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。

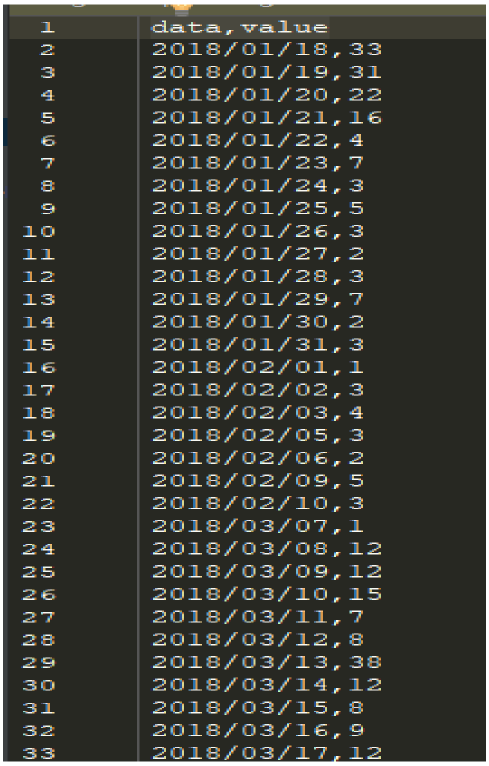
MSE 均方误差介绍

MSE 用于衡量预测值和真实值之间的离差度其公式如下

$$\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

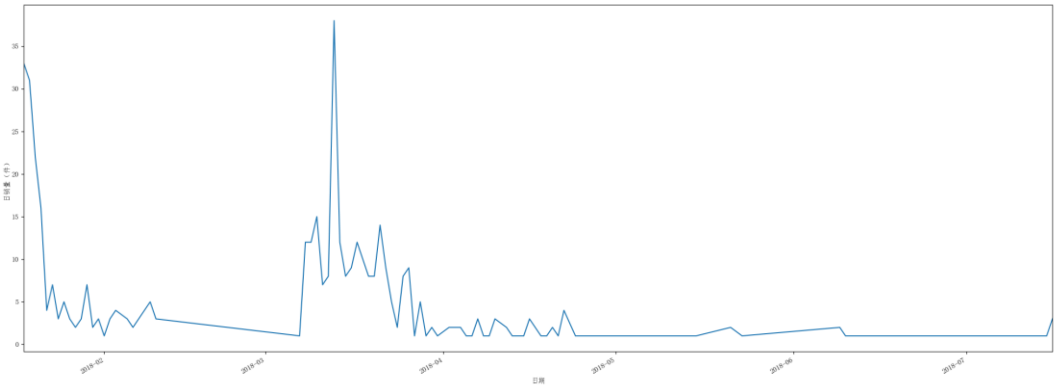
y_i是预测值，$\hat{y}_i$为真实值，m 为数据大小

样本数据



```
data,value
2018/01/18,33
2018/01/19,31
2018/01/20,22
2018/01/21,16
2018/01/22,4
2018/01/23,7
2018/01/24,3
2018/01/25,5
2018/01/26,3
2018/01/27,2
2018/01/28,3
2018/01/29,7
2018/01/30,2
2018/01/31,3
2018/02/01,1
2018/02/02,3
2018/02/03,4
2018/02/05,3
2018/02/06,2
2018/02/09,5
2018/02/10,3
2018/03/07,1
2018/03/08,12
2018/03/09,12
2018/03/10,15
2018/03/11,7
2018/03/12,8
2018/03/13,38
2018/03/14,12
2018/03/15,8
2018/03/16,9
2018/03/17,12
```

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$\{x_i\}$做单位根检验(ADF)，检验统计量结果如图 3 所示。

		t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic		-7.575968	0.0000
Test critical values:	1% level	-3.986996	
	5% level	-3.423930	
	10% level	-3.134966	

