


中国研究生创新实践系列大赛
“华为杯”第十八届中国研究生
数学建模竞赛

学　　校	华中科技大学
参赛队号	**No.21104870136**
队员姓名	**1.** 孙晨
	2. 刘川
	3. 曹云康

中国研究生创新实践系列大赛
“华为杯”第十八届中国研究生
数学建模竞赛

题　　目　**基于多层感知机的最小二乘优化定位模型**

摘　　要：

在智能化时代，基于位置的服务逐渐成为人们关注的热点，其中室内定位技术在城市场景中应用更为广泛。基于超频宽带 (Ultra-Wideband,UWB) 的技术通过发送纳秒级的极窄脉冲来传输数据，具有传输功耗低，带宽大，传输能力强等优点，为室内定位技术的研究热点。但在复杂的室内环境下，UWB 的信号由于时间延时、多路径效应等因素不可避免地产生测量误差，因而信号干扰下的 UWB 精确定位成为了亟需解决的问题。

对于任务一，首先分析给定的数据文件格式，提取所需的测量值数据，并按要求转换为二维表格式。之后通过设定判定准则，重复值、缺失值、异常值、相似值等除冗余及无用数据，得到最终所需的数据，其中重复值为不同组之间四个锚点观测距离完全相同的数据值；缺失值为某一锚点观测距离缺失的数据；异常值为 3σ **准则**以外的数据；并依据设定的预测精度（1cm）建立相似度判定准则，通过对数据进行相似度度量及聚类，保留可信度高的数据，删除相似数据。

对于任务二，首先将“锚点定位”问题抽象建模为最小二乘拟合问题。**针对正常数据的定位问题，本文提出了基于多层感知机的最小二乘优化定位模型（LS-MLP）**，将该最小二乘问题建模为一个优化问题，并使用神经网络进行解决。通过设计神经网络的输入、输出、损失函数以及网络结构，实现了初步的位置预测。对已生成的若干预测值，使用 K-means 算法进行聚类和选择，保留一致性较高的结果并进行统计学分析，进一步提高定位精度。为了衡量所提方法的定位有效性和精确性，本文使用了多维度的评价方式，在 x, y 轴上定位的相对误差为 3%,3%；在 X-Y 平面上定位的 RMSE 为 6.95cm；在三维空间中，定位的 RMSE 为 37.04cm。相较于传统数值计算方法，所提方法具有不依赖于初始解、精度高、抗噪声能力强的特点。**针对异常数据的定位问题，通过数据可视化分析，做出合理假设：任一时刻最多只有一个基站的信号受到干扰。基于所提假设，本文在 LS-MLP 模型的基础上，提出了置信度赋权的联合定位模型（CE-CL）。**所提 CE-CL 模型通过模拟不同锚点异常的情况，对异常数据进行切片采样，使用 LS-MLP 模型得到多种位置估计数据。构建基于交叉熵损失的 Softmax 分类

器，训练位置估计数据，得到其置信度水平，并对估计结果及其置信度进行带权重的聚类，以最大簇的均值和方差作为最终预测。将所提 CE-CL 模型在异常数据上进行评估，在 x, y 轴的相对误差分别为 7%,8%，X-Y 平面定位的 RMSE 为 18.24cm, 三维定位的 RMSE 为 46.21cm，在信号传输受到干扰的情况下仍能保持一定的精度。

对于任务三，首先分析了所提 LS-MLP 模型的内在特性：本质为最小二乘拟合优化器，通过将实验场景信息编码到损失函数中，设计可微的损失函数来指导网络进行优化，实现了网络结构与实验场景信息的剥离。因此，在解决场景 2 下的定位问题时，将更新的场景信息重编码到模型中，使用更新后的损失函数对 LS-MLP 进行训练完成任务场景的迁移。由于每个对应靶点的测量数据仅包含一组，为此无法使用任务二所提的聚类方法，因此在数据异常的情况下，提出了**加权聚合方法**，对不同抽样下得到的预测值，使用置信度进行加权平均，得到最终结果。

对于任务四，首先确定了任务实质为有监督分类问题。为提升分类的性能，本文采用了**特征工程、优化器选择、自动机器学习、投票法**等方法。在特征工程阶段，利用任务二构建的 LS-MLP 模型，构建**观测距离、预测距离、两距离平方差等共计 12 个维度的特征**，极大地挖掘了距离信息的内在表征。选择极限梯度提升算法 (XGBoost) 作为分类器，**并使用基于贝叶斯的自动机器学习技术优化分类器的关键参数**。在模型预测阶段，利用集成学习的思想，基于多模型的输出，进行投票获得最终结果。通过以上方法，分类器在测试集精度可达 99.26%。任务四所给的测试数据，对应的分类结果为：正常、异常、正常、异常、异常、正常、异常、异常、异常、正常。

对于任务五，首先分析任务二所提正常数据定位模型 $LS-MLP$ 与异常数据定位模型 $CE-CL$ 的关联，对两者进行归纳统一，提出了可同时处理正常数据、异常数据的**统一置信度赋权定位框架**，并基于该框架得到了靶点的初步运动轨迹。由于任务五中的靶点为运动靶点，故用卡尔曼滤波，考虑靶点的运动因素，结合初步预测结果对动态靶点的运动轨迹进行联合优化。滤波后结果波动较小，轨迹较为平滑。

最后，本文对所提模型的优缺点进行评价，并提出改进建议，介绍了模型的推广前景。

关键字：最小二乘　多层感知机　聚类　模型聚合　卡尔曼滤波

目录

1. 问题重述

1.1 问题背景

伴随着城市的建设和发展，基于位置的服务逐渐成为人们关注的热点。在智能化的时代，位置服务与日常生活密切相关，如车辆导航、人员定位等，这些功能为人们的生产生活提供了便利，也推动了定位技术的不断发展。

定位技术主要分为室外定位和室内定位两种。在室外定位中，卫星定位导航系统 (Global Navigation Satellite Syste, GNSS)，如美国的 GPS、中国的北斗、欧盟的 Galileo 等，具备在全球范围内提供三维精确定位、授时及测速的能力，能满足大部分室外定位需求，在军事、交通、测绘等领域获得了广泛应用。但在城市场景中，大量的建筑物遮挡和复杂的电磁环境会对卫星信号产生严重干扰，导致定位精度下降或者失效。因此，对于室内定位技术的需求不断增长。

目前，已经出现了多种针对室内定位的技术，比如红外线定位、超声波、蓝牙定位、WIFI 定位、射频识别定位（Radio Frequency Identification，RFID）等，但都无法满足室内定位精度高和环境适应性好的要求。而基于超频宽带 (Ultra-Wideband, UWB) 的技术通过发送纳秒级的极窄脉冲来传输数据，拥有 GHz 量级的带宽，具有传输功耗低（仅有几十 μW），穿透力强，定位精度高（可达厘米级甚至毫米级）等特点，成为室内定位技术的研究热点。

超宽带（UWB）定位的测距方法包括：到达角方法（Angle of Arrival，AOA），接收信号强度方法（Received Signal Strength，RSS），基于飞行时间的方法（Time of Flight，TOF）等。其中，基于飞行时间的方法是最常见的定位测距方法之一。该方法主要根据信号在定位目标和基站之间的传播时间，计算定位目标和基站之间的距离。依据定位目标和多个基站之间的距离，实现定位。

但在复杂室内环境下，UWB 信号在传播过程中存在遮挡、反射等情况，产生时间延时、非视距传播、多路径效应等问题，导致距离测量值不可避免地出现波动，影响定位算法性能，甚至无法完成定位。因此，研究信号干扰下的 UWB 精确定位成为了亟需解决的问题。

1.2 问题要求

本研究通过实际场景实测来解决信号干扰下的超宽带精确定位问题。在 $5000mm \times 5000mm \times 3000mm$ 的测试环境中，放置锚点 A0，A1，A2，A3 在 4 个角落，向外发送信号。靶点接收到 4 个锚点的信号后，利用 TOF 定位技术，获得锚点（anchor）与靶点（tag）之间的距离，建立数学模型或者算法，使得无论测距信号是否受到干扰，都可以给出目标靶点的精确定位。问题示意图如图1-1。

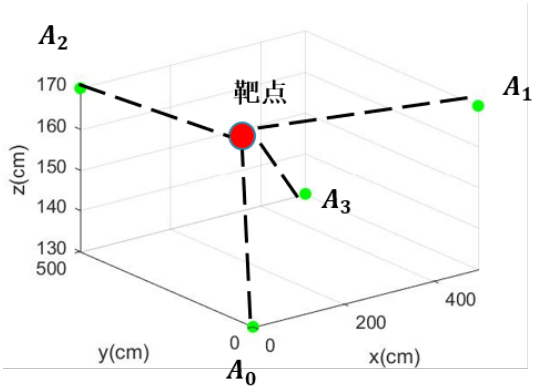


图 1-1　问题示意图

实验采集了 324 个不同位置靶点的数据，每个位置个采集两次，一次在无信号干扰的情况下，一次在有信号干扰下的情况下。由于靶点在同一位置会稍作停留，而锚点与靶点之间每间隔 0.2-0.3 秒就会收发信号一次，因而在同一位置点，UWB 会采集到多组数据，得到 UWB 数据集。现根据以上背景以及所提供数据完成以下任务：

1. 数据预处理（清洗）。针对靶点在信号正常和异常采集的数据，设计读取方式并转换成二维表（矩阵）的形式，每一行代表一组数据。对数据文件进行预处理，删掉异常、缺失、相同或相似等“无用”数据，保留经处理后的数据，重点列出 24. 正常.txt，109. 正常.txt，1. 异常.txt，100. 异常.txt 四个数据文件保留下来的数据。
2. 定位模型。利用处理后的数据，分别对“正常”和“异常”数据，设计合适的数学模型，估计靶点的精确位置，所建立的定位模型定位模型必须体现实验场景信息，并说明有效性。同时利用所建立的定位模型分别对 5 组无信号干扰数据和 5 组有信号干扰数据进行精确定位，并给出定位模型的 3 维（x，y，z）精度、2 维（x，y）精度以及 1 维的各自精度。
3. 在不同场景下测试所建立模型的有效性。分别用任务 2 中建立的“正常数据”定位模型及“异常数据”定位模型，对实验场景 2 下的无信号干扰数据及有干扰信号数据进行精确定位。
4. 分类模型。建立数学模型，对“正常”和“异常”数据进行区分，以判断在 UWB 数据采集阶段是否受到信号干扰，并说明所建立的分类模型的有效性。同时利用所建立的分类模型判断场景 1 下的 10 组数据是否为信号干扰数据。
5. 运动轨迹定位。利用静态点的定位模型，加上靶点自身运动规律，建立运动轨迹定位模型，对存在随机信号干扰的动态靶点的运动轨迹进行定位。对场景 1 下采集的动态靶点运动轨迹进行精确定位，并画出运动轨迹图。

2. 模型假设与符号说明

2.1 模型假设

假设 1：假设任意时刻至多一个基站出现异常；
假设 2：假设出现的异常只导致时延误差，即会使得测量值增大；
假设 3：假设任务五中数据的采集频率固定，即相邻组之间的采集间隔固定。

2.2 符号说明

表 1　符号说明

符号	意义
S^1, S^2, S^3, S^4	4 个锚点在某一位置下的时序信号
$\mathbf{A^j} \in \mathbb{R}^{N_j \times 3}, j = 1, 2, \cdots, 324$	4 个锚点的坐标矩阵
$\mathbf{T^j} \in \mathbb{R}^{N_j \times 3}, j = 1, 2, \cdots, 324$	靶点的坐标矩阵
$\mathbf{S^j} \in \mathbb{R}^{N_j \times 1}, j = 1, 2, \cdots, 324$	原始观测距离矩阵
$\mathbf{E^j} \in \mathbb{R}^{N_j \times 1}, j = 1, 2, \cdots, 324$	残差矩阵
$\mathbf{D} \in \mathbb{R}^{N \times 4}$	成组后的实际观测距离矩阵
$\mathbf{D'} \in \mathbb{R}^{N \times 4}$	N 个预测的靶点位置与 4 个锚点间的距离矩阵
$\mathbf{T} \in \mathbb{R}^{N \times 4}$	神经网络的输出，即 N 个靶点的位置估计值
$\mathbf{D_0} \in \mathbb{R}^{N \times 4}$	实际观测距离矩阵 **D** 的抽样矩阵
$\mathbf{D_1} \in \mathbb{R}^{N \times 4}$	实际观测距离矩阵 **D** 的抽样矩阵

$\mathbf{D_2} \in \mathbb{R}^{N\times 4}$	实际观测距离矩阵 **D** 的抽样矩阵
$\mathbf{D_3} \in \mathbb{R}^{N\times 4}$	实际观测距离矩阵 **D** 的抽样矩阵
$\mathbf{D_4} \in \mathbb{R}^{N\times 4}$	实际观测距离矩阵 **D** 的抽样矩阵
$\mathbf{T_0} \in \mathbb{R}^{N\times 4}$	输入为 $\mathbf{D_0}$ 时的神经网络输出
$\mathbf{T_1} \in \mathbb{R}^{N\times 4}$	输入为 $\mathbf{D_1}$ 时的神经网络输出
$\mathbf{T_2} \in \mathbb{R}^{N\times 4}$	输入为 $\mathbf{D_2}$ 时的神经网络输出
$\mathbf{T_3} \in \mathbb{R}^{N\times 4}$	输入为 $\mathbf{D_3}$ 时的神经网络输出
$\mathbf{T_4} \in \mathbb{R}^{N\times 4}$	输入为 $\mathbf{D_4}$ 时的神经网络输出
$\mathbf{A_1} \in \mathbb{R}^{N\times 3}$	锚点 1 的坐标矩阵
$\mathbf{A_2} \in \mathbb{R}^{N\times 3}$	锚点 2 的坐标矩阵
$\mathbf{A_3} \in \mathbb{R}^{N\times 3}$	锚点 3 的坐标矩阵
$\mathbf{A_4} \in \mathbb{R}^{N\times 3}$	锚点 4 的坐标矩阵

3. 任务一的分析与求解

3.1 任务一分析

任务一中的数据来自于实验场景 1，采自 324 个不同的靶点位置，其中每个位置都采集了两个数据文件（1 个正常，1 个异常），共 648 个文件，每个文件记录了靶点在某个特定位置下的连续时间内 UWB 采集到的多组时序数据，每 4 行为一组。一行数据的格式如下所示（每个数据之间用“:”分隔）：

靶点标识：时间戳：Range Report 的缩写：靶点 ID：锚点 ID：该锚点的测距值 (mm)：测距值的校验值：数据序列号：数据编号

UWB 采集的一组完整数据如下所示：

T:144235622:RR:0:0:950:950:118:1910

T:144235622:RR:0:1:2630:2630:118:1910

T:144235622:RR:0:2:5120:5120:118:1910

T:144235622:RR:0:3:5770:5770:118:1910

对数据进行建模分析之前，首先需要将每个文件中的数值抓取出来。由于文件中给出的数据是统一格式，因而直接利用编程工具（python、matlab 等）读.txt 文件，每四行为一组对其进行处理，将其转换为二维表的形式即可。该二维表的形式如表2：

表 2　数据抓取二维表

index	d1	d2	d3	d4
0	950	2630	5120	5770
⋮	⋮	⋮	⋮	⋮

得到原始数据后，需要对数据进行预处理，清洗“无用”数据。由于每个数据文件为靶点在同一位置下采集的多组数据，除了噪声，还包含较多的冗余数据。因此，本文主要考虑缺失数据、异常数据、相同数据、相似数据四种情况，对其分别制定处理方式。任务一的解决流程如图3-1所示。

