AIS 轨迹数据重建、压缩与预测

摘要

近年来随着水上经济活动日益频繁，对繁忙水域交通信息处理能力提出了更高要求。随着船舶自动识别系统 (AIS) 的广泛使用，大量与船舶有关的信息得以自动收集，为水上交通管理提供了必要的数据支撑。但收集得到的 AIS 数据仍存在数据异常、数据缺失、数据量冗余等问题。另一方面，如何利用 AIS 数据对船舶轨迹进行预测也存在挑战。本文针对这些问题，从 AIS 原始数据出发，完成了数据重建，数据压缩和数据预测。

针对问题一，由于原始的 AIS 数据存在大量的异常点，在进行了必要的数据预处理之后，我们使用了 **DBSCAN 聚类分析**对原始数据中存在的异常点进行了检测并加以清除。接着，针对原始数据中大量的数据缺失，我们以秒为单位，利用**三次样条插值法**完成了缺失值的补全。经过数据重建后的 AIS 数据能够正常而精确地反映船舶的运动轨迹。

针对问题二，由于 AIS 数据量庞大（重建之后的数据量更为庞大），大大提高了数据存储成本和数据使用难度，我们基于 **Douglas-Peucker 算法**对 AIS 二维轨迹数据进行了压缩。针对 DP 算法存在超参数的问题，我们以压缩率和平均误差为评价指标，利用经过**熵权法优化的 TOPSIS 综合评价模型**对不同参数下 DP 算法的结果进行评估。在重建之后的数据集上，经过遴选的数据压缩模型能够将数据集压缩 99.6%，同时保持平均误差小于 10 米，在极大减少数据量的同时保持了原有轨迹。

针对问题三，我们首先构建了长江大桥的断面方程，并基于压缩之后的数据集进行历史各时段的长江大桥流量统计。得到统计结果之后，考虑历史数据较少，各时段通行流量情况复杂，我们采用了$GM(1,1)$ **灰色预测模型**对 10 月 26 日长江大桥各时段的流量进行了预测。稳健的灰色预测模型为预测结果提供了保障。

关键字： AIS 轨迹数据　DBSCAN 聚类分析　三次样条插值　Douglas-Peucker 算法　熵权法　Topsis 法　灰色预测

目录

一、 问题重述

1.1 问题背景

近年来随着国家在水路运输领域重大战略决策的出台和实施，水上经济活动日益频繁，但相应地也对繁忙水域交通信息处理能力提出了更高要求。随着船舶自动识别系统 (AIS) 的广泛使用，大量与船舶有关的信息得以自动收集，为有效进行水上交通管理提供了必要的数据支撑。但收集得到的 AIS 数据仍存在数据异常、数据缺失、数据量大等问题。另一方面，如何利用 AIS 数据对船舶轨迹进行预测也存在挑战。

1.2 问题要求

根据上述题目背景与数据集，题目要求建立数学模型以讨论以下问题：

1. 对收集到的原始 AIS 数据进行异常点去除以及丢失数据补全，以完成数据重建；

2. 对重建后的 AIS 数据集进行数据压缩，并评价压缩质量；

3. 利用重建后的 AIS 数据集，以长江大桥为观测断面，统计并预测 2016 年 10 月 26 日在不同时间段内通过该观测断面的船舶交通流量。

二、 模型假设

- 在数据收集到预测的时间范围内，没有船只发生损毁等异常情况;
- 被收集数据的船只不会在短时间内大幅度改变运动状态，包括但不限于速度、方位等;
- AIS 所提供的指标主要在定位指标上出现异常;
- AIS 数据的冗余主要是出现在二维轨迹的坐标描述上;
- 预测日期，即 10 月 26 日，长江大桥没有出现意外情况;

三、 符号说明

符号	意义
X_i	指标向量
Eps	DBSCAN 半径参数
$MmPts$	DBSCAN 点数参数
ϕ	经度
λ	纬度
D_{max}	Douglas-Peucker 最大阈值
R	数据压缩率
$MErr$	平均误差
w	权重
Z	归一化矩阵
X_{ij}	第 i 个模型第 j 个指标的数据

此处仅列出全文通用符号，个别符号将会在第一次使用时进行说明

四、 问题分析

4.1 问题一

在原始的 AIS 轨迹数据集中，存在较多的异常数据，也存在较多数据丢失的情况，问题一要求我们对原始 AIS 轨迹数据集进行异常值清理与检测，并在此基础上进行缺失值的填充。在异常值检测方面，我们假设数据异常的原因来自于经度和纬度，利用**DBSCAN 聚类分析**方法，我们能较好地检测出数据中的“跳跃点”并将其删去。在缺失值填充方面，我们考虑使用**三次样条插值**对缺失值进行插值填补，在填充缺失值的同时保持了船舶轨迹较好的平滑性。

4.2 问题二

由于 AIS 原始数据量就已经很庞大，重建之后的数据集为了精确描述各个时间的运行轨迹将更为庞大，大大提高了数据存储成本和数据使用难度，必须要在数据重建的基础上对新的数据集进行压缩。压缩的主要思路是从二维轨迹的描述切入，我们可

以使用原数据集中的部分关键点代表原有的行驶轨迹。另一方面，我们也将建立起数据压缩的评估模型，主要是从压缩率和数据的失真率出发，评估我们的数据压缩模型，从而遴选出性能最优的压缩方法，在极大减少数据量的同时保持原有轨迹。

4.3 问题三

针对问题三，我们将构建长江大桥的断面方程，从而判断船只是否有经过长江大桥截面的行为。基于压缩之后的数据集，我们将进行历史各时段的长江大桥流量统计。得到统计结果之后，考虑历史天数数据较少，各时段通行流量情况复杂，我们拟采用$GM(1,1)$ **灰色预测模型**对 10 月 26 日长江大桥各时段的流量进行了预测。稳健的灰色预测模型能够为预测结果提供保障。

五、 模型建立与求解

5.1 问题一:AIS 数据集重建

首先，对于每一份文件内记录的 AIS 原始数据，我们采用 **DBSCAN 聚类分析** [1] 技术检测出其中的异常点并将其去除。接着，对于缺失时间段内的数据，我们采用了**三次样条插值** [2] 方法进行插值，较为平滑地完成了缺失值补全。最终，将数据集保存，完成了数据集的重建。