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：

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \epsilon_t$$

上式中 y_t 是当前值, u 是常数项, p 是阶数 γ_i 是自相关系数, E_t 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：

$$y_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

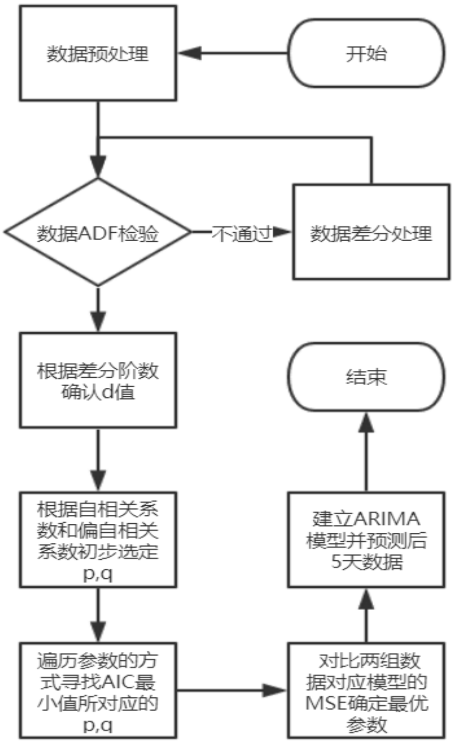
移动平均法能有效地消除预测中的随机波动。

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：

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