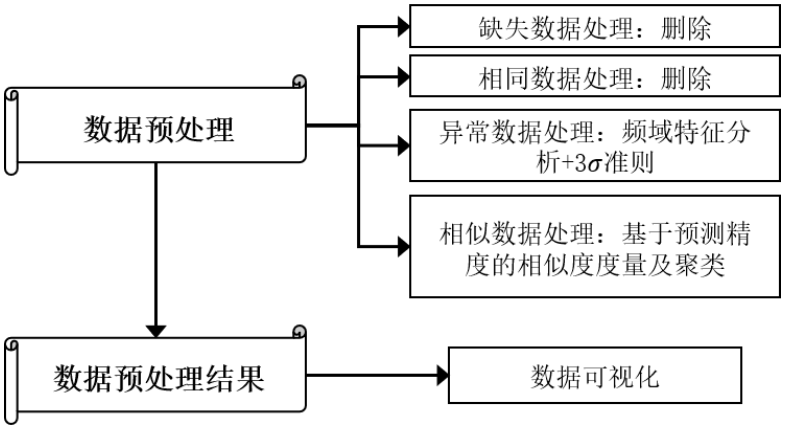


图 3-1　任务一流程图

3.2 数据预处理

3.2.1 缺失数据、相同数据处理

(1) 缺失数据判定准则：原始数据中，存在一个或多个锚点值为空的情况即判定为缺失数据。

(2) 相同数据判定准则：原始数据中，两组数据间 4 个锚点值全部相同的数据即判定为相同数据。

由于每个数据文件采集到的是同一位置下的多组冗余信息，因而对缺失数据及相同数据直接进行删除。

3.2.2 异常数据处理

由于每个数据文件为靶点在同一位置下采集到的一段连续时间内的多组数据，因而本文将每个锚点（锚点）采集到的数据看成一维时序信号，对信号进行分析。

(1) 快速傅里叶变换分析信号特征

傅里叶变换（Fourier Transform，FT）是一种线性积分变换，用于信号在时域（或空域）和频域之间的转换，是信号处理中的一种常用工具。而快速傅里叶变换（Fast Fourier Transform，FFT）是一种可在时间内完成离散傅里叶变换的高效的计算方法。

首先，本文将同一数据文件下的 4 个锚点数据进行可视化，得到 4 个一维的时序信号，如下图3-2所示（以 142. 正常.txt 为例）。

可以看出信号在时域下，整体较为平稳，有轻微的波动，偶有异常值出现。接着，本文对时域下的信号进行快速傅里叶变换，分析其在频域下的特征，判断是否有周期性的噪声。对上图四个一维信号分别进行傅里叶变换得到频域下的特征，如图3-3所示。

可以看到信号频率集中在 0，没有高频噪声，因而不需要滤波。

(2) 3σ 准则剔除异常数据

本文利用 3σ 准则对 4 个锚点的信号（记为 S^1, S^2, S^3, S^4）分别进行异常值的筛除。设 $\left|S^1\right| = \left|S^2\right| = \left|S^3\right| = \left|S^4\right| = N$，则：

$$\mu_i = \frac{1}{N}\sum_{n=1}^{N} S_n^i \tag{1}$$

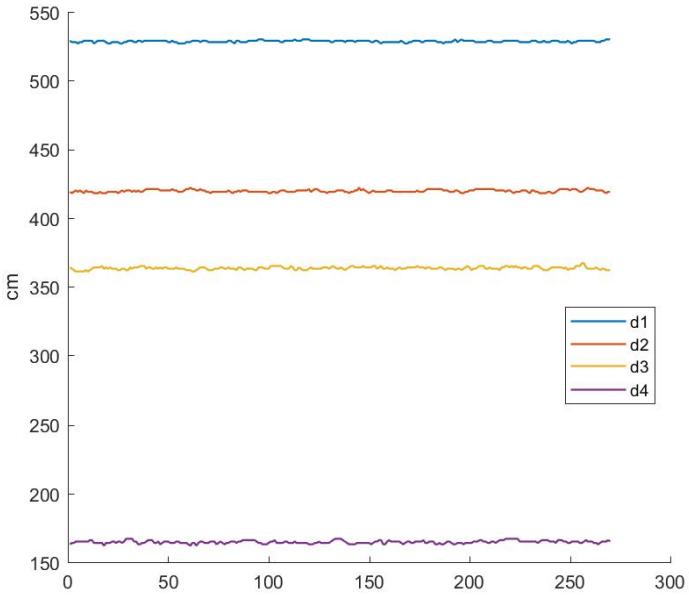


图 3-2　信号可视化图

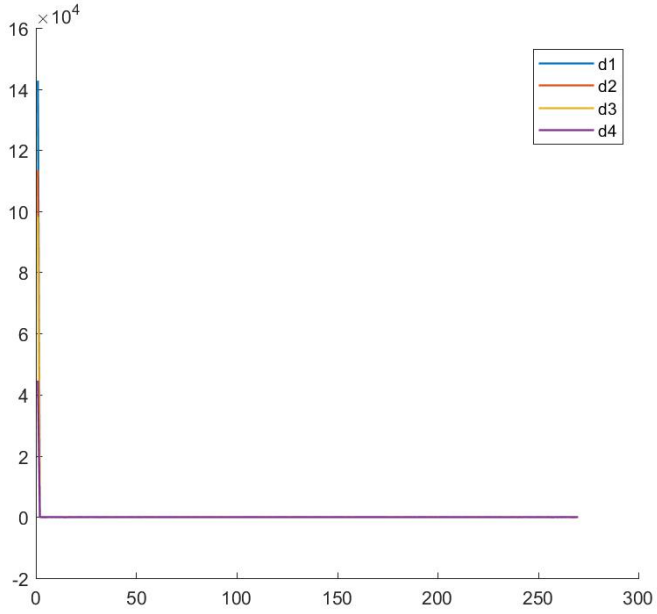


图 3-3　信号傅里叶变换图

$$\sigma_i = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(S_n^i - \mu_i)^2} \tag{2}$$

根据 3σ 准则，删除 3σ 以外的数据，记剔除异常数据后的信号为 C^1, C^2, C^3, C^4：

$$C^i \leftarrow \left\{ S_n^i \middle| \quad \mu_i - 3\sigma \le S_n^i \le \mu_i + 3\sigma \right\}, \quad i = 1, 2, 3, 4. \tag{3}$$

3.2.3 相似数据处理

经过上述处理后，每个靶点位置下仍保留有较多相似的冗余数据，因此本文依据测量精度制定相似数据判定准则，删除相似数据。记上述处理后的数据为 $\mathbf{X}=[\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_{N'}]^T$，$\mathbf{X}\in\mathbb{R}^{N\times 4}$。其中，$\mathbf{x}_i=[x_{i1},x_{i2},x_{i3},x_{i4}]$，$\mathbf{x}_i\in\mathbb{R}^4$，单位为 mm。设定靶点每个维度（一共有 x、y、z 三个维度）的定位精度为 1cm，即 10mm，则相似数据定义为：

$$|x_{ic}-x_{jc}|<M,\quad i\neq j,\quad i,j\in\{1,2,3,\cdots,N'\},\quad c=1,2,3,4. \tag{4}$$

其中 $M=10mm$。即对 $\forall i,j\in 1,2,3,\cdots,N'$，若 $|x_{i1}-x_{j1}|<10mm$ 且 $|x_{i2}-x_{j2}|<10mm$ 且 $|x_{i3}-x_{j3}|<10mm$ 且 $|x_{i4}-x_{j4}|<10mm$，则认为 $\mathbf{x}_i$ 与 $\mathbf{x}_j$ 为相似的数据样本。对 $\mathbf{X}$ 中的数据样本进行两两遍历，得到多个相似的数据团，**每团只保留一个与均值距离最近的样本**。

3.3 数据预处理结果

根据上述处理流程，分别对“正常数据”与“异常数据”下的 324 个文件进行处理，得到“正常数据”和“异常数据”文件夹下每个数据文件最后保留下来的样品组数。“正常数据”与“异常数据”文件夹下每个数据文件的数据组数如表 3、表 4 所示，详细结果见附件。

表 3　“正常数据”文件夹下每个数据文件的数据组数

文件编号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数据组数	15	19	19	39	41	33	46	36	19	22	39	30	21	16	20
文件编号	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
数据组数	24	24	36	25	23	27	12	20	14	22	24	21	25	17	16
文件编号	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
数据组数	23	33	18	11	22	38	35	31	17	42	14	37	30	37	29
文件编号	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

表 4　“异常数据”文件夹下每个数据文件的数据组数

文件编号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数据组数	52	54	53	89	89	88	101	100	62	73	87	76	56	44	49
文件编号	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
数据组数	71	80	79	65	69	59	37	50	51	64	68	62	64	47	46
文件编号	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
数据组数	55	85	61	41	68	97	75	70	50	87	43	70	75	82	72
文件编号	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

其中，“24. 正常.txt”文件经处理后保留的数据为：

$$\begin{bmatrix} 3280 & 4660 & 2600 & 3910 \\ 3280 & 4670 & 2610 & 3920 \\ 3270 & 4660 & 2590 & 3880 \\ 3280 & 4650 & 2590 & 3900 \\ 3290 & 4660 & 2590 & 3920 \\ 3270 & 4680 & 2590 & 3910 \\ 3270 & 4660 & 2590 & 3940 \\ 3280 & 4660 & 2600 & 3890 \\ 3280 & 4670 & 2590 & 3900 \\ 3270 & 4660 & 2590 & 3920 \\ 3290 & 4650 & 2610 & 3900 \\ 3290 & 4650 & 2610 & 3920 \\ 3280 & 4670 & 2610 & 3900 \\ 3270 & 4660 & 2610 & 3880 \end{bmatrix}$$

其余“109. 正常.txt”文件、“1. 异常.txt”文件、“100. 异常.txt”文件经处理后保留的数据详见附录 A。

4. 任务二的分析与求解

4.1 任务二分析

在本题中，需要利用任务 1 处理后的数据，分别对“正常数据”和“异常数据”，设计合适的数学模型（或算法），预测出靶点的精确位置。模型要求能够体现实验场景的信息，并对其有效性进行评估。对于正常数据而言，任务本质为通过四个锚点到靶点之间的带噪声的观测距离，预测靶点在锚点坐标系中所处的三维空间坐标，为经典的超定方程组优化问题（通过四个方程组解三个未知量）。针对这类问题，最小二乘法及相关特定方法 Chan 为经典解决方案，但这类解决方案通常优化难度较大，且当噪声为非高斯分布时，预测误差较大。本文提出了一种基于多层感知机的定位模型，将该最小二乘拟合问题建模为一个优化问题，并巧妙地用神经网络进行解决。模型与传统方法相比，具有较好的抗噪能力。并通过后接聚类算法，对预测结果进行进一步的聚类，选择较好的一类作为最终结果，因而模型具有较高的精度。

针对异常数据，建立了基站异常的联合定位模型。首先，对观测数据进行可视化，并根据题中陈述，合理进行假设——任一时刻最多只有一个基站的信号受到干扰。基于本假设，对观测到的异常数据进行采样，得到五个数据集合。用采样后的得到的五个数据集合，分别输入上述正常数据下的定位模型，得到五个相应的靶点预测集合。进而建立置信度评估模型，对预测结果进行置信度评估。最后，设计基于置信度的结果聚类算法，对预测结果进行聚类，选择较好的一类作为最终的结果。

为了验证方法的有效性，使用 RMSE、相对误差、绝对误差等多种精度指标，分别在三维空间、XY 平面和 3 个坐标轴方向衡量了模型的定位精度，证明了所提方法在信号传输有干扰的情况下，仍能保持一定的定位精度。任务二的解决流程如图4-1所示。

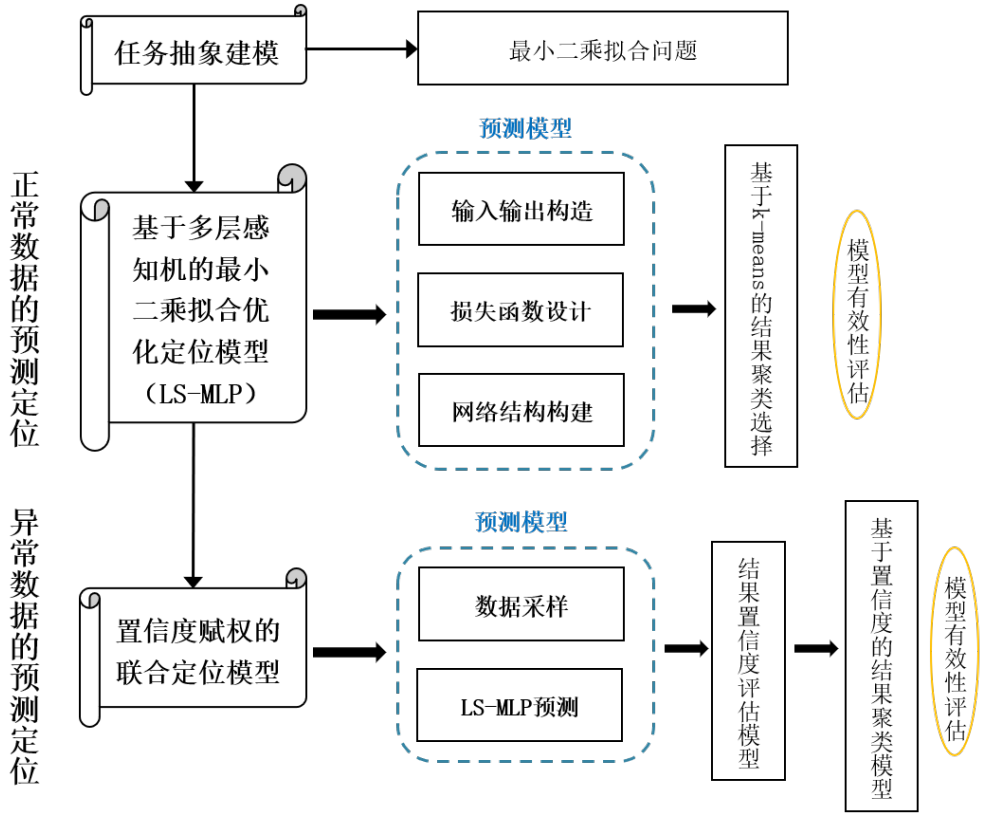


图 4-1　任务二流程图

4.2 任务抽象

已知实测环境 1 靶点范围为 $500cm \times 500cm \times 300cm$，四个基站的位置坐标（单位：cm）分别为：A0（0，0，130）、A1（500，0，170）、A2（0，500，170）、A3（500，500，130）。在该场景下进行基于 UWB 的三维定位，可建立定位模型如下。

以某一个靶点位置下的一组观测值定位为例，设已知基站位置为 (x_i, y_i, z_i)，$i = 1, 2, 3, 4$，以及各个基站距离靶点的空间距离 d_i，$i = 1, 2, 3, 4$。需要求的靶点位置为 (x, y, z)，则可以建立距离观测方程为：

$$\begin{cases} d_1 = \sqrt{(x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2} \\ d_2 = \sqrt{(x_2 - x)^2 + (y_2 - y)^2 + (z_2 - z)^2} \\ d_3 = \sqrt{(x_3 - x)^2 + (y_3 - y)^2 + (z_3 - z)^2} \\ d_4 = \sqrt{(x_4 - x)^2 + (y_4 - y)^2 + (z_4 - z)^2} \end{cases} \tag{5}$$

问题即转化为为求解该超定方程组的问题。该方程组的理论解为所有以基站为球心，以相应基站空间距离为半径的球的交点。但实际测量中，由于每个距离的观测值都不可避免会受到影响，产生一定的误差，进而导致四个球体的空间交点不唯一，如下图4-2所示（以二维空间为例）。

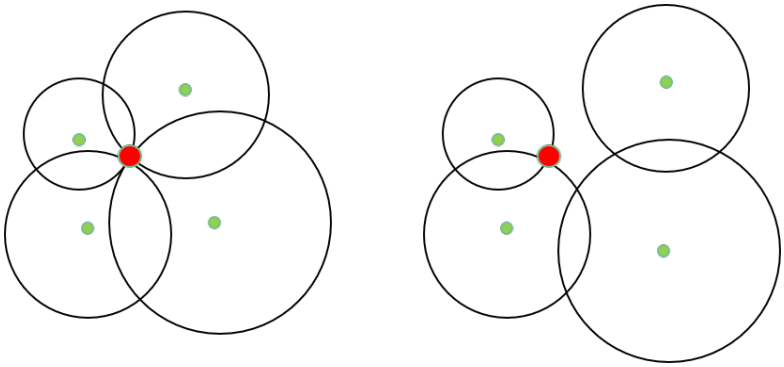


图 4-2　准确定位与不准确定位的二维示意图。左图为无噪声准确定位时，若干观测数据可汇聚于一点。右图为观测距离带噪声时，若干观测数据无统一交点，需要通过优化方法进行定位。

因而，依据上述理论定位模型，建立观测定位模型，并用参数估计的方法对靶点的位置进行估计。观测定位模型为：

$$\begin{cases} s_1 = d_1 + e_1 \\ s_2 = d_2 + e_2 \\ s_3 = d_3 + e_3 \\ s_4 = d_4 + e_4 \end{cases} \tag{6}$$

