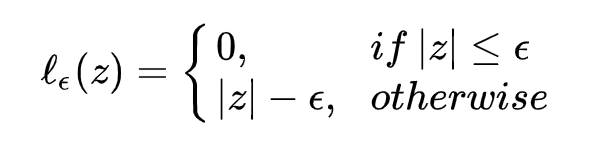


SVR问题可以形式化为



其中min1/2||w||^2为超平面的最小距离，C为正则化常数

Le为函数如下



可以理解为当点落在距离超平面的|z|中时，不损失，若落在之外，则执行相应损失处理

## 获取时间序列数据

*def* read\_csv(*path*):  
 csv\_data = pd.read\_csv(*path*) # 读取训练数据  
 print(csv\_data.data.size)  
 data=[]  
 value = []  
 *for* i *in* range(0,csv\_data.data.size):  
 data.append(i)  
 value.append(csv\_data.value[i])  
 *return* data,value

利用pandas中的read\_csv读取数据，将时间序列存储至date，销量存储值value并返回

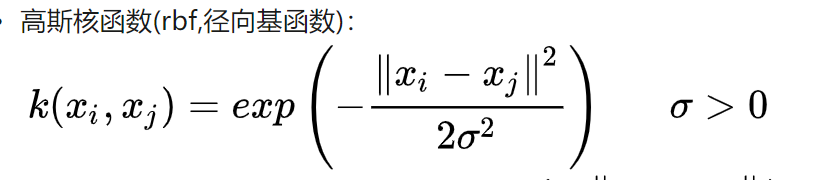
## 建立SVR模型

svr\_rbf = SVR(kernel='rbf', C=*c\_parameter*, gamma=*gamma\_paramenter*)

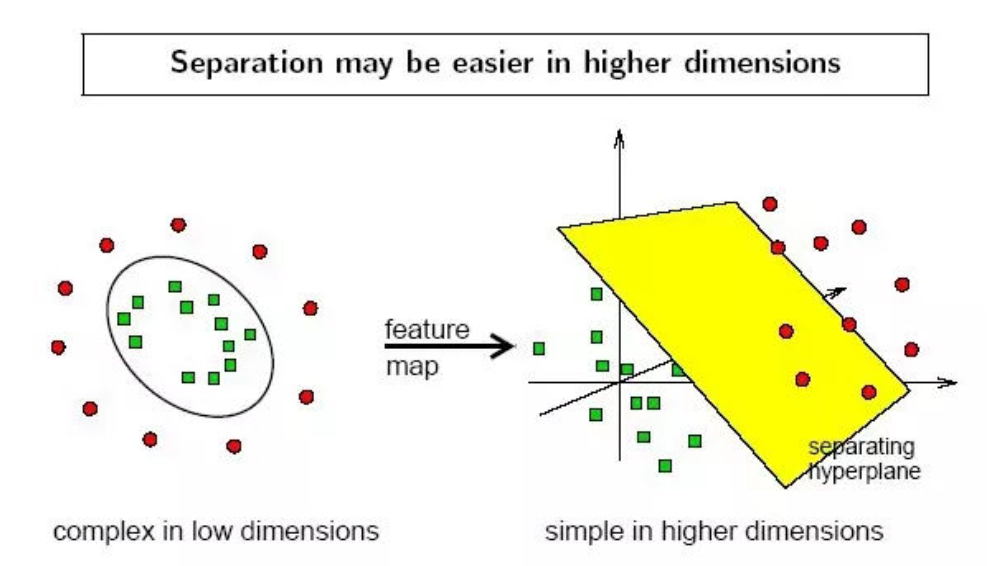
利用sklearn。Svm中的SVR函数构建SVR模型对象，需要三个参数分别是kernel、gamma以及C

### 关于核函数的选取

这其中kernel参数指定要在算法中使用的内核类型。它必须是'linear'，'poly'，'rbf'，'sigmoid'，'precomputed'或者callable之一。如果没有给出，将使用'rbf'。如果给出了callable，则它用于预先计算内核矩阵。在这里我选择了rbf核函数



核函数可以使得数据被映射到高维空间中，使其变得线性可分

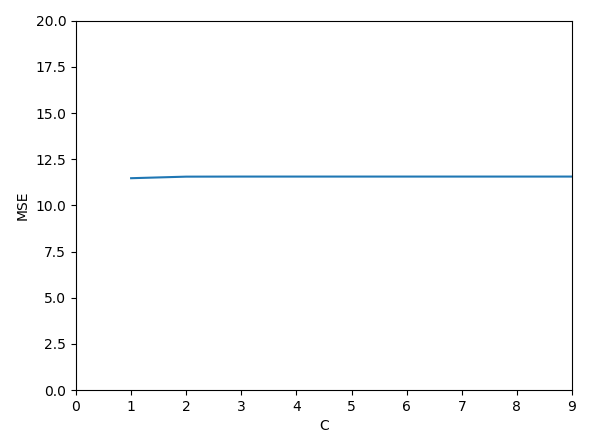


### C的取值

C是错误惩罚参数，这里取值为1.0，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

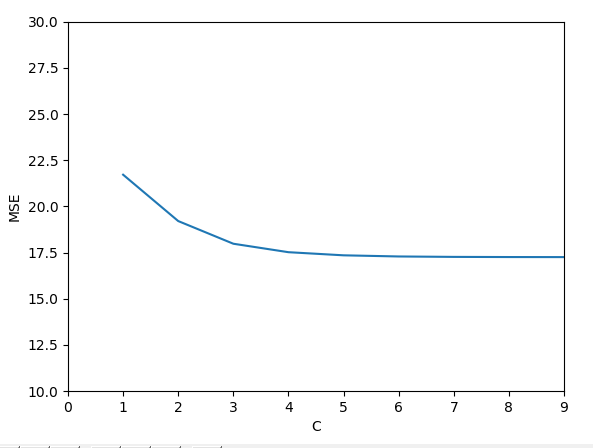
下表为通过测试咋在gamma恒定为1的情况下各C取值情况下的mse值

|  |  |
| --- | --- |
| C | Mse |
| 1 | 11.472 |
| 2 | 11.556 |
| 3 | 11.560 |
| 4 | 11.560 |
| 5 | 11.560 |
| 6 | 11.560 |
| 7 | 11.560 |
| 8 | 11.560 |
| 9 | 11.560 |

**下图为可视化结果**

可以看到在gamma恒定为1的情况下，随着惩罚指数变大的情况下，mse值并没有明显变化，这样数据也就无观测意义

此时将gamma 恒定为10，再测试一遍



可以看到随着C的增大MSE值在逐渐减少，最后趋于平稳，

**可以得出结论：**惩罚系数C不可取的过小，过小会导致模型无法正常拟合，也不可取得过大。且C对于结果的影响在Gamma取值过小的情况下几乎为零

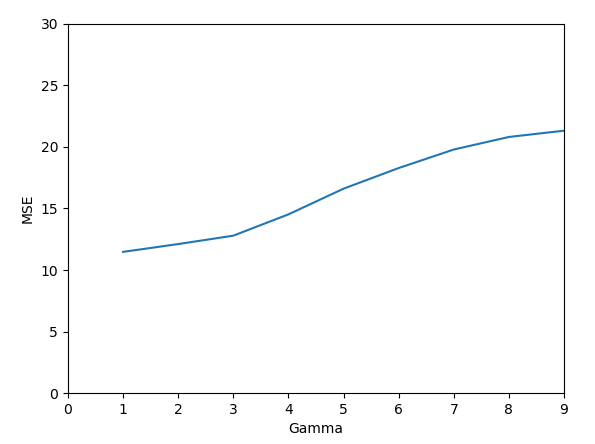
### Gamma取值

Gamma是选择RBF函数作为kernel核函数后，gamma为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。

下表为通过测试咋在C恒定为1的情况下各Gamma取值情况下的mse值

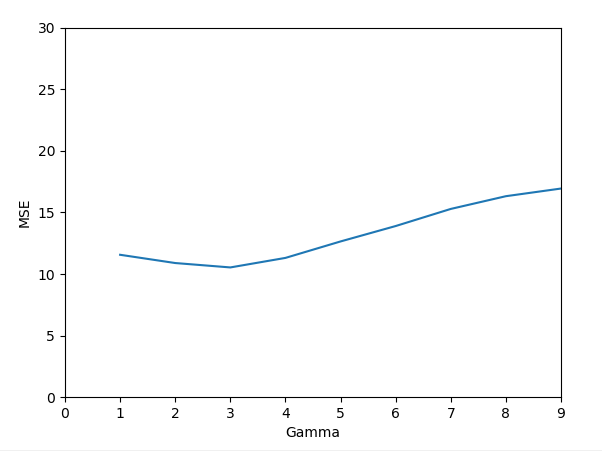
|  |  |
| --- | --- |
| Gamma | Mse |
| 1 | 11.472 |
| 2 | 12.106 |
| 3 | 12.786 |
| 4 | 14.516 |
| 5 | 16.602 |
| 6 | 18.274 |
| 7 | 19.779 |
| 8 | 20.798 |
| 9 | 21.308 |

**将表中数据可视化结果如下**



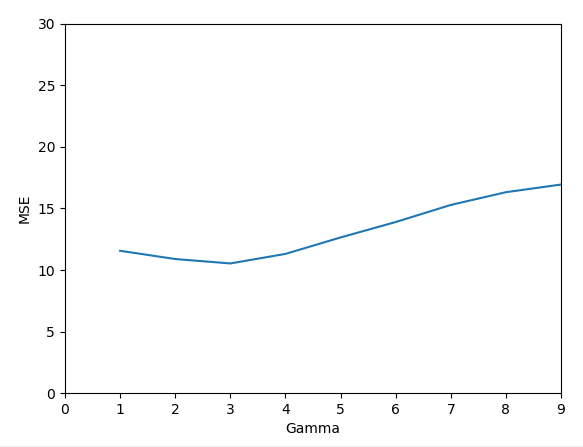
若此时将C的取值增大至10

**可视化结果如下**



C增大至100

**可视化结果如下**



可以看到图中mse值随着gamma取值的增大而增大。

**可以得出结论：**预测结果的准确度随着gamma取值的增大而减小，且gamma对结果的影响不随着C的改变而改变

此部分代码如下：

*def* testC(*gamma*):  
 print("when Gamma=%d "%*gamma*)  
 cs= []  
 mses = []  
 *for* c *in* range(1,10):  
 X\_data, Y\_data, X\_prediction, y\_prediction, error, mse = sv.svm\_timeseries\_prediction(data, value, *gamma*, c)  
 print("C= %.3f" %c)  
 cs.append(c)  
 print("mse = %.3f" %mse)  
 mses.append(mse)  
 plt.plot(cs,mses)  
 plt.axis([0,9,10,30])  
 plt.xlabel('C')  
 plt.ylabel('MSE')  
 plt.show()  
  
*def* testGamma(*c*):  
 print("when c=%d "%*c*)  
 gammas = []  
 mses = []  
 *for* gamma *in* range(1,10):  
 X\_data, Y\_data, X\_prediction, y\_prediction, error, mse = sv.svm\_timeseries\_prediction(data, value, gamma, *c*)  
 print("Gamma= %d" %gamma)  
 print("mse = %.3f" %mse)  
 gammas.append(gamma)  
 mses.append(mse)  
 plt.plot(gammas, mses)  
 plt.axis([0, 9, 0, 30])  
 plt.xlabel('Gamma')  
 plt.ylabel('MSE')  
 plt.show()

## 选取最优的C与gamma

经过上述步骤初步探究了C与gamma取值对于结果的影响，这一步就要选择最优的参数组合了。通过遍历比较MSE的方式可以得出最佳的参数组合

**此部分代码如下：**

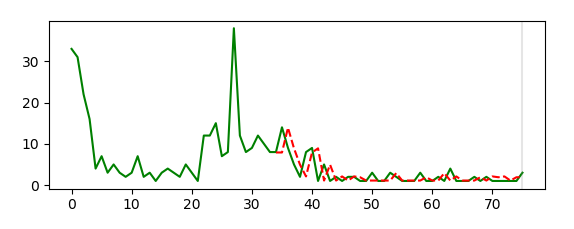
*import* SVM.svmprediction *as* sv  
  
data,value = sv.read\_csv("../data/testData.csv")  
  
temp\_mse = 10000  
*for* c\_paramenter *in* range(1,10):  
 *for* gamma\_paramenter *in* range(1,10):  
 X\_data,Y\_data,X\_prediction,y\_prediction,error,mse = sv.svm\_timeseries\_prediction(data,value,c\_paramenter,gamma\_paramenter)  
 *if*(mse<temp\_mse):  
 temp\_mse = mse  
 temp\_c = c\_paramenter  
 temp\_gamma = gamma\_paramenter  
  
print("best mse:")  
print(temp\_mse)  
print("best c:")  
print(c\_paramenter)  
print("best gamma:")  
print(gamma\_paramenter)

最终的出的最优参数为C=9，gamma = 9，此参数组合得出的mse值为10.533（数据未归一化）

## 模型**拟合并预测**

采用前30份数据进行训练后对后40份数据进行预测，并求出拟合值。

可视化结果如下



mse为0.010，比最优参数的ARIMA模型低

# 结论

在同一数据，且进行平稳性检验的前提下SVR模型得出的拟合值为0.010.ARIMA模型得出的拟合值为0.139，综合得出结论：在相同平稳时间序列的情况下，经过同同样的归一化处理后SVR模型的拟合值误差更小

# 参考文献

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html