ARIMA 构建流程图

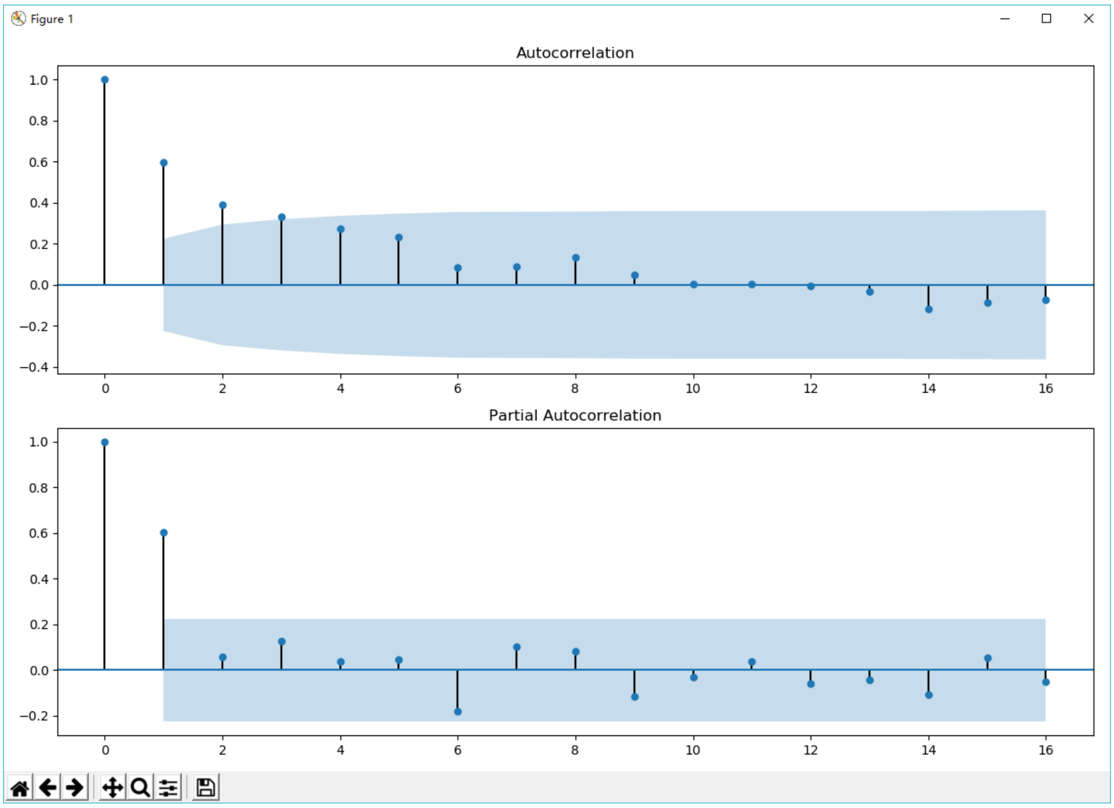


模型建立流程图

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

模型（序列）	AR(p)	MA(q)	ARMA(p, q)
自相关函数	拖尾	第q个后截尾	拖尾
偏自相关函数	第p个后截尾	拖尾	拖尾

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

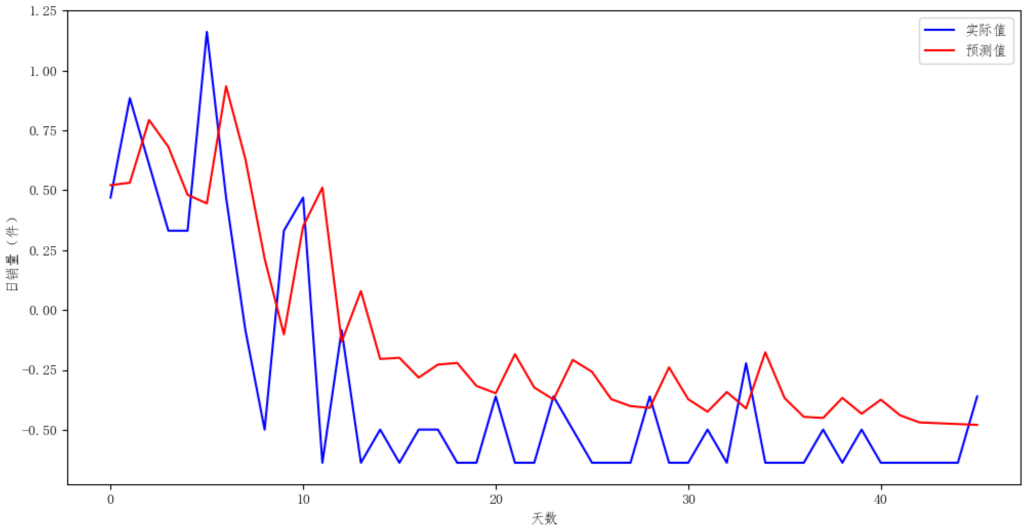
由表可看出，当 p=1，q=0 时，MSE 值为 0.085；当 p=2，q=0 时，MSE 值为 0.063；当 p=6，q=0 时，MSE 值为 0.066。当 p=2 时，MSE 值较小，所以确立模型为 ARIMA(2,0,0)。

