数据驱动的异常检测与预警

摘要

。

**针对全图障碍信息已知的情况**，。

**针对全图障碍信息未知的情况**，。

。

**关键词： 算法 蚁群算法 动态局部最优 模拟人类视觉扫描法**

一、问题重述

**1.1 问题背景**

近年来，随着社会经济水平的提升，企业的生产也在迅速地发展。但随着生产企业数量的增多，其生产质量与安全状况呈现了下滑趋势，生产事故频频爆发。究其原因，这不仅受到了生产企业硬性的限制，如生产人员密集、生产规模及能力较弱、生产中的薄弱环节较多等，还受到了生产企业管理层和一线生产员工安全意识的影响。许多管理人员和一线职工防范意识较弱，导致原本安全问题就频发的生产环节发生安全事故的概率大大加强。因此，保证生产安全、防范系统性风险在高质量推动生产企业发展中就显得尤为重要。

目前，解决生产企业高质量发展问题仍然是中国制造业所需要改进的方向。安全的生产环境才能保证产品的质量和生产人员的人身安全。建立风险预测和评定机制也就成为了当务之急，只有制定标准化的管理、评定机制才可以使生产企业的安全得到保障。因此，我们考虑从生产中常见的非风险性异常和对生产安全产生威胁的风险性异常入手，建立预测和风险性异常程度评价模型，用适当的数学模型对生产系统的安全性进行分析和评价，并进行敏感性分析检验模型的合理性。

**1.2 问题提出**

**问题一**：根据问题一的要求，具有规律性、特征性、偶发性等特点的波动属于正常波动，不会对生产安全产生影响，属于非风险性异常；具有持续性、联动性特点的波动属于异常性波动，会对生产安全造成威胁，属于风险性异常。要求我们建立合适的数学模型，给出评判这风险性异常数据和非风险性异常数据的方案。

**问题二**：在问题一结果的基础上，需要建立风险性异常数据的量化评价模型，对每个时刻的数据进行量化评价，通过建立的数学评价模型和附件数据信息，寻找异常分值最高的5个时刻和这5个时刻所对应的异常传感器编号，并通过模型对最终的结果进行合理的分析和评价。

**问题三：问题三需要**建立风险性异常预警模型来针对在未来生产过程中可能存在的安全风险。首先需要对当日23:00:00-23:59:59时间段进行风险性异常预测，并结合问题二的数据量化评价模型，将预测所得的结果进行量化评价，给出每个时间段内的最高异常分和对应的传感器编号。

**问题四：**结合问题二和问题三的结果，建立生产系统安全行评估模型，以30分钟为时间间隔单位对当日00:00:00-23:59:59的时间段内的整个生产安全系统进行量化评价，量化结果的最终分值在0-100分的区间范围内，并对量化评价后的结果进行敏感性分析和评价。

二、问题分析

**2.1 问题一的分析**

对于问题一而言，我们根据题目的条件，主要考虑通过建立筛选的模型来区分风险性异常数据与非风险性异常数据。我们的模型先进行了数据预处理，利用离群点检测和等级性数据划分的方式，将所有的传感器的数据属性进行初步的划分和筛选；再利用白噪声检测的方法，找到只存在白噪声的时序数据，将其归纳为非风险性异常数据。我们采用夏皮洛-威尔逊检验和KS检验，通过白噪声时序数据满足正态分布或均匀分布的特点进一步筛选数据属性种类；最后我们利用丕尔逊相关系数法，对剩余未能被筛选出的传感器进行相关性矩阵计算，并以丕尔逊相关系数等于0.65为划分依据，当丕尔逊相关系数大于0.65，则认为该传感器间相关性显著，属于风险性异常数据，反之，则属于非风险性异常数据。最终可以将所有传感器数据分为风险性异常数据和非风险性异常数据。

**2.2 问题二的分析**

问题二要求我们对风险性异常数据建立一个衡量异常程度的量化指标模型。我们主要考虑采用密度聚类的方法来获取指标权重、利用自相关系数获取每一传感器任一时刻的异常得分，并以此为基础计算某一时刻所有传感器数据异常程度的量化值。以丕尔逊相关系数矩阵为基础，以矩阵中数据计算相对距离，调整Mabel number，使得其既满足于传感器数据相关性，又满足簇分类的基本要求。之后求解出各个簇的核心对象和最小半径，簇中的每个风险性异常数据的权重即为簇的最小半径。每个传感器任意时刻的数值通过计算其当前时刻的自相关系数得到。最终可以得到生产系统任意时刻的总异常程度数值以及该时刻下各传感器的异常程度数值。

**2.3 问题三的分析**

问题三题目要求我们建立风险性异常预警模型，以提前发现未来生产过程中可能发生的隐患。我们考虑的是通过ARIMA(差分自回归滑动平均模型)。先利用单次或多次的差分运算，使时间序列数据能够满足ARMA模型的平稳性要求；再以AIC准则函数的解最小化为目标函数，在有限区域内枚举ARMA模型中的两个参数值和，以此改变ARMA模型中的参数向量和残差，最终改变AIC准则函数的解。枚举有限区域中的所有参数值和，进行循环运算，求出所有情况的函数最终解。最终选取AIC准则函数最小值情况下的参数向量，对传感器的未来数据进行风险性预测，给出异常程度最高的5个传感器编号。

**2.4 问题四的分析**

问题四要求我们根据问题二和问题三的结果，建立一个对整个生产系统安全性能进行评估的量化模型。我们考虑通过分别计算非风险性和风险性两种属性的异常程度的得分，以一定的权重组合获得异常得分，反向求解安全性能评分。我们将非风险性中传感器分为存在白噪声和不可评价两类，对存在白噪声的时序数据进行权重分析。通过时间段的方差比确定传感器在时间段的权重比值；通过传感器间的归一化标准差比，确定传感器间的权重分配。通过非风险异常中的传感器和风险性异常的传感器数量之比，确定它们两者间的权重比值关系。风险性异常传感器除了最大权重转化为其具体数量外，其余参数信息与问题二模型一致。最终可以计算出统一标准下非风险性和风险性异常两种情况下的异常程度得分，反向求解，可以对任意时间段内的整个生产系统的安全行进行合理的评估，并对结果进行评价和灵敏度分析。

三、模型假设

1. 假设单值、双值及三值数据为属性值，即这些值不反映异常情况，无需对其异常程度进行评分。
2. 假设时间序列连续时，非单值、双值及三值数据连续变化。
3. 假设安全生产过程中不会产生离群数据。

四、符号说明

|  |  |
| --- | --- |
| 符号 | 符号意义 |
| *T* | 表示 |
|  | 表示坐标为的栅格的成本 |
|  | 表示栅格所接受到的所有成本的最小值 |
| *S* | 表示此栅格的总像素点个数 |
|  | 表示某个栅格内障碍物所包含的像素点个数 |
|  | 表示每个栅格的状态 |
|  | 表示从起点（S）到终点（D）的估计代价 |
|  | 表示从起点到状态n的成本 |
|  | 表示从状态n到终点（D）的估计代价 |
| *K* | 表示蚁群算法中决定信息素含量的因子 |
|  | 表示障碍物栅格的默认通过率 |
|  | 表示栅格的通过率 |
|  | 表示栅格中非障碍物所占面积 |
|  | 表示栅格的总面积 |
|  | 表示从移动机器人当前所处的栅格到终点的任意路径的成本 |
|  | 表示任意两条射线之间的夹角 |
|  | 表示整个地图的长与宽 |

注：未列出符号及重复的符号以出现处为准

五、模型的建立与求解

**5.1问题一的模型建立与求解**

**5.1.1 模型的准备**

根据题目的要求，我们建立的问题一模型将主要通过离群点检测与划分、夏皮洛-威尔逊检验法、KS检验法和丕尔逊相关系数方法对附件1给出的波动性数据进行分析，判定其为非风险性异常数据或为风险性异常数据。

1).夏皮洛-威尔逊检验（Shapiro–Wilk test）

夏皮洛-威尔逊检验是一种检验数据是否服从正态分布的方法，此方法的目的在于判断单一传感器的数据波动是否属于正态分布。

其检验的原假设如下

H0: 是来自一个正态分布总体的样本，其统计量是：

其中表示的来自母体的样本，表示样本个数，是个样本的平均值。

常量符合以下条件：

a).是的最佳线性无偏估计

b).是样本来源的正态分布的标准差

可以得到的准确值

其中矩阵是一个协方差矩阵，属于个标准正态分布的随机变量的顺序统计量，是这些变量的期望组成的一个向量。我们通过设定显著性水平=0.05，之后获得它的临界值，若求得，则拒绝原假设，认为其不服从正态分布，反之，则接受原假设，认同其服从正态分布。