5.1.1 异常检测——DBSCAN 聚类分析

数据集重建第一步为异常点检测并删除。异常点检测上就是无监督分类的一种，基于 AIS 数据集中的经纬度指标，我们将正常与异常性质的数据点分为两类类。常见的聚类方法包括 K-means 聚类、层次聚类等，但以上的聚类方法一般只适用于凸样本集的聚类，在轨迹类的点集聚类中效果较差。为此，我们提出一种改进的基于 DBSCAN 的聚类算法进行异常值检测。

步骤一：数据预处理

DBSCAN 聚类算法中最重要的参数就是区域半径的选取，它对于数据的尺度是很敏感的，所以我们首先需要对数据进行标准化以消除量纲带来的影响。对于每一个文件，我们取出 AIS 数据中的经度与纬度数据，分别记为记为 X_1, X_2，标准化的公式如下：

$$X_{ij} = \frac{X_{ij} - e_i}{s_i} \tag{1}$$

$$e_i = \frac{1}{n}\sum_{j=1}^{n} X_{ij} \tag{2}$$

$$s_i = \sqrt{\frac{\sum_{j=1}^{n}(X_{ij} - e_i)^2}{n-1}} \tag{3}$$

步骤二：DBSCAN 聚类分析

DBSCAN(Density-Based Spatial Clustering of Applications with Noise）是一种基于密度的空间聚类算法。它将簇定义为密度相连的点的最大集合，该算法将具有足够密度的区域划分为簇 (即要求聚类空间中的一定半径区域内所包含对象的数目不小于某一给定阈值), 并在具有噪声的空间数据库中发现任意形状的簇。DBSCAN 算法中有两个重要参数：Eps 和 $MmPtS$。Eps 是定义密度时的邻域半径，$MmPts$ 为定义核心点时的阈值。并基于此将数据点分为 3 类：

- 核心点：半径 Eps 范围内含有超过 MmPts 数目的点;
- 边界点：其半径 Eps 内含有点的数量小于 MinPts，但是该点落在核心点的邻域内;
- 异常点：既不是核心点也不是边界点。

DBSCAN 聚类分析的算法示意图与算法流程图分别如图 1 和图 2 所示。

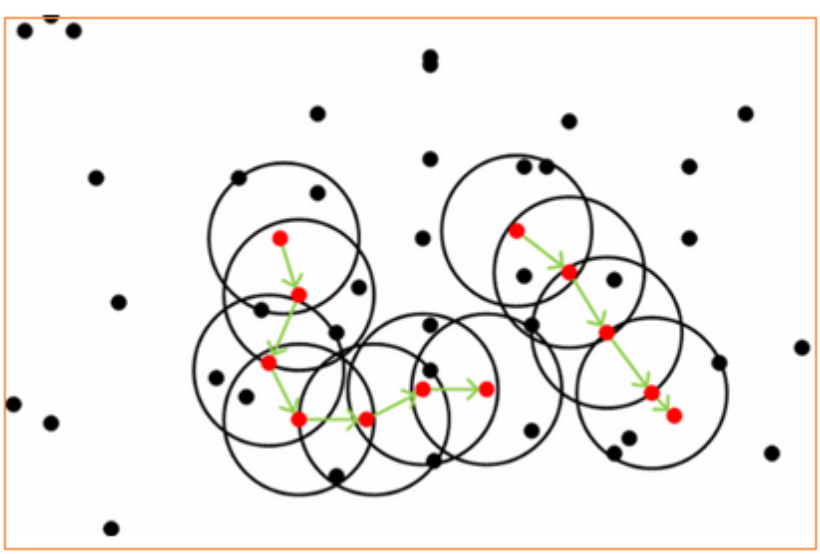


图 1　DBSCAN 算法示意图

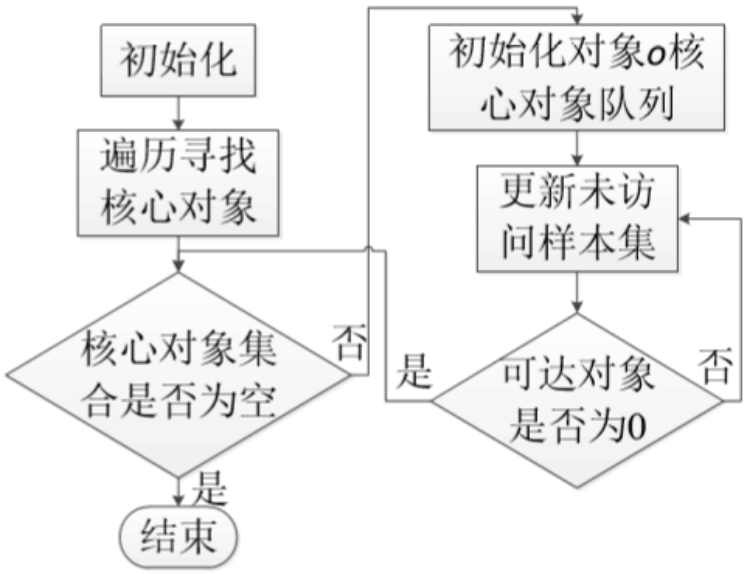


图 2　DBSCAN 算法流程图

5.1.2 补全缺失数据——三次样条插值

在删除数据中的异常点之后，我们需要对数据集的缺失数据进行填充，缺失情况主要来自两个方面：

- 原本数据收集的缺失;
- 删除异常点导致的缺失。

我们采用了三次样条插值技术对数据集进行了补全，保持了船只各个指标变化的平滑性，效果较好。

步骤一：数据预处理

我们将个指标看成关于时间 t 的函数，首先要将文件中“时 (h)-分 (m)-秒 (s)”的字符串时间戳转化为实数 t（单位：秒），转化公式为：

$$t = 3600h + 60m + s \tag{4}$$

步骤二：三次样条插值

所谓样条本来是工程设计中使用的一种绘图工具，它是富有弹性的细木条或细金属条。绘图员利用它把一些已知点连接成一条光滑曲线（称为样条曲线），并使连接点处有连续的曲率。

数学上将具有一定光滑性的分段多项式称为样条函数。具体地说，给定区间 $[a, b]$ 上的一个分划 $\triangle : a = x_0 < x_1 < \cdots < x_n = b$，如果函数 $s(x)$ 满足：

- 在每个小区间 $[x_i, x_{i+1}](i = 0, 1, \ldots, n-1)$ 上 $s(x)$ 是 k 次多项式;