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

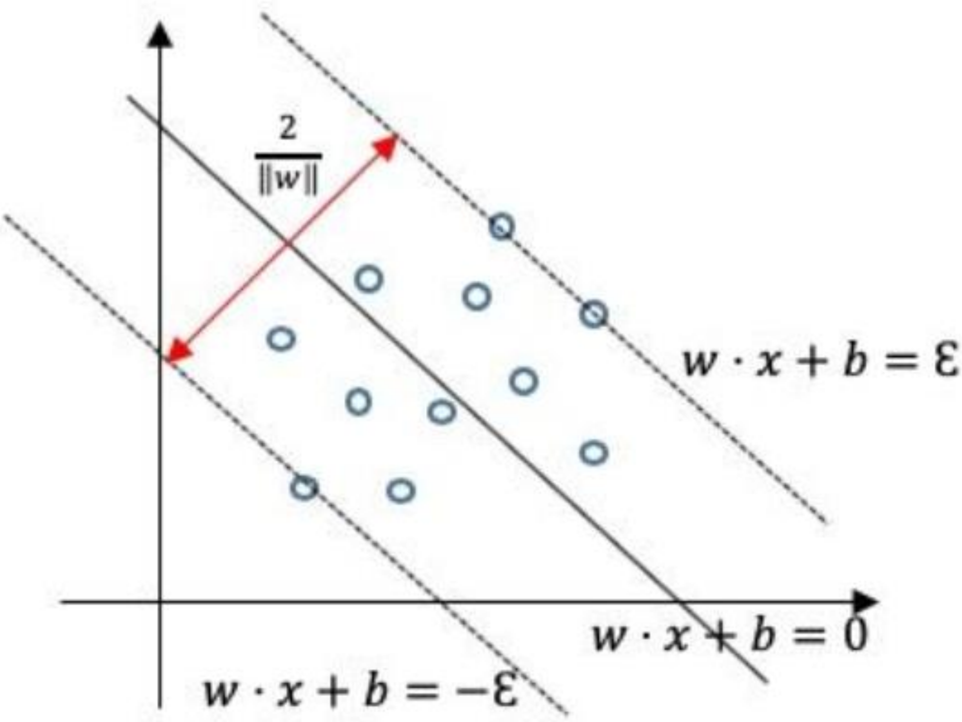
选取出最优参数之后，将模型拟合并采用测试样本前 30 条数据预测后 40 条数据



ARIMA 拟合图

拟合结果MSE = 0.139

SVR 模型建立



使得到超平面最远的样本点的距离最小

SVR 问题可以形式化为

$$\min_{\omega,b} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m} \ell_\epsilon(f(\boldsymbol{x}_i) - y_i)$$

其中 min1/2||w||^2 为超平面的最小距离，C 为正则化常数

Le 为 ε-不敏感损失 函数如下

$$\ell_\epsilon(z) = \begin{cases} 0, & if\ |z| \le \epsilon \\ |z| - \epsilon, & otherwise \end{cases}$$

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

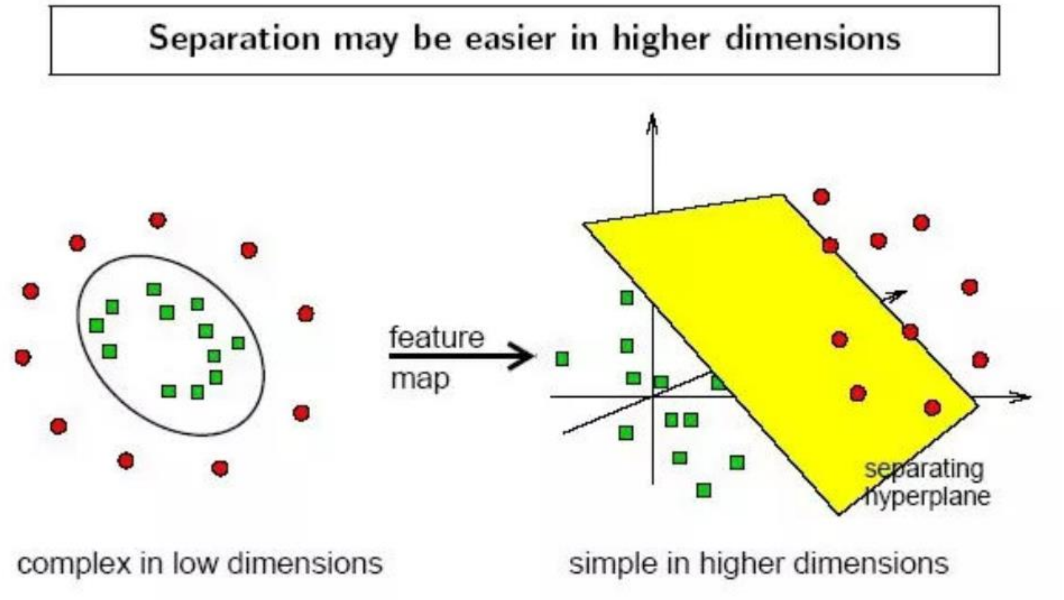
关于核函数的选取

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

- 高斯核函数(rbf,径向基函数):