利用Shapiro–Wilk test，我们可以对传感器数据或残差是否为白噪声进行系统性检验。

2).丕尔逊相关系数（Pearson correlation coefficient）

本文通过计算丕尔逊相关系数，确定两个传感器之间的线性相关性。其具体的计算公式如下：

其中表示传感器1的时序数据，表示传感器1时序数据的平均值；表示传感器2的时序数据，表示传感器2时序数据的平均值。

在丕尔逊相关系数的取值范围区间内。存在以下含义：

a).当Pearson correlation coefficient为0时，两个向量不相关

b).当Pearson correlation coefficient为负数时，两个向量呈负相关关系

c).当Pearson correlation coefficient为正数时，两个向量呈正相关关系

最终我们可以通过定义的相关系数的值，来筛选出时序数据不满足相关关系情况下的传感器编号。

3).克尔莫克洛夫-斯米洛夫检验（Kolmogorov–Smirnov test，K-S test）

我们利用克尔莫克洛夫-斯米洛夫检验方法是一种非参数统计检验方法，判断传感器时序数据是否满足均匀分布。

K-S test的原假设：总体的具有的分布，构造检验统计量：

其中累积分布函数有：

为指示函数，

若当原假设为真时，依分布收敛于假设的理论分布，即

因此，当时，趋向于0，原假设H0成立，即样本集的累积分布函数满足均匀分布；反之累积分布函数不满足均匀分布。

**5.1.2 模型工作流程**

1)目标的确定

根据题目要求，我们建立合理的数学模型对存在波动的数据进行分析处理，定量化地区分具有规律性、独立性、偶发性特点的非风险性异常数据波动和具有持续性、联动性特点的风险性异常数据波动。最终给出非风险性异常数据和风险性异常数据的判定方案。

2)数据预处理

a).离群点检测

通过查阅文献，我们发现，在外界电流和其他环境因素不稳定的情况下，传感器的值可能会有较大波动，可能使分析结果产生较大误差。因此我们首先要对数据中的传感器误报数据进行筛选和剔除。

我们采用分位数法剔除误报数据。我们以第一个四分位数、中位数、第三个四分位数为输入数据，计算最大和最小非离群点值，作为一维数据的离群区分点。

例如，我们将按下方箱图剔除错误数据：

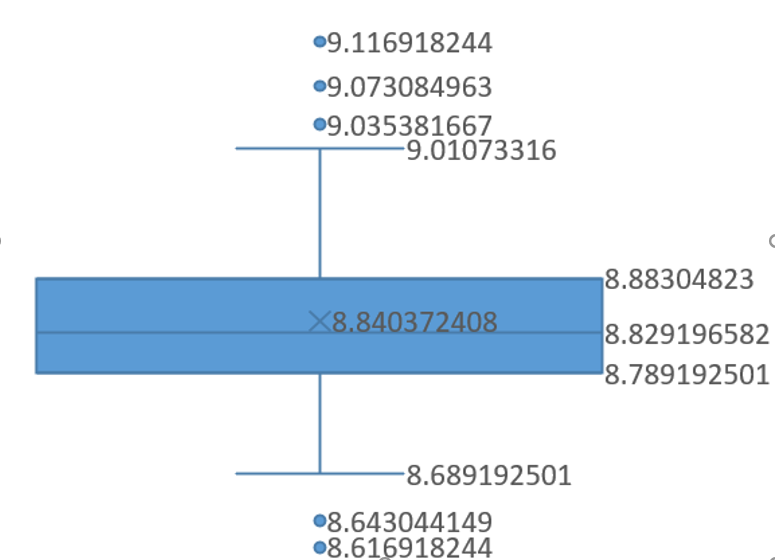


图1 box plot呈现

因为我们可以通过观察传感器时序数据的分布特征，从而了解时序数据的大致分布，并且该数据是针对单一属性的，且数据十分充分。故而选取这种建立在标准统计学原理基础上的方案能够使检验非常有效。最终通过离群点检测筛选出来的离群点都是作为风险性异常数据的传感器，对没被该方案筛选出来的传感器继续进行筛选处理。

b).等级性数据划分

经过离群点检测筛选方案后，我们再通过等级性数据划分进行第二步数据预处理。在所有的100个传感器中，我们观察可以知道，传感器提供的数据存在单值或多值，即某些传感器其时序数据只存在一个值或存在明显区别且个数有限的数个数据值。我们将传感器时序数据的值定义为该传感器的等级性，因此每个传感器都有与之对应的唯一等级性数据。我们认为，在实际生产中，该属性值可能反应了机器在不同档位运行时的工作特征。

通过统计，我们发现，传感器数据取值个数在5以下，且其时序数据信息具有明显差异的数据，可以认为是等级性数据，属于非风险性异常数据。

3)方案设定

*Step1*.对100个传感器的时序数据判断，寻找只存在白噪声的时间序列的传感器编号，并将其归纳到非风险性异常传感器中。因此我们对所有传感器的时序数据单独进行夏皮洛-威尔逊检验方法和检验方法。若传感器的时间序列数据存在白噪声，则其应该满足正态分布或均匀分布，接受原假设。

只存在白噪声的时间序列数据传感器属于正常波动，即其数据波动属于非风险性异常。

我们可以通过 夏皮洛-威尔逊检验方法 和 克尔莫克洛夫-斯米洛夫检验方法 ，从100个传感器中筛选出时序数据不满足正态分布或不满足均匀分布的传感器，对其进行下一步操作。

*Step2.*对通过 夏皮洛-威尔逊检验方法 筛选后的剩余数据进行丕尔逊相关系数的计算，利用这些传感器时序数据间的丕尔逊相关系数的求解值和我们设定的值进行比较

其中表示传感器的编号。

最终，我们能够通过丕尔逊相关系数的值来确定传感器是具有风险性异常数据还是非风险性异常数据。

若丕尔逊相关系数大于0.65，则这对应的两个传感器之间相关性显著，具有较强的联动性，都属于风险性异常数据的传感器；若丕尔逊相关系数小于0.65，则对应的两个传感器之间相关性较弱，关联性较差，属于非风险性异常数据的传感器。

**5.1.4 模型的结果**

通过计算机编程求解，我们得到的结果传感器编号和数据属性的分类情况如下所示。

表1 问题一结果呈现表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 非风险性  异常数据 | | | | 风险性  异常数据 |
| 1 | 2 | 3 | 5 | 4 |
| 6 | 8 | 9 | 10 | 7 |
| 12 | 13 | 14 | 15 | 11 |
| 16 | 18 | 19 | 22 | 17 |
| 24 | 25 | 26 | 27 | 20 |
| 28 | 29 | 30 | 31 | 21 |
| 32 | 33 | 34 | 35 | 23 |
| 36 | 37 | 38 | 39 | 41 |
| 40 | 42 | 43 | 44 | 47 |
| 45 | 46 | 48 | 49 | 57 |
| 50 | 51 | 52 | 53 | 65 |
| 54 | 55 | 56 | 58 | 71 |
| 59 | 60 | 61 | 62 | 72 |
| 63 | 64 | 66 | 67 | 73 |
| 68 | 69 | 70 | 74 | 76 |
| 75 | 77 | 78 | 79 | 94 |
| 80 | 81 | 82 | 83 | 96 |
| 84 | 85 | 86 | 87 | / |
| 88 | 89 | 90 | 91 | / |
| 92 | 93 | 95 | 97 | / |
| 98 | 99 | 100 | / | / |

**5.2问题二的模型建立与求解**

**5.2.1 模型的工作原理**

根据题目的要求，我们以密度聚类思想为核心建立风险性异常数据异常程度的量化评价模型，采用百分制方法对每个时刻数据异常程度打分，并通过模型对异常分值最高的5个时刻进行评价与分析。

1).密度聚类思想

密度聚类的核心思想是只要样本点的密度大于阙值，便可以将该样本点添加到最近的簇中。我们考虑通过密度聚类的思想来计算风险性异常数据异常程度量化模型中的权重指标。

在数据集，参数的密度聚类模型中存在着几个基本概念：

a).-邻域：数据集中除了外的其它样本与距离小于的样本集合；

b).对象：若的-邻域中样本数量大于，则样本被叫做核心对象；

c).密度直达：若样本是的-邻域中的样本，则由为密度直达；

d).密度可达：存在样本序列，其中的核心对象是，的核心对象是，且由密度直达，则由为密度可达；

e).密度相连：，使和均由密度可达，则和密度相连。

因此，我们可以从密度可达的关系中到处最大密度相连的样本集合。应该有给定参数和簇，应该满足两个条件：

连接性:和密度相连，用数学符号表示为：，

最大性：，由为密度可达，可以推出

通过上述方法，可以确定一个簇中的核心对象点，并且以此对象核心点出发，不断向密度可达的-邻域扩张，得到一个包含核心点和边界点的最大化区域，区域中的任意两点是密度相连的。

**5.2.2 模型**

1)目标函数的确定

依据题目要求，我们要对每个时刻的数据异常程度进行打分，建立量化评价模型，确立目标函数：

其中表示第个簇指标的量化权重，表示第个簇中传感器指标的数值,为计算出的簇的数量。

2)初始方案的策划

*Step1*.在问题一模型丕尔逊相关系数的基础上，确定所有传感器之间的相关系数，并建立相关特性矩阵，以此为基础计算出传感器点在空间之间的相对距离；

*Step2*.由于问题要求我们对风险性异常数据的异常程度作量化评价，因此在之后的模型设计考虑过程中，需要将与核心对象传感器之间存在较强相关性，但并非属于风险性异常数据的传感器指标进行及时更改，防止指标权重信息产生误差；