其中 $s_i(i=1,2,3,4)$ 表示靶点与 4 个基站间的观测距离，$e_i(i=1,2,3,4)$ 表示观测误差。

上述模型为一组观测值下的观测定位模型，由题可知，每个靶点位置下有多组观测值，将同一靶点位置下的多组超定方程组整理为矩阵的形式：

$$\mathbf{S^j} = diag\left(\sqrt{(\mathbf{A^j}-\mathbf{T^j})(\mathbf{A^j}-\mathbf{T^j})^T}\right) + \mathbf{E^j}, \quad j = 1,2,\cdots,324. \tag{7}$$

其中，矩阵 $\mathbf{A^j} \in \mathbb{R}^{N_j \times 3}$ 表示 4 个锚点的坐标矩阵，该矩阵由 4 个锚点的坐标矩阵 $[[x_1,y_1,z_1],[x_2,y_2,z_2],[x_3,y_3,z_3],[x_4,y_4,z_4]]$ 纵向堆叠而来。矩阵 $\mathbf{T^j} \in \mathbb{R}^{N_j \times 3}$ 表示靶点的坐标矩阵，矩阵每个行向量都相等，由向量 $[x,y,z]$ 纵向堆叠而来。$diag$ 操作表示取矩阵对角线元素，将其变为一个一维的列向量。矩阵 $\mathbf{S^j} \in \mathbb{R}^{N_j \times 1}$ 表示观测距离矩阵，矩阵 $\mathbf{E^j} \in \mathbb{R}^{N_j \times 1}$ 表示残差矩阵。N_j 表示第 j 个靶点位置下的观测数据量，$j = 1,2,3,\cdots,324$。

因而，任务即归结为对上述实测环境 1 下测量的 324 个靶点位置，分别针对正常情况与异常情况建立的定位模型，给出正常情况与异常情况下 324 个靶点的精确位置坐标，并说明所建立模型的有效性。

4.3 正常数据的预测定位

4.3.1 基于多层感知机的最小二乘拟合优化定位模型

神经网络（Neural Networks）是计算智能领域发展迅猛的一门理论与技术，其本质为学习一个从输入空间到目标空间的非线性映射，具有较强的非线性拟合能力。其无需知道准确的系统模型，便可自动学习输入输出之间的关系，常用与难以精确建模的系统。目前神经网络已经在分类、预测、回归任务中取得了较为瞩目的结果，近年来，越来越多的研究者尝试将神经网络用于优化任务，通过建立合适的目标函数，利用神经网络对其进行优化。

神经网络进行优化的核心为梯度下降与反向传播技术。在前向传播时保留计算图，求出梯度，在反向传播时利用梯度对网络参数进行优化，在解空间内进

行寻优，使目标函数不断降低。因此，本文建立基于多层感知机的最小二乘拟合优化定位模型（简称为 LS-MLP），通过制定合适的目标函数，利用神经网络对其进行优化，进而得到观测定位模型。下面从输入输出构造、目标函数设计、网络结构构建三个方面阐述本文的定位模型。

- 输入输出构造

本文通过四个锚点的观测距离，利用神经网络，得到对应的靶点坐标。由上可知，同一靶点位置下的观测距离矩阵 $\mathbf{S_j} \in \mathbb{R}^{N_j \times 1}$ 每四行对应一组锚点的观测距离，因而本文每四行为一组将其变为距离矩阵 $\mathbf{D_j} \in \mathbb{R}^{\frac{N_j}{4} \times 4}$。由于在该实验场景下，观测了 324 个不同的靶点位置，因此可以得到 324 个距离矩阵 $\mathbf{D_j}, j = 1, 2, 3, \cdots, 324$。将这 324 个矩阵纵向堆叠，得到模型的输入 $\mathbf{D} \in \mathbb{R}^{N \times 4}$，其中 $N = \sum_{j=1}^{324} N_j$。

本文神经网络通过 4 个观测距离，拟合出一个靶点的坐标。因而模型的输出为 $\mathbf{T}^{N \times 3}$，其中 $N = \sum_{j=1}^{324} N_j$。

- 损失函数设计

损失函数（loss function）是神经网络优化的关键，要想通过神经网络对目标函数进行优化，目标函数需要满足：（1）目标函数包含待优化的未知量。在本题中，待优化的未知量即为模型的输出 **T**。（2）目标函数关于待优化的未知量可微。即可利用反向传播回传梯度，改进网络参数，进而优化模型输出。

基于上述原则，本文设计出本模型的损失函数为：

$$\min \left(\mathbf{D}' - \mathbf{D}\right)^2$$

$$\mathbf{D}'^T = \begin{bmatrix} diag\left(\sqrt{(\mathbf{T}-\mathbf{A_1})(\mathbf{T}-\mathbf{A_1})^T}\right)^T \\ diag\left(\sqrt{(\mathbf{T}-\mathbf{A_2})(\mathbf{T}-\mathbf{A_2})^T}\right)^T \\ diag\left(\sqrt{(\mathbf{T}-\mathbf{A_3})(\mathbf{T}-\mathbf{A_3})^T}\right)^T \\ diag\left(\sqrt{(\mathbf{T}-\mathbf{A_4})(\mathbf{T}-\mathbf{A_4})^T}\right)^T \end{bmatrix} \tag{8}$$

其中，**T** 为神经网络的输出，即为 N 个靶点位置估计值。$\mathbf{A_1}, \mathbf{A_2}, \mathbf{A_3}, \mathbf{A_4} \in \mathbb{R}^{N \times 3}$ 为 4 个锚点的坐标进行广播后得到的坐标矩阵。$diag$ 操作表示取矩阵的对角线元素，将其组成一个列向量。$\mathbf{D}' \in \mathbb{R}^{N \times 4}$ 为 N 个估计的靶点位置与 4 个锚点间的距离矩阵，其由四个列向量纵向拼接得到。本文约束网络估计的距离矩阵 $\mathbf{D}'$ 与实际观测的距离矩阵 **D** 尽可能接近，即最小化上述观测定位模型的残差向量，得到观测定位模型的最小二乘解。

- 网络结构构建

神经网络根据连接方式的不同，可以分为：前馈网络（无反馈的前向神经网络）和反馈网络（相互连接型网络）两大类。前馈神经网络包含输入层、若干隐藏层和输出层，每一层的神经元只接收前一层神经元的输出。除了输入层，隐藏层和输出层的神经元都承担一定的运算，被称为计算节点。反馈神经网络是指拓扑结构中有反馈回路的神经网络，输入信号在神经元之间反复往返传递，逐渐趋于某一稳定状态或进入周期震荡。

多层感知机（Multilayer perception，MLP）是一种前馈人工神经网络，其层与层之间是全连接的。最底层是输入层，中间是隐藏层，最后是输出层，最简

单的 MLP 只包含一个隐层，即三层结构。本文即选用最简单的多层感知机模型，网络结构如图4-3所示。

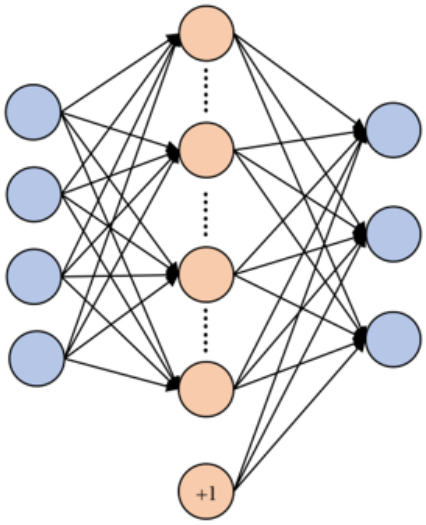


图 4-3　网络结构图

如图4-3所示，输入层神经元个数为 4，隐藏层神经元个数为 32，输出层神经元个数为 3。隐藏层和输出层的激活函数都选用 GeLU 函数，即高斯误差线性单元激活函数，函数形式为：

$$\mathrm{GeLU}(\mathrm{x}) = 0.5\mathrm{x}\left(1 + \tanh\left(\sqrt{2/\pi}(\mathrm{x} = 0.044715\mathrm{x}^3)\right)\right). \tag{9}$$

函数图形如图4-4所示。

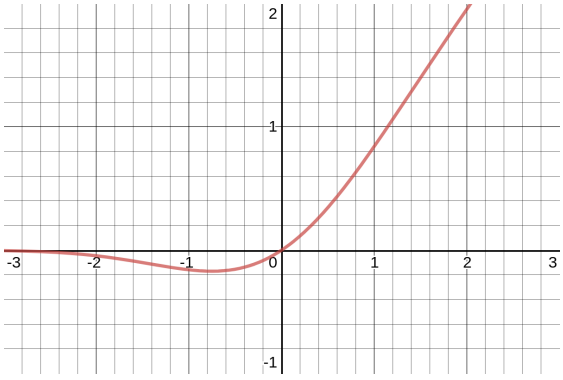


图 4-4　GeLU 激活函数

综上所述，网络可表示为：

$$\mathbf{H} = \mathrm{GeLU}\left(\mathrm{MLP}\left(\mathbf{D}\right)\right) = \sigma\left(\mathbf{D}\theta^1\right). \tag{10}$$

$$\mathbf{T} = \mathrm{GeLU}\left(\mathrm{MLP}\left(\mathbf{H}\right)\right) = \sigma\left(\mathbf{H}\theta^2\right). \tag{11}$$

得到了神经网络架构与相应的损失函数之后，神经网络需要通过一定的学习算法才能对损失函数进行优化，常见的学习算法有：Hebb 学习算法、竞争式学习算法、梯度下降法等。本文选用梯度下降法，其基本思想为沿着损失函数的

负梯度方向不算修正网络参数，直至损失函数达到最小。在此，本文通过选用优化器来实现梯度的自动反向传播。常见的优化器有：SGD、Momentum、ASGD、Rprop、RMSprop、Adam、Adamax、AdaGrad、AdaDelta、AdamW 等。通过实验进行优化器的选择，在本任务中，不同优化器的效果如图4-5所示。从图中可以看出，Rprop 能够取得较快的收敛速度，且寻优效果好（在 20 个 epoch 内，取得了最小的 loss）。因而本文选择 Rprop 优化器进行梯度的自动计算及反向传播。

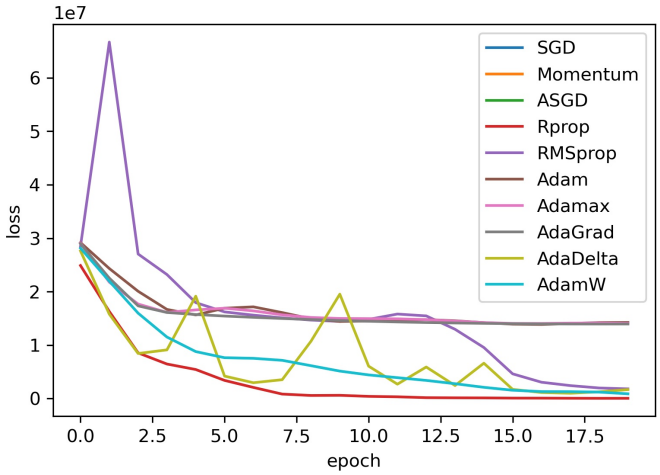


图 4-5　优化器的效果图

4.3.2 结果的聚类选择

通过上述观测定位模型（LS-MLP），本文可以得到同一靶点位置下，多个观测值所对应的预测值。虽然任务 1 已经对数据进行了预处理，但观测值难免会存在随机误差，进而导致得到的多个预测值之间的差距。因而需要从中选取质量较高的预测值作为最终的靶点位置。

在此，本文选择经典的 k-means 方法进行聚类。本文假设正常数据中准确的观测值占大多数，存在随机误差的观测值占少数。因而，本文将每个靶点位置的多个预测值聚成两类，选择数量多的一类的聚类中心为预测的靶点的位置。

k-means 是一种典型的无监督聚类算法，也是一种常用的去除离群点的方法。其依据样本之间的相似性，将样本划分为指定的 k 个类别。其中相似性度量的方法有：欧式距离法，余弦相似度法，曼哈顿距离法等。由于本文距离都为欧式距离，因此基于欧氏距离进行相似度度量，(x_i, y_i, z_i) 与 (x_j, y_j, z_j) 之间的相似度定义如下：

$$s_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{12}$$

聚类算法流程如下：

Step1：设定聚类个数为 2。

Step2：随机选择两个样本作为聚类中心。

Step3：对每个样本点，根据其与两个聚类中心的相似度，确定所属类别。

Step4：依据每个类中的样本值更新两个聚类中心，即 $\left(\frac{\sum_{i=1}^{n_1} x_i}{n_1}, \frac{\sum_{i=1}^{n_1} y_i}{n_1}, \frac{\sum_{i=1}^{n_1} z_i}{n_1}\right)$，$\left(\frac{\sum_{i=1}^{n_2} x_i}{n_2}, \frac{\sum_{i=1}^{n_2} y_i}{n_2}, \frac{\sum_{i=1}^{n_2} z_i}{n_2}\right)$。

Step5：判定是否满足收敛条件。若满足，则输出聚类结果；反之，则转 Step3 继续进行迭代。

4.3.3 结果分析及可视化

首先对模型聚类前后的结果进行可视化，如图4-6所示。可以看到预测结果在空间中较为明显地聚成两类，类内较为集中，类间差异较大，聚类帮助模型选择数据量较多的蓝色类别作为模型最终的预测结果，有助于进一步提高模型预测准确度。

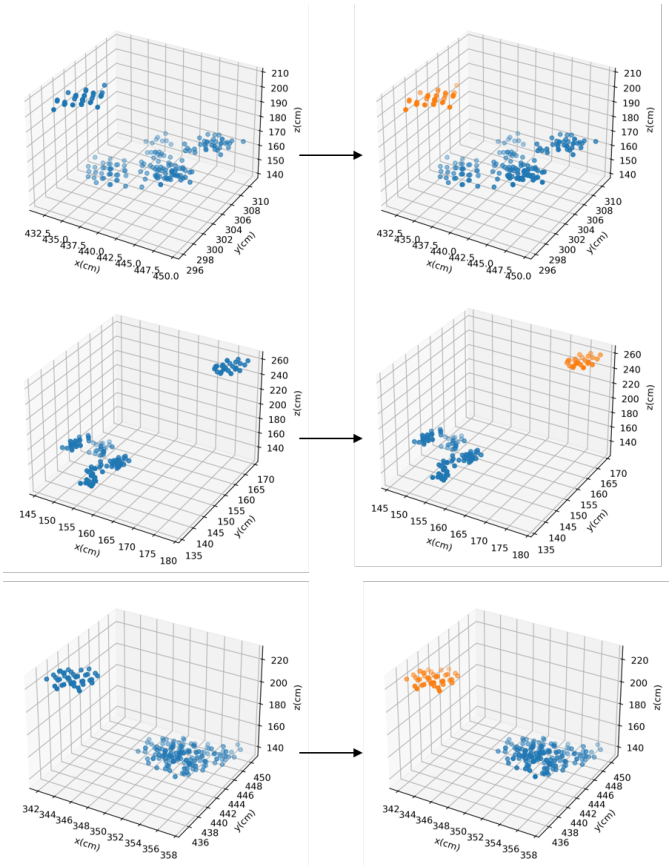


图 4-6　聚类前后结果对比图。左列为聚类前的结果可视化结果，右列为聚类后的结果可视化结果。

在实际场景中，由于系统误差和障碍物等因素的存在，有必要对定位算法的性能进行评估。本文分别从三维精度、X-Y 平面的二维精度和一维精度，衡量模型定位的有效性，同时给出定位模型在三维空间和 XY 平面的可视化结果。

针对三维空间和二维平面，本文选取均方根误差 (RMSE) 作为定位精度的衡量指标，RMSE 可以反映模型定位的真实情况，在空间、和平面的计算公式分别如公式14和13所示。

$$RMSE_{XY} = \sqrt{E[(x-\hat{x})^2 + (y-\hat{y})^2]} \tag{13}$$

$$RMSE_{XYZ} = \sqrt{E[(x-\hat{x})^2 + (y-\hat{y})^2 + (z-\hat{z})^2]} \tag{14}$$

上式中，x、y、z 为靶点的实际坐标值，$\hat{x}$、$\hat{y}$、$\hat{z}$ 为定位算法求解的目标节点坐标值。

针对一维空间，本文选择绝对误差和相对误差作为衡量定位精度的指标，以X 轴为例，公式分别如15和16所示：

$$\Delta x = |x - \hat{x}| \tag{15}$$

$$\delta x = \frac{|x - \hat{x}|}{x} \tag{16}$$

表 5　正常数据的定位精度

衡量指标	RMSE_{XYZ}/cm	RMSE_{XY}/cm	$\Delta x/cm$	δx	$\Delta y/cm$	δy	$\Delta z/cm$	δz
结果	37.04	6.95	4.76	0.03	4.11	0.03	35.85	0.31

由表5可知，本节所提的 LS-MLP 方法在 X-Y 二维空间、X 轴、Y 轴的定位精度较高；由于锚点在 Z 轴分布较近，导致距离测量误差较大，因此在 Z 轴的定位误差交大。将 X-Y 平面和三维空间的部分定位案例可视化，结果分别如图4-7和4-8所示.

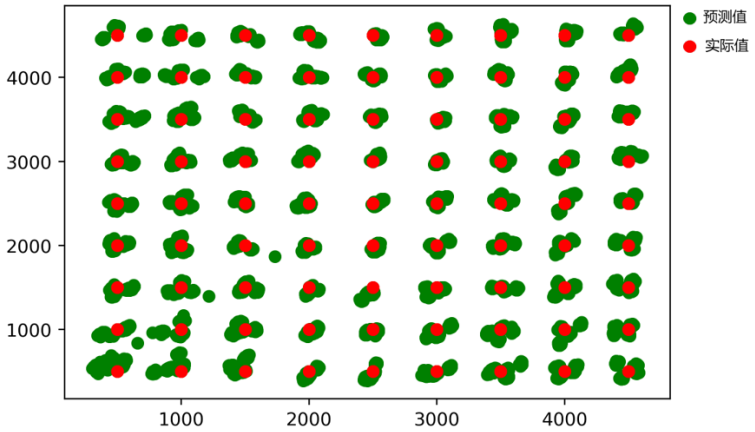


图 4-7　X-Y 平面定位结果可视化

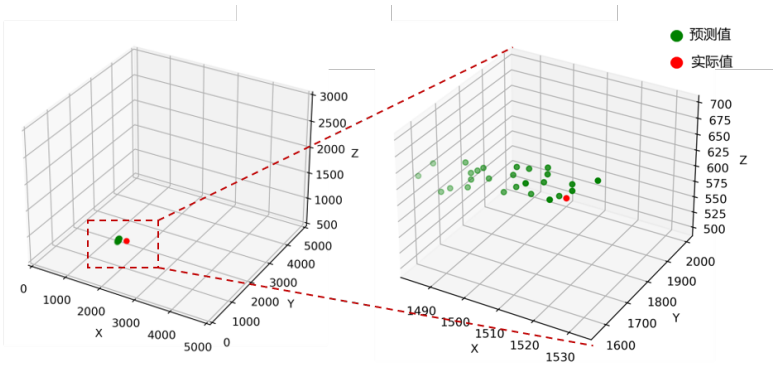


图 4-8　三维空间定位结果可视化

4.4 异常数据的预测定位

4.4.1 置信度赋权的联合定位模型

基于 UWB 的室内定位系统，当某一基站出现异常情况时，该异常基站与定位标签之间的信号会受到影响，造成观测距离出现较大偏差。由题可知，本题中信号干扰的情况主要是锚点与靶点间有遮挡造成的非视距传播（Non-Line Of Sight，NLOS），示意图如图4-9所示。

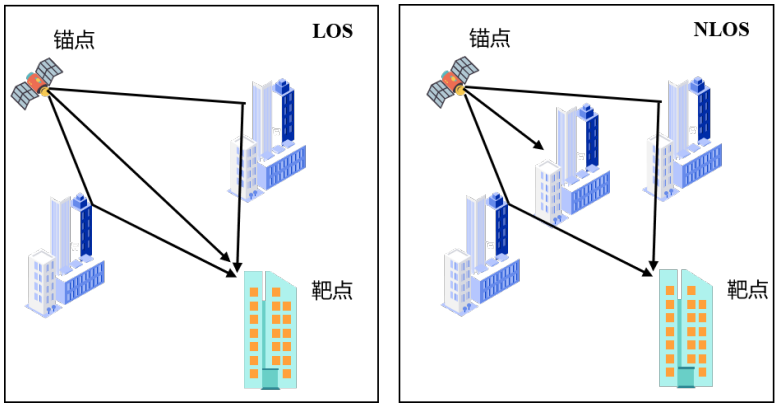


图 4-9　视距（LOS）与非视距（NLOS）传播示意图

在非视距传播的情形下，基站的观测距离大于实际距离。基于此，本文对异常数据进行可视化，如图4-10所示。

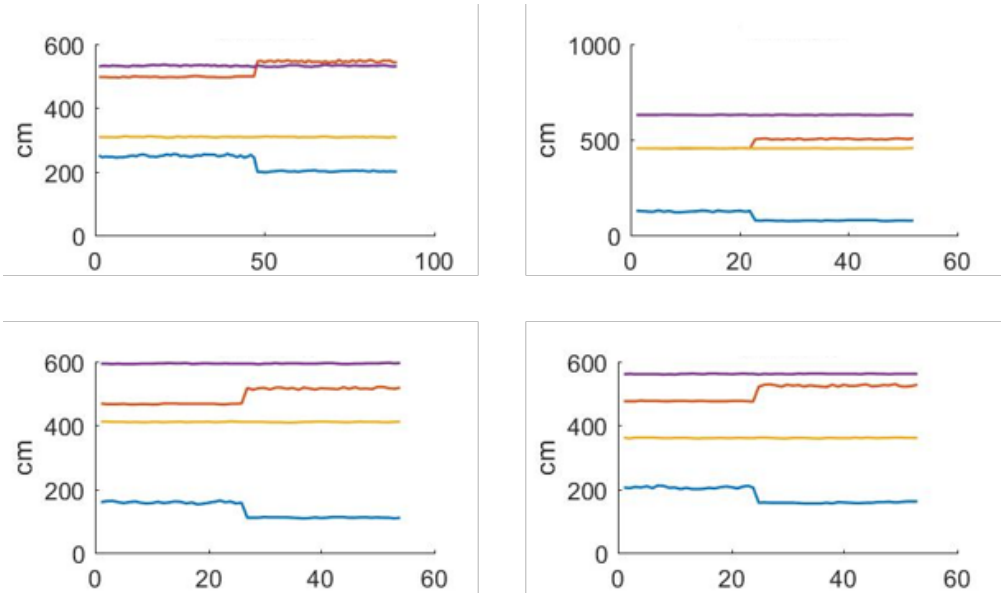


图 4-10　多个异常数据可视化图

根据数据可视化结果，合理假设某一时刻只有一个基站出现异常，即任一时刻至少有 3 个基站的数据是正常的。基于本假设，本文可以对信号干扰下获得的距离矩阵 **D** 进行数据的切片采样。将采样后得到的样本数据分别送入 4.3 节中建立的定位模型，得到相应的输出。通过建立结果的置信度评估模型，得到每个输出的置信度值，基于该置信度值再进行结果的聚类选择，得到最终的预测值。下面本文将详细阐述置信度赋权的联合定位模型（简称为 CE-CL）数据的采样及预测方法，以及基于置信度的结果聚类模型。

• 数据采样及预测

由于某一时刻至少有 3 个基站的数据是正常的，因而本文对信号干扰下获得的距离矩阵 **D** 进行切片处理，构造出 5 个抽样距离矩阵，$\mathbf{D_0}$，$\mathbf{D_1}$，$\mathbf{D_2}$，$\mathbf{D_3}$，$\mathbf{D_4}$。其中，$\mathbf{D_0} = \mathbf{D}$，$\mathbf{D_1}$ 由 **D** 保留第 2、3、4 列得到（代表第一个基站出现异常），$\mathbf{D_2}$ 由 **D** 保留第 1、3、4 列得到（代表第二个基站出现异常），$\mathbf{D_3}$ 由 **D** 保留第 1、2、4 列得到（代表第三个基站出现异常），$\mathbf{D_4}$ 由 **D** 保留第 1、2、3 列得到（代表第一 = 四个基站出现异常）。本文分别用这 5 个距离矩阵，输入 4.3 节中建立的定位模型，得到 5 个预测的坐标矩阵，分别记为 $\mathbf{T_0}$、$\mathbf{T_1}$、$\mathbf{T_2}$、$\mathbf{T_3}$、$\mathbf{T_4}$。接下来需要对这 5 个矩阵中的每个预测值进行置信度评估。

4.4.2 基于置信度的结果聚类模型

基于上述方法，对任一个观测样本 $(d_{i1}, d_{i2}, d_{i3}, d_{i4})$，本文能够得到 5 个靶点坐标的预测值，记为 (t_{0x}, t_{0y}, t_{0z})，(t_{1x}, t_{1y}, t_{1z})，(t_{2x}, t_{2y}, t_{2z})，(t_{3x}, t_{3y}, t_{3z})，(t_{4x}, t_{4y}, t_{4z})。这五个预测值中，有一个是在锚点都正常的情况下得到的靶点位置预测值，其余四个则是在锚点异常情况下得到的靶点位置预测值。本文建立置信度评估模型，给与锚点都正常的情况下得到的靶点位置预测值较大的置信度，其余四个较小的置信度，为下文基于置信度的结果聚类选择提供基础。

通过 4.2 节，可得正常数据样本下的靶点位置预测值，通过前述数据采样及预测，可得异常数据样本下的靶点位置预测值。基于此，本文可以有监督地训练一个分类器，对一个观测样本 $(d_{i1}, d_{i2}, d_{i3}, d_{i4})$，及其相应的 (t_x, t_y, t_z) 输出其该预测属于正常数据样本下输出结果的概率，以此作为 (t_x, t_y, t_z) 的置信度水平。

为了在有监督的条件下训练分类器，判断由观测样本 $(d_{i1}, d_{i2}, d_{i3}, d_{i4})$ 得到的坐标信息 (t_x, t_y, t_z) 的置信度水平，本文需要给每一个观测样本赋予“正常”或“异常”的标签。

基于 4.4.1 的假设与分析，本文做出如下判断：首先，正常数据及其切片都为正常，其次，针对异常数据及其切片，同一个时刻下只有一个切片的结果是在“正常”情况下接收的。由于遮挡、反射等因素的存在，测距信号在传播过程中容易出现延时误差，因此如果接收的距离出现较大变动，则认为在传播过程中出现了异常情况。具体来说，如果接收的距离数据突然变小，则变动前为异常数据，变动后为正常数据；如果接收的距离数据突然变大，则变动前为正常数据，变动后为异常数据。

为了精确定位发生变动的数据点，本文对数据进行平滑操作，过滤掉微小的波动，并采用了多个尺度的滑动窗口，对存在异常的切片数据进行片段分析，找到数据波动幅度最大的节点，以此为分界点，根据数据变动情况分别对前后的数据打上“正常”或“异常”的标签。

本文采用一个输入维度为 7 的 3 层多层感知机模型，配合 Softmax 分类器和交叉熵损失函数，公式分别为17和18，输出切片数据的置信度水平。多层感知机的网络结构如图4-3所示。Softmax 函数输出一个二维向量 $[y_1, y_2]$，分别表征数据为正常和异常的置信度。

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{2} e^{x_i}}, i = 1, 2 \tag{17}$$

$$Loss = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \tag{18}$$

基于上述操作，可得信号干扰情况下的 324 个靶点位置带置信度的预测值。需要从每个靶点位置多个带置信度的预测值中，选取置信度高且空间距离距离

相对紧密的集团作为最终的聚类结果。首先对置信度进行降序排列，选择前百分之五十的预测值进行聚类。在这里，依旧选择 k-means 聚类算法进行聚类，聚成两类，选择数量多的的一类作为最终的预测结果。

4.4.3 结果分析

表 6　异常数据的定位精度

衡量指标	RMSE_{XYZ}/cm	RMSE_{XY}/cm	$\Delta x/cm$	δx	$\Delta y/cm$	δy	$\Delta z/cm$	δz
结果	46.21	18.24	11.59	0.07	11.72	0.08	40.13	0.36

由表6可知，本节所提的 CE-CL 方法在 X-Y 二维空间、X 轴、Y 轴仍然保持较高精度；由于锚点在 Z 轴分布较近，导致距离测量误差较大，因此在 Z 轴的定位误差交大。X-Y 平面和三维空间的可视化结果分别如图4-7和4-8所示

4.5 预测结果

- 信息无干扰下的靶点位置预测结果

表 7　前五组靶点，正常数据位置预测坐标表

数据编号	x/cm	y/cm	z/cm
1	110±4	67±5	145± 18
2	316±4	170±4	163± 11
3	271±3	114±4	157± 130
4	248±3	103±5	170±74
5	148±7	254±6	157 ±96

- 信息有干扰下的靶点位置预测结果

表 8　后五组靶点，异常数据位置预测坐标表

数据编号	x/cm	y/cm	z/cm
1	209±8	79±9	157 ±13
2	437±12	170±20	158±12
3	195±1	141±6	173±9
4	359±4	206±2	193±12
5	484±8	205±15	159 ±12

5. 任务三的分析与求解

5.1 任务三分析

任务二中训练模型所用的数据都采集于同一实验场景下，但所建立的定位模型应该能够用于不同的场景，即模型有一定的泛化能力。本节旨在进行模型泛

化能力的测试。不同于实验场景 1 下采集到的数据，实验场景 2 中只有 10 组数据（即 10 个靶点位置），其中前五组数据无信号干扰，后五组数据有信号干扰，每组数据只有一个样本点。

由任务二可知，UWB 精确定位问题实质上为超定方程组的求解问题。本文所建立的基于神经网络的观测定位模型，可以看作一个求解该超定方程组的求解器。即利用一个简单的多层感知机，来拟合观测得到的距离矩阵 **D** 与所求的靶点位置坐标矩阵 **T** 之间的（非线性）关系，通过梯度下降法优化残差，进而优化靶点位置坐标矩阵 **T**，得到超定方程组的最小二乘解。模型通过巧妙地设计矩阵形式的损失函数来指导网络进行优化，损失函数既包含了实验场景信息，且关于优化目标 **T** 可微。通过将实验场景信息编码到损失函数中，模型实现了网络结构与实验场景信息的剥离。网络采用简单的双层感知机模型（4->32->3），参数量较少，训练时不容易出现过拟合的情况，模型本身即具有较好的泛化能力。

因此，本文通过更新模型的损失函数，将实验场景 2 的信息编码到模型中，实现模型的迁移，进而实现实验场景 2 下的靶点定位。任务三的解决流程如图5-1所示，主要分为模型迁移，其中包含重新编码场景信息与加权聚合结果两个部分，具体描述如下。

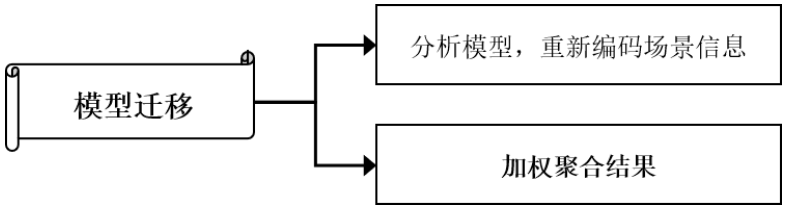


图 5-1　任务三流程图

5.2 场景信息重编码

- 损失函数更新

由前可知，模型的损失函数为：

$$\min\left(\mathbf{D}'-\mathbf{D}\right)^2, \tag{19}$$

其中，**T** 为神经网络的输出，即靶点位置估计值矩阵。$\mathbf{A_1},\mathbf{A_2},\mathbf{A_3},\mathbf{A_4}$ 为 4 个锚点的坐标进行广播后得到的坐标矩阵。$diag$ 操作表示取矩阵的对角线元素，将其组成一个列向量。则 $\mathbf{D}'$ 为估计的靶点位置与 4 个锚点间的距离矩阵。因而模型的迁移只需要更新 $\mathbf{A_1},\mathbf{A_2},\mathbf{A_3},\mathbf{A_4}$ 即可，即：$\mathbf{A_1}=\begin{bmatrix}0 & 0 & 120\\ \vdots & \vdots & \vdots\\ 0 & 0 & 120\end{bmatrix}$，$\mathbf{A_2}=\begin{bmatrix}500 & 0 & 160\\ \vdots & \vdots & \vdots\\ 500 & 0 & 160\end{bmatrix}$，$\mathbf{A_3}=\begin{bmatrix}0 & 300 & 160\\ \vdots & \vdots & \vdots\\ 0 & 300 & 160\end{bmatrix}$，$\mathbf{A_4}=\begin{bmatrix}500 & 300 & 120\\ \vdots & \vdots & \vdots\\ 500 & 300 & 120\end{bmatrix}$。

- 信息无干扰下的靶点预测

更新模型损失函数后，本文用任务二中建立的基于神经网络的观测定位模型来进行信号无干扰下的靶点定位。模型的输入的观测距离矩阵：

$$\mathbf{D} = \begin{bmatrix} 422 & 258 & 373 & 145 \\ 450 & 194 & 442 & 146 \\ 355 & 251 & 341 & 214 \\ 330 & 313 & 290 & 279 \\ 72 & 452 & 305 & 538 \end{bmatrix}。$$

- 信息有干扰下的靶点预测

更新模型损失函数后，本文用任务二中建立的基站异常的联合定位模型来进行信号有干扰下的靶点定位。

模型的输入的观测距离矩阵：$\mathbf{D_0} = \begin{bmatrix} 510 & 222 & 497 & 80 \\ 290 & 321 & 314 & 289 \\ 238 & 353 & 232 & 376 \\ 215 & 322 & 314 & 364 \\ 162 & 395 & 258 & 444 \end{bmatrix}$，

$\mathbf{D_1} = \begin{bmatrix} 222 & 497 & 80 \\ 321 & 314 & 289 \\ 353 & 2320 & 376 \\ 322 & 314 & 364 \\ 3950 & 258 & 444 \end{bmatrix}$，$\mathbf{D_2} = \begin{bmatrix} 510 & 497 & 80 \\ 290 & 314 & 289 \\ 238 & 232 & 376 \\ 215 & 314 & 364 \\ 162 & 258 & 444 \end{bmatrix}$，

$\mathbf{D_3} = \begin{bmatrix} 510 & 222 & 80 \\ 290 & 321 & 289 \\ 238 & 353 & 376 \\ 215 & 322 & 364 \\ 162 & 395 & 444 \end{bmatrix}$，$\mathbf{D_4} = \begin{bmatrix} 510 & 222 & 497 \\ 290 & 321 & 314 \\ 238 & 353 & 232 \\ 215 & 322 & 314 \\ 162 & 395 & 258 \end{bmatrix}$。

5.3 加权聚合结果

送入基站异常的联合定位模型，可得四个观测矩阵对应的四个靶点位置估计矩阵 $\mathbf{T_0}$，$\mathbf{T_1}$，$\mathbf{T_2}$，$\mathbf{T_3}$，$\mathbf{T_4}$。然后，利用 4.4 节中训练好的置信度评估模型，对四个矩阵共 20 个预测结果进行置信度的评估。可得每个靶点位置下的 5 个位置估计值，记为 $(t_{0x}, t_{0y}, t_{0z}), (t_{1x}, t_{1y}, t_{1z}), (t_{2x}, t_{2y}, t_{2z}), (t_{3x}, t_{3y}, t_{3z}), (t_{4x}, t_{4y}, t_{4z})$ 的置信度值，记为 z_0, z_1, z_2, z_3, z_4。利用置信度值对 5 个位置估计值进行加权平均，得到最终的预测值，即：

$$\begin{aligned} t_x &= \frac{z_0 * t_{0x} + z_1 * t_{1x} + z_2 * t_{2x} + z_3 * t_{3x} + z_4 * t_{4x}}{z_0 + z_1 + z_2 + z_3 + z_4} \\ t_y &= \frac{z_0 * t_{0y} + z_1 * t_{1y} + z_2 * t_{2y} + z_3 * t_{3y} + z_4 * t_{4y}}{z_0 + z_1 + z_2 + z_3 + z_4} \\ t_x &= \frac{z_0 * t_{0z} + z_1 * t_{1z} + z_2 * t_{2z} + z_3 * t_{3z} + z_4 * t_{4z}}{z_0 + z_1 + z_2 + z_3 + z_4} \end{aligned} \tag{20}$$

5.4 预测结果

表 9　前五组 Tag 位置预测坐标表

数据编号	x/cm	y/cm	z/cm
1	366± 3	221± 4	125± 5
2	419± 9	172± 4	112± 7
3	317±2	175± 3	124± 5
4	256±6	191±10	77±20
5	553±6	20±3	56 ±33

表 10　后五组 Tag 位置预测坐标表

数据编号	x/cm	y/cm	z/cm
1	480±7	219±25	150 ±13
2	255±9	156±19	104±24
3	146±18	145±34	14±22
4	195±1	103±2	45±33
5	106±9	88±18	47 ±32

6. 任务四的分析与求解

6.1 任务四分析

在任务二中，分别针对正常数据和异常数据建立了（根据前述方法扩充）定位模型，根据靶点与四个锚点的距离，计算其空间坐标。但在实际场景下，无法预知 UWB 信号传输过程中数据的质量，想要针对性地根据数据质量选择定位模型，首先需要判断信号是否受到干扰。该问题为典型的有监督分类问题，数据集按照标签信息可分为“正常”（记作 1）或“异常”（记作 0）两类，而输入数据为靶点与 4 个锚点的距离信息。

由于输入数据维度较低，分类器可能无法充分学习到距离信息内在表征，导致过拟合、分类精度差等问题，因此，本节采用多种特征工程方法，增加输入特征，充分挖掘距离信息和信号质量的内在联系，增强模型的鲁棒性。此外，通过分析任务 1 处理后的数据发现，所有 324 个靶点生成的“正常”数据共有 8000 条，“异常”数据共有 30000 条，存在严重的数据分布不平衡问题。使用传统分类算法训练不平衡数据时，模型的决策边界容易偏向多数类数据，而忽略了少数类数据。因此，本节采用基于集成学习的极限梯度提升算法 (eXtreme Gradient Boosting，XGBoost) 方法，通过交叉验证的训练方式，消除数据分布不平衡对分类精度的影响，同时引入贝叶斯优化方法，自动调整极限梯度提升算法的重要超参数，提升模型的分类精度。任务四的解决流程如图6-1所示，具体分为数据预处理、模型选择、自动机器学习、投票法集成，具体描述如下。

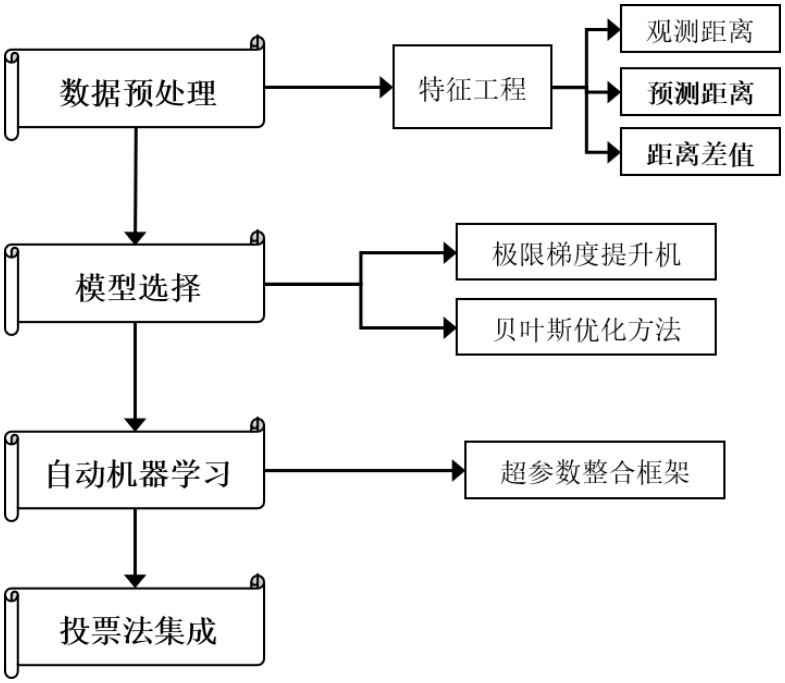


图 6-1　任务四流程图

6.2 数据预处理

6.2.1 特征工程

为了挖掘数据的内在特征，在正常数据和异常数据之间获得明显的区分，本节对距离数据 (d_1, d_2, d_3, d_4) 进行了如下特征挖掘：

1. 根据任务 2 提出的模型，计算每组距离数据 (d_1, d_2, d_3, d_4) 对应的**预测坐标** (x, y, z)

2. 根据所得预测坐标 (x, y, z)，计算靶点与锚点之间的**预测距离** $(\hat{d}_1, \hat{d}_2, \hat{d}_3, \hat{d}_4)$

3. 计算**实际距离** (d_1, d_2, d_3, d_4) 和**预测距离** $(\hat{d}_1, \hat{d}_2, \hat{d}_3, \hat{d}_4)$ 的差值，并取绝对值，得到**距离预测误差** $(|d_1 - \hat{d}_1|, |d_2 - \hat{d}_2|, |d_3 - \hat{d}_3|, |d_4 - \hat{d}_4|)$

综上所述，本节将原始的仅包含距离信息的 4 维度特征扩充为了 12 维特征，借助任务 2 提出的定位模型，能够扩大正常数据与异常数据的内在差距，有助于后续的分类。

6.2.2 数据归一化

在进行特征工程后，对所提取特征进行数据归一化，消除数值差异太大带来的影响，加快模型收敛速度数据标准化过程如式21所示。

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{21}$$

6.3 极限梯度提升简介

极限梯度提升算法是陈天奇等人提出的机器学习算法，是梯度提升决策算法 (Gradient Boosting Decision Tree, GBDT) 的工程实现，它将弱学习器模型以迭代的方式组合成强学习器。相较于 GBDT，极限梯度提升算法主要改进在于对损失函数进行归一化，加入了模型复杂度的正则化项，减小了模型的方差，降低了建模复杂性和模型过度拟合的可能性。此外，GBDT 方法只能处理机器学习中的

一阶导数，而极限梯度提升算法可以通过泰勒展开处理高阶损失函数。极限梯度提升算法可以处理稀疏数据，灵活实现分布式并行计算，已经被广泛应用于分类、回归预测等多种机器学习问题。图6-2给出了极限梯度提升算法的流程图。

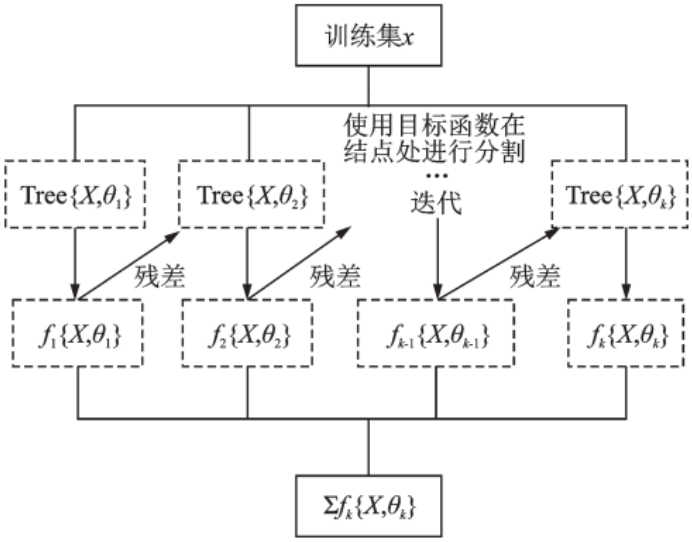


图 6-2　极限梯度提升算法流程图

极限梯度提升算法的基本模型是决策树，首先定义单个决策树的输出，如公式22所示。

$$f(x) = \omega_{q(x)}, \omega \in \mathbf{R}^T, q : \mathbf{R}^d \to \{1, 2, \cdots, T\} \tag{22}$$

上式中，x 是输入向量，q 表示树的结构，结构函数 $q(x)$ 表示把输入映射到叶子的索引号，ω 表示对应于每个索引号的叶子的分数，T 是树中叶节点的数量，d 为特征维数。极限梯度提升算法可看成是由 K 棵决策树的集合，则 K 棵树集合的输出为:

$$y_i = \sum_{k=1}^{K} f_k(x_i) \tag{23}$$

由决策树的模型可知，单棵决策树的复杂度计算公式为：

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda\|\omega\|^2 \tag{24}$$

类似地，集成树的复杂度可以表示为：

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T}\omega_j^2 \tag{25}$$

上式中，T 是叶节点的数目，γ 是范围在 0 和 1 之间的学习速率，γ 乘以 T 等于树修剪，防止过度拟合，λ 是一个正则化参数，ω 是叶子的质量，$\Omega(f_k)$ 是极限梯度提升算法的正则项。

此外，极限梯度提升算法的目标函数在第 i 步的迭代可以表示为

$$\mathbf{Obj}^{(t)} = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{t} \Omega(f_k) \tag{26}$$

上式包含两个部分：第一部分代表真实值 y_i 和预测值 $\hat{y}_i$ 的误差之和，L 为误差函数。第二部分代表单棵决策树的复杂度之和。已知：$\hat{y}^t$ 与 $\hat{y}^{t-1}$ 的函数关系为:$\hat{y}^t = \hat{y}^{t-1} + f_t(x_i)$，其中 $f_t(x_i)$ 为第 t 轮需要学习的决策树，因此式26中目标函数可以转化为:

$$\mathbf{Obj}^{(t)} = \sum_{i=1}^{n} L(y_i, \hat{y}_i^t) + \sum_{i=1}^{t} \Omega(f_i) = \sum_{i=1}^{n} L(y_i j, \hat{y}_i^{(t-1)} + f(x_i)) + \sum_{i=1}^{t} \Omega(f_i) \tag{27}$$

6.4 贝叶斯优化方法简介

在使用极限梯度提升算法时，需要根据具体数据集调整多种超参数的组合，基于人工经验选择超参数组合的过程不仅费时费力，而且依赖大量的专家经验，因此本节采用自动机器学习技术中的贝叶斯优化模型，选择树形结构 Parzen 估计器（Tree-structured Parzen Estimator，TPE）为代理模型，用于构建极限梯度提升算法的训练与优化过程。TPE 模型能有效应对多种复杂的超参数配置情况。代理模型和采集函数是 TPE 算法的两大核心，以下详细介绍本节所使用的超参数优化算法细节。

（1）代理模型：本文选择的代理模型是 TPE。有别于传统的基于 $p(y|x)$ 建模的贝叶斯优化模型，如基于高斯过程和基于随机森林的贝叶斯优化，TPE 对 $p(x|y)$ 和 $p(y)$ 建模，能够处理更加复杂的超参数分布。TPE 将每个参数的先验分布转换成高斯混合分布，再基于已观测值修改后验分布，从已有的 y 计算分位数 y^*，对大于 y^* 和小于 y^* 的数据，分别建立条件概率密度公式，如公式28所示。

$$p(x|y) = \begin{cases} l(x), \ y < y^* \\ g(x), \ y \geq y^* \end{cases} \tag{28}$$

上式中，y^* 是基于已观测 y 计算得到的分位数，本文取 $\gamma = 0.25$，即 $\gamma = p(y < y^*) = 0.25$；$l(x)$ 和 $g(x)$ 为服从高斯过程的概率密度函数，其中 $l(x)$ 表示 $y < y^*$ 时的概率密度，$g(x)$ 表示 $y \geq y^*$ 时的概率密度。

（2）采集函数：TPE 模型选择的采集函数为 EI（Expected improvement）定义如公式29所示。

$$EI(x) = E_{\max}(f(x) - f(\hat{x}), 0) = \begin{cases} (\mu(x) - f(\hat{x}))\Phi(Z) + \sigma(x)\varphi(Z), \ \sigma(x) > 0 \\ 0 \qquad\qquad\qquad\qquad\qquad\quad , \ \sigma(x) \leq 0 \end{cases} \tag{29}$$

$$Z = \begin{cases} \dfrac{\mu(x) - f(\hat{x})}{\sigma(x)}, \sigma(x) > 0 \\ 0 \qquad\quad , \sigma(x) = 0 \end{cases} \tag{30}$$

上式中，$f(\hat{x})$ 是目标函数的当前最优值，$\hat{x}$ 是对应的最优参数组合，$\mu(x)$ 和 $\sigma(x)$ 分别是高斯过程在 x 处后验预测的均值和标准差，Φ 和 φ 分别是标准正态分布的累计分布函数和概率密度函数。

$$\max EI_{y^*}(x) = \frac{\gamma y^* l(x) - l(x)\int_{-\infty}^{y^*} p(y)dy}{\gamma l(x) + (1-\gamma)g(x)} \propto \left(\gamma + \frac{g(x)}{l(x)}(1-\gamma)\right)^{-1} \tag{31}$$

$$\gamma = p(y < y^*),\ p(x) = \int_R p(x|y)p(y)dy = \gamma l(x) + (1-\gamma)g(x) \tag{32}$$

搜寻下一个采样点的过程可以通过最大化 EI 函数实现。由公式31可知，在 TPE 模型中，最大化 EI 的过程也是最大化 $\frac{l(x)}{g(x)}$ 的过程。$\frac{l(x)}{g(x)}$ 越高，则下一个采样点 x 在目标函数上取值大于 y^* 的可能性越大，超参数的性能就越好。

因此，本节取每次训练的损失为贝叶斯优化的目标函数，经过多次迭代，选择其中性能最优的超参数组合作为最终模型。

6.5 模型训练及超参数优化

本节对靶点的实测距离数据进行特征工程，得到 12 维的特征数据，使用极限梯度提升算法判断传输数据的异常与否，并使用贝叶斯优化提升模型的预测精度。整体算法流程图如图 6-2 所示。

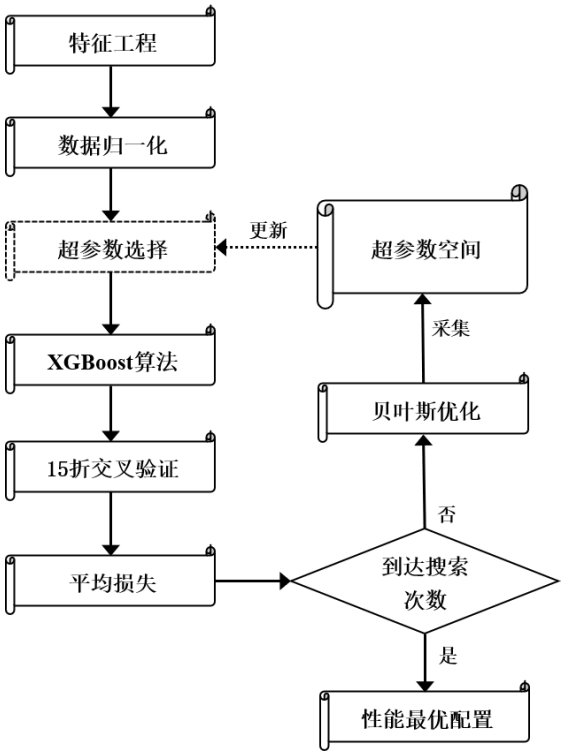


图 6-3　超参数优化流程图

6.5.1 模型训练

为了衡量算法在实测数据集上的分类精度和鲁棒性，实验采用了 15 折交叉验证法，原始样本分为 15 个数量相同且不重叠的子集，每次选择其中 1 个作为验证集，剩余 14 个作为训练集，得到 15 组训练-测试数据集。选择精度 (accuracy) 为评价指标，公式如式 (33) 所示。

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{33}$$

上式中，TP 表示预测的正真例，TN 表示真负例，FP 表示假正例,FN 表示假负例

6.5.2 超参数优化设置

在使用极限梯度提升算法时，需要根据具体数据集调整多种超参数的组合，以获得最优的分类结果。本节选择多个重要的超参数，设置如表11所示的超参数范围，借助贝叶斯优化方法，自动选择最优的超参数配置。

表 11　超参数空间

参数名称	作用描述	参数范围
learning_rate	学习率	[0.01 0.2]
n_estimators	决策树数量	[100,200,300,400,500]
max_depth	每棵树最大深度	[4,5,6,7,8,9,10]
tree_method	树方法	[”exact”,”auto”,”approx”,”hist”]
gamma	树枝修剪率	[0 1]

贝叶斯优化的目标函数为 15 折交叉验证的平均损失，公式如式 (34) 所示。通过最小化分类损失的方式，找到最优的超参数配置，迭代次数设置为 50 次。

$$A_{\theta^*} = \arg\min_{\theta_i \in \Theta} L(A_{\theta_i}, D) \tag{34}$$

上式中，D 表示训练所用数据集，Θ 表示待搜索的超参数空间，θ_i 表示该超参数空间中的一组超参数配置，A 表示所用极限梯度提升算法，A_{θ_i} 表示超参数为 θ_i 的极限梯度提升算法，$A_{\theta*}$ 为经过优化的最优结果

6.5.3 训练结果

模型的训练结果如表12所示，可以看出使用特征工程和超参数优化方法可以在原有模型基础上有效提升分类的精度。

表 12　结果分析

模型	使用特征	准确率 (%)
XGBoost	(d_1, d_2, d_3, d_4)	88.34
XGBoost	$+(\hat{d}_1, \hat{d}_2, \hat{d}_3, \hat{d}_4)$	91.1
XGBoost	$+(\hat{d}_1, \hat{d}_2, \hat{d}_3, \hat{d}_4)$ $+(\lvert d_1 - \hat{d}_1\rvert, \lvert d_2 - \hat{d}_2\rvert, \lvert d_3 - \hat{d}_3\rvert, \lvert d_4 - \hat{d}_4\rvert)$	95.6
XGBoost+ 贝叶斯优化	$+(\hat{d}_1, \hat{d}_2, \hat{d}_3, \hat{d}_4)$ $+(\lvert d_1 - \hat{d}_1\rvert, \lvert d_2 - \hat{d}_2\rvert, \lvert d_3 - \hat{d}_3\rvert, \lvert d_4 - \hat{d}_4\rvert)$	99.2

6.6 基于投票法的集成预测

由于单个模型的输出随机性较大，故本文采用了基于投票法的集成学习对测试数据进行预测。具体而言，利用 15 折交叉训练，获得 15 个不同的分类模型。单个分类模型的训练超参数如前所述进行整合。在训练获得不同的分类模型

f_i 后，对测试数据进行预测，得到若干分类结果。最后使用投票法进行最终的集成。若干模型的分类结果以及最终的结果如表13所示。

表 13　任务四使用投票法后的预测结果。1 代表预测为正常，0 代表预测为异常。

模型序号	1	2	3	4	5	6	7	8	9	10
1	1	0	1	0	0	1	0	0	0	1
2	1	0	1	0	0	1	0	0	0	1
3	1	0	1	0	0	1	0	0	0	1
4	1	0	1	0	0	1	0	0	0	1
5	1	0	1	0	0	1	0	0	0	1
6	1	0	1	0	0	0	0	0	0	1
7	1	0	1	0	0	1	0	0	0	1
8	1	0	1	0	0	1	0	0	0	1
9	1	0	1	0	0	1	0	0	0	1
10	1	0	1	0	0	0	0	1	0	1
11	1	0	1	0	0	0	0	0	0	1
12	1	0	1	0	0	1	0	0	0	1
13	1	0	1	0	0	0	0	0	0	1
14	1	0	1	0	0	0	0	0	0	1
15	1	0	1	0	0	1	0	0	0	1
最终结果	1	0	1	0	0	1	0	0	0	1

7. 任务五的分析与求解

7.1 任务五分析

任务五要求利用任务二所得的静态点的定位模型，结合靶点自身的运动规律，对附件 5 给出的带随机干扰的动态靶点进行运动轨迹精确定位。由前所述，本文提供了 LS-MLP 方法以及 CE-CL 方法分别处理正常数据、异常数据的定位，并输出定位的结果及其空间位置的方差。为了得到更加精确的运动轨迹，本文对靶点的运动状态进行隐式建模，并将单个点的位置计算结果作为传感器观测值，采用卡尔曼滤波器进行二者的联合优化，得到最终的精确定位结果。任务五的解决流程如图7-1所示，具体分为数据读取、观测值预测、卡尔曼滤波、结果可视化，具体描述如下。

首先利用与任务一中类似的方法对任务五所给数据进行字符串处理，整理得到所需的数据矩阵；之后建立了能同时适用于正常数据与异常数据的统一处理框架**统一置信度赋权定位框架**，对观测值进行初步的预测；由于靶点处于动态，故利用卡尔曼滤波对初步估计得到的运动轨迹结合靶点运动状态进行联合优化；最后对结果进行可视化，分别展示三维以及二维的可视化结果。

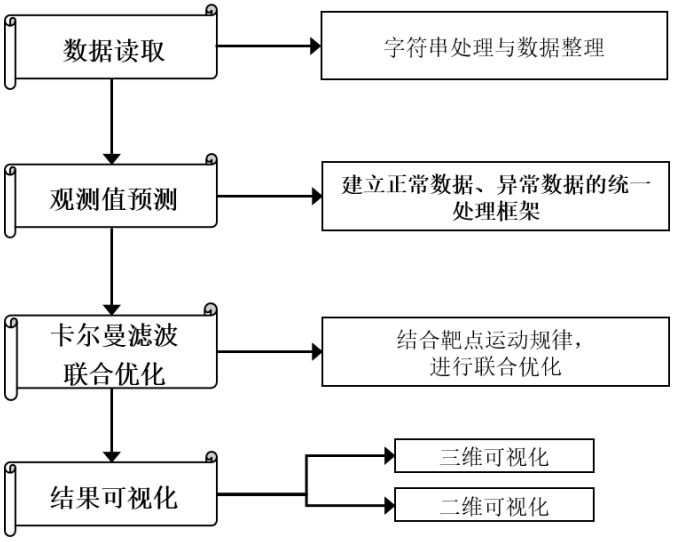


图 7-1　任务 5 的流程图

7.2 数据读取

本文采用与任务一相同的方法对数据进行读取。读取过程中发现，相邻组数据之间的时间戳差值接近相同，为此本文假设传感器的采样频率固定，从而无需对时间戳进行额外处理。最终处理得到的数据仅保存每组的 $d_i, i = 1, 2, 3, 4$ 四个距离值。部分数据如表14所示。

表 14　任务五数据样例表

数据编号	d1	d2	d3	d4
0	81	465	457	652
1	81	465	457	651
2	81	464	458	651
3	81	464	458	652
4	80	464	461	653
5	79	464	460	653

7.3 统一置信度赋权定位模型

LS-MLP 正常模型与 CE-CL 异常模型是分别在数据有无异常的情况下建立的，通常而言，针对数据有无异常需要使用不同的模型进行计算，但这会带来额外的计算步骤。且由于对于数据异常判断存在错误的可能，会进一步导致模型的错误选择，无法取得最优解。本节首先使用统一的框架对 LS-MLP 正常模型以及 CE-CL 异常模型进行描述，从而可使用该框架对任意数据进行计算，而无需先对数据进行分类再选择模型。

CE-CL 异常模型实质为 LS-MLP 正常模型的改进版本。LS-MLP 正常模型在通过预测模型对输入的距离值组合进行预测后会得到相应坐标，之后通过选择最大聚类簇得到最终的结果。异常数据中使用不同的数据组合计算相应坐标，

由于异常数据中，仅有单个距离组合是正常组合，即该组合对应的输出置信度最高，而其他组合的输出置信度相对较低。前述采用 CE 分类模型衡量输入距离矩阵 **D** 与预测结果 **T** 的一致性，即置信度。从而可通过五个距离组合，得到五组距离输出以及其对应的置信度，之后通过基于置信度的聚类模型得到最终的结果（详细说明参考 4.4 节）。与正常数据的处理仅有的区别在于最终的五组坐标包含置信度。为统一流程，所有的数据均使用总结得到的**统一置信度赋权定位框架**进行定位，其流程可总结为图7-2。

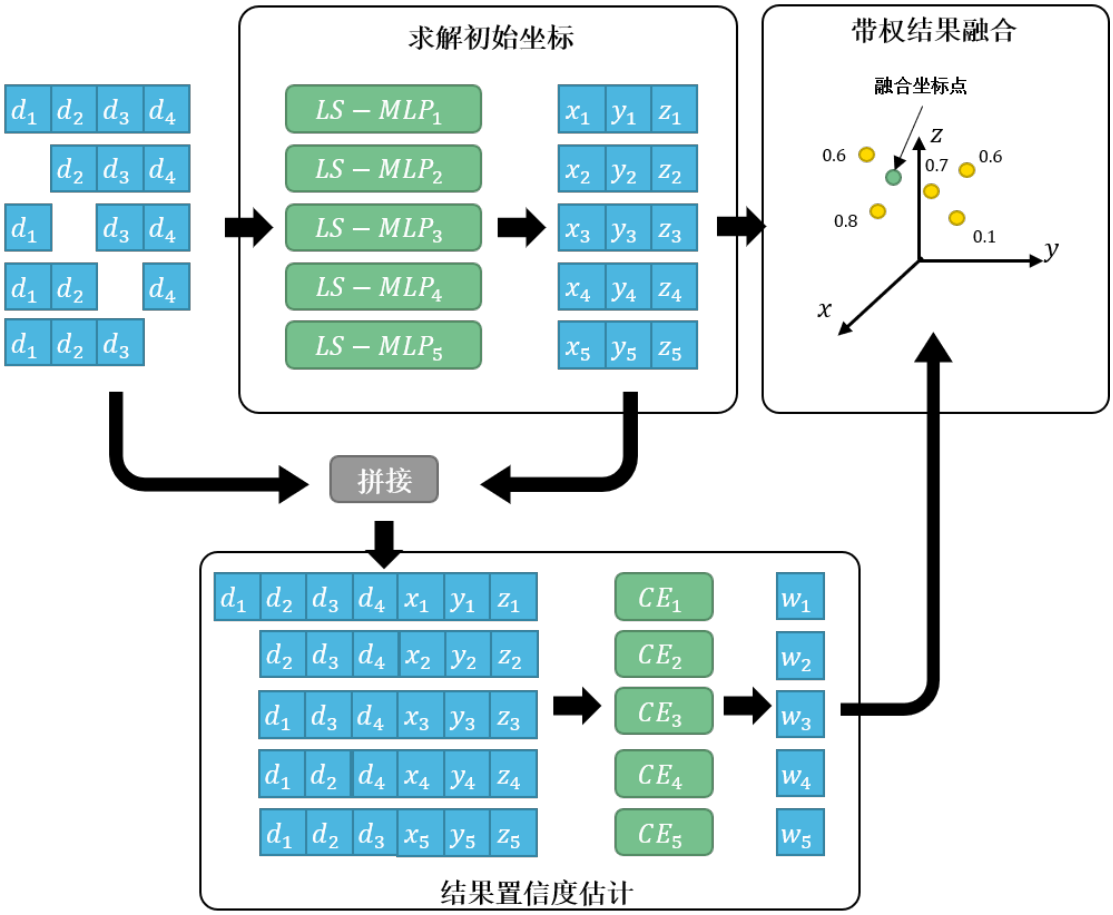


图 7-2　统一置信度赋权定位框架。框架主要包含求解初始坐标、结果置信度估计、带权结果融合三个步骤。该框架无需对数据进行分类，正常数据、异常数据均可通过该框架进行计算。其中，CE 为任务二中提到的置信度估计器。

如图所示，该框架首先使用在正常数据训练获得的模型组合 LS-MLP 进行求解初始坐标，得到预测定位结果；之后使用分类模型对观测距离和预测定位结果进行一致性预测，获得每个预测结果对应的权重，从而每组距离值可获得 5 个初步定位结果及其权重，其中，正常数据输入获得的结果对应的权重理论上偏高，异常数据输入获得的结果对应权重偏低；最后，利用基于置信度的聚类模型对所获得的五个初步定位结果进行融合解算，得到最终的预测结果。使用该框架对附件 5 给出的数据进行初步静态定位，结果为 $\mathbf{T_s}$。

7.4 卡尔曼滤波

随着无线定位逐渐发展，卡尔曼滤波器被应用于移动靶点的定位。卡尔曼滤波可通过一系列不完全或者包含噪声的测量数据中，估计动态系统的状态，本质为一种自回归滤波器。仅需上一时刻的状态估计值以及当前状态的观测值即

可计算当前时刻状态的估计值。基本的卡尔曼滤波器仅使用于线性条件，本文认为测量系统为非线性系统，故使用扩展卡尔曼滤波器（Extended Kalman Filter, EKF）对系统状态进行估计。