$$k(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \qquad \sigma > 0$$

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

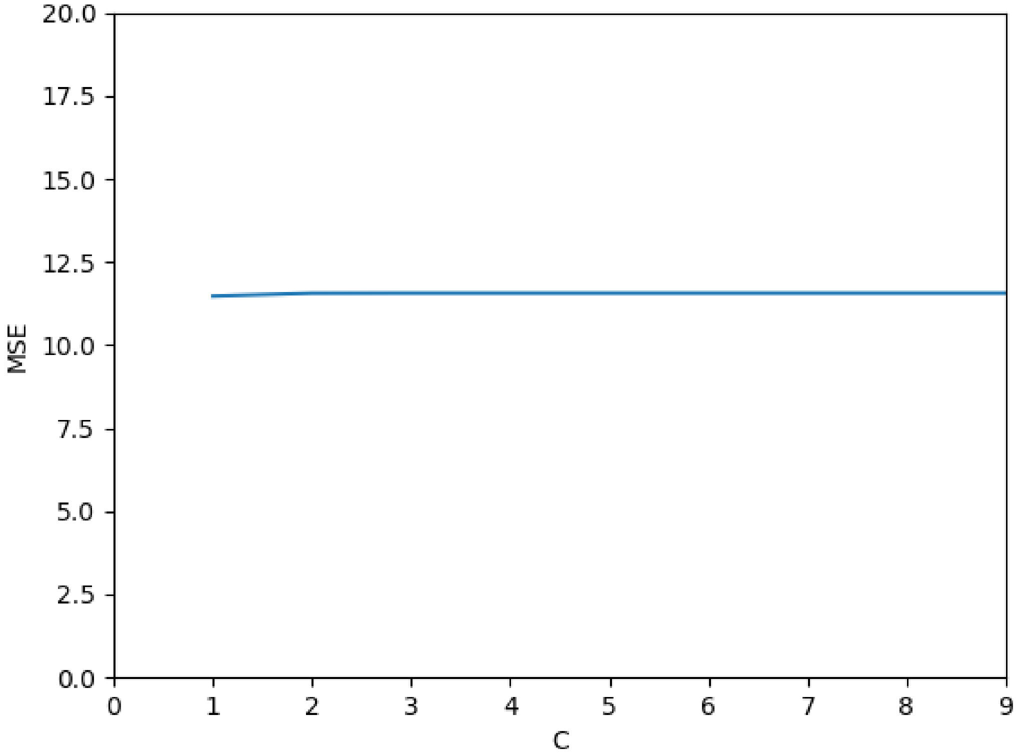


C 的取值

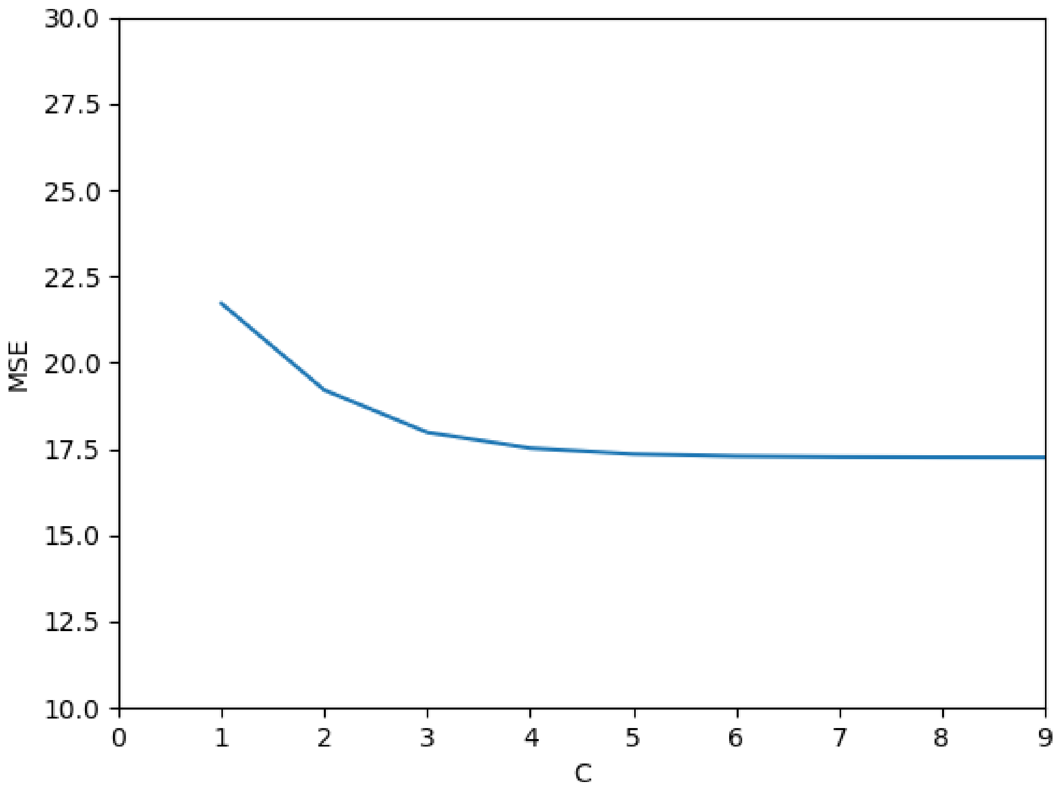
C 是错误惩罚参数，这里取值为 1.0，c 越高，说明越不能容忍出现误差,容易过拟合。C 越小，容易欠拟合。C 过大或过小，泛化能力变差
下表为通过测试咋在 gamma 恒定为 1 的情况下各 C 取值情况下的 mse 值

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


20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
MSE
0
1
2
3
4
5
6
7
8
9
C