*Step3*.通过第一步中建立的相关特性矩阵，我们进一步寻找彼此间存在较强相关性的传感器。利用密度聚类的思想，把所有在相关特性矩阵中反应的相对距离呈现在二维平面中，寻找出一个既能满足于传感器数据之间相关性筛选关系，又能满足簇的个数等基本要求的簇的分类方法和方式。以该种簇的分类方式将二维图像中所包含的传感器聚成一个簇。这样就可以形成几个互相之间并无显著关联特性的簇，并进一步对每个簇分析。

*Step4*.求解每个簇的最小半径。运用密度聚类思想中的密度直达、密度可达、密度相连、邻域和核心对象的概念，以满足连接性和最大性为目标条件，寻找每个簇中以任意一个传感器点为核心对象所求出的对应最小半径。

最终求出的每个簇的最小半径是所有最小半径中的最小值，核心对象是求出的最小半径所对应的核心对象。

*Step5*.以每个簇中的最小半径作为每个簇的权重，若簇中含有非风险性异常数据的传感器，则将该传感器的权重定义为0，不对其所在簇中的其它传感器产生任何影响。在最后计算任意时刻数据异常程度评价时，以每个簇的权重作为该簇中的所有含有风险性异常数据的传感器的权重。这里的每个簇的权重也就是模型的目标函数中的量化权重。

*Step6*.计算目标函数中每个传感器指标的数值，我们采取自相关系数表示每个传感器指标在任意时刻的数值，得到公式

其中常数表示滞后数，即两个同一时序数据中的时序差值。由于滞后数h在一个范围内都是成立的，此处我们取平均值代表自相关系数，也即：

*Step6*.最终通过密度聚类方法计算出的簇权重和每个传感器指标中的任意时刻自相关系数求解出最终量化后的风险性异常数据异常程度的量化指标，并选出异常分值最高的5个时刻及这5个时刻对应的5个异常程度最高的传感器编号。

**5.2.3 模型的求解**

根据问题二数学模型的描述，我们通过算法DBSCAN来解决密度聚类问题，通过矩阵和迭代运算来进行传感器的任意时刻自相关系数计算和风险性异常数据异常程度量化数值。

以每个传感器任意时刻的数据为目标点位，利用迭代方法求解出传感器在某一时刻的自相关系数。

传感器自相关系数作为传感器任意时刻的数值。利用目标函数的求解公式，带入传感器任意时刻的数值和各个簇的异常程度数值权重。量化求解出各时刻风险性异常数据异常程度的具体数值和5个异常程度最高时刻中5个传感器异常程度最高的编号。

**5.2.4 模型的结果**

通过计算机编程求解(详见代码文件 第一题.cpp)，得到的结果如下所示，可以看到最高异常分值的5个时刻以及其所对应的5个异常程度最高的传感器编号。

表2 问题二结果呈现表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 第一高分 | 第二高分 | 第三高分 | 第四高分 | 第五高分 |
| 异常程度得分 | 72.84799914 | 72.81424496 | 72.76768961 | 72.76160024 | 72.75144977 |
| 异常时刻编号 | 1425 | 1424 | 1423 | 1426 | 1422 |
| 异常传感器编号 | 4 | 4 | 4 | 4 | 4 |
| 异常传感器编号 | 7 | 7 | 7 | 7 | 7 |
| 异常传感器编号 | 20 | 20 | 20 | 20 | 20 |
| 异常传感器编号 | 41 | 41 | 41 | 41 | 41 |
| 异常传感器编号 | 94 | 94 | 94 | 94 | 94 |

**5.2.5 模型的结果分析**

先分析结果的特征，可得到以下结论：

1). 异常程度得分最高的时刻为相邻的时刻；

2). 异常程度得分最高的传感器包含了三个簇的传感器；

3). 五个时刻异常程度得分最高的传感器相同。

由此可得，在这个时间段内，三个簇所包含的传感器均产生了较大波动，使得此时间段的异常程度得分较高。

**5.3问题三的模型建立与求解**

**5.3.1模型的工作原理**

根据题目要求，建立风险性异常预警模型，提前发现当日23时至24时可能产生的风险性异常，并结合问题二的量化评价模型，给出四个时段的异常程度数值。我们运用ARIMA模型(自回归综合滑动平均模型)来预测和分析未来的1h内异常程度数值的变化情况。

1).ARIMA模型

ARIMA模型是差分运算和ARMA模型的结合，而ARMA模型又可以细分成AR模型和MA模型。

a).差分运算

因为ARMA模型必须满足平稳性要求，我们需要通过差分运算将不平稳的数据转化为能够被ARMA模型所用的平稳性数据。当序列蕴含多项式曲线的趋势时，我们运用多阶差分实现平稳的特性；若序列含有固定的周期，我们能用步长为周期的长度差分，从原序列中过滤出周期的影响，实现趋势的平稳。

b).AR模型

AR模型是自回归模型，其变量当前的取值与过去的取值相关。模型蕴含延迟系数数，阶自回归模型可以表示为：

对于中心化的AR()模型而言，当时成立，即有

c).MA模型

MA模型为滑动平均模型，其当前序列的数据变量的记忆是关于过去外部干扰的记忆，即当前时间序列可以表示成现在干扰值和过去干扰之的线性组合关系。其也存在参数阶数，阶移动平均模型可以表示为：

对于中心化的MA()模型而言，当时成立，即有

同时对AR()模型和MA()模型引入延迟算子，可以得到中心化AR()和中心化的MA()模型

**5.3.2模型**

1)模型参数的确定

我们通过信息准则函数法确定ARIMA模型的参数p和q。准则函数既要考虑模型对原始数据的预测准确性，也考虑了模型中所含变量的个数，即既要防止欠拟合，又要防止过拟合。通过数次尝试，我们发现以下准则函数（AIC）能够很好地确定参数：

其中是残差的方差，表示白噪声，为样本容量，和为模型的阶数，为常数，当其值在[2,3]的区间内时效果良好。

2)初始方案的策划

*Step1*.观察题目附件1中所给出的函数时序数据序列值，对100个传感器的时序数据有一个大致的认识，以便于接下来能够寻找更好的方法对这一组的数据进行处理和分析；

*Step2*.差分和平稳性检验。我们采用菲利普斯-佩伦测试（Phillips-Perron test ,PP-test）来检测其平稳性。如果测试结果显示序列不具备平稳性的特性，则进行差分，直到得到都序列具备平稳性为止。

*Step3*.白噪声检验。我们通过对残差进行检验来确定传感器的时序数据是不是存在白噪声，具体检验的方法是问题一模型中的夏皮洛-威尔逊检验和克尔莫克洛夫-斯米洛夫检验来进行，通过检测时序数据是否满足正态分布或均匀分布来判断。

*Step4*.若传感器时序数据参数通过白噪声检验，则可以直接根据白噪声的特征得到传感器的在未来1h内的预测数据值；反之，则需要重新改变参数数据，重新进行白噪声检验，寻找最优的满足*AIC*准则函数的最小值，从而求解传感器的预测数据量。

*Step5*.当时序数据未通过检验后，需要重新确定参数和的数据，因为参数和均为整数，因此我们选择采用有限区域枚举的方案，对参数和在的区间中任意取整数对，寻找*AIC*准则函数的最小值；

*Step6*.通过上一步确定的参数和，进一步确定ARMA模型中的系数向量，从而进一步计算残差值。我们将假设枚举的参数和的值以及以此计算出的残差的方差代入*AIC*准则函数中，计算出该参数和取值情况下的目标函数值；

*Step7*.进一步对参数和进行枚举，得到很多种情况下的系数向量和残差值，利用这些数据计算出参数和在各个取值情况下的最终*AIC*准则函数解。对所有取值情况下的最终函数值进行筛选，选取最小情况时的参数和及系统向量。并以此为依据，求解各传感器的预测数据数值。