- $s(x)$ 在 $[a, b]$ 上具有 $k-1$ 阶连续导数。

则称 $s(x)$ 为关于分划 $\triangle$ 的 k 次样条函数，其图形称为 k 次样条曲线。当 $k = 3$ 的情况下，便是我们插值算法采用的三次样条函数，其一般形式为：

$$s_3(x) = a_0 + a_1 x + \frac{a_2}{2!}x^2 + \frac{a_3}{3!}x^3 + \sum_{j=1}^{n-1} \frac{b_j}{3!}(x - x_j)_+^3 \tag{5}$$

$$(x - x_j)_+^3 = \begin{cases} (x - x_j)^3 & x \geq x_j \\ 0 & x \leq x_j \end{cases} \tag{6}$$

我们使用三次样条插值对原始数据集进行了填充，完成了对数据集的重构。

5.1.3 模型求解与结果分析

(1)DBSCAN 模型求解

我们基于 Python 语言，利用功能优秀的机器学习库 Sklearn[3]，对每一个数据文件完成了数据标准化和 DBSCAN 聚类分析，实现了异常点的检测与清除。由于文件数量过多，难以展示全部异常点数据。在此，我们仅以“20161017(55).txt”与“20161017(88).txt”两个数据为例进行展示，DBSCAN 聚类分析的结果如图 3 和图 4 所示 (其中黄色点为正常点，紫黑色点为异常点)，可以看出 DBSCAN 算法对异常点（跳跃点）具有较好的检测功能。

(2) 三次样条插值模型求解

基于已去除异常点的文件，我们首先利用 Python 语言将每个文件中的时间戳转化为数值坐标，并将文件保存。之后利用 MATLAB 软件进行三次样条插值算法的应用，补全了缺失数据集，完成了数据集的重建。由于文件与指标数量过多，我们仅选取“20161017(1).txt”和“20161017(98).txt”的经纬度进行展示插值之后的结果，如图 5 和图 6 所示。

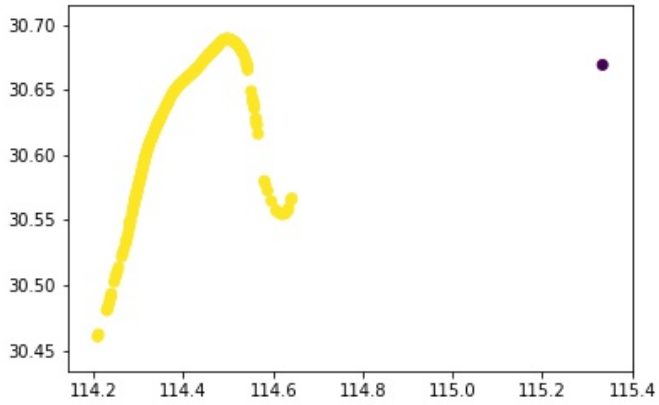


图 3　20161017(55) 结果图

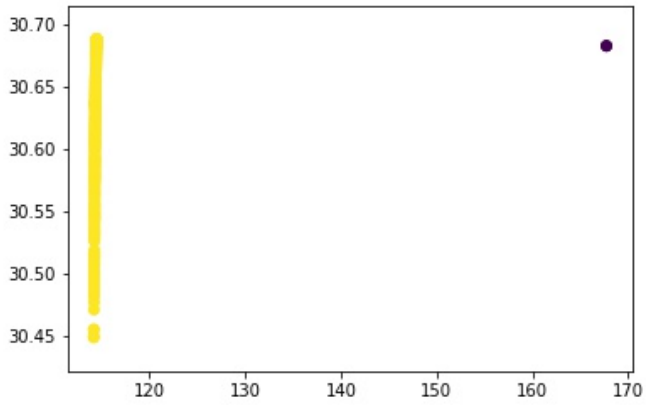


图 4　20161017(88) 结果图

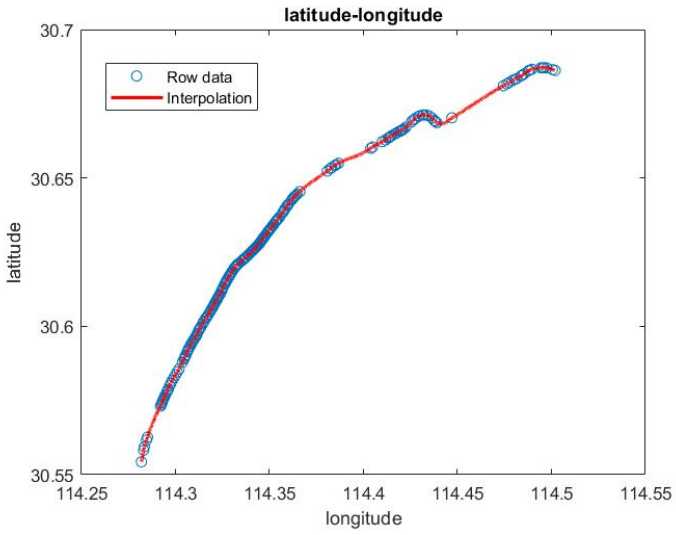


图 5　20161017(98) 插值结果图

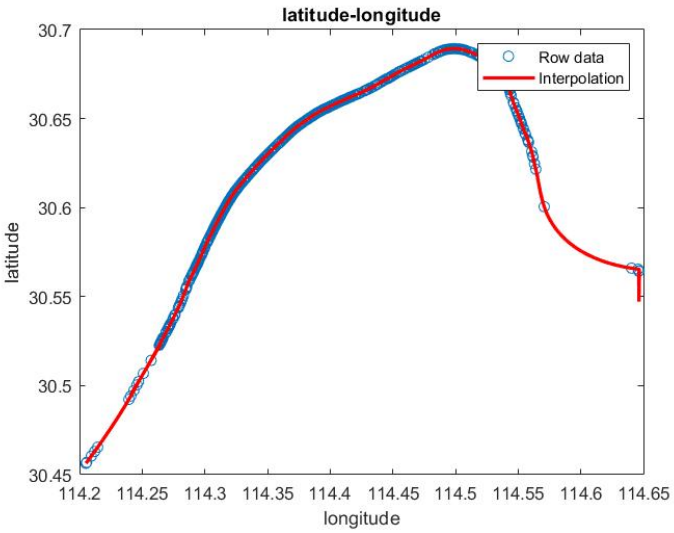


图 6　20161017(1) 插值结果图

5.2 问题二：AIS 轨迹数据压缩

不论是收集到的原始 AIS 轨迹数据，还是经过第一问数据重建后的 AIS 轨迹数据集，都存在内存过于冗余的情况，大大提高了数据存储成本，降低了 ECDIS 平台回放和再现船舶轨迹的效率。在大量调研前人研究的基础上，我们采用了 **Douglas-Peucker 算法** [4, 5] 进行船舶 AIS 航迹数据压缩。并采取压缩率作为与平均误差作为评价准则，基于**熵权法和 TOPSIS 法**建立了较为科学的评价体系。经遴选得到的压缩模型在很好保证数据质量的同时，实现了数据压缩，效果优异。

