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	
参赛队号	
队员姓名	1.
	2.
	3.

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 空气质量预报二次建模

摘 要：

大气污染危害人体健康、破坏生态环境，精准预测污染物浓度并及时采取有效措施可以大幅减少其危害。然而，现有空气质量预报模型的预测结果并不理想，因此对于空气质量预报进行二次建模拥有重要意义。

问题一：按照已有方法计算指定日期的 AQI 和首要污染物。首先，检查计算所需原始数据是否符合格式类型规范和问题要求规范。其次，依据附录公式计算每日 IAQI（各污染物空气质量分指数）。接着，选取每日最大 IAQI 作为当日 AQI（空气质量指数）。然后，确定当日空气质量等级和当日首要污染物。最终，得到 8 月 25 日到 8 月 28 日的{AQI，首要污染物}依次为{60，O_3}、{46，无}、{109，O_3}、{138，O_3}。

问题二：根据气象条件对天气质量的影响，现将气象条件合理分类。本文建立了**监督学习初分类—无监督学习再聚类模型**。考虑到聚类往往需要指定类别数目，因此本文采用**监督学习初分类**，获取气象条件**大致分类数目**。首先，对数据进行缺失值处理、异常值处理、去量纲化等**数据预处理**操作。其次，将有效分数向量模型、布朗过程、一阶指数平滑模型三种模型组合使用，**提取出影响 AQI 的气象事件**，并将每一气象事件表示成二维表结构的数据集以便后续使用。接着，依据现存气象条件划分别标准或 Fisher 线性判别模型对每一**气象条件因素**进行**等级划分**。然后，通过核函数、同源生成概率对划分等级后的气象条件因素进行计算，以此来衡量各气象事件的相似性，将各**相似气象事件合并**。通过以上**监督学习初分类**，获得气象条件大致可分为 8 类左右，接下来按照 6 类、7 类、8 类、9 类、10 类进行**无监督学习再聚类**。在此使用深度注意力嵌入聚类模型，其特征是可以在聚类样本中嵌入样本属性，这样便于将污染物浓度和气象条件两类不同指标放在一起聚类。本文还设计了一个聚类效果评估函数，经过计算将气象条件分为八类是最佳的分类方案。最终，所获得的详细气象条件分类情况请参见正文部分。

问题三：对于空气质量预报二次建模，预测指定日期的空气质量情况。本文分别建立了**大气过程动力学—卡尔曼滤波模型**和 **NARX 神经网络模型**来预测，并且使用后者来修正前者以获得更为精确的结果。对于**大气过程动力学—卡尔曼滤波模型**，本文引入污染传输矩阵表达化学过程引导的污染传送模式，在污染传送模式的作用下，污染物浓度次态表

达为污染物浓度现态与排放清单现态的线性组合，在一阶 Markov 条件下，将原大气过程动力学问题转化为卡尔曼滤波模型进行求解。对于**NARX 神经网络模型**，本文将神经网络结构设计为输入层节点数为 3，滞后阶数为 1，隐含层数为 1，隐含层节点数为 12。将一个预测周期（3 天）所有相同时刻的 6 种污染物浓度一次预报值和 5 种气象条件因素作为输入，将预测时刻的 6 种污染物浓度实测值作为输出，选取约 68%的有效数据作为训练集，另 32%的有效数据作为测试集，选取贝叶斯正则化算法训练神经网络，选取交叉熵损失函数计算损失函数值。获得预测结果后根据其相关评价指标对**大气过程动力学—卡尔曼滤波模型**的预测结果进行修正。最终，获得更为精确的检测点 A、B、C 在 7 月 13 日到 7 月 15 日空气质量情况，具体结果请参见正文部分。

问题四：探究增添三个监测点进行区域协同预报是否会提高空气质量预测的准确度。本文建立了**基于图卷积神经网络的区域协同预测模型**来预测。首先，采用迭代法将坐标图平面坐标划分成不同区域网格，将网格作为图的节点，构建拓扑结构。然后，综合考虑风速，风向，距离和浓度差等物理因素构建大气动力学模型，对各个节点的连接权重进行初始化。接着，构建图卷积神经网络，将一次预报污染物浓度和初始化的拓扑结构作为输入，将修正的一次预报污染物浓度作为输出，输入到问题 3 的预测模型，进行反向传播。随后，更新拓扑结构和图卷积神经网络的参数，直到参数更新完成。最后，可以得到基于**基于图卷积神经网络的区域协同预测模型预测**的空气质量状况。相较于问题 3 的预测模型，加入区域协同预测后的预测模型拥有更高的准确率：对监测点 A 的误差较之前相比减小了 42.37%，具体结果请参见正文部分。

回顾整个建模流程，总结出本文共有**四大创新**：

①解决问题二时，本文将天气条件分类转换成对于气象条件成因相同概率的衡量，即将数据集（气象条件）的合并转换成对于数据集的生成概率（气象条件成因）计算，利用核函数和隐变量进行变换，然后合并生成概率值等大范围的数据集。

②解决问题二时，本文先用监督学习方法进行初分类，获得了大致气象条件分类的分类数量，之后在该分类数量附近做多种聚类方案，避免盲目聚类。

③解决问题三时，本文建立了以知识驱动的大气过程动力学—卡尔曼滤波模型和以数据驱动的神经网络模型，并用神经网络模型修正气过程动力学—卡尔曼滤波模型结果，综合的大模型由知识、数据双驱动，提升模型效果、增强可解释性。

④解决问题四时，本文建立了区域协同预测模型，将地理位置信息，风向，风速以及浓度差等信息全都考虑进去，然后用图卷积神经网络对该过程进行修正。是一个利用人工知识为初始条件，数据驱动为后续调整的模型。提出了一种人工知识迁移的方法。

最后，本文对模型进行了评价、改进和推广。

关键词：空气质量预测　气象条件分类　卡尔曼滤波　神经网络　协同预测

目录

1. 问题重述

1.1 问题背景

近年来，由于快速的城市化和工业化，空气质量已成为重要的环境和健康问题。空气质量对人们日常生活的具有潜在影响，准确地预测空气质量具有重大的现实意义。然而，空气质量系统是一个高噪声耦合，高非线性和强时变的复杂巨系统，这给空气质量预测带来了巨大的挑战性。