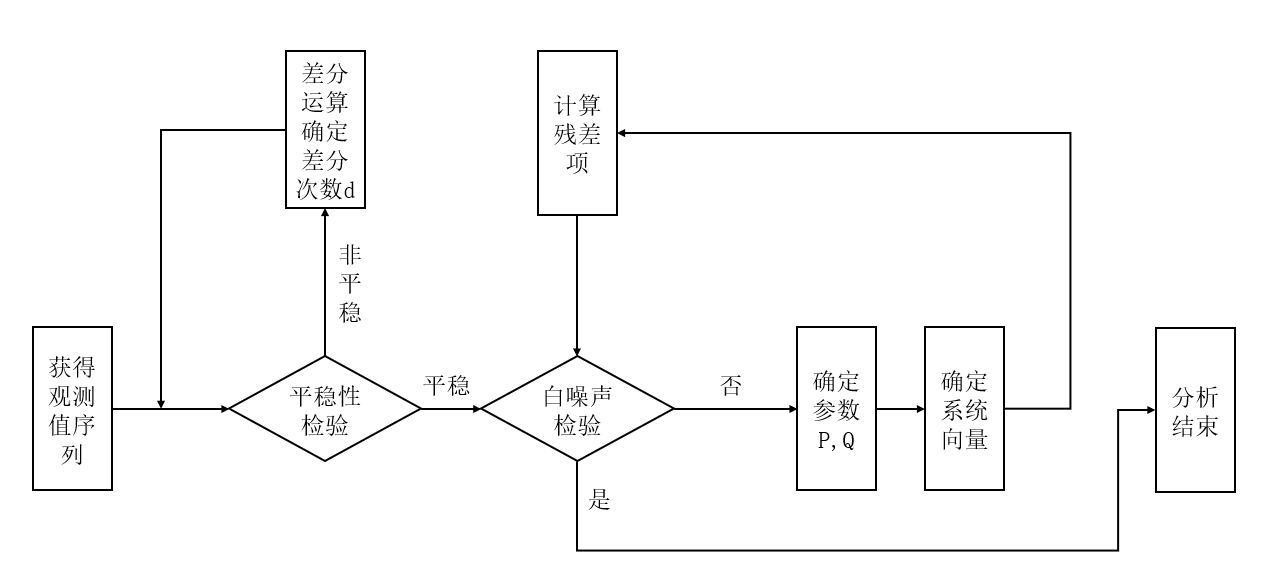


图 求解步骤流程图呈现

**5.3.3模型的求解**

我们在运用数学模型求解传感器预测结果的过程中，所面临的两个问题分别是*AIC*准则函数和自回归移动平均模型中的系数求解。

1).利用*AIC*准则函数确定参数p与q

我们选择采用有限区域枚举的方案，对参数和在的区间中任意取整数对，寻找*AIC*准则函数的最小值。也即：

2).自回归移动平均模型中的系数求解

由ARMA(p,q)模型中的特征，我们可以知道中心化ARMA(p,q)模型是当时成立，与AR模型的判定条件相一致，这时，满足序列为：

可以从上述表达式看出，当前时间序列值，与过去的时间序列值乘线性关系。当假设ARMA(p,q)模型中的(p,q)解值后，可以通过附件1中的数据和ARMA模型中的(p,q)数值，进一步求解出自回归移动平均模型中的系数以及残差。并以此作为白噪声检验的依据，若满足白噪声检验，则通过该系数向量求解序列数据的预测数值；若不满足白噪声检验，则在有效区域内枚举可能的(p,q)解值对，直到最终的准则函数满足白噪声检验为止。

**5.3.4模型的结果**

求解模型(ARIMA计算.m)。我们通过预测数学模型，求解得到的结果表如下所示

表3 问题三结果呈现表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 时间 | 23:00:00-23:14:59 | 23:15:00-23:29:59 | 23:30:00-23:44:59 | 23:45:00-23:59:59 |
| 异常程度得分 | 54.04454892 | 73.26893413 | 73.93539558 | 73.93733735 |
| 异常时刻编号 | 5564 | 5639 | 5672 | 5727 |
| 异常传感器编号 | 11 | 7 | 4 | 4 |
| 异常传感器编号 | 71 | 11 | 17 | 17 |
| 异常传感器编号 | 72 | 20 | 96 | 96 |
| 异常传感器编号 | 76 | 76 | 76 | 76 |
| 异常传感器编号 | 94 | 94 | 94 | 94 |

**5.4问题四的模型建立与求解**

**5.4.1 模型的工作原理**

根据题目的要求，我们需要建立数学模型对该生产系统的安全性进行0-100分量化的评估。因此我们在问题二中的风险性异常数据异常程度量化评价模型的基础上，引入非风险性异常数据异常程度的评价指标，计算风险性和非风险性的异常程度，最终转化为安全性评分。在考虑本问时，主要需要关注风险性数据异常程度模型的权重变化和非风险性数据异常程度模型的权重分配。

1).风险性异常数据异常程度量化模型

在问题二的异常程度量化模型中，我们没有考虑非风险数据的影响，但在本问中，我们需要考虑其所可能带来的对安全性能的影响。因此我们需要将第二问风险性异常程度模型的权重进行调整。利用风险性和可定量化处理的非风险性传感器数据的个数作为问题四模型中这两种属性数据的最终权重比。因此可以得到风险性异常数据异常程度模型公式为：

其中表示风险性传感器个数。其余指标计算方式和意义与问题二模型中一致。

我们可以通过这个方法使问题二模型能够满足整个生产系统的大背景的条件

2).非风险性异常数据异常程度模型

对于非风险性异常数据异常程度模型而言，我们首先需要考虑其是否具有异常。将非风险的传感器分为两类。其一是阶梯型数据，其特征为数据只有单值、双值，这类数据我们无法给出量化的评判，因为其状态稳定，无风险因素波动影响；其二是剩余的存在白噪声的时序数据，我们需要对这类满足均匀或正态分布的数据进行评分。

其次需要考虑关于异常程度数值的计算，我们采用权重分配相乘的方式对非风险异常程度进行量化。总的异常得分计算形式如下

其中表示只存在白噪声的非风险性异常数据传感器个数，表示针对于某个传感器而言，其某个时间段的异常程度分数值与总的时间段分数值的权重比；表示针对于非风险性传感器间而言，它们权重比例的分配数值。

对于某个传感器而言，我们通过某个时间段的方差和总时间的方差的关系，建立权重比，关系如下：

其中表示某个时间段内的传感器数据方差对全天方差的贡献，表示总的一天时间内，传感器数据的方差。易知：

对于不同传感器间而言，将所有平均值作归一化处理，目的是将不同传感器数据间的标准差放在同一标准下进行对比，以它们传感器间的标准差数值大小作为权重进行计算，

最终可以通过计算得到非风险性异常数据异常程度的分数值。

**5.4.3 模型**

1)目标函数的确定

根据题目需求，我们最终需要对整个生产系统的不同时段安全性进行量化评估，因此我们通过非风险性异常程度量化模型和风险性异常程度量化模型的求解函数，建立最终的目标函数：

用总分减去在权重合理分配下的两种情况下异常程度得分，可以得到最终的安全行评分。

2)初始方案的策划

*Step1*.首先将非风险性异常程度评价模型中无法评价的阶梯形状时序数据的量化分数值定义为0，即其是安全的，不会对总的生产系统安全性能评估造成任何负面的影响；其次对剩余的满足均匀或正态分布的，即含有白噪声的时序数据传感器进行权重分配的计算；

*Step2*.对于权重分配，我们先考虑针对单一传感器权重值计算的情况。我们通过时间段分布的方差的比值，来计算某个传感器其所在时间段在一天总的时间段中的权重比例分配；

*Step3*.其次考虑不同传感器间权重分配的情况。为了保证权重分配的合理性，我们将所有非风险性异常程度的传感器进行平均归一化处理，目的是将各组存在白噪声的数据传感器的标准差放在统一的标准下进行对比，防止因未标准化而导致权重不正确的情况发生；

*Step4*.计算非风险性异常数据异常程度的数值，考虑到阶梯型时序数据的异常得分为0，因此我们仅对存在白噪声的时序数据作权重乘积运算即可，将满足均与分布或正态分布的传感器数据个数乘以上述步骤中求解的两个权重比值和，就可以求解出非风险性异常程度的数值。

*Step5*.对于风险性异常数据异常程度量化模型而言，可以在问题二模型的基础上进行改进。因为问题二仅针对风险性异常程度评价指标，因此风险性异常程度权重指标在问题二模型中之和为100。而问题四是针对整个生产系统，因此需要考虑非风险性异常程度权重的影响，我们将这两个属性的权重分配与它们异常传感器个数挂钩。非风险性异常程度的权重就是其属性所对应的传感器个数；风险性异常程度的权重也是其属性所对应的传感器个数。

*Step6*.通过求出某个时间段内非风险异常数据异常程度量化模型的量化数值和风险性异常数据异常程度量化模型的数值。可以将这两个数值带入生产系统安全性的目标函数求解出，整个系统在某个时间段的安全性能评分。依照此方法可以求出一天时间内任意时间段的安全性能评分。

**5.4.4 模型的结果**

利用算法求解模型(详见代码文件 第二题K.cpp)。我们通过数学模型，计算出整个生产系统的安全性能评分，题目中所给的各个具体时间段的安全性能评分结果呈现如下所示。