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

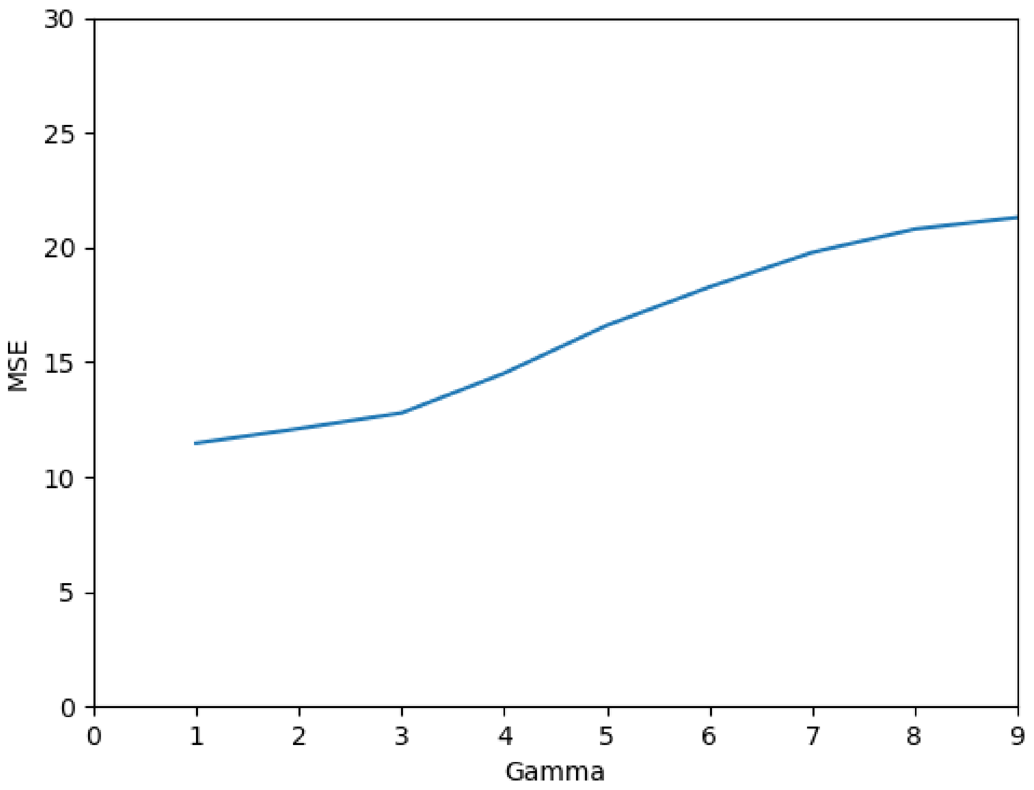
Gamma 取值

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

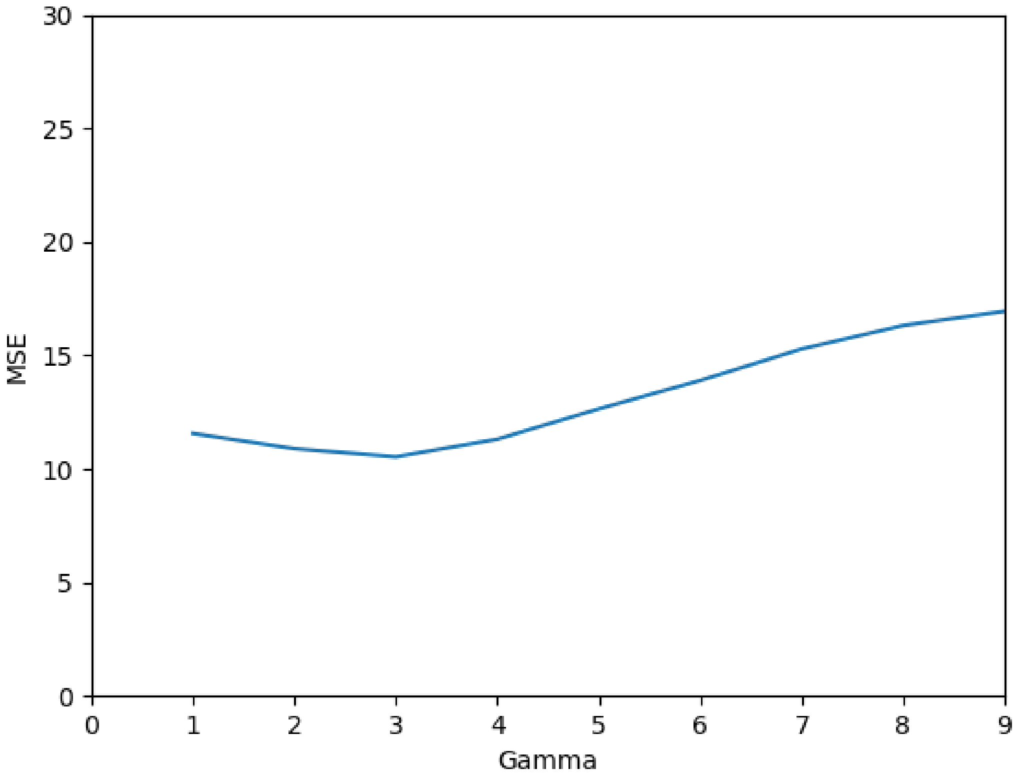
8	20.798
9	21.308

将表中数据可视化结果如下

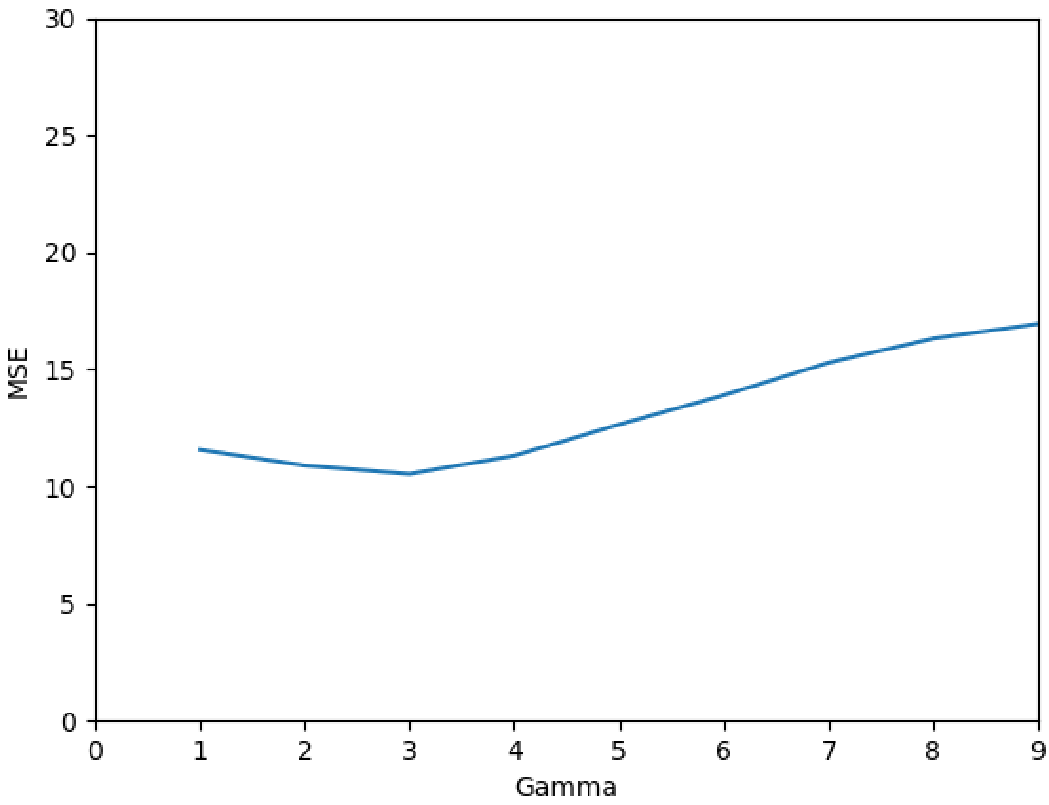


若此时将 C 的取值增大至 10

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

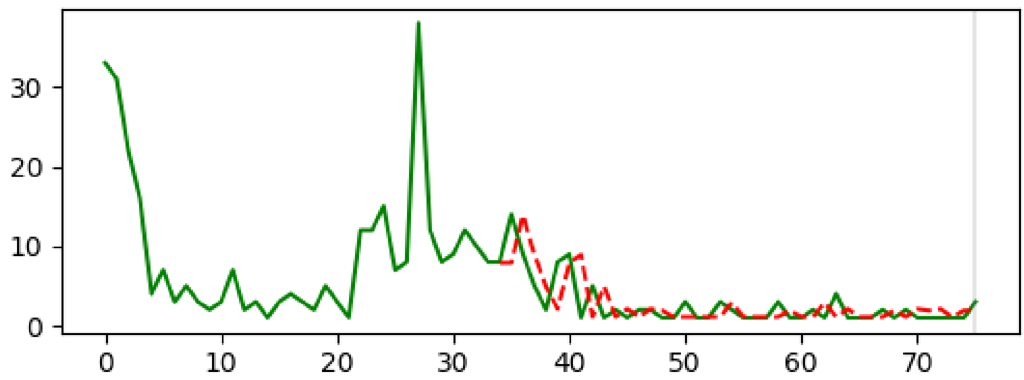
此部分代码如下：

```
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_parame
nter)
        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