事实上，由于空气污染物数量众多，且可主动或被动结合形成二次污染物来源，各国都采用 WRF-CMAQ 模拟体系和空气质量指数(AQI)来预报空气质量。然而，受制于气象条件模拟困难，排放清单不确定和空气污染物转化路径不明晰等现实问题，通常使用气象实测数据对已有的空气质量预测模型进行校正，即二次建模。实测数据具有更强的可靠性和鲁棒性，更能够反应真实的空气质量情况。因此，利用实测数据驱动的二次建模技术，对发现潜在的首要污染物，分析各类气象条件对于污染形成和扩散的作用机制，建立高效且鲁棒的空气质量预测模型，进而构建高度协同的区域联合预报机制具有重大理论价值和实践支撑。

1.2 问题提出

问题一：按给定方法计算数值

题目已给出各污染物空气质量分指数(IAQI)和空气质量指数(AQI)的计算方法以及空气质量等级和首要污染物的判定方法，现要求根据已知的 2020 年 8 月 25 日到 8 月 28 日每天的气象条件实测值，计算这四天的 AQI 和首要污染物，将结果以表格形式放入正文中。

问题二：气象条件分类

气象条件会对地区内的污染物排放情况会产生影响。利用题目中给出的附件 1 的数据，来对气象条件进行合理的分类。判断哪些气象条件有利于污染物的扩散和沉降，造成该地区的 AQI 下降，哪些气象条件不利于污染物的扩散和沉降，造成该地区的 AQI 上升。并阐述各类气象条件的特征。

问题三：二次建模预测空气质量

由于一次预报采用的 WRF-CMAQ 模型难以精确的预测臭氧浓度的变化，所以题目要求我们建立一个可以用来预测未来三天 6 种常规污染物单日浓度值的二次预报模型，该二次预报模型同时适用于 A、B、C 三个监测点，且要求该模型对 AQI 的预报值和首要污染物的预测情况足够准确。利用该模型预测出 A、B、C 三个监测点在 2021 年 7 月 13 日至 7 月 15 日 6 种常规污染物的单日浓度值，AQI 和首要污染物。并将结果按照附录要求的格式展示在论文中。

问题四：区域协同预测对预测准确性的影响

题目中给出了 A、A1、A2、A3 四个相邻的区域的信息，包括它们的位置坐标，各自的气象条件，污染物浓度等信息。现要求利用这四个相邻区域的信息建立一个模型，该模型要求 AQI 的预报值和首要污染物的预测准确度尽量高，且能依据该模型来判断邻近区域的信息能否有效提高预测准确性。利用该模型预测出 A、A1、A2、A3 三个监测点在 2021 年 7 月 13 日至 7 月 15 日 6 种常规污染物的单日浓度值，AQI 和首要污染物。并将结果按照附录要求的格式展示在论文中。

2. 研究总述

2.1 问题分析

本题主要有以下四个子问题：

问题一，需要根据题目给出的式子进行指定日期的空气质量情况的计算。在计算之前应该考虑是否所需要使用的数据会存在缺失、错误、异常等问题，另外还需要考虑所需使用数据是否满足题目要求，比如考虑到问题要求臭氧（O_3）最大 8 小时滑动平均浓度值高于 800 $\mu g/m^3$ 的情况，或其余污染物浓度高于 AQI=500 对应限值时，都不再进行其空气质量分指数计算，需要对所有实测值进行数据预处理和筛选。所以该题目在计算的时候需要先做一下数据预处理。然后，根据问题指定计算方法，依次计算 IAQI 和 AQI 指标。最后，根据 AQI-空气质量等级阈值表进行判断并根据 IAQI 的最大值确定首要污染物。

问题二，需要根据题目所给出的 Excel 表格里的数据，找出使得 AQI 发生较大变化的气象数据，并且将这些气象条件进行分类，并且描述一下每一类气象条件的特征，以及分辨一下那些类的气象条件是有利于污染物扩散或沉降的，哪些类的气象条件是不利于污染物扩散或沉降的。针对这一问题，我们需要考虑如何从 AQI 的变化下手，找出这些使其发生变化的气象条件，再将其进行分类。同时，还要考虑这么大规模的数据，是否每一个数据都完整、干净、正常呢？这里还是面临着数据预处理的问题，首先，是否表格内的数据有空缺。如果有哪些是可以填充后保留使用的，哪些是无法填充的需要剔除的。其次，对于保留下来的数据，填充应该怎么填充。然后，当所有数据都完整后，还应该再考虑是否有数据是异常的，比如有的百分比出现的负数，根据实际情况我们都知道这是不可能发生的，比如有的数据是离群数据，这种的应该如何处理，这些都是我们应该思考的问题。数据预处理过后提取数据气象条件，这里应该如何从繁杂的数据中尽可能简练地提取出气象条件。之后是气象条件的分类，一般思想是直接聚类，可是聚类往往需要指定聚类的类数，对于该题气象条件分几类我们是不知道的，为了避免盲目聚类，是否可以考虑通过某种方式大致知道可以聚几类。然后在聚类的大致类数附近做出多个聚类方案。最后多个聚类方案又该如何确定聚类效果最好的方案，这些都是我们需要探讨的问题。

问题三，需要根据题目所给出的 Excel 表格里的数据，对现有的一次预报进行二次建模，使得新的空气质量情况预测模型能更加精准地对空气质量状况进行预测，并且还需要对 A、B、C 三个监测点在 2021 年 7 月 13 日至 7 月 15 日 6 种常规污染物的单日浓度值，计算相应的 AQI 和首要污染物。建模思路主要分为两种。第一种是考虑建立以知识驱动的模型，该类模型的建立需要有比较扎实的物理基础、气象学基础、数学基础。第二种是考虑建立以数据驱动的模型，该类模型的建立需要有比较扎实的计算机基础和数学基础。对于以知识驱动的模型，要从现实机理出发，分析原来的一次预报模型有什么问题，为什么会导致预测不准确，在该模型基础上应该如何利用已知条件或者是数据进行建模，比如各种气象条件是如何变化的，这种变化导致了什么状况的发生等等。以数据驱动的模型也可以用来解决本问题，因为本问题提供了大量的数据。不过数据驱动的模型由于属于黑盒问题，虽然有结果，但是不清楚过程是怎样的，因此使用数据驱动模型写数学式、模型式是比较困难的。本文可以考虑使用两种模型组合起来的综合模型，这样可以使两个子模型相互弥补对方缺点，共同发挥优势，从而提高模型的性能。