表4 问题四结果呈现表

|  |  |  |  |
| --- | --- | --- | --- |
| 时间 | 安全性评分 | 时间 | 安全性评分 |
| 00:30:00 | 90.8264785035472 | 12:30:00 | 62.0605178821818 |
| 01:00:00 | 61.5855640045683 | 13:00:00 | 64.3362560107371 |
| 01:30:00 | 61.7845923596261 | 13:30:00 | 69.9782379057836 |
| 02:00:00 | 68.8814190989694 | 14:00:00 | 72.4286711498034 |
| 02:30:00 | 66.5935616324952 | 14:30:00 | 60.5129892002528 |
| 03:00:00 | 64.5358008884087 | 15:00:00 | 59.2248424621715 |
| 03:30:00 | 67.6109144177722 | 15:30:00 | 64.7324658691658 |
| 04:00:00 | 61.0251837793729 | 16:00:00 | 57.8982774466103 |
| 04:30:00 | 56.7541607223285 | 16:30:00 | 62.4732259625621 |
| 05:00:00 | 57.9644123631374 | 17:00:00 | 62.3104319508221 |
| 05:30:00 | 61.2314886141373 | 17:30:00 | 69.2307298935948 |
| 06:00:00 | 71.149373439921 | 18:00:00 | 62.680811338123 |
| 06:30:00 | 67.5509971990349 | 18:30:00 | 68.5293585514074 |
| 07:00:00 | 64.0987223421491 | 19:00:00 | 67.5948967534762 |
| 07:30:00 | 64.2209509586209 | 19:30:00 | 68.0457844837606 |
| 08:00:00 | 57.8063432196527 | 20:00:00 | 58.3564173036089 |
| 08:30:00 | 55.5631724394828 | 20:30:00 | 66.0652266087341 |
| 09:00:00 | 62.4129556971205 | 21:00:00 | 57.4012657671471 |
| 09:30:00 | 59.3092812102277 | 21:30:00 | 64.1221033071493 |
| 10:00:00 | 61.7124089541373 | 22:00:00 | 60.9585067978608 |
| 10:30:00 | 60.8814669049633 | 22:30:00 | 61.3323178296972 |
| 11:00:00 | 57.2684252411184 | 23:00:00 | 61.1096472612839 |
| 11:30:00 | 59.8536720297076 | 23:30:00 | 67.0231558474388 |
| 12:00:00 | 66.4687411800495 | 23:59:59 | 68.4049189419218 |

**5.4.5 模型的灵敏度分析**

我们对问题四安全性能评估的结果作灵敏度分析，(详见代码文件 Ktest.cpp)

如图所示，对于比例系数k而言的结果呈现图。在考虑实际应用时，需要谨慎考虑比例系数k值，尽可能得取到平均开销最低值所对应的k值，以保证模型的效率。

六、模型的综合评价和推广

**6.1 模型的综合评价**

**6.1.1 模型的优点**

本文在划分模型方面，利用了离群点检验、分布检验以及相关系数方法，对数据的属性进行了有效的划分；

本文在量化评价打分模型方面，利用了密度聚类的思想、各参数间的关联性和概率论权重分配模型，准确、定量化地反映了具体数值。这些数值有明显的区分度，可以较好地反应该数值对应的参数性能是否稳定；

本文在预测模型方面，既考虑了单个参数受时间的影响，也考虑了受其它参数趋势变化的影响，能够有效地对未来数据做出准确的数值预测；

**6.1.2 模型的缺点分析**

本文在构造区分数据属性模型时，不能通过模型判断单值或双值时序数据，并将这些数据信息归类；

本文在假设时的情况相对较理想，没有完整考虑实际工业上可能出现的问题，模型和现实条件可能具有一定的差距。

**6.2模型的推广**

模型在实际的生产工业中实用性较强，我们考虑了指标间的相关性影响，并用定量的方法评价其相关性，能够较为准确应对多元复杂的环境；

在工业中面对风险性异常数据和非风险性异常数据的区分状况下，我们考虑从持续性和联动性出发，利用白噪声来判断数据是否会对整个生产系统产生危害，考虑了数学模型在实际中的具体应用。

在面对工业生产中的未来数据预测情况中，我们考虑利用时间序列的数学模型进行计算，建立严谨的数学模型，对未来时序数据的预测具有良好的现实参考依据和价值。

整个数学模型和代码计算过程体系完整、严密，逻辑性强，对于工业生产具有一定的参考作用和价值，并满足当今社会的发展趋势，能够为工业生产的高效发展和进步贡献一定的力量。

七、参考文献

[1]刘璋温.赤池信息量准则 AIC 及其意义[J].数学的实践与认识,1980(03):64-72.

[2]方涛,姚应生.夏皮罗—威尔克方法对不等精度观测列正态性的检验[J].矿山测量,1990(02):17-19.

[3]刘振宇. 基于密度峰值的聚类算法优化研究[D].哈尔滨理工大学,2021.

[4]陈寿宏,易木兰,张雨璇,尚玉玲,杨平.基于优化DBSCAN聚类算法的晶圆图预处理研究[J/OL].控制与决策:1-9[2021-08-14].https://doi.org/10.13195/j.kzyjc.2020.0738.

[5]王晓辉,宋学坤,王晓川.基于邻域密度的异构数据局部离群点挖掘算法[J].计算机仿真,2021,38(07):281-285.

[6]许德合,丁严,张棋,黄会平.EEMD-ARIMA在干旱预测中的应用——以新疆维吾尔自治区为例[J].中国农村水利水电,2021(07):1-11.

[7]丁文绢.基于股票预测的ARIMA模型、LSTM模型比较[J].工业控制计算机,2021,34(07):109-112+116.

[8]王晨博,吴晓龙,谢智成,张艺,时统宇.基于ARMA的航班延误时间序列预测[J].科学技术创新,2021(19):76-77.

[9]张景肖,刘史诗,王伟华,李浩成,胡镜清.基于AIC准则的函数型数据主成分联合选择研究[J/OL].数理统计与管理:1-13[2021-08-14].https://doi.org/10.13860/j.cnki.sltj.20210722-010.

[10]刘银萍. 均匀分布的相关性质以及检验[D].扬州大学,2016.

[11]张慧娟,宋向东,王永茂.均匀分布下的异常数据检验[J].辽宁工程技术大学学报(自然科学版),2013,32(02):277-280.

[12]周纲,黄瑞,刘度度,张芝敏,胡军华,高云鹏.基于改进K-means聚类和皮尔逊相关系数户变关系异常诊断[J/OL].电测与仪表:1-13[2021-08-14].http://kns.cnki.net/kcms/detail/23.1202.th.20210531.1707.013.html.

[13]张纪泉.总体分布的正态性检验——介绍夏皮罗-威尔克的W检验法[J].中国纤检,1982(05):34-40.

附录

附录1：Autocorrelation\_coefficient-1.cpp

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

double a[5520][105]; //预处理后的数据

double ans[5520][105]; //answer

int main(){

memset(ans,0,sizeof(ans));

freopen("Autocorrelation\_coefficient\_in.txt","r",stdin);

freopen("Autocorrelation\_coefficient\_out.txt","w",stdout);

int tar[16]={4,7,11,17,20,21,23,41,47,57,65,71,72,73,76,96};

//编号从1开始

for(int i=1;i<=5519;i++){

for(int j=1;j<=100;j++)scanf("%lf",&a[i][j]);

}

//读入数据

for(int J=1;J<=100;J++){

//处理第J列数据

int c=1;

double sum=0,y=0;

//sum 用于计算前c个和，以便求平均数

for(;c<=15;c++){

sum+=a[c][J];

}

//从十六开始

for(;c<=5519;c++){

sum+=a[c][J];

y=sum/c;

//求出分母上的var\*c

double deno=0;

for(int i=1;i<=c;i++){

deno+=( a[i][J] - y )\*( a[i][J] - y );

}

for(int k=1;k<=15;k++){

double tmp=0;

//tmp保留当前k延迟条件下的自相关系数

for(int i=1;i<=c-k;i++){

tmp+=(a[i][J]-y)\*(a[i+k][J]-y)/deno;

}

ans[c][J]+=tmp/15;

}

}

}

for(int i=1;i<=5519;i++){

for(int j=1;j<=100;j++){

printf("%.10lf\t",ans[i][j]);

}

printf("\n");

}

return 0;

}

附录2：Autocorrelation\_coefficient-2.cpp

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

double a[5760][105]; //预处理后的数据

double ans[5760][105]; //answer

int main(){

memset(ans,0,sizeof(ans));

freopen("Autocorrelation\_coefficient\_3\_in.txt","r",stdin);

freopen("Autocorrelation\_coefficient\_3\_out\_100.txt","w",stdout);

int tar[16]={4,7,11,17,20,21,23,41,47,57,65,71,72,73,76,96};

//编号从1开始

for(int i=1;i<=5759;i++){

for(int j=1;j<=100;j++)scanf("%lf",&a[i][j]);

}

//读入数据

for(int J=1;J<=100;J++){

//处理第J列数据

int c=1;

double sum=0,y=0;

for(;c<=100;c++){

sum+=a[c][J];

}

//sum 用于计算前c个和，以便求平均数

//从十六开始

for(c=101;c<=5759;c++){

sum=sum+a[c][J]-a[c-100][J];

y=sum/100;

//求出分母上的var\*c

double deno=0;

for(int i=c-99;i<=c;i++){

deno+=( a[i][J] - y )\*( a[i][J] - y );

}

for(int k=1;k<=15;k++){

double tmp=0;

//tmp保留当前k延迟条件下的自相关系数

for(int i=c-99;i<=c-k;i++){

tmp+=(a[i][J]-y)\*(a[i+k][J]-y)/deno;

}

ans[c][J]+=tmp/15;