卡尔曼滤波器包含两个阶段：预测和更新，其示意图如图7-3所示。预测阶段，滤波器基于上一时刻状态的估计当前时刻的状态；更新阶段，滤波器利用当前状态的观测值对预测阶段的预测值进行优化，以提升观测值的精度。卡尔曼滤波器的状态方程和测量方程如下：

$$\begin{aligned} s_{k+1} &= \mathbf{A}s(k) + \mathbf{Q}w(k) \\ z_k &= \mathbf{G}(k)s(k) + v(k) \end{aligned} \tag{35}$$

其中，$s(k+1)$ 为状态向量，z_k 为测量向量，$\mathbf{A}$ 为状态转移矩阵，$\mathbf{Q}$ 为噪声耦合矩阵，$\mathbf{G}(k)$ 为测量矩阵，$w(k)$、$v(k)$ 为噪声分量。

基本的卡尔曼滤波器仅使适用于线性系统，扩展卡尔曼滤波器用非线性函数的局部线性特征来近似非线性。设 k 时刻的靶点坐标为 (x_k, y_k, z_k)，对应四个基站坐标为为 (x_i, y_i, z_i)，的观测距离为 S^1, S^2, S^3, S^4。取 k 时刻被估计状态 $s(k) = [(x_k, y_k, z_k]^T$，系统状态方程可定义为：

$$s(k) = \mathbf{A} + w(k-1) \tag{36}$$

其中，$\mathbf{A}$ 为一步转移矩阵：

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{37}$$

系统测量方程为：

$$z_k = f(s(k)) + v(k) \tag{38}$$

其中，$z_k = [S^1, S^2, S^3, S^4]^T$,$f(s(k)) = [f_1(s(k)), f_2(s(k)), f_3(s(k)), f_4(s(k))]$。$f_i(s(k))$ 实质为距离函数，求取当前预测值与四个基站之间的距离，$f_i(s(k)) = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2 + (z_k - z_i)^2}$。基于所建立的状态方程与测量方程，可对系统的定位进行联合优化，得到最终结果。

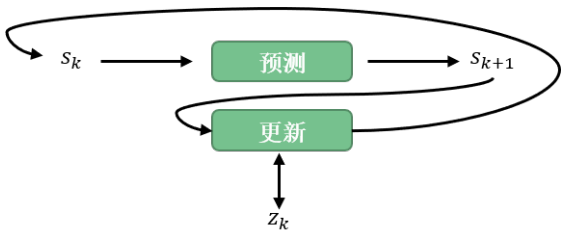
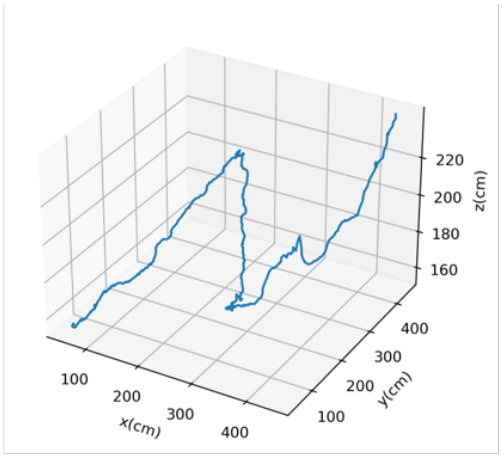


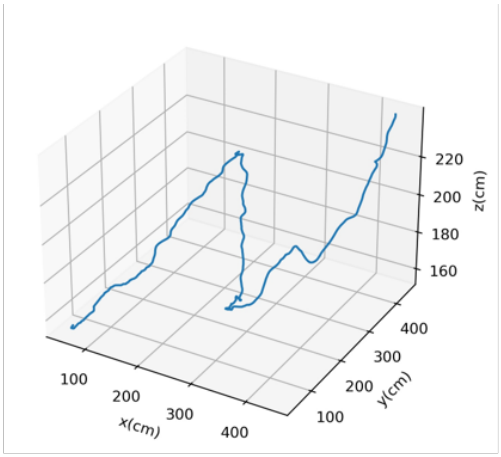
图 7-3　卡尔曼滤波器示意图。卡尔曼滤波分为预测与更新两个阶段，预测时滤波器基于上一时刻状态的估计当前时刻的状态；更新时滤波器利用当前状态的观测值对预测阶段的预测值进行优化。

7.5 结果可视化

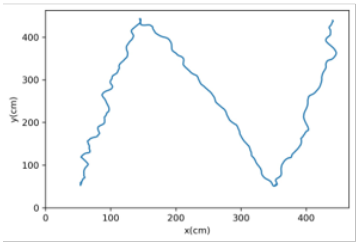
使用卡尔曼滤波对统一定位框架得到的静态点估计以及靶点自身的运动规律，可获得更加准确的运动轨迹跟踪结果。其结果如图 x 所示。从图中可见，原始的预测结果组成的运动轨迹相对而言波动较大，经过卡尔曼滤波后所获得的最终运动轨迹相对平滑，说明了靶点运动规律对运动轨迹预测具有明显的改善效果。从 xy 平面观察，靶点的运动类似于正弦运动，且由前述可知，本文所提模型对于 xy 方向的预测精度较高，说明结果合理；从 xz 与 yz 平面观察，由于模型对于 z 轴预测的误差较大，预测得到的靶点运动规律不明显。



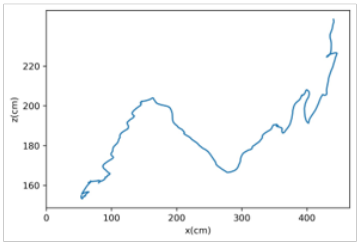
(a) 滤波前运动轨迹



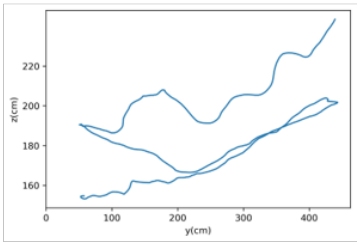
(b) 滤波后运动轨迹



(c) xy运动轨迹



(d) yz运动轨迹



(e) xz运动轨迹

图 7-4　运动轨迹可视化结果

8. 模型的评价与推广

8.1 模型的评价

8.1.1 模型的优点

- 文中提出了针对正常观测数据定位的 LS-MLP 方法，代替传统的最小二乘法对数据进行优化，可通过损失函数的设计，使得模型优化更加抗噪声。
- 文中提出了针对异常观测数据定位的 CE-CL 方法，使用置信度模型预测 $LS-MLP$ 的输入与输出的一致性，从而判断输入数据是否异常，进而使用多模型结果对异常数据进行精确定位。
- 文中建立了正常、异常数据的统一置信度赋权定位框架，该框架无需对正常、异常数据先进行分类之后应用不同模型，而可以输入任意数据并获得高精度结果。

8.1.2 模型的缺点

所建立的模型需迭代计算，相比较传统的具有闭合解的方法，需要更长的运行时间。

8.2 模型的推广和改进

8.2.1 模型的推广

模型可推广至不同环境，仅需锚点坐标，便可进行模型的迁移与结果聚合。

8.2.2 模型的改进

由前所述，模型在 z 轴的误差较大，说明该模型对于锚点的构型较为敏感，对构型范围较小的轴无法精确预测。后续可通过合适的构型选择或者更优的损失函数设计，增加模型的精度。

参考文献

[1] Xu J, Ma M, Law C L. Position estimation using UWB TDOA measurements[C]//2006 IEEE International Conference on Ultra-Wideband. IEEE, 2006: 605-610.

[2] 黄泽华. 基于 UWB 的室内定位系统研究 [D]. 山东大学,2021.

[3] 严嘉祺. 基于 UWB 的室内定位系统的算法与误差分析 [D]. 哈尔滨工业大学,2020.

[4] Chen T, He T, Benesty M, et al. Xgboost: extreme gradient boosting[J]. R package version 0.4-2, 2015, 1(4): 1-4.

[5] Alavi B, Pahlavan K. Modeling of the TOA-based distance measurement error using UWB indoor radio measurements[J]. IEEE communications letters, 2006, 10(4): 275-277.

[6] Bergstra J, Bardenet R, Bengio Y, et al. Algorithms for hyper-parameter optimization[J]. Advances in neural information processing systems, 2011, 24.

[7] 韩中庚. 数学建模方法及其应用 [M]. 高等教育出版社, 2005.

[8] 周志华，机器学习 [M]，清华大学出版社，2016

[9] Pal S K, Mitra S. Multilayer perceptron, fuzzy sets, classifiaction[J]. 1992.

[10] Li W, Zhang T, Zhang Q. Experimental researches on an UWB NLOS identification method based on machine learning[C]//2013 15th IEEE International Conference on Communication Technology. IEEE, 2013: 473-477.

[11] 吴绍华, 张乃通. 基于 UWB 的无线传感器网络中的两步 TOA 估计法 [J]. 软件学报,2007(05):1164-1172.

[12] Kietlinski-Zaleski J, Yamazato T. TDoA UWB positioning with three receivers using known indoor features[J]. IEICE transactions on fundamentals of electronics, communications and computer sciences, 2011, 94(3): 964-971.

附录 A 任务 1 结果

“109. 正常.txt”文件经处理后保留的数据为：

$$\begin{bmatrix}
4910 & 5310 & 2020 & 2880 \\
4890 & 5310 & 2020 & 2880 \\
4920 & 5310 & 2030 & 2870 \\
4910 & 5320 & 2020 & 2860 \\
4870 & 5330 & 2020 & 2870 \\
4860 & 5320 & 2030 & 2870 \\
4890 & 5330 & 2030 & 2880 \\
4880 & 5300 & 2030 & 2880 \\
4890 & 5310 & 2020 & 2860 \\
4880 & 5310 & 2010 & 2870 \\
4880 & 5320 & 2030 & 2870 \\
4890 & 5330 & 2010 & 2880 \\
4910 & 5330 & 2030 & 2880 \\
4880 & 5330 & 2050 & 2880 \\
4880 & 5330 & 2020 & 2890 \\
4870 & 5320 & 2010 & 2890 \\
4870 & 5320 & 2030 & 2890 \\
4890 & 5320 & 2040 & 2890 \\
4910 & 5330 & 2010 & 2880 \\
4930 & 5330 & 2040 & 2870 \\
4910 & 5340 & 2060 & 2880 \\
4890 & 5340 & 2040 & 2890 \\
4910 & 5320 & 2040 & 2890 \\
4910 & 5330 & 2030 & 2900 \\
4910 & 5340 & 2020 & 2890 \\
4930 & 5340 & 2030 & 2880 \\
4910 & 5340 & 2040 & 2890 \\
4910 & 5350 & 2030 & 2880 \\
4910 & 5340 & 2040 & 2870
\end{bmatrix}$$

“1. 异常.txt”文件经处理后保留的数据为：

$$\begin{bmatrix}
1280 & 4550 & 4550 & 6300 \\
1260 & 4550 & 4550 & 6300 \\
1240 & 4550 & 4550 & 6300 \\
1220 & 4550 & 4550 & 6300 \\
1300 & 4550 & 4550 & 6300 \\
1230 & 4540 & 4550 & 6310 \\
1270 & 4540 & 4550 & 6310 \\
1200 & 4550 & 4550 & 6300 \\
1200 & 4550 & 4570 & 6300 \\
1220 & 4550 & 4570 & 6300 \\
1240 & 4550 & 4570 & 6300 \\
1250 & 4550 & 4560 & 6290 \\
1290 & 4550 & 4560 & 6290 \\
1270 & 4550 & 4560 & 6290 \\
1210 & 4550 & 4560 & 6290 \\
1290 & 4540 & 4540 & 6310 \\
1250 & 4540 & 4550 & 6310 \\
1230 & 4550 & 4540 & 6290 \\
1260 & 4550 & 4570 & 6300 \\
1280 & 4560 & 4560 & 6310 \\
1230 & 4560 & 4550 & 6310 \\
1270 & 4540 & 4550 & 6280 \\
770 & 5030 & 4540 & 6290 \\
760 & 5050 & 4550 & 6290 \\
780 & 5070 & 4550 & 6300 \\
770 & 5010 & 4550 & 6280 \\
750 & 5030 & 4550 & 6310 \\
760 & 5010 & 4550 & 6310 \\
740 & 5050 & 4550 & 6300 \\
750 & 5010 & 4550 & 6290 \\
750 & 5070 & 4550 & 6290 \\
770 & 5000 & 4550 & 6300 \\
770 & 5020 & 4550 & 6300 \\
770 & 5030 & 4560 & 6280 \\
740 & 5070 & 4550 & 6310 \\
780 & 5010 & 4540 & 6300 \\
770 & 5070 & 4540 & 6290 \\
760 & 5060 & 4550 & 6310 \\
770 & 5040 & 4550 & 6300 \\
790 & 5070 & 4540 & 6310 \\
790 & 5030 & 4540 & 6310 \\
790 & 5010 & 4550 & 6310 \\
790 & 5030 & 4550 & 6290 \\
790 & 5050 & 4550 & 6300 \\
790 & 5010 & 4550 & 6290
\end{bmatrix}$$

$$\begin{bmatrix} 750 & 5030 & 4550 & 6290 \\ 740 & 5010 & 4540 & 6300 \\ 750 & 5050 & 4540 & 6310 \\ 770 & 5070 & 4540 & 6310 \\ 770 & 5050 & 4540 & 6310 \\ 760 & 5030 & 4560 & 6300 \\ 760 & 5070 & 4560 & 6300 \end{bmatrix}$$

“100. 异常.txt”文件经处理后保留的数据为：

$$\begin{bmatrix}
1520 & 3720 & 4680 & 5720 \\
1530 & 3710 & 4690 & 5720 \\
1510 & 3730 & 4690 & 5720 \\
1530 & 3760 & 4690 & 5700 \\
1510 & 3770 & 4690 & 5700 \\
1510 & 3750 & 4690 & 5700 \\
1510 & 3760 & 4680 & 5710 \\
1510 & 3790 & 4680 & 5700 \\
1510 & 3840 & 4670 & 5710 \\
1520 & 3860 & 4680 & 5710 \\
1510 & 3850 & 4680 & 5700 \\
1510 & 3830 & 4680 & 5700 \\
1500 & 3780 & 4680 & 5710 \\
1490 & 3790 & 4670 & 5690 \\
1500 & 3770 & 4670 & 5700 \\
1520 & 3770 & 4670 & 5700 \\
1510 & 3810 & 4680 & 5700 \\
1500 & 3800 & 4680 & 5710 \\
1520 & 3800 & 4680 & 5720 \\
1510 & 3760 & 4660 & 5700 \\
1510 & 3740 & 4660 & 5700 \\
1510 & 3680 & 4690 & 5700 \\
1510 & 3700 & 4680 & 5700 \\
1510 & 3740 & 4680 & 5690 \\
1500 & 3750 & 4670 & 5710 \\
1490 & 3740 & 4680 & 5710 \\
1510 & 3740 & 4680 & 5710 \\
1500 & 3730 & 4690 & 5700 \\
1510 & 3720 & 4680 & 5700 \\
1520 & 3760 & 4680 & 5690 \\
1520 & 3750 & 4670 & 5710 \\
1530 & 3760 & 4680 & 5720 \\
1510 & 3560 & 5290 & 5710 \\
1510 & 3560 & 5230 & 5710 \\
1510 & 3560 & 5190 & 5700 \\
1510 & 3560 & 5110 & 5700 \\
1510 & 3560 & 5060 & 5710 \\
1510 & 3560 & 4970 & 5710 \\
1510 & 3550 & 4950 & 5710 \\
1510 & 3560 & 4930 & 5720 \\
1510 & 3560 & 4930 & 5700
\end{bmatrix}$$

$$\begin{bmatrix}
1530 & 3560 & 4840 & 5720 \\
1520 & 3560 & 5090 & 5710 \\
1520 & 3560 & 5000 & 5710 \\
1510 & 3560 & 5030 & 5710 \\
1510 & 3560 & 5140 & 5710 \\
1510 & 3550 & 5080 & 5710 \\
1520 & 3550 & 5020 & 5710 \\
1530 & 3560 & 4970 & 5710 \\
1510 & 3560 & 4910 & 5720 \\
1510 & 3560 & 4850 & 5710 \\
1520 & 3560 & 4860 & 5700 \\
1520 & 3560 & 4910 & 5690 \\
1510 & 3570 & 4910 & 5680 \\
1510 & 3570 & 4890 & 5680 \\
1530 & 3570 & 4910 & 5700 \\
1510 & 3570 & 4910 & 5700 \\
1510 & 3570 & 4890 & 5700 \\
1520 & 3560 & 4880 & 5700 \\
1510 & 3560 & 4830 & 5700 \\
1510 & 3560 & 4810 & 5700 \\
1520 & 3560 & 5240 & 5700 \\
1520 & 3560 & 5130 & 5700 \\
1500 & 3560 & 4990 & 5700 \\
1510 & 3560 & 4970 & 5690 \\
1520 & 3570 & 4960 & 5690 \\
1530 & 3550 & 4880 & 5690 \\
1530 & 3550 & 4860 & 5710 \\
1520 & 3550 & 4840 & 5700 \\
1510 & 3560 & 4870 & 5710 \\
1520 & 3570 & 4810 & 5710 \\
1520 & 3570 & 4840 & 5710 \\
1510 & 3560 & 4790 & 5700 \\
1520 & 3550 & 4800 & 5700 \\
1510 & 3550 & 4910 & 5700 \\
1530 & 3560 & 4930 & 5710 \\
1530 & 3550 & 4940 & 5700 \\
1500 & 3560 & 5060 & 5690 \\
1510 & 3560 & 5000 & 5690 \\
1500 & 3560 & 4940 & 5690 \\
1520 & 3550 & 4930 & 5690 \\
1500 & 3550 & 4920 & 5710
\end{bmatrix}$$