问题四，需要根据题目所给出的 Excel 表格里的数据，在问题三模型的基础上，基于

监测点 A1、A2、A3 的实测数据构建针对监测点 A 的协同预报模型，并分析协同预报模型对于单点预测模型的性能。协同预测是添加多个监测点与原有监测点实行数据共享、共同预测的方式。这种方式可以保证在某个监测点发生情况或者是监测数据异常的时候，其他监测点可以帮助修复该监测点的数据。因此想办法找出被修复监测点与其他监测点测试数据之间的关系。进一步可以利用其他监测点的一次预报数据拟合出被修复点的全新的一次预报数据，代入模型三中进行预测，根据相关性能指标讨论协同预测的方式是否能提高空气质量预测准确度。

2.2 数据—模型流程总图

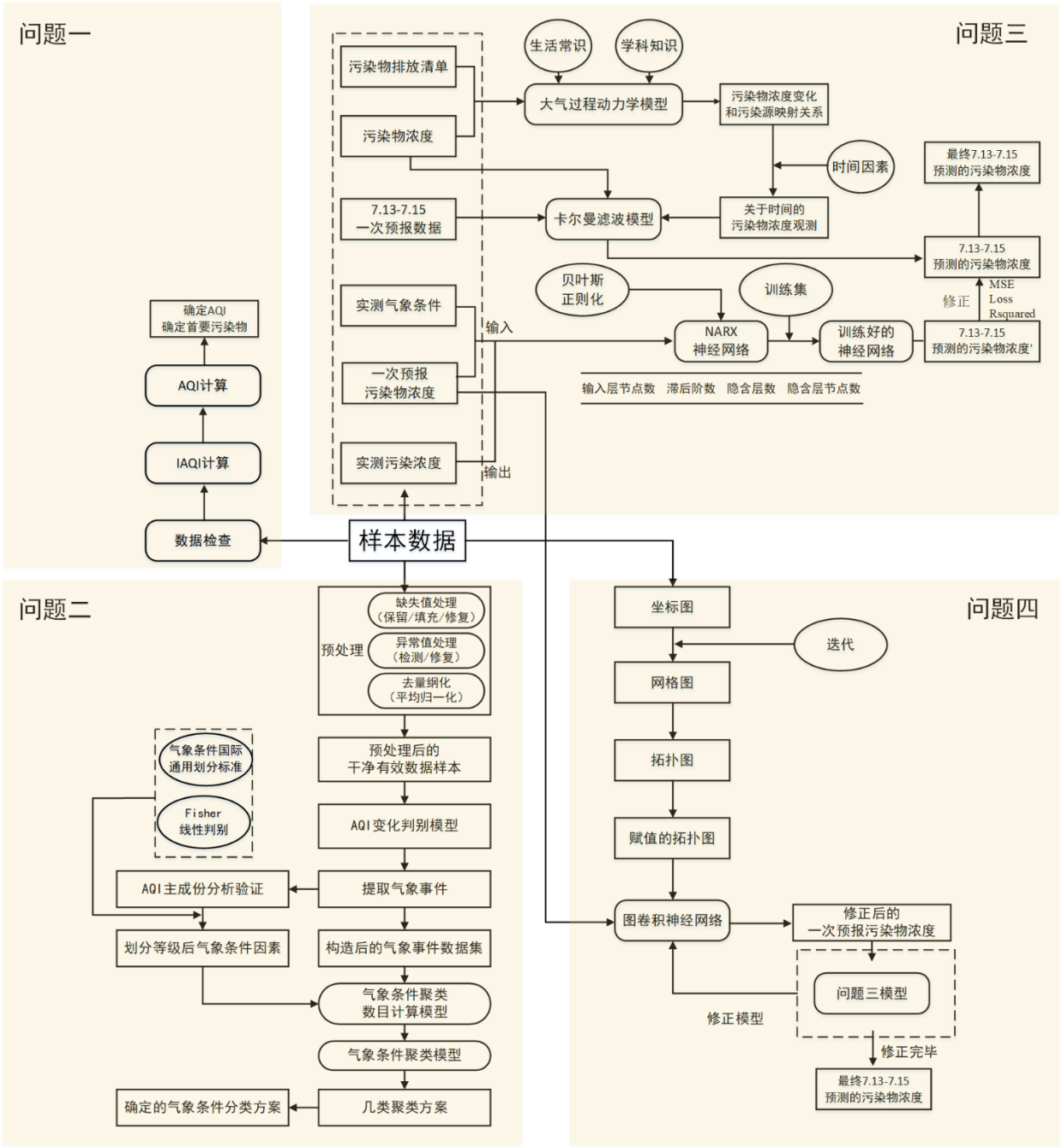


图 1 数据—模型流程总图

3. 模型假设

假设一：站点气象监测设备在数据记录期间的性能不变。
假设二：不考虑除气象条件外的因素对于污染物浓度的影响。
假设三：不考虑地形地势等地理因素对地区之间污染物浓度的影响。
假设四：大气动力学过程可视为一阶 Markov 过程。
假设五：污染物浓度的先验概率服从正态分布。

4. 符号说明

符号	意义
AQI	空气质量指数
$IAQI$	空气质量分指数
$\boldsymbol{C}$	各组分污染浓度的向量
$\boldsymbol{M}$	反应机理引导的污染传送模式矩阵
$\boldsymbol{Q}^{b}$	污染源相关的污染物清单向量
$\boldsymbol{q}$	污染传送模式的误差向量
$\boldsymbol{H}$	污染物观测模式矩阵
$\boldsymbol{r}$	污染物观测模式的误差向量
t_k	污染物的观测时刻
$\boldsymbol{y}^{(j)}$	气象实测数据的观测值向量
$\boldsymbol{C}^{a}$	各组分污染浓度的清单向量的预测
$\boldsymbol{Q}^{a}$	污染源相关的污染物清单的预测

5. 问题一的建模与求解

5.1 问题一的解析

本问题的目的是根据附录给定的计算方法计算 2020 年 8 月 25 日到 8 月 28 日每天实测的 AQI 和对应的首要污染物。该问题的解决主要可以分为以下两个步骤：