}

}

}

for(int i=1;i<=5759;i++){

for(int j=1;j<=100;j++){

printf("%.10lf\t",abs( ans[i][j] ));

}

printf("\n");

}

return 0;

}

附录3：SW\_test数据类型转换.cpp

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

//int find(const char \*s, int pos = 0) const;//从pos开始查找字符串s在当前串中的位置

//string substr(int pos = 0,int n = npos) const;//返回pos开始的n个字符组成的字符串

int main(){

string s;

freopen("SW\_test\_in.txt","r",stdin);

freopen("SW\_test\_out.txt","w",stdout);

for(int i=0;i<100;i++){

getline(cin,s);

string s1,s2,s3;

int a1=s.find('=');

int a2=s.find(',');

s1=s.substr(a1,a2-a1);

cout<<s1;

printf("\t");

s=s.substr(a2);

a1=s.find('=');

a2=s.find(')');

s1=s.substr(a1,a2-a1);

cout<<s1<<endl;

}

return 0;

}

附录4：test1.cpp

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int INF=0x3f3f3f3f;

const long long mod=100000007;

const double e=2.718281828459045;

const double pi=3.1415926535;

#define CK cout<<"OK\n";

int main(){

int a95=00,a90=0,a85=0,a80=0;

freopen("1.txt","r",stdin);

freopen("output1.txt","w",stdout);

for(int i=0;i<100;i++){

double a;

cin>>a;

if(a>0.95)a95++;

if(a>0.90)a90++;

if(a>0.85)a85++;

if(a>0.80)a80++,cout<<i+1<<endl;

}

cout<<a95<<endl<<a90<<endl<<a85<<endl<<a80<<endl;

return 0;

}

附录5：第2题.cpp

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

double ans[5520][105]; //answer

struct A{

double s1,s2,s3;

double score;

int rank;

friend bool operator<(A x,A y){

return x.score>y.score;

}

A(double s1=0,double s2=0,double s3=0,double score=0,int r=0):s1(s1),s2(s2),s3(s3),score(score),rank(r){}

}a[5520];

int main(){

freopen("score\_2\_in.txt","r",stdin);

freopen("score\_2\_out.txt","w",stdout);

for(int i=1;i<=5519;i++){

for(int j=1;j<=100;j++)scanf("%lf",&ans[i][j]);

}

double p1=1.151329,p2=3.0625,p3=1.331716;

for(int i=1;i<=5520;i++){

a[i].s1=a[i].s2=a[i].s3=0;

//分别计算s1 s2 s3 的平均值

a[i].s1=( ans[i][4]+ans[i][17] )/2;

a[i].s2=(

+ans[i][7]

+ans[i][20]

+ans[i][21]

+ans[i][41]

+ans[i][65]

+ans[i][72]

+ans[i][73]

+ans[i][76]

+ans[i][71]

+ans[i][96]

+ans[i][23]

+ans[i][47]

+ans[i][57]

)/13;

a[i].s3=( ans[i][11]+ans[i][94] )/2;

a[i].score=p1\*a[i].s1 + p2\*a[i].s2 + p3\*a[i].s3;

a[i].rank=i;

}

sort(a+1,a+5520);

for(int i=1;i<=5;i++){

printf("%d\t%.8lf\n",a[i].rank,a[i].score);

}

return 0;

}

附录6：第3题.cpp

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

double ans[5760][105]; //answer

struct A{

double s1,s2,s3;

double score;

int rank;

friend bool operator<(A x,A y){

return x.score>y.score;

}

A(double s1=0,double s2=0,double s3=0,double score=0,int r=0):s1(s1),s2(s2),s3(s3),score(score),rank(r){}

}a[5760];

int main(){

freopen("Autocorrelation\_coefficient\_3\_out\_100.txt","r",stdin);

freopen("score\_3\_out.txt","w",stdout);

for(int i=1;i<=5759;i++){

for(int j=1;j<=100;j++)scanf("%lf",&ans[i][j]);

}

double p1=1.151329,p2=3.0625,p3=1.331716;

for(int i=1;i<=5759;i++){

a[i].s1=a[i].s2=a[i].s3=0;

//分别计算s1 s2 s3 的平均值

a[i].s1=( ans[i][4]+ans[i][17] )/2;

a[i].s2=(

+ans[i][7]

+ans[i][20]

+ans[i][21]

+ans[i][41]

+ans[i][65]

+ans[i][72]

+ans[i][73]

+ans[i][76]

+ans[i][71]

+ans[i][96]

+ans[i][23]

+ans[i][47]

+ans[i][57]

)/13;

a[i].s3=( ans[i][11]+ans[i][94] )/2;

a[i].score=p1\*a[i].s1 + p2\*a[i].s2 + p3\*a[i].s3;

a[i].rank=i;

}

sort(a+5520,a+5580);

printf("%d\t%.8lf\n",a[5520].rank,a[5520].score);

sort(a+5580,a+5640);

printf("%d\t%.8lf\n",a[5580].rank,a[5580].score);

sort(a+5640,a+5700);

printf("%d\t%.8lf\n",a[5640].rank,a[5640].score);

sort(a+5700,a+5760);

printf("%d\t%.8lf\n",a[5700].rank,a[5700].score);

return 0;

}

附录7：ARIMA计算.m

clear

%% 数据读取

DATA = xlsread('matlab1.xlsx')

fin = zeros(300,100)

pq=zeros(2,100)

load H\_safe.mat H

%%

for i = 99:100

y=DATA(:,i);

T = length(y);

%H里保存了差分的次数（10的差0次）

if (H(i)==1)

d=1

else

d=0

end

tar=10000

tar\_p=1

tar\_q=1

for p=1:5

for q=1:5

mdl = arima(p, q, d);

EstMdl = estimate(mdl, y);

res = infer(EstMdl,y);

tmp=var(res)

aic=log(tmp)+170\*(p+q+1)/5519

if (tar>aic)

tar=aic

tar\_p=p

tar\_q=q

end

end

end

mdl = arima(tar\_p, tar\_q, d);

EstMdl = estimate(mdl, y);

res = infer(EstMdl,y);

[yF,yMSE] = forecast(EstMdl,300,'Y0',y);

UB = yF + 1.96\*sqrt(yMSE); % 95%置信区间上限

LB = yF - 1.96\*sqrt(yMSE); % 95%置信区间下限

fin(:,i) = yF

pq(1,i)=tar\_p

pq(2,i)=tar\_q

end

附录8：ARIMA枚举求参.m

clear

%% ARIMA(p,d,q)

DATA = xlsread('matlab1.xlsx')

fin = zeros(300,100)

%%

y = DATA(:,1);

T = length(y);

tar=10000

tar\_p=0

tar\_q=0

for p=1:5

for q=1:5

mdl = arima(p, q, 0);

EstMdl = estimate(mdl, y);

res = infer(EstMdl,y);

tmp=var(res)

aic=log(tmp)+170\*(p+q+1)/5519

if (tar>aic)

tar=aic

tar\_p=p

tar\_q=q

end

end

end

mdl = arima(tar\_p, tar\_q, 0);

EstMdl = estimate(mdl, y);

res = infer(EstMdl,y);

[yF,yMSE] = forecast(EstMdl,300,'Y0',y);

UB = yF + 1.96\*sqrt(yMSE); % 95%置信区间上限

LB = yF - 1.96\*sqrt(yMSE); % 95%置信区间下限

figure(6)

h4 = plot(y,'Color',[.75,.75,.75],'LineWidth',2);

hold on

h5 = plot(T+1:T+300,yF,'r','LineWidth',2);

h6 = plot(T+1:T+300,UB,'k--','LineWidth',1.5);

plot(T+1:T+300,LB,'k--','LineWidth',1.5);

h7 = gca;

plot((T+0.5)\*[1;1], h7.YLim, 'k--','LineWidth',1.5), % 样本点与预测点的时间分割线

h7.XLim = [0,T+302];

legend([h4,h5,h6],'CPI','预测值',...

'95置信区间','Location','Northwest')

title('CPI预测值')

hold off

fin(:,1) = yF

%%

y = DATA(:,2);

T = length(y);

tar=10000

tar\_p=0

tar\_q=0

for p=1:5

for q=1:5

mdl = arima(p, q, 0);

EstMdl = estimate(mdl, y);

res = infer(EstMdl,y);

tmp=var(res)

aic=log(tmp)+170\*(p+q+1)/5519

if (tar>aic)

tar=aic

tar\_p=p

tar\_q=q

end

end

end

mdl = arima(tar\_p, tar\_q, 0);

EstMdl = estimate(mdl, y);

res = infer(EstMdl,y);

[yF,yMSE] = forecast(EstMdl,300,'Y0',y);

UB = yF + 1.96\*sqrt(yMSE); % 95%置信区间上限

LB = yF - 1.96\*sqrt(yMSE); % 95%置信区间下限

figure(6)

h4 = plot(y,'Color',[.75,.75,.75],'LineWidth',2);

hold on

h5 = plot(T+1:T+300,yF,'r','LineWidth',2);

h6 = plot(T+1:T+300,UB,'k--','LineWidth',1.5);

plot(T+1:T+300,LB,'k--','LineWidth',1.5);

h7 = gca;

plot((T+0.5)\*[1;1], h7.YLim, 'k--','LineWidth',1.5), % 样本点与预测点的时间分割线

h7.XLim = [0,T+302];

legend([h4,h5,h6],'CPI','预测值',...