5.2.1 算法数据压缩模型——Douglas-Peucker 算法

我们选取数据集的经纬度作为研究对象，压缩目标是压缩运动轨迹，在能够真实反映船舶原始运动轨迹的情况下，选用尽可能少的点作为我们的保留点，通过减少数据点实现压缩目标。我们选取了 Douglas-Peucker 算法作为我们的数据压缩方式，在基本保留运行特征的同时极大压缩了数据。

(1) 数据预处理

AIS 数据中坐标以经纬度的形式进行存储，再压缩前要先将经纬度坐标转化为墨卡托坐标，然后转化为屏幕坐标显示。根据等角正圆柱投影定理，若某点经纬度坐标为 (ϕ, λ)，则其转化为屏幕坐标 (x, y) 的公式为：

$$f(x)=\begin{cases} r_0 = & \dfrac{a}{1-e^2sin^2\phi}cos\phi \\ q = & lntan(\dfrac{\pi}{2}+\dfrac{\phi}{4})-\dfrac{e}{2}ln\dfrac{1+esin\phi}{1-esin\phi} \\ r_0 = & r_0\lambda \\ r_0 = & r_0 q \end{cases} \tag{7}$$

其中 a 为地球椭球长轴半径，e 为椭球第一偏心率。

(2) Douglas-Peucker 压缩算法

Douglas-Peucker 算法正是基于压缩二维运动轨迹设计的，以图 7 为例说明 Douglas-Peucker 算法的原理。将首点和尾点分别作为初始锚点和初始漂浮点, 并将其连成的直线称为基线。依次计算中间各点到基线的垂直距离, 并找出最大距离对应的点。若最大距离小于设定的阈值, 则用基线代替原始曲线。若最大距离大于设定的阈值, 则将最大距离对应的点作为分裂点, 并将该分裂点作为前向点集的漂浮点和后向点集的锚点。依次递归选取分裂点和分段, 直到每一个子集中都不再出现新的分裂点。

由 Douglas-Peucker 算法提取出的特征点之所以能够反映原始曲线的总体和局部特征, 是因为活跃的基线始终由当前最有特点的 2 个点连接而成, 而后以基线作为“视平线”去“端详”最大局部的区段, 并选取该区段的主要特征点。

我们选取了一系列不同的最大距离阈值使用 Douglas-Peucker 算法进行数据压缩，得到了一系列的压缩率和平均误差。

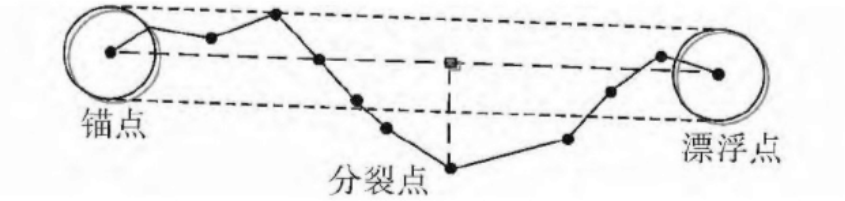
5.2.2 压缩质量评价模型——熵权法优化的 TOPSIS 算法

(1) 熵权法获取评价指标权重

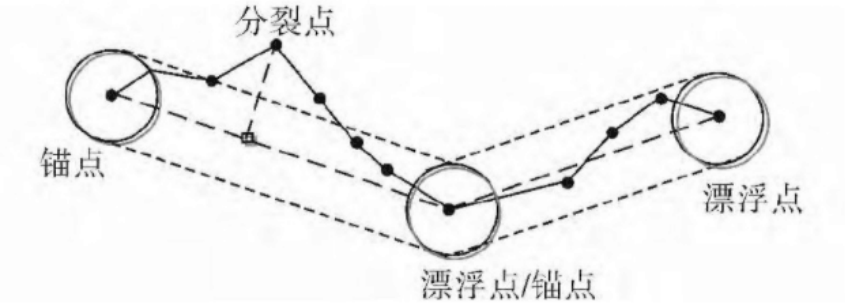
熵权法 [6] 是可以根据各种指标的数据直接生成各指标权重的一种方法，可以避免其它权重获得方法（如层次分析法）中人为的主观偏差，完全客观地得到各指标权重。假设评估指标为 X_1, X_2, 其中 $X_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$，$n$ 为我们选取不同阈值进行实验的次数。

步骤一：指标正向化

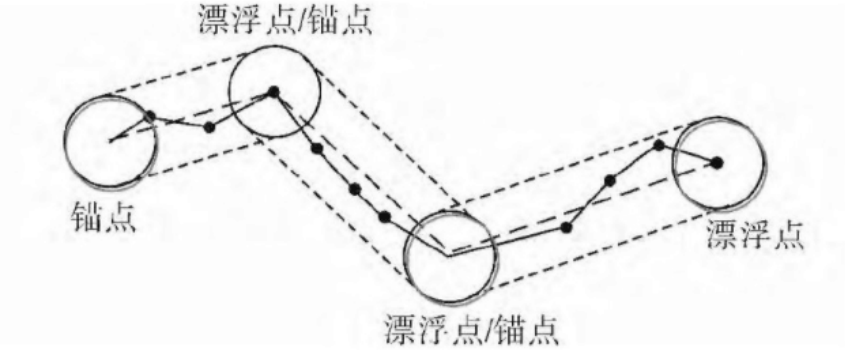
- if X_j 为收益型，则 $X_{ij} = X_{ij} - min(X_j)$
- if X_j 为成本型，则 $X_{ij} = max(X_j) - X_{ij}$
- if X_j 为中间型, 则 $X_{ij} = 1 - \frac{|X_{ij}-1|}{max(|X_j-1|)}$