步骤一：数据预处理与筛选，包括缺失值处理，异常值处理和数据筛选；

步骤二：AQI 数值计算和首要污染物判断，包括以下三个子步骤：

子步骤 1：计算 IAQI；

子步骤 2：计算 AQI；

子步骤 3：根据阈值表进行空气质量等级判断并根据 IAQI 确定首要污染物。

本问题的处理流程可以用如下流程图所示的方式表达：

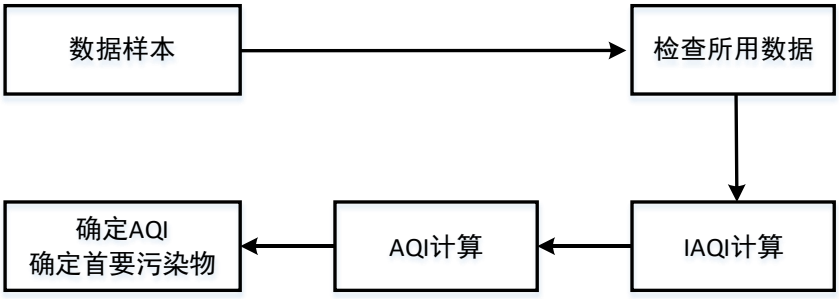


图 2 问题一的处理流程图

5.2 数据的预处理与筛选

本问题的数据来源于某监测点长期空气质量预报基础数据。数据包括污染物浓度一次预报数据、气象一次预报数据、气象实测数据和污染物浓度实测数据。一次预报数据的时间跨度约为一年，即 2020 年 7 月 23 日到 2021 年 7 月 13 日。实测数据的时间跨度为 2019 年 4 月 16 日到 2021 年 7 月 13 日，数据总量在十万量级。本问题对用于计算的数据提出了以下两点要求：

① 臭氧（O_3）最大 8 小时滑动平均浓度值高于 800 $\mu g/m^3$ 的，不再进行其空气质量分指数计算。

② 其余污染物浓度高于 IAQI =500 对应限值时，不再进行其空气质量分指数计算。

经检查，用于计算的数据中臭氧（O_3）不包含最大 8 小时滑动平均浓度值高于 800 $\mu g/m^3$ 的情况；也不存在 IAQI ＞500 的情况。

除了以上两点要求外，还检查了缺失值和离群值，满足计算需求。因此，本问题的数据可以直接用于 AQI 和对应的首要污染物的计算。

5.3 计算模型

根据《环境空气质量指数（AQI）技术规定（试行）》（HJ633-2012），空气质量指数（AQI）可用于判别空气质量等级。首先需得到各项污染物的空气质量分指数（IAQI），其计算公式如下：

$$\mathrm{IAQI_P}=\frac{\mathrm{IAQI_{Hi}}-\mathrm{IAQI_{LO}}}{\mathrm{BP_{Hi}}-\mathrm{BP_{Lo}}}\cdot\left(\mathrm{C_P}-\mathrm{BP_{Lo}}\right)+\mathrm{IAQI_{Lo}} \tag{5.1}$$

其中，IAQI_P 是污染物 P 的空气质量分指数（进位取整）；C_P 是污染物 P 的质量浓度值；BP_{Hi} 和 BP_{Lo} 分别是与 C_P 相近的污染物浓度限值的高位值与低位值；IAQI_{Hi} 和 IAQI_{Lo} 分别是与 BP_{Hi} 和 BP_{Lo} 对应的空气质量分指数。

空气质量指数（AQI）取各分指数中的最大值，即

$$\mathrm{AQI}=\max\left\{\mathrm{IAQI_1},\mathrm{IAQI_2},\mathrm{IAQI_3},\ldots,\mathrm{IAQI}_n\right\} \tag{5.2}$$

空气质量等级与空气质量指数 AQI 的对应关系如表 X 所示：

表 1 空气质量等级及对应空气质量指数（AQI）范围

空气质量等级	优	良	轻度污染	中度污染	重度污染	严重污染
AQI 范围	[0,50]	[51,100]	[101,150]	[151,200]	[201,300]	[301,+∞)

5.4 结果和分析

根据如 5.3 节所述的计算过程，得到监测点 A 在 2020 年 8 月 25 日到 8 月 28 日 AQI 计算结果，如表 X 所示。其中，8 月 25 日到 8 月 28 日的 AQI 值分别为 60，46，109 和 138，即空气质量对应为良，优，轻度污染和轻度污染。除对应空气质量为优的 8 月 26 日，其余三天的首要污染物均为臭氧 $\mathrm{O_3}$。

表 2　2020 年 8 月 25 日到 8 月 28 日 AQI 计算结果表

监测日期	地点	AQI 计算	
		AQI	首要污染物
2020/8/25	监测点 A	60（良）	$\mathrm{O_3}$
2020/8/26	监测点 A	46（优）	无
2020/8/27	监测点 A	109（轻度污染）	$\mathrm{O_3}$
2020/8/28	监测点 A	138（轻度污染）	$\mathrm{O_3}$

依据模型计算结果，有以下结果分析：

（1）臭氧 $\mathrm{O_3}$ 是与大气中其他组分（如氮氧化物 NO_x 和挥发性有机物 VOC_s）发生化学或光化学反应而产生的二次污染物，与光照强度相关。现有研究表明，臭氧污染具有明显的时间规律特征，超标时段集中于高温强日晒的午后至傍晚。监测点 A 在 8 月 25 日到 8 月 28 日为夏季时段，紫外线照射一般偏强，由此导致臭氧 $\mathrm{O_3}$ 更易在夏季成为首要污染物。这与计算所得到的结论是一致的。

（2）挥发性有机物 VOC_s 位于生成臭氧 $\mathrm{O_3}$ 的反应链条前端，其在大气组分中的浓度受到温度，湿度和风速的影响。根据实测数据，8 月 25 日傍晚（18 时）监测点 A 附近的

风速从 2.2m/s 陡升到 3m/s。同时，较高的风速会降低挥发性有机物 VOC_s 在大气组分中的浓度。由此，8 月 26 日的空气质量将有好转的倾向。根据计算数据，空气质量在 8 月 26 日相应地转为优。在 8 月 26 日夜间，风速快速地低降到 0.4m/s，在其他条件不变的情况下，空气质量倾向于变差。根据计算数据，空气质量在 8 月 27 日相应地转为轻度污染。

（3）基于以上两点，以实测数据驱动进行天气质量预报的二次建模具有较高的可信度和鲁棒性，且可以帮助发现各类潜在的空气质量关联性和模式。

6. 问题二的建模与求解

6.1 问题二的解析

问题二的目标是对影响 AQI 的各类气象因素进行分类，并阐释其潜在特征。该问题的解决主要可以分为以下五个步骤：

步骤一：数据预处理，包括 ①缺失值处理 ②异常值处理 ③去量纲归一化；

步骤二：气象事件提取，包括 ①QAQI 变化判别模型 ②气象事件数据集构造 ③关于 AQI 的气象事件主成分析 ④气象条件等级划分；

步骤三：气象条件聚类，包括 ①气象条件聚类数目计算 ②气象条件聚类模型；

步骤四：评估多个聚类数目下的类内相似度和类间距离性能；

步骤五：并分析所得类别的潜在特征。

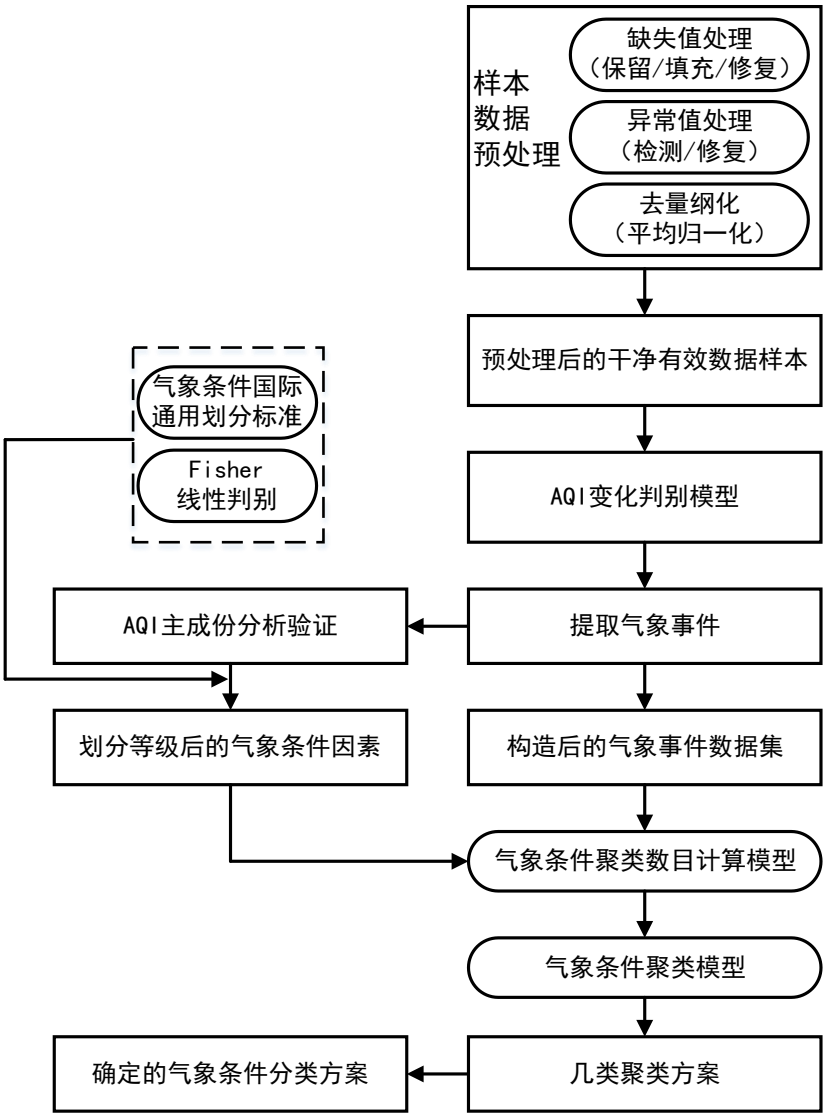


图 3 问题二的处理流程图

6.2 数据预处理

空气质量系统是一个高噪声耦合，高非线性和强时变的复杂巨系统，这给高精度空气质量数据记录和预报工作带来了巨大挑战。其中，空气质量监测站点实际监测得到的数据（实测数据）和 WRF-CMAQ 模型运行产生的数据（一次数据）往往直接影响到基于数据驱动的空气质量预报性能。在本题中，出于利用实测数据校正一次数据预报模型的目的，需要对实测数据进行预处理。也即通过缺失值处理，异常数据处理和去量纲归一化来提升二次建模的鲁棒性。数据预处理过程可以用如下的流程图表达。

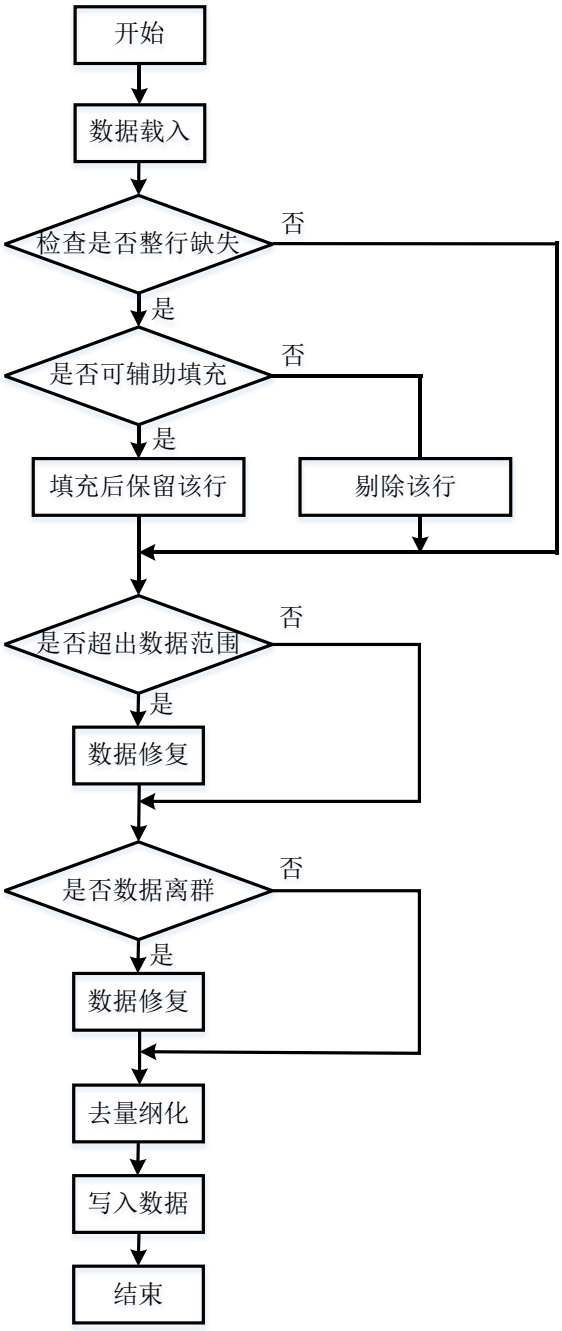


图 4 问题二的数据预处理流程图

6.2.1 缺失值处理

在天气质量预报过程中，由于监测站点设备调试、维护、停电和站点监测指标差异等情况影响，将导致部分时间出现连续性或间断性的变量值缺失为空值。

原则（1）：对于连续性变量缺失，如监测点 A 逐小时污染物浓度与气象实测数据在 2019 年 5 月 6 日 21 时到 2019 年 5 月 7 日 9 时所记录，所有气象指标数据全部缺失。针对此类问题，考虑以下两种处理方式 ①根据一次数据填充缺失实测数据，并且保留该条目；②无法通过一次数据填充的，删除该条目。

原则（2）：对于间断性变量缺失，如监测点 A 逐小时污染物浓度与气象实测数据在 2019 年 4 月 23 日 13 时到 2019 年 4 月 23 日 17 时所记录，部分气象指标数据缺失。针对此类问题，对于这种情况主要采取利用平均化的方式来处理。然而对本问题而言，某自然日内空气质量预测的最大窗口时间为 8 小时，时间颗粒度为 1 时。因此，首先考虑用一次数据填充，再考虑将连续 4 小时以上空缺的实测数据删除。定义站点实测数据为序列 $D_n=\left\{I_n^1,I_n^2,\cdots,I_n^{\ i},\cdots\right\}(i=1,2,3,\cdots)$，其中，$I_n^{it}$ 为第 n 项气象指标在 it 时刻的实测数据值。根据上述原则（1）和原则（2），则可定义如下所示的缺失值判据：

$$\left|I_n^{it}-I_n^{(i-1)t}\right|\geq 4 \tag{6.1}$$

如前所述，设定为可接受的时间上限阈值 Tr，超出阈值的首先考虑利用一次数据填充，此时将一次数据填充标志位 F 置位，即 $F=1$。如果一次数据不能填充，即 $F=0$，考虑到天气系统可视为简化的一阶大惯性环节，可将空缺前后 N 小时（共 $2N$ 小时）的平均值作为填充数据。定义 $i(t-1)$ 到 it 时刻之间缺失值个数与数值的二元组为 $\left(k,s_n^{(i-1)t}\right)$，记任意两个数据的时间间隔为 T，则满足**原则（1）和原则（2）**的缺失值处理如下：

$$\left(k,s_n^{(i-1)t}\right)\Big|_{F=0}=\begin{cases}(0,\sim),T<Tr\\ \left(\dfrac{T}{4},\Omega\left(\sum\limits_{j=i\text{-}N}^{i\text{-}1}I^{j}{}_{n}+\sum\limits_{j=i}^{i+N\text{-}1}I^{j}{}_{n}\right)\right),Tr\leqslant T\leqslant 4\\ \text{delete}，\ T>4\end{cases} \tag{6.2}$$

$$\left(k,s_n^{(i-1)t}\right)\Big|_{F=1}=\left(0，\ \Delta\right) \tag{6.3}$$

其中，$\Omega(\bullet)$ 为算数平均算子，Δ 是代表一次数据填充实测数据的算子。

6.2.2 异常值处理

受监测站点及其附近的自然和人为因素的影响，实测数据会出现异常值，主要体现在①数据超出了合理范围，不再具有合理的物理意义；②偏离数据正常分布。

对于①类情况，定义集合的映射 Γ_1：$I_{\ n}^{*}\leftarrow I_n$ 和第 n 项气象指标的下物理边界和上物理边界对应的区间为 $\left(B_n^L,B_n^H\right)$，则 $\forall I_n^{\ it}\in I_n$，有以下的操作：

$$\Gamma_1\left(I_n\right)=I_n^*=\begin{cases}I_n,B_n^L\leq I_n^{it}\leq B_n^H\\ I_n-\left\{I_n^{it}\right\},I_n^{it}\leq B_n^L or I_n^{it}\geq B_n^H\end{cases} \tag{6.4}$$

这样的操作确保了湿度不会超过 100%，风速和气压不会出现负值，风向不会超过360°，即对于不满足物理含义的气象指标数据，进行删除操作。

对于②类情况，当变量值超出3σ区间范围时，可以认为站点对应时间的实测数据为离群值。因此，根据3σ准则，考虑将这样的跳跃值剔除出去。即在利用实测数据进行空气质量二次建模的过程中，对这些在3σ范围外的数据不予考虑。定义集合的映射Γ_2：$I_n^* \leftarrow I_n$进行离群值处理如下：

$$I_n^{it} = \frac{1}{S}\sum_{S}^{i=1} I_n^{it} \tag{6.5}$$

$$\sigma = \sqrt{\frac{1}{S-1}\sum_{S}^{i=1}\left(I_n^{it} - \mu(I_n^{it})\right)^2} \tag{6.6}$$

$$\Gamma_2\left(I_n\right) = I_n^* = \begin{cases} I_n, I_n - 3\sigma \le I_n^{it} \le I_n + 3\sigma \\ I_n - \left\{I_n^{it}\right\}, otherwise \end{cases} \tag{6.7}$$

6.2.3 去量纲化

对于量纲不同的实测数据，一般取值范围也会存在较大的差异，这会加剧数据本身的波动性并导致模型的可用性（收敛速度）和鲁棒性（模型精度）衰减。因此，考虑进行均值归一化进行去量纲处理，则第n项气象指标在it时刻的实测数据值I_n^{it}的均值归一化为：

$$Nom(I_n^{it}) = \frac{I_n^{it} - \mu(I_n^{it})}{\max(I_n^{it}) - \min(I_n^{it})} \tag{6.8}$$

其中，$\mu(\bullet)$是均值算子。

6.3 气象事件提取

6.3.1 气象事件的识别

根据题目可知，空气污染物会根据气象条件的不同而扩散或沉降。根据生活常识，大风、大雨会让空气中的污染物浓度降低，AQI 下降。所以，想要将气象条件进行分类，就需要先将有代表性的气象事件进行提取。

本文依照 AQI 指数的变化进行气象事件的判定和提取。对于一个完整的气象事件，需要确定其起始时刻和结束时刻。

① 气象事件开始时刻判定

在这里，将 AQI 发生骤变的时间点当做一个气象事件的开始时刻，我们可以利用传统的基于有效分数向量（ESV）对数据样本进行骤变点的检测。

设$x_1, x_2, ...x_n$是一组待检测骤变关于 AQI 的时间序列，其密度函数为$f(x;\theta,\eta)$，其中$\theta \in \Omega_1 \subset R^d$，$d \geqslant 1$的时候表示算法中的“interest”的参数变量，$\eta \in \Omega_1 \subset R^p$，$p \geqslant 0$表示算法中的“uninterest”的参数变量，$\Omega=\Omega_1 \times \Omega_2$为参数变量所在空间。

现假设：$H_0:\theta=\theta_0$，对于所有观测值η未知；H_A：对于$x_1,x_2,...x_{\tau\text{-}1}$，满足$f(x;\theta_0,\eta)$，$\eta$未知,对于$x_\tau,x_{\tau+1},...x_n$，满足$f(x;\theta_A,\eta)$，$\theta_A$未知；其中$\theta_0$为骤变点发生前序列服从分布$f$中的参数，$\theta_A$为骤变点发生后序列服从分布$f$中的参数，$\tau$为骤变点所在时刻，则 ESV 定义表达式为：

$$V_k(\theta,\eta)=\sum_{i=1}^{k}\nabla_\xi \log f(x_i;\theta,\eta),\xi=(\theta,\eta) \tag{6.9}$$

其中，uninterest 参数变量η的估计值$\hat{\eta}$由以下式子进行计算：

$$\sum_{i=1}^{k}\nabla_\eta \log f(x_i;\theta_0,\eta)=0 \tag{6.10}$$

所以，ESV 定义表达式的变式为：

$$V_k(\theta_0,\hat{\eta}_k)=\sum_{i=1}^{k}\nabla_\xi \log f(x_i;\theta,\hat{\eta}_k)=\sum_{i=1}^{k}\nabla_\theta \log f(x_i;\theta_0,\hat{\eta}_k) \tag{6.11}$$

由该式可以看出，当假设H_A为真是，ESV 时间序列V_k具有零均值，当假设H_A为真时，ESV 时间序列V_k的值将会逐渐变大，而且其变化程度会随着骤变点后的数据增多呈线型比例增大。

假设存在某一过程$W(k)$满足：

$$\|W_k-W(k)\|=\sigma(t^{1/(2+\delta)}) \tag{6.12}$$

称这个过程为布朗过程，其中$W(k)=(W^1(k),W^2(k),...,W^d(k))$。

对于时间序列中，若出现

$$k^{-\frac{1}{2}}W_k \geqslant C_1(\alpha) \tag{6.13}$$

这说明，时间序列在k处出现骤变点，其中$\alpha=[\alpha^1,\alpha^2,...,\alpha^i,...,\alpha^d]$为置信度向量，该向量一般取值 0.05$C_1(\alpha)=[C_1^1(\alpha^1),C_1^2(\alpha^2),...,C_1^i(\alpha^i),...,C_1^d(\alpha^d)]$。

以上是通过基于传统的 ESV 算法检测骤变点的过程，通过该方法，我们可以找出所有的 AQI 骤变点，识别出气象事件的初始时刻。

② 气象事件结束时刻判定

对于气象事件的结束时刻的判定，我们可以等价于检测查找 AQI 变化趋于平缓的时刻，在时间序列检测方法中，我们可以利用一阶指数平滑法来解决，其计算公式为：

$$S_t^{(1)}=\alpha y_t+(1-\alpha)S_{t-1}^{(1)}t \tag{6.14}$$

其中，α为平滑指数，$S_t^{(1)}$为t时刻的一次平滑指数，y_t为t时刻的实际值，即 AQI 实测值，一般我们认为，当平滑指数α小于 0.1 的时候，说明时间序列变化比较缓慢，因此我们可以将该平滑系数所对应的时刻作为气象事件结束时刻。

6.3.2 气象事件数据集重构

通过上述方法对气象事件进行了提取，可以将每一件气象事件看做若干一维向量的集

合，为了便于后面研究，我们统一为所有具体的气象事件构造一个代表其自身特性的数据集。数据集的形式是一张二维表，其中包含了气象事件开始时刻到结束时刻的若干一维向量集合成的二维表子表，在此基础上再添加最大值、最小值、初末差（开始时刻指标与结束时刻指标差值）还有最值差，二维表具体形式如下表所示。

表 4 数据集重构表

指标 参数	SO2 浓度	NO2 浓度	PM10 浓度	PM2.5 浓度	O3 浓度	CO 浓度	温度	湿度	气压	风速	风向
时刻 i	7	51	57	18	9	0.7	29.2	78	1006.6	1.4	36.3
……											
时刻 t	4	11	11	0	17	0.4	25.3	94	999.4	5.8	70.6
最大值	47	211	217	163	405	2.5	38.2	99	1029.2	5.8	360
最小值	0	0	0	0	0	0	5.8	14	993.5	0.1	0
初末差 it	3	40	46	18	-8	0.3	3.9	-16	7.2	-4.4	-34.3
最值差	47	211	217	163	405	2.5	32.4	85	35.7	5.7	0

6.3.3 气象条件指标等级划分

本问题中涉及到的气象条件有温度，风速，风向，湿度和气压。其中，温度，风速和风向可以根据经验进行划分，如表 5 所示：

表 5 温度、风速和风向的等级划分

温度等级	温度℃	风向分类	角度°	风速等级	风速 m/s
奇热	35.0~39.9	东北	22.5~67.5	无风	0~0.2
酷热	30~34.9	东	67.5~112.5	软风	0.3~1.5
暑热	28~29.9	东南	112.5~157.5	轻风	1.6~3.3
炎热	25～27.9	南	157.5~202.5	微风	3.4~5.4
热	22～24.9	西南	202.5~247.5	和风	5.5~7.9
暖	20～21.9	西	247.5~292.5	劲风	8.0~10.7
温暖	18～19.9	西北	292.5~382.5	强风	10.8~13.8
微温	16～17.9	北	382.5-360 和 0~22.5	-	-
温和	14～15.9			-	-
微温凉	12～13.9	-	-	-	-
温凉	10～11.9	-	-	-	-
凉	5～9.9	-		-	-
微寒	0～4.9	-	-	-	-

然而，湿度和气压在很大程度上是相互耦合的。因此，考虑对气象条件的潜在特征进行挖掘。常用的模式识别分类算法大致可以分为两类：①基于贝叶斯理论的分类算法；②非参数化分类算法。其中，基于贝叶斯理论的分类算法强烈依赖于样本数据的先验知识，如先验概率和概率密度函数等，这与实测数据驱动的二次建模范式相悖。Fisher 线性判别模型是一类非参数化的分类方法，可以提取气象条件的潜在特征，同时可以避免过拟合。Fisher 方法的目的是寻找一个最佳的投影方向，使得样本在该投影空间具有最大的类间散度和最小的类内散度。

已知样本$\{x_1, x_2, \ldots, x_n\}$，$x_i \in \square^n$，这些样本属于$c$类$x_i \in \{X_1, X_2, \ldots, X_k\}$。计算类间散度矩阵$S_\omega$和类内散度矩阵$S_b$：

$$S_{\omega}=\sum_{c}^{i=1}\sum_{x_k\in X_i}\left(x_k-\mu_i\right)\left(x_k-\mu_i\right)^{\mathrm{T}} \tag{6.15}$$

$$S_b=\sum_{c}^{i=1}N_i\left(\mu_i-\mu\right)\left(\mu_i-\mu\right)^{\mathrm{T}} \tag{6.16}$$

其中，定义 $N_j(j=1,2\ldots k)$ 为第 j 类样本的个数，$\mu_j(j=1,2\ldots k)$ 为第 j 类样本的均值向量。希望样本在投影空间具有最大的类间散度和最小的类内散度，由此定义 Fisher 函数：

$$J_{\mathrm{F}}(\boldsymbol{w})=\frac{\sum_{i=1,j=1}^{N_j}\left(\mu_i-\mu_j\right)^2}{\sum_{j=1}^{N_j}S_b^2} \tag{6.17}$$

为了清晰说明问题，取 $N_j=n$，即原问题退化为一个 n 分类问题，此时使得 Fisher 函数取得第 i 个最大值的最佳投影向量 ω_i^* 为：

$$\omega_i^*=S_{\omega_i}^{-1}(\mu_i-\mu_{i+1}) \tag{6.18}$$

分类阈值 y_i 通常用如下公式计算：

$$y_i=\frac{N_1\mu_1+\ldots+N_{i+2}\mu_{i+2}}{N_1+\ldots+N_{i+2}} \tag{6.19}$$

由此，得到 Fisher 判别规则：

$$\begin{cases}Y>y_i\Leftrightarrow X\in\omega_{i+1}\\ Y<y_i\Leftrightarrow X\in\omega_{i+2}\end{cases} \tag{6.20}$$

6.4 气象条件聚类

6.4.1 气象条件聚类数目计算模型

先前工作提取了所有的气象事件，由于按照 6.3.3 中对气象条件的划分标准划分气象事件会使得气象条件划分的类别过多，所以我们可以先将一些类似的气象条件进行合并，比如常识指出气象事件的产生是有固定的成因导致的。因此，可以合并相类似的气象事件转换成衡量两种气象事件成因为同一成因的可能性。

设置一个概率生产模型 $p(\cdot)$，定义一个核函数 $k(A,B)=p(A)p(B)$，我们把两种气象条件是否相似转换成衡量他们是否由同一概率生成模型所生产，如果相似，则生成概率和核函数的取值都比较大。

我们再引入一个隐变量 ϑ 来扩展上述核函数，即

$$k(A,B)=p(A|\vartheta)p(B|\vartheta)p(\vartheta) \tag{6.21}$$

为了让式子有泛化性，我们假设隐变量 ϑ 是连续的，因此核函数可以转化为积分：

$$k(A,B)=p(A|\vartheta)p(B|\vartheta)p(\vartheta) \tag{6.22}$$

在这里隐变量相当于联系两种气象事件的变量，对于两种气象事件所对应的数据集，只有在成因相同的情况下才予以合并，因此，在这里隐变量ϑ可以通过气象条件组成的一维向量来计算。

通过上述计算方法，两两计算两种气象事件的生成概率和核函数取值，然后按照时间顺序将某气象事件与和其做概率和核函数计算值最大的气象事件合并，最终可以获得最终的气象事件合并集合数量。

6.4.2 气象条件聚类模型

气象条件存在高度非线性，强耦合和时变等特征，传统聚类模型难以表达气象条件之间每个维度的关联特性。图是一种数据结构，图中的每个节点可以视作气象条件中的对应子维度。通过图中节点的权重描述气象条件维度之间的相关关系，适合于气象条件的聚类分析。

图可表示为$G=(V,E,X)$，其中$V=\{v_i\}_{i=1,\cdots,n}$是图的节点，$E=\{e_{ij}\}$是图的边集。图G的拓扑结构可以用近邻矩阵A来表达：

$$A_{i,j}=\begin{cases}1,(v_i,v_j)\in E\\0,otherwise\end{cases} \tag{6.23}$$

此外，定义$X=\{x_1;\ldots;x_n\}$为图G各节点的属性值向量，显然，$x_i\in\mathbb{R}^m$。

通常而言，对于给定图G的聚类通常满足以下两个属性：①在图G的拓扑结构上临近；②在图G的各个节点具有更类似的X属性值向量。

深度注意力嵌入聚类模型是一种先进的图聚类技术，其技术框架如图所示，由图注意自动编码器和自训练聚类模块两部分组成。

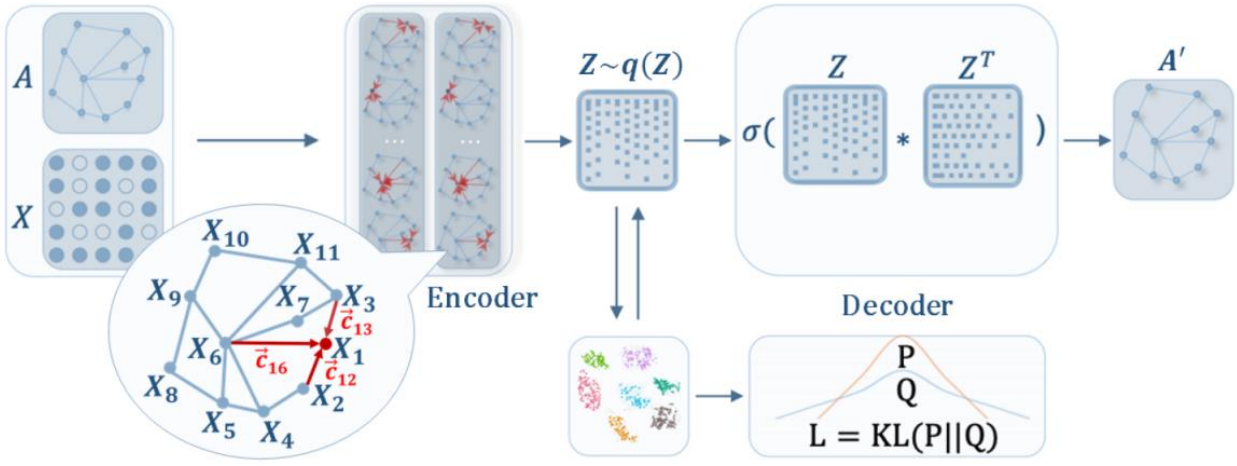


图 5 深度注意力嵌入