'95置信区间','Location','Northwest')

title('CPI预测值')

hold off

fin(:,2) = yF

附录9：ARIMA作图.m

clear

%% ARIMA(p,d,q)消费指数时间序列数据

% Australian Consumer Price Index (CPI) measured from 1972 and 1991,

load Data\_JAustralian

y = DataTable.PAU;

T = length(y);

%% 平稳性的单位根检验

% adftest

[h, p] = adftest(y)

% pptest

[h, p] = pptest(y)

%% 差分

diff\_y = diff(y); % 一阶差分运算

%% 平稳性的单位根检验

[h, p] = adftest(diff\_y) % 平稳性adf检验

%% autocorr

figure(3), autocorr(diff\_y); %自相关系数图

%% parcorr

figure(4), parcorr(diff\_y); %偏自相关系数图

%% 建立选定的较优模型ARIMA(2,1,0)

mdl = arima(2, 1, 0);

EstMdl = estimate(mdl, y);

%% 残差检验

% 可视化分析

res = infer(EstMdl,y);

%% Ljung-Box Q (lbq) 检验

[h, p] = lbqtest(res)

%% 预测

[yF,yMSE] = forecast(EstMdl,20,'Y0',y);

UB = yF + 1.96\*sqrt(yMSE); % 95%置信区间上限

LB = yF - 1.96\*sqrt(yMSE); % 95%置信区间下限

figure(6)

h4 = plot(y,'Color',[.75,.75,.75],'LineWidth',2);

hold on

h5 = plot(T+1:T+20,yF,'r','LineWidth',2);

h6 = plot(T+1:T+20,UB,'k--','LineWidth',1.5);

plot(T+1:T+20,LB,'k--','LineWidth',1.5);

h7 = gca;

plot((T+0.5)\*[1;1], h7.YLim, 'k--','LineWidth',1.5), % 样本点与预测点的时间分割线

h7.XLim = [0,T+22];

legend([h4,h5,h6],'CPI','预测值',...

'95置信区间','Location','Northwest')

title('CPI预测值')

hold off

附录10：H\_safe.mat

（由于该数据不能直接用txt格式展示，故此处只展示其数值）

0 1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 1 1 0 1 0 0 1 0 0 0 1 1 1 1 1 0 0 1 0 0 1 1 10 0 0 0 0 1 0 1 1 0 10 1 1 0 1 1 0 0 0 10 1 1 0 1 1 0 0 0 1 1 1 1 0 0 1 0 1 1 0 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 1 0 10 1 1

附录11：第4题计算.py

#!/usr/bin/env python  
# coding: utf-8  
  
# In[175]:  
  
  
import numpy as np  
import pandas as pd  
  
  
# In[176]:  
  
  
data = pd.read\_excel('全时段数据.xlsx')  
  
  
# In[177]:  
  
  
data.drop(data.columns[0], axis=1, inplace=True)  
data  
  
  
# In[178]:  
  
  
data\_np=np.array(data)  
  
  
# In[179]:  
  
  
AC = pd.read\_excel('Autocorrelation\_coefficient\_100.xlsx')  
  
  
# In[180]:  
  
  
AC.drop(AC.columns[0], axis=1, inplace=True)  
  
  
# In[181]:  
  
  
AC  
  
  
# In[182]:  
  
  
ac\_np=np.array(AC)  
ac\_np.shape  
  
  
# In[183]:  
  
  
type2=np.array([1,2,3,10,15,16,18,19,30,31,32,34,38,39,54,56,60,63,69,74,79,81,84,85,86,87,89,91,92,97,99])  
type3=np.array([4,7,11,17,20,21,23,41,47,57,65,71,72,73,76,96])  
  
  
# In[184]:  
  
  
type3.shape[0]  
  
  
# In[185]:  
  
  
type2.shape[0]  
  
  
# ### 数据导入完成  
  
# In[186]:  
  
  
score1=np.zeros(48\*100).reshape(48,100)  
score1.shape  
  
  
# In[187]:  
  
  
data\_ave=np.mean(data\_np,axis=0)  
data\_var=np.var(data\_np,axis=0)  
data\_ave.shape[0]  
  
  
# In[188]:  
  
  
n=5760  
  
  
# In[189]:  
  
  
data\_var=data\_var\*n  
  
  
# In[190]:  
  
  
data\_div=np.zeros(n\*100).reshape(n,100)  
  
  
# In[191]:  
  
  
import math   
  
  
# data\_power 表示正态间的权重关系  
  
# In[192]:  
  
  
data\_power=np.zeros(100)  
for i in range(100):  
    data\_power[i]=math.sqrt(data\_var[i])/data\_ave[i]  
  
  
# In[193]:  
  
  
for I in range(type2.shape[0]):  
    J=type2[I]  
    for  i in range(n):  
        data\_div[i,J]=(data\_np[i,J]-data\_ave[J])\*(data\_np[i,J]-data\_ave[J])/(data\_var[J]+0.00001)  
data\_div  
  
  
# data\_div 表示一个时刻正态的分数  
  
# In[194]:  
  
  
data\_score=np.zeros(48\*100).reshape(48,100)  
data\_score.shape  
  
  
# In[195]:  
  
  
for j in range(100):  
    for i in range(48):  
        for k in range(60):  
            c=i\*60+k  
            data\_score[i,j]=data\_score[i,j]+data\_div[c,j]  
  
  
# In[196]:  
  
  
data\_score  
  
  
# ### 正态分布处理完成  
  
# In[197]:  
  
  
ac\_score=np.zeros(48\*100).reshape(48,100)  
data\_score.shape  
  
  
# In[198]:  
  
  
for j in range(100):  
    for i in range(48):  
        sum=0  
        for k in range(60):  
            c=i\*60+k  
            sum=sum+ac\_np[c,j]  
        sum=sum/60  
        ac\_score[i,j]=sum  
  
  
# In[208]:  
  
  
tmp2=pd.DataFrame(ac\_score)  
tmp2.to\_excel('ac\_score.xlsx')  
  
  
# In[200]:  
  
  
data\_score=data\_score\*48  
  
  
# ### 自相关处理完成  
  
# In[202]:  
  
  
data\_power\_sum=np.sum(data\_power)  
  
  
# In[203]:  
  
  
score\_fin=np.zeros(48)  
  
  
# In[204]:  
  
  
for i in range(48):  
    for J in range(type2.shape[0]):  
        j=type2[J]  
        score1[i,j]=data\_score[i,j]  
        score\_fin[i]=score\_fin[i]+data\_score[i,j]\*data\_power[j]/data\_power\_sum\*31  
        print(i,j,data\_score[i,j],data\_power[j])  
    for J in range(type3.shape[0]):  
        j=type3[J]  
        score1[i,j]=ac\_score[i,j]  
        score\_fin[i]=score\_fin[i]+ac\_score[i,j]  
  
  
# In[205]:  
  
  
#tmp2=pd.DataFrame(data\_power)  
#tmp2.to\_excel('data\_power.xlsx')  
  
  
# In[206]:  
  
  
score\_fin  
  
  
# In[211]:  
  
  
score2=100-score\_fin  
  
  
# In[212]:  
  
  
S=(score2-np.min(score2))/( np.max(score2)-np.min(score2) )\*100  
tmp2=pd.DataFrame(S)  
tmp2.to\_excel('S.xlsx')  
  
  
# In[ ]:

附录12：灵敏度测试.py

#!/usr/bin/env python  
# coding: utf-8  
  
# In[1]:  
  
  
import numpy as np  
import pandas as pd  
  
  
# In[2]:  
  
  
data = pd.read\_excel('全时段数据.xlsx')  
  
  
# In[3]:  
  
  
data.drop(data.columns[0], axis=1, inplace=True)  
data  
  
  
# In[4]:  
  
  
data\_np=np.array(data)  
  
  
# In[5]:  
  
  
AC = pd.read\_excel('Autocorrelation\_coefficient\_100.xlsx')  
  
  
# In[6]:  
  
  
AC.drop(AC.columns[0], axis=1, inplace=True)  
  
  
# In[7]:  
  
  
AC  
  
  
# In[8]:  
  
  
ac\_np=np.array(AC)  
ac\_np.shape  
  
  
# In[9]:  
  
  
type2=np.array([1,2,3,10,15,16,18,19,30,31,32,34,38,39,54,56,60,63,69,74,79,81,84,85,86,87,89,91,92,97,99])  
type3=np.array([4,7,11,17,20,21,23,41,47,57,65,71,72,73,76,96])  
  
  
# In[10]:  
  
  
type3.shape[0]  
  
  
# In[11]:  
  
  
type2.shape[0]  
  