附录 B 程序代码

```
# 任务1
def outlier_remove(distance):
    mean_distance = np.mean(distance, axis=0)
    std_distance = np.std(distance, axis=0)

    mask_temp = distance < (mean_distance + 3 * std_distance)
    mask_temp = mask_temp & (distance > (mean_distance - 3 *
        std_distance))
    mask = mask_temp[:, 0] & mask_temp[:, 1] & mask_temp[:, 2] &
        mask_temp[:, 3]

    return distance[mask,:]

def duplicate_samples_remove(distance):
    unique,ori_index =
        np.unique(distance,axis=0,return_index=True)

    sort_index = np.argsort(ori_index)
    unique = unique[sort_index,:]
    return unique

def similar_samples_remove(distance,margin = 20):

    # 计算均值，按照距离均值的距离计算可信度
    mu = np.mean(distance, axis=0)
    ddistance = np.linalg.norm(distance - mu, 2, axis=1)

    index = np.argsort(ddistance)

    reverse_index = np.arange(0,len(distance))

    # 按照可信度将样本进行排序，距离小的样本可信度高
    sort_distance = distance[index, :]

    sort_index = reverse_index[index] # 用于恢复正常顺序

    # 遍历搜索，优先添加可信度高的样本。
    selected_distance = []
    time_index = []
    for d,i in zip(sort_distance,sort_index):
        if selected_distance == []:
            selected_distance.append(d)
            time_index.append(i)

        append_flag = True
        for sd in selected_distance:
            temp = np.linalg.norm(sd - d, 2)
            if temp < margin:
                append_flag = False
                break
```

```
            if append_flag:
                selected_distance.append(d)
                time_index.append(i)

        index = np.argsort(time_index)

        return np.array(selected_distance)[index,:]

# 任务2
import numpy as np
from tqdm import tqdm
import os
from sklearn.metrics import mean_squared_error
from torch import optim
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

def read_data(file_path:str):
    data = pd.read_csv(file_path,index_col=0)
    return data

def preprocess_data(csv_data):
    distance = csv_data[['d1', 'd2', 'd3', 'd4']].values
    is_normal = csv_data['is_normal'].values[0]

    tag_location = csv_data[['x','y','z']].values
    tag_location = np.array(tag_location)

    return distance,tag_location,is_normal

###------SGNN for Maximum Independent Set------###
class SimpleGCN(nn.Module):
    '''
    The GCN layer refers to the paper (Graph Clustering with
        Graph Neural Networks)
    '''

    def __init__(self, in_features, out_features, bias=True):
        super(SimpleGCN, self).__init__()
        self.in_features = in_features
        self.out_features = out_features
        self.weight = Parameter(torch.FloatTensor(in_features,
            out_features))
        if bias:
            self.bias = Parameter(torch.FloatTensor(out_features))
        else:
            self.register_parameter('bias', None)
        self.reset_parameters()

    def reset_parameters(self):
        ### Random ###
        stdv = 1. / math.sqrt(self.weight.size(1))
        self.weight.data.uniform_(-stdv, stdv)
```

```
        if self.bias is not None:
            self.bias.data.uniform_(-stdv, stdv)

    def forward(self, x):
        output = torch.mm(x, self.weight)
        if self.bias is not None:
            return output + self.bias
        else:
            return output

    def __repr__(self):
        return self.__class__.__name__ + ' (' +
            str(self.in_features) + ' -> ' + str(self.out_features)
            + ')'

class EnsembleModels(nn.Module):

    def __init__(self,model_save_root,pentix='best_fine_tine'):
        super(EnsembleModels, self).__init__()

        self.indices = np.array([[0, 1, 2, 3], [0, 1, 2], [0, 1,
            3], [1, 2, 3], [0, 2, 3]])
        self.models = []
        for ind in self.indices:
            self.models.append(torch.load(
            f'{model_save_root}/model_d{ind}_{pentix}.pkl'))

    def forward(self,x):
        output_list = []
        for m,ind in zip(self.models,self.indices):
            output_list.append(m(x[:,ind]))

        return output_list

    def calculate_distance(self,output,anchor_location,trainA):

        loss_list = []
        for T,ind in zip(output,self.indices):
            D = T.unsqueeze(1) - anchor_location[ind, :]
            D = torch.sqrt(torch.sum(D ** 2, dim=2))

            Dif = (D - trainA[:,ind]) ** 2
            loss = torch.mean(torch.sum(Dif, dim=1))

            loss_list.append(loss)

        return loss_list

class UWBLocate(nn.Module):
    def __init__(self,nfeat=3,nhid=10,nout=3,nhid_layer=3):
        super(UWBLocate, self).__init__()
        self.GCN1 = SimpleGCN(nfeat, nhid)
        self.hidLayer = []
        for i in range(nhid_layer):
```

```
            self.hidLayer.append(SimpleGCN(nhid, nhid))
        self.GCN2 = SimpleGCN(nhid, nout)

    def forward(self, x):
        x = F.gelu(self.GCN1(x))
        for hid in self.hidLayer:
            x = F.gelu(hid(x))
        x = F.gelu(self.GCN2(x))
        return x

def train(model, optimizer, A_all,
    anchor_location,epoch=50,indices = np.array([0,1,2,3])):
    loss_list = []
    # for e in tqdm(range(0, epoch), desc='Epoch'):
    for e in (range(0, epoch)):
        trainA = A_all[:,indices]

        T = model(trainA)

        D = T.unsqueeze(1) - anchor_location[indices,:]
        D = torch.sqrt(torch.sum(D ** 2, dim=2))

        Dif = (D - trainA) ** 2
        loss = torch.mean(torch.sum(Dif, dim=1))

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        loss_list.append(loss.item())

    return model,loss_list

def validate(data_root,model_path,is_normal = True):
    if is_normal:
        path = 'cleaned_normal'
        path1 = 'normal'
    else:
        path = 'cleaned_abnormal'
        path1 = 'normal'

    for i in tqdm(range(1, 325), desc=path1):
        file_path = os.path.join(data_root, path, f'{i}.csv')
        data = read_data(file_path)

        distance, tag_location, is_normal = preprocess_data(data)
        tag_location = torch.from_numpy(tag_location)

        A = torch.from_numpy(distance)
        A = A.float()

        model = torch.load(model_path)
        tag_predict = model(A)
```

```
        rmse = torch.sum((tag_predict - tag_location)**2,dim=1)

        relevant_error = torch.abs(tag_predict -
            tag_location)/tag_location
        relevant_error =
            torch.cat((torch.tensor([[0,0,0]]),relevant_error),dim=0)

        location =
            torch.cat((tag_location[0:1,:],tag_predict),dim=0)
        rmse = torch.cat((torch.tensor([0]),rmse),dim=0)


        data = torch.cat((location,relevant_error,rmse.unsqueeze(1
        )),dim=1)

        df = pd.DataFrame(data.detach().numpy(),
        columns=['x','y','z','rx','ry','rz','rmse'])
        df.to_csv(f'data/task2/{path1}/{i}.csv')

def predict_per_target(A,model):
    output = model(A)
    output_npy = torch.cat(output, dim=0).detach().numpy()

    cluster_model = KMeans(n_clusters=2)
    cluster_model.fit(output_npy)

    if np.sum(cluster_model.labels_ == 0) >
        np.sum(cluster_model.labels_ == 1):

        final_cluster = output_npy[cluster_model.labels_ == 0, :]
    else:
        final_cluster = output_npy[cluster_model.labels_ == 1, :]

    mu = np.mean(final_cluster, axis=0)
    std = np.std(final_cluster, axis=0)

    return mu,std

# 任务3
def train_model(search_iter=100):
    model_save_root = 'models'
    data_root = f'data/附件1: UWB数据集'

    os.makedirs(model_save_root,exist_ok=True)

    lr = 0.03
    anchor_location = np.array([[0, 0, 1300], [5000, 0, 1700],
                          [0, 5000, 1700], [5000, 5000, 1300]])
    anchor_location = torch.from_numpy(anchor_location)

    A_all = get_full_dataset(data_root)


    indices =
```

```
        np.array([[0,1,2,3],[0,1,2],[0,1,3],[1,2,3],[0,2,3]])
    for ind in indices:
        loss_abs_best = 1e9

        for i in tqdm(range(search_iter)):

            model_Rprop = UWBLocate(nfeat=len(ind), nhid=32,
                nout=3, nhid_layer=1)
            opt_Rprop = optim.Rprop(model_Rprop.parameters(), lr=lr)
            model, loss_absolute = \
                train(model_Rprop, opt_Rprop, A_all, anchor_location,
                      epoch=50,indices=ind)

            loss_abs = loss_absolute[-1]

            if (loss_abs < loss_abs_best):
                model_best = model
                loss_abs_best = loss_abs
                torch.save(model_best,
                    f'{model_save_root}/model_d{ind}_best.pkl')

                print(f'final loss: {loss_abs_best}')
                print(f'save folder:
                    {model_save_root}/model_d{ind}_best.pkl')

def predict_per_target(A,model):
    output = model(A)
    output_npy = torch.cat(output, dim=0).detach().numpy()

    cluster_model = KMeans(n_clusters=2)
    cluster_model.fit(output_npy)

    if np.sum(cluster_model.labels_ == 0) >
        np.sum(cluster_model.labels_ == 1):

        final_cluster = output_npy[cluster_model.labels_ == 0, :]
    else:
        final_cluster = output_npy[cluster_model.labels_ == 1, :]

    mu = np.mean(final_cluster, axis=0)
    std = np.std(final_cluster, axis=0)

    return mu,std

# 任务4
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn import preprocessing

model_save_root = 'models'
data_root = f'data/附件1：UWB数据集'

ensemblemodel = EnsembleModels(model_save_root)
```

```
normalA = get_full_dataset(data_root,is_normal=True)
abnormalA = get_full_dataset(data_root, is_normal=False)
normalXYZ = ensemblemodel(normalA)
abnormalXYZ = ensemblemodel(abnormalA)

normal_input_list = []
abnormal_input_list = []

anchor_location = np.array([[0, 0, 1300], [5000, 0, 1700],
                     [0, 5000, 1700], [5000, 5000, 1300]])
anchor_location = torch.from_numpy(anchor_location)

D_normal,Dif_normal =
   ensemblemodel.calculate_distance(normalXYZ,anchor_location,
normalA)
D_abnormal,Dif_abnormal =
   ensemblemodel.calculate_distance(abnormalXYZ,anchor_location,
abnormalA)

for (normal,abnormal,dnormal,difnormal,dabnormal,difabnormal,
   ind) in \
      zip(normalXYZ,abnormalXYZ,
         D_normal,Dif_normal,
         D_abnormal,Dif_abnormal,
         ensemblemodel.indices):

   normal_input_list.append(

   torch.cat((normalA[:,ind],dnormal,difnormal),dim=1))

   abnormal_input_list.append(

   torch.cat((abnormalA[:,ind],dabnormal,difabnormal),dim=1))

# d1-d4 d_hat_1-d_hat_4 dif1-dif4
train_data =
   torch.cat((normal_input_list[0],abnormal_input_list[0]),dim=0)
train_label = torch.cat((
   torch.ones((normal_input_list[0].shape[0],)),
   torch.zeros((abnormal_input_list[0].shape[0],))),dim=0)


train_data_np = train_data.detach().numpy()
train_label_np = train_label.detach().numpy()

normalizer = preprocessing.Normalizer().fit(X=train_data_np)
norm_train_data_np = normalizer.transform(train_data_np)

task4_A = [[2940,4290,2840,4190],
        [5240,5360,2040,2940],
        [4800,2610,4750,2550],
        [5010,4120,3810,2020],
        [2840,4490,2860,4190],
        [5010,5320,1990,2930],
```

```
        [5050,3740,3710,2070],
        [5050,4110,3710,2110],
        [4840,2600,4960,2700],
        [2740,2720,4670,4790]]
task4_A = torch.tensor(task4_A).float()

model_save_root = 'models'
data_root = f'data/附件1: UWB数据集'

ensemblemodel = EnsembleModels(model_save_root)
XYZ = ensemblemodel(task4_A)

input_list = []

anchor_location = np.array([[0, 0, 1300], [5000, 0, 1700],
                    [0, 5000, 1700], [5000, 5000, 1300]])
anchor_location = torch.from_numpy(anchor_location)

D,Dif =
   ensemblemodel.calculate_distance(XYZ,anchor_location,task4_A)

for (xyz,d,dif, ind) in \
      zip(XYZ,D,Dif,ensemblemodel.indices):
   sum_feature =
      torch.cat((torch.sum(task4_A[:,ind],dim=1,keepdim=True),
           torch.sum(d,dim=1,keepdim=True),
           torch.sum(dif,dim=1,keepdim=True)),dim=1)
   input_list.append(torch.cat((task4_A[:,ind],d,dif,sum_feature)

   ,dim=1))


# d1-d4 d_hat_1-d_hat_4 dif1-dif4
test_data = input_list[0].detach().numpy()

test_data = normalizer.transform(test_data)


from xgboost import XGBClassifier

xgboostC = XGBClassifier(n_estimators=800,max_depth=7)

predict_result_list = []
score_list = []
for k,(train_index,validate_index) in
   enumerate(kf.split(norm_train_data_np)):
   X_train = norm_train_data_np[train_index,:]
   X_validate = norm_train_data_np[validate_index,:]

   y_train = train_label_np[train_index]
   y_validate = train_label_np[validate_index]

   xgboostC.fit(X_train,y_train)
```

```
    score = xgboostC.score(X_test,y_test)
    score_list.append(score)
    predict_result = xgboostC.predict(test_data)
    predict_result_list.append(predict_result)
    print(70*'-')

    print(f'{k}: validate score:{score}')
    print(f'predict result(test): {predict_result}')


# 任务5
with open(f'data/附件5：动态轨迹数据.txt') as f:
    data = f.read()


data_per_line = data.split('\n')

distance = []
# 首行、尾行包含无效信息
for l in data_per_line[1:-1]:
    d = l.split(':')
    distance.append(int(d[5]))

distance = np.array(distance)

distance1 = distance[0::4]
distance2 = distance[1::4]
distance3 = distance[2::4]
distance4 = distance[3::4]

data = np.stack((distance1,distance2,distance3,distance4),axis=1)

df_task5 = pd.DataFrame(data,columns=['d1','d2','d3','d4'])
df_task5.to_csv('results_per_task/task5/data.csv')

A = np.stack((distance1,distance2,distance3,distance4),axis=1)
A = torch.from_numpy(A).float()

model_save_root = 'models'
ensemblemodel = EnsembleModels(model_save_root)

XYZ = ensemblemodel(A)

from pykalman import KalmanFilter

kf = KalmanFilter(initial_state_mean=choose_data[0,:],
                  n_dim_state=3,n_dim_obs=3)
kf.em(XYZ)

(filter_mu,filter_covar) = kf.filter(choose_data)
(smooth_mu,smooth_covar) = kf.smooth(choose_data)
```