(a) 第 1 步: 选取最大距离点



(b) 第 2 步: 根据分裂点分段执行



(c) 第 3 步: 循环执行,直至无分裂点

图 7　Douglas-Peucker 算法示意图

步骤二：指标规范化

$$Z_{ij} = \frac{X_{ij}}{\sqrt{\sum_{i=1}^{n} X_{ij}^2}} \tag{8}$$

步骤三：计算第 i 个样本在第 j 个指标下的比例，并将其视为相对熵计算中使用的概率。

$$p_{ij} = \frac{Z_{ij}}{\sum_{i=1}^{n} Z_{ij}} \tag{9}$$

步骤四：计算各指标的信息熵，得到各指标的权重。对于第 j 个指标，其信息熵和权重的计算公式为：

$$e_j = \frac{1}{lnn} \sum_{i=1}^{n} p_{ij} ln(p_{ij}), j = 1, 2, 3, ...m \tag{10}$$

$$w_j = \frac{1 - e_j}{m - \sum_{i=1}^{m} e_j}, j = 1, 2, ..., m \tag{11}$$

(2) TOPSIS 法对各个压缩模型进行排序

TOPSIS 法 [7] 是系统工程中一种多目标决策方法，找出有限目标中的最优与最劣目标，当某个可行目标最靠近最优目标同时又远离最劣目标，这个方案解的向量集就是最优目标。

TOPSIS 法作为一种综合指标评价方法，区别于如模糊综合评判法，层次分析评价，它不需要目标函数，也不需要通过相应的检验，即限制要求大大降低，使得适用范围较为广泛。

TOPSIS 方法的流程如下所示:

- 基于之前建立的评价指标体系，将原始得分进行正向化，与熵权法中的正向化一致。

- 进行正向化后，再将数据进行标准化，建立归一化矩阵 Z, 即

$$Z_{ij} = \frac{X_{ij}}{\sqrt{\sum_{i=1}^{n} X_{ij}^2}} \quad i = 1, 2, ...n; j = 1, 2, ..m$$

- 由上述得到的 Z 矩阵，得到最优矩阵 $Z_j^+ = max_i Z_{ij}$，和最劣矩阵 $Z_j^- = min_i Z_{ij}$;

- 计算所选取或构建的每个模型向量与最优向量的欧式距离。$D_i^+ = \sqrt{\sum_{j=1}^{m}(Z_{ij} - Z_j^+)^2}$ 和最劣向量的距离 $D_i^- = \sqrt{\sum_{j=1}^{m}(Z_{ij} - Z_j^-)^2}$;

- 最后得到与最优值的相对接近程度 $Score_i = \frac{D_i^-}{D_i^+ + D_i^-}$, 并且进行排名。

5.2.3 模型求解与结果分析

我们利用 Python 语言，首先将经纬度坐标转化为墨卡托坐标，然后采用不同阈值的 Douglas-Peucker 算法进行点集压缩得到各组评价指标数据。之在 MATLAB 软件上，使用我们建立的评估模型对各个压缩模型进行了评估，选取了其中最优的模型。它在原始 AIS 数据上进行压缩，压缩率可以到达 75%；在重建后庞大而精确的 AIS 数据上，我们的压缩率可以达到 99.65%，且平局误差小于 5 米。经过遴选的 Douglas-Peucker 算法较好地完成了压缩任务。

由于所有文件数据集过大，我们此处仅选择文件“20161017(1).txt”的压缩情况进行展示，其余文件的压缩方式是一致的。另一方面，由于重建之后的数据更能精确反

映船舶运动轨迹的变化情况，我们此处的展示是基于重建的数据集 (“20161017(1).txt”重建后数据点由 369 个变为 59339 个) 进行压缩的。由以上建模流程得到的各个数据以及结果如表 1 和图 8 所示。

表 1　各压缩模型数据和指标权重

D_{max}	R	$MErr$
1	0.9796	0.222
5	0.9931	1.210
10	0.9959	3.531
15	0.9965	3.960
20	0.9969	4.551
30	0.9976	6.478
50	0.9981	8.293
70	0.9984	9.539
100	0.9989	28.392
权重 w	0.4937	0.5063

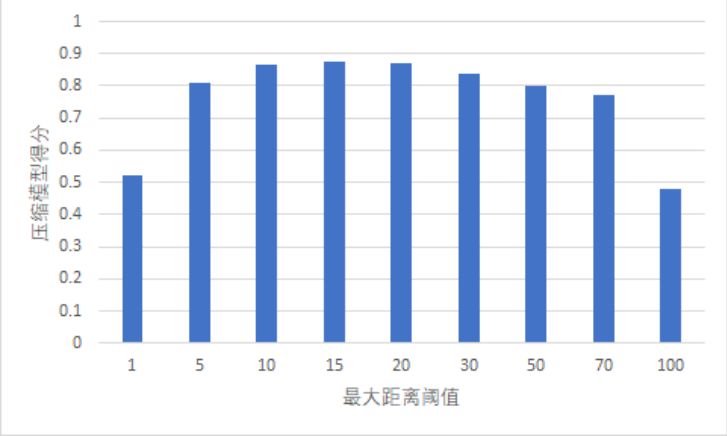


图 8　TOPSIS 得分排序图

可以看出，随着 D_{max} 的不断增大，数据的压缩率越来越大，数据占用的内存甚至可以缩小为原有的 0.02%。但相应的，数据的失真率也越来越高，平均误差由 0.2 变为

28.3，扩大了近 100 倍。基于熵权法优化的 TOPSIS 评价法正是权衡两个指标，完成了压缩模型的取舍。最后我们选择了 D_{max} 为 15 的情况，此时模型压缩率为 0.9965，平均误差为 3.960。如图 9 和 10 所示，此时压缩后的剩余点集仍能较好地反映船舶的运动轨迹，性能优异。

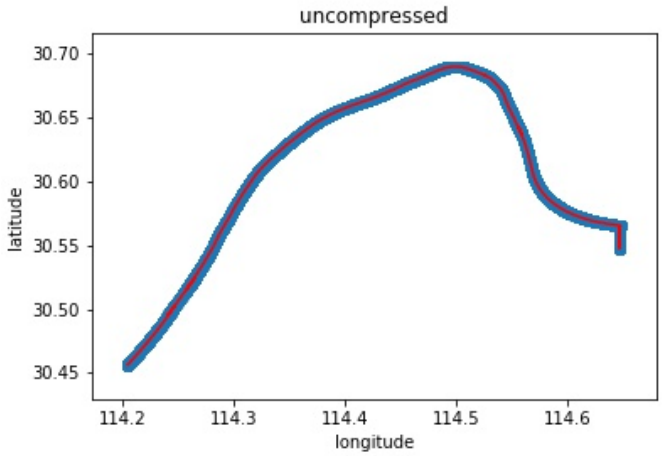


图 9　压缩前数据点图

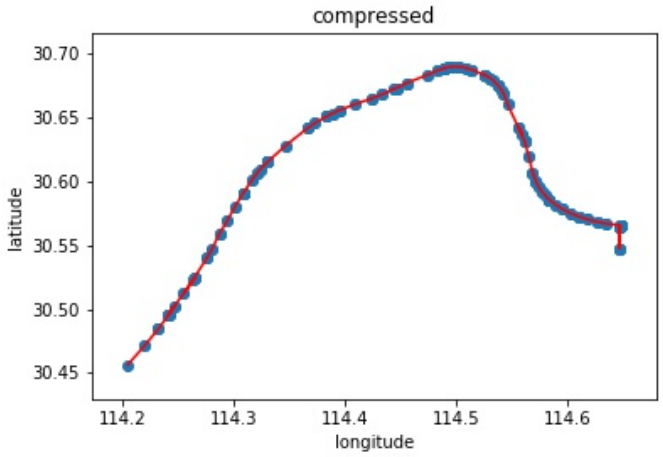


图 10　压缩后数据点图

5.3 问题三：长江大桥流量预测

经过问题一的数据重建和问题二的数据压缩，我们已经获得了轻量级的、能够反映船舶运行轨迹的数据集，现在，我们将统计各日各时段长江大桥的流量，并且对 10 月 26 日各时段长江大桥的流量进行预测。在统计方面，我们查询到武汉长江大桥的两个端点的经纬度分别为 $(30°33'07.4"N, 114°17'02.5"E)$ 和 $(30°32'50.1"N, 114°17'33.8"E)$[8]，通过构建断面方程进行相关流量判断与统计；在预

测方面，由于每个时段的大桥流量是一个复杂的灰色系统，我们采用 **GM(1,1) 灰色预测** [9] 进行了 10.26 日的大桥流量预测。

5.3.1 大桥流量统计

步骤一：由大桥的两个端点坐标构建大桥的断面方程，结果如下：

$$\lambda = -0.5462\phi + 130.7369 \tag{12}$$

步骤二：遍历数据进行统计，设某船只在某一时间段内有数据点集 $x_1, x_2, ..., x_n$，其中 $x_i = (\phi_i, \lambda_i)$, 若存在 i, j，满足：

$$(\lambda_i + 0.5462\phi_i - 130.7369)(\lambda_j + 0.5462\phi_j - 130.7369) < 0 \tag{13}$$

则认为此船只在此时间段内通过武汉长江大桥。

5.3.2 大桥流量预测——GM(1,1) 灰色预测模型

通过大桥流量统计之后，我们获得了长江大桥各时段流量的历史数据。本文考虑运用灰色预测模型 $GM(1,1)$ 预测 10 月 26 日的长江大桥流量。灰色预测是对灰色系统进行的预测, 识别系统因素之间发展趋势的差异程度，对原始数据进行生成和处理，找出系统变化的规律，生成规律性强的数据序列，然后建立相应的微分方程模型来预测事物未来的发展。该方法利用等时间间隔观测到的预测对象特征的一系列定量值构造灰色预测模型，预测未来某一时刻。该模型所需建模信息少，操作方便，建模精度高。它在各种预测领域有着广泛的应用，是处理小样本预测问题的有效工具。

定义历史某一时段的长江大桥流量序列为 $x^{(0)} = x^{(0)}(1), x^{(0)}(2), ..., x^{(0)(n)}$。

步骤一：计算级比

$$\lambda(k) = \frac{x^{(0)}(k-1)}{x^{(0)}(k)}, k = 2, 3, ...n \tag{14}$$

所有级比都落在 $(e^{-\frac{2}{n+1}}, e^{\frac{2}{n+2}})$ 内，可以使用 $GM(1,1)$ 进行灰色预测。

步骤二：对原始序列进行累加，得到新的序列 $x^{(1)} = x^{(1)}(1), x^{(1)}(2), ..., x^{(1)(n)}$。

步骤三：建立序列

$$z^{(1)}(k) = 0.5(x^{(1)}(k-1) + x^{(1)}(k)) \tag{15}$$

步骤四：建立微分方程

$$x^{(0)}(k) + az^{(1)}(k) = u, k = 2, 3, ..., n \tag{16}$$

步骤五：求解得：

$$\hat{x}^{(0)}(k+1)=\hat{x}^{(1)}(k+1)-\hat{x}^{(1)}(k)=\left(1-e^{\hat{a}}\right)\left[x^{(1)}(1)-\frac{\hat{u}}{\hat{a}}\right]e^{-\hat{a}k} \tag{17}$$

步骤六：检验预测值，包括残差检验和级比偏差值检验。

5.3.3 模型求解与结果分析

我们利用 Python 语言进行了大桥流量的统计，并且使用 MATLAB 软件进行了大桥流量的灰色预测，但由于数据量过大，电脑并没有运行出最终的结果，在此表示歉意。但模型建立流程应当是比较合理的。

六、 模型评价与扩展

6.1 问题一模型

6.1.1 模型评价

优点：

(1) 能够较好地将轨迹上的异常点，尤其是跳跃点检测出；

(2) 与常见的聚类方法包括 K-means 聚类、层次聚类等相比，DBSCAN 算法突破了凸样本集的聚类，在轨迹类的点集聚类中效果好；

(3) 插值算法补充缺失值后，使得数据点能够精确反映运动轨迹；

缺点：仅从经纬度坐标，船只运动轨迹的方面定义了异常点，而其他方面没有过多考虑；

6.1.2 模型扩展

将其它特征也考虑进来，包括速度、航向等等，突破数据点本身信息，充分考虑前后关联，建立多元而科学的异常点分类 [10]。

6.2 问题二模型

6.2.1 模型评价

优点：

(1) 在重建后的数据集上压缩率很大

(2) 平均误差率很小，压缩后仍能反映运动轨迹

缺点：小范围的折返情况可能难以捕捉

6.2.2 模型扩展

对于小范围折返较多的船舶提出更为科学的算法（比如渔船）。

参考文献

[1] Ester, M., H. P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press,1996:226–231.

[2] 张黎翔, 朱怡安, 陆伟等. 基于 AIS 数据的船舶轨迹修复方法研究. 西北工业大学学报,2021,39(01),119-125.

[3] Pedregosa et al. Scikit-learn: Machine Learning in Python. JMLR 12, 2011: 2825-2830.

[4] Alan Saalfeld. Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm. Cartography and Geographic Information Science, 1999: 7-18.

[5] 张树凯, 刘正江, 张显库等. 基于 Douglas-Peucker 算法的船舶 AIS 航迹数据压缩. 哈尔滨工程大学学报,2015,36(05):595-599.

[6] ZH Zou, Y Yi, JN Sun.Entropy method for determination of weight of evaluating indicators in fuzzy synthetic evaluation for water quality assessment. Journal of Environmental sciences, 2006.

[7] YJ Lai, TY Liu, CL Hwang. Topsis for MODM. European journal of operational research, 1994.

[8] https://www.google.com/maps/place/Wuhan+Yangtze+River+Bridge/@30.549622,114.2862273,17z/data 2020,4,5

[9] 姜启源, 谢金星, 叶俊.《数学模型（第五版）》：高等教育出版社，2003 年 8 月.

[10] 邓磊. 基于数据挖掘的船舶航行轨迹异常检测方法研究. 武汉理工大学, 2016.

附录 A　DBSCAN 聚类分析 –Python 源程序

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

COLUMNS = ['date','time','ID','longitude','latitude',
          'longitude_last','latitude_last','v','direct','examine']
table = pd.read_csv('DATA/20161017 (1).txt', header = None,
    delim_whitespace=True, names = COLUMNS)
table_size1 = table.shape[0]
print('table_size1: %d' %table_size1)
#table.plot.scatter(x = 'longitude', y= 'latitude')
#plt.show()

X = table[['longitude','latitude']]
X = StandardScaler().fit_transform(X)
labels = DBSCAN(eps= 0.5).fit_predict(X)

plt.figure()
plt.scatter(table['longitude'].values, table['latitude'].values, c = labels)
#plt.savefig('result3.jpg')
plt.legend(['normal','abnormal'])
plt.show()

abnormal = np.argwhere(labels == -1).squeeze()
print('abnormal:' ,abnormal)
table = table.drop(index= abnormal)
table = table.reset_index(drop = True)

table_size2 = table.shape[0]
print('table_size2: %d' %table_size2)
print('Abnormal points number: %d' %(table_size1 - table_size2))
```

附录 B　时间戳处理 –Python 源程序

```
for i in range(len(table)):
    h,m,s = list(map(int, table['time'][i].split(':')))
    table['time'][i] = 3600 * h + 60*m + s

table_save.to_excel('qs1.xlsx', index= False)
```

附录 C　三次样条插值填充数据 –MATLAB 源程序

```
clear;clc;
load DATA/1.mat

Minimum = min(X(:,1));
Maximum = max(X(:,1));
disp(Minimum);
disp(Maximum);
t = Minimum:Maximum;

pp1 = csape(X(:,1), X(:,2));
y1 = fnval(pp1, t);
figure(1);
plot(X(:,1), X(:,2), 'o');
hold on;
plot(t, y1, 'r-','LineWidth',2);
legend('Row data','Interpolation');
xlabel('t');
ylabel('longitude');
title('longitude-t');

pp2 = csape(X(:,1), X(:,3));
y2 = fnval(pp2, t);
figure(2);
plot(X(:,1), X(:,3), 'o');
hold on;
plot(t, y2, 'r-','LineWidth',2);
```

```
legend('Row data','Interpolation');
xlabel('t');
ylabel('latitude');
title('latitude-t');



figure(3);
plot(X(:,2), X(:,3), 'o');
hold on;
plot(y1, y2, 'r-','LineWidth',2);
legend('Row data','Interpolation');
xlabel('longitude');
ylabel('latitude');
title('latitude-longitude');

Y = [t',y1',y2'];
xlswrite('qs1/re1.xlsx', Y);
```

附录 D　Douglas-Peucker 算法 –Python 源程序

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math

def Rad(d):
    return d * math.pi / 180

def Geodist(point1,point2):
    radLat1 = Rad(point1[1])
    radLat2 = Rad(point2[1])
    delta_lon = Rad(point1[0] - point2[0])
    top_1 = math.cos(radLat2) * math.sin(delta_lon)
    top_2 = math.cos(radLat1) * math.sin(radLat2) - \
    math.sin(radLat1) * math.cos(radLat2) * math.cos(delta_lon)
    top = math.sqrt(top_1 * top_1 + top_2 * top_2)
```

```
    bottom = math.sin(radLat1) * math.sin(radLat2) + \
    math.cos(radLat1) * math.cos(radLat2) * math.cos(delta_lon)
    delta_sigma = math.atan2(top, bottom)
    distance = delta_sigma * 6378137.0

    return round(distance,3)

def get_vertical_dist(pointA,pointB,pointX):
    a=math.fabs(Geodist(pointA,pointB))

    #当弦两端重合时点到弦的距离变为点间距离,
    if a==0:
        return math.fabs(Geodist(pointA,pointX))

    b=math.fabs(Geodist(pointA,pointX))
    c=math.fabs(Geodist(pointB,pointX))
    p=(a+b+c)/2
    S=math.sqrt(math.fabs(p*(p-a)*(p-b)*(p-c)))

    vertical_dist=S*2/a

    return vertical_dist

def DP_compress(point_list,output_point_list,Dmax):
    start_index=0
    end_index=len(point_list)-1

    #起止点必定是关键点但是作为递归程序此步引入了冗余数据后期必须去除,,
    output_point_list.append(point_list[start_index])
    output_point_list.append(point_list[end_index])

    if start_index<end_index:
        index=start_index+1     #工作指针遍历除起止点外的所有点,
        max_vertical_dist=0     #路径中离弦最远的距离
        key_point_index=0     #路径中离弦最远的点即划分点,

        while(index<end_index):
```

```
            cur_vertical_dist=get_vertical_dist(
            point_list[start_index],
            point_list[end_index],
            point_list[index]
            )
            if cur_vertical_dist>max_vertical_dist:
                max_vertical_dist=cur_vertical_dist
                key_point_index=index    #记录划分点
            index+=1

        #递归划分路径
        if max_vertical_dist>=Dmax:
            DP_compress(point_list[start_index:key_point_index],output_point_list,Dmax)
            DP_compress(point_list[key_point_index:end_index],output_point_list,Dmax)

def get_MeanErr(point_list,output_point_list):
    Err=0

    start_index=0
    end_index=len(output_point_list)-1

    while(start_index<end_index):   #遍历所有关键点
        #选取两相邻关键点
        pointA_id=int(output_point_list[start_index][2])
        pointB_id=int(output_point_list[start_index+1][2])

        index=pointA_id+1     #工作指针用于遍历非关键点,
        while(index<pointB_id):    #遍历两关键点之间的非关键点
            Err+=get_vertical_dist(
            output_point_list[start_index],
            output_point_list[start_index+1],
            point_list[index]
            )
            index+=1

        start_index+=1
```

```
    return Err/len(point_list)

COLUMNS = [ 'time','longitude','latitude']
data = pd.read_excel('qs1/re1.xlsx', header= None, names= COLUMNS)
ID = pd.Series(data.index, name = 'index')
data = data.drop('time', 1)
data = pd.concat([ID, data], axis = 1)
data.head()

point_list=[]
output_point_list=[]

for i in range(len(data)):
    index = int(data.iloc[i][0])
    longitude = data.iloc[i][1]
    latitude = data.iloc[i][2]
    point_list.append((longitude, latitude, index))

DP_compress(point_list,output_point_list,Dmax=100)
output_point_list=list(set(output_point_list)) #去除递归引入的冗余数据
output_point_list=sorted(output_point_list,key=lambda x:x[2]) #按照排序id

print("compression rate={}/{}={}".format(len(point_list),
len(output_point_list),
1 -len(output_point_list)/len(point_list)))
print("mean error:{}".format(get_MeanErr(point_list,output_point_list)))

uncompressed=[[],[]]
for point in point_list[:]:
    uncompressed[0].append(point[0])
    uncompressed[1].append(point[1])

plt.scatter(uncompressed[0],uncompressed[1])
plt.plot(uncompressed[0], uncompressed[1], 'r')
plt.xlabel("longitude")
plt.ylabel("latitude")
plt.title("uncompressed")
```

```
plt.show()

compressed=[[],[]]
for point in output_point_list:
    compressed[0].append(point[0])
    compressed[1].append(point[1])

plt.scatter(compressed[0],compressed[1])
plt.plot(compressed[0], compressed[1], 'r')
plt.xlabel("longitude")
plt.ylabel("latitude")
plt.title("compressed")
plt.show()

out = np.array(output_point_list)
out = pd.DataFrame(out, columns= ['longitude', 'latitude', 'index'])
out = out.drop('index', 1)
out.head()
```

附录 E　基于熵权法优化的 TOPSIS 算法 –MATLAB 源程序

```
%% Loda data
[r,c] = size(X);

%% Positivation & normalization

ifpositive = input('Do you need Positivzation? (0/1):');

if ifpositive ~= 1 && ifpositive ~=0
    disp('Error,Please examine your input.');
end

if ifpositive == 1

    columns = input('Please enter the columns you want to be positive, like
        [1,2,3]:');
```

```
    types = input('Please enter the type of transformation for the columns you
        have selected:(1:min2max,2:mid2max,inter2max):');
    for i = 1:size(columns,2)
        X(:,columns(i)) = Positivation(X(:,columns(i)) , types(i));
    end
end

%It depends on reality.
columns = input('Please enter the columns without positivation:');

for i = 1 : size(columns,2)
    X(:,columns(i)) = ( X(:,columns(i)) - min(X(:,columns(i))) )
    / ( max(X(:,columns(i))) - min(X(:,columns(i))) );
end

disp('X_positivation: ');
disp(X);


%% Standardization

Z = X ./ repmat(sum(X .* X) .^ 0.5 , r ,1);
disp('Z = X_standard: ');
disp(Z);

%% Set weights vector

weights = get_entropy_weights(Z);
disp('Weights by entropy method:');
disp(weights);

%% Compute each vector's distance to best/worst vector

D_P = sum(((Z - repmat(max(Z) , r , 1)) .^ 2) .* repmat(weights , r , 1) , 2)
    .^ 0.5;
D_N = sum(((Z - repmat(min(Z) , r , 1)) .^ 2) .* repmat(weights , r , 1) , 2)
    .^ 0.5;
```

```
disp('Distance to the best vector:');
disp(D_P);
disp('Distance to the worst vector:');
disp(D_N);

%% Compute scores

scores = D_N ./ (D_P + D_N);
scores_stand = scores ./ sum(scores);
disp('scores_standard:');
disp(scores_stand);

%% sort scores

[scores_sorted , index] = sort(scores_stand , 'descend');
disp('scores_sorted:');
disp(scores_sorted);
disp('rank:')
disp(index);
```

附录 F　大桥流量灰色预测 –MATLAB 源程序

```
n=length(y);
yy=ones(n,1);
yy(1)=y(1);
for i=2:n
    yy(i)=yy(i-1)+y(i);
end
B=ones(n-1,2);
for i=1:(n-1)
    B(i,1)=-(yy(i)+yy(i+1))/2;
    B(i,2)=1;
end
BT=B';
for j=1:n-1
    YN(j)=y(j+1);
```

```
end
YN=YN';
A=inv(BT*B)*BT*YN;
a=A(1);
u=A(2);
t=u/a;
i=1:n+5;
yys(i+1)=(y(1)-t).*exp(-a.*i)+t;
yys(1)=y(1);
for j=n+5:-1:2
    ys(j)=yys(j)-yys(j-1);
end
x=2011:2020;
xs=2021:2025;
yn=ys(n+1:n+5);
disp(yn);
figure(1);
plot(x,y,'o',xs,yn,'*');
xlabel('Year');
ylabel('Research Funding/GDP(%)');
grid('on');
det=0;

sum1=0;
sumpe=0;
for i=1:n
    sumpe=sumpe+y(i);
end
pe=sumpe/n;
for i=1:n
    sum1=sum1+(y(i)-pe).^2;
end
s1=sqrt(sum1/n);
sumce=0;
for i=2:n
    sumce=sumce+(y(i)-yn(i));
end
```

```
ce=sumce/(n-1);
sum2=0;
for i=2:n
    sum2=sum2+(y(i)-yn(i)-ce).^2;
end
s2=sqrt(sum2/(n-1));
c=(s2)/(s1);
disp(['The posterior difference ratio is:',num2str(c)]);
if c<0.35
    disp('Good system prediction accuracy.')
elseif c<0.5
    disp('System prediction accuracy is qualified')
elseif c<0.65
    disp('System prediction accuracy is barely')
else
    disp('Unqualified system prediction accuracy')
end
% disp下个拟合值为([' ',num2str(ys(n+1))]);
% disp再下个拟合值为(['',num2str(ys(n+2))]);
```