  
# ### 数据导入完成  
  
# In[12]:  
  
  
score1=np.zeros(48\*100).reshape(48,100)  
score1.shape  
  
  
# In[13]:  
  
  
data\_ave=np.mean(data\_np,axis=0)  
data\_var=np.var(data\_np,axis=0)  
data\_ave.shape[0]  
  
  
# In[14]:  
  
  
n=5760  
  
  
# In[15]:  
  
  
data\_var=data\_var\*n  
  
  
# In[16]:  
  
  
data\_div=np.zeros(n\*100).reshape(n,100)  
  
  
# In[17]:  
  
  
import math   
  
  
# data\_power 表示正态间的权重关系  
  
# In[18]:  
  
  
data\_power=np.zeros(100)  
for i in range(100):  
    data\_power[i]=math.sqrt(data\_var[i])/data\_ave[i]  
  
  
# In[19]:  
  
  
for I in range(type2.shape[0]):  
    J=type2[I]  
    for  i in range(n):  
        data\_div[i,J]=(data\_np[i,J]-data\_ave[J])\*(data\_np[i,J]-data\_ave[J])/(data\_var[J]+0.00001)  
data\_div  
  
  
# data\_div 表示一个时刻正态的分数  
  
# In[20]:  
  
  
data\_score=np.zeros(48\*100).reshape(48,100)  
data\_score.shape  
  
  
# In[21]:  
  
  
for j in range(100):  
    for i in range(48):  
        for k in range(60):  
            c=i\*60+k  
            data\_score[i,j]=data\_score[i,j]+data\_div[c,j]  
  
  
# In[22]:  
  
  
data\_score  
  
  
# ### 正态分布处理完成  
  
# In[23]:  
  
  
ac\_score=np.zeros(48\*100).reshape(48,100)  
data\_score.shape  
  
  
# In[24]:  
  
  
for j in range(100):  
    for i in range(48):  
        sum=0  
        for k in range(60):  
            c=i\*60+k  
            sum=sum+ac\_np[c,j]  
        sum=sum/60  
        ac\_score[i,j]=sum  
  
  
# In[25]:  
  
  
tmp2=pd.DataFrame(ac\_score)  
tmp2.to\_excel('ac\_score.xlsx')  
  
  
# In[26]:  
  
  
data\_score=data\_score\*48  
  
  
# ### 自相关处理完成  
  
# In[27]:  
  
  
data\_power\_sum=np.sum(data\_power)  
  
  
# In[28]:  
  
  
score\_fin=np.zeros(48)  
  
  
# In[29]:  
  
  
for i in range(48):  
    for J in range(type2.shape[0]):  
        j=type2[J]  
        score1[i,j]=data\_score[i,j]  
        score\_fin[i]=score\_fin[i]+data\_score[i,j]\*data\_power[j]/data\_power\_sum\*31  
        print(i,j,data\_score[i,j],data\_power[j])  
    for J in range(type3.shape[0]):  
        j=type3[J]  
        score1[i,j]=ac\_score[i,j]  
        score\_fin[i]=score\_fin[i]+ac\_score[i,j]\*0.33333  
  
  
# In[30]:  
  
  
#tmp2=pd.DataFrame(data\_power)  
#tmp2.to\_excel('data\_power.xlsx')  
  
  
# In[31]:  
  
  
score\_fin  
  
  
# In[32]:  
  
  
score2=100-score\_fin  
  
  
# In[33]:  
  
  
S=(score2-np.min(score2))/( np.max(score2)-np.min(score2) )\*100  
tmp2=pd.DataFrame(S)  
tmp2.to\_excel('S.xlsx')  
  
  
# In[ ]:

附录13：数据审视.py

#!/usr/bin/env python  
# coding: utf-8  
  
# In[3]:  
  
  
import numpy as np  
import pandas as pd  
data = pd.read\_excel('附件1.xlsx', index\_col=1, header=0, parse\_dates=True)  
data  
  
  
# In[4]:  
  
  
data.drop(data.columns[0], axis=1, inplace=True)  
  
  
# In[5]:  
  
  
data\_np=np.array(data)  
  
  
# 已经转为np类型了。  
#   
  
# In[6]:  
  
  
# np.mean(a, axis=0) # axis=0，计算每一列的均值  
  
  
# In[7]:  
  
  
# 求出平均值  
data\_ave=np.mean(data\_np,axis=0)  
  
  
# In[8]:  
  
  
# 求出方差  
data\_var=np.var(data\_np,axis=0)  
data\_var  
  
  
# 第一步预处理，先用处理离群点的方法处理冲击性数据  
  
# In[9]:  
  
  
import copy  
  
  
# In[10]:  
  
  
for i in range(100):  
    slct=data\_np[:,i]  
    a1=np.percentile(slct,25,interpolation='midpoint')  
    a2=np.percentile(slct,50,interpolation='midpoint')  
    a3=np.percentile(slct,75,interpolation='midpoint')  
    up\_lim=a3\*2-a2  
    down\_lim=a1\*2-a2  
    for j in range(5519):  
        if (data\_np[j][i]>up\_lim or data\_np[j][i]<down\_lim):  
            data\_np[j][i]=a2  
  
  
# 第二步预处理，处理符合正态分布的列  
  
# In[11]:  
  
  
import scipy.stats as stats  
  
  
# In[12]:  
  
  
for i in range(100):  
    if (data\_var[i]<1e-4):  
        for j in range(5519):  
            data\_np[j][i]=data\_ave[i]  
  
  
# In[13]:  
  
  
ans1=np.zeros(300).reshape(3,100)  
for i in range(100):  
    slct=data\_np[:,i]  
    sta,pvalue=stats.shapiro(slct)  
    ans1[0,i]=sta  
    ans1[1,i]=pvalue  
    if (sta>0.9):  
        ans1[2:i]=1  
  
  
# In[14]:  
  
  
ans2=pd.DataFrame(ans1)  
ans2.to\_excel('A.xlsx')  
  
  
# In[ ]:  
  
  
  
  
  
# In[ ]:

附录14：数据预处理.py

#!/usr/bin/env python  
# coding: utf-8  
  
# In[20]:  
  
  
import numpy as np  
import pandas as pd  
data = pd.read\_excel('data.xlsx', index\_col=1, header=0, parse\_dates=True)  
data  
  
  
# In[21]:  
  
  
data.drop(data.columns[0], axis=1, inplace=True)  
  
  
# In[22]:  
  
  
data\_np=np.array(data)  
  
  
# 已经转为np类型了。  
#   
  
# In[23]:  
  
  
# np.mean(a, axis=0) # axis=0，计算每一列的均值  
  
  
# In[24]:  
  
  
# 求出平均值  
data\_ave=np.mean(data\_np,axis=0)  
  
  
# In[25]:  
  
  
# 求出方差  
data\_var=np.var(data\_np,axis=0)  
data\_var  
  
  
# 第一步预处理，先用处理离群点的方法处理冲击性数据  
  
# In[26]:  
  
  
import copy  
  
  
# In[27]:  
  
  
for i in range(100):  
    slct=data\_np[:,i]  
    a1=np.percentile(slct,25,interpolation='midpoint')  
    a2=np.percentile(slct,50,interpolation='midpoint')  
    a3=np.percentile(slct,75,interpolation='midpoint')  
    up\_lim=a3\*2-a2  
    down\_lim=a1\*2-a2  
    for j in range(5519):  
        t=0  
        if (data\_np[j][i]>up\_lim or data\_np[j][i]<down\_lim):  
            data\_np[j][i]=a2  
            t=1  
    if(t==1):  
        print(i)  
              
  
  
# 第二步预处理，处理符合正态分布的列  
  
# In[19]:  
  
  
ans2=pd.DataFrame(data\_np)  
ans2.to\_excel('A.xlsx')  
  
  
# In[ ]:

附录15：原始数据作图.py

#!/usr/bin/env python

# coding: utf-8

# In[1]:

# -\*- coding: utf-8 -\*-

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.graphics.tsaplots import plot\_acf

from matplotlib.pyplot import acorr

from scipy.stats import t

def drawing(data):

'''

画出数据随着时间的变化图

'''

cols = data.columns

time\_series = range(1,data.shape[0]+1)

for col in cols:

fig = plt.figure()

plt.plot(time\_series, data[col])

plt.xlabel('Time Index', fontsize=16)

plt.ylabel('Value', fontsize=16)

plt.title(f'Sensor\_{col}', fontsize=20)

fig.savefig(f'传感器{col}.png')

plt.grid()

plt.show()

# In[2]:

def save\_medium(verbose=True):

'''

数据的统计描述，并保存

'''

print('数据的统计信息如下： \n', data.describe())

data\_summary = data.describe()

data\_summary.to\_excel('数据统计描述.xlsx')

# In[3]:

data = pd.read\_excel('附件1(Appendix 1)2021-51MCM-Problem C.xlsx', index\_col=1, header=0, parse\_dates=True)

# 删除第一列（时间编号）

data.drop(data.columns[0], axis=1, inplace=True)

# In[4]:

a = data.iloc[:, 5].autocorr(lag=1)

save\_medium(verbose=True)

# 画出传感器读数-时间图

drawing(data)

# In[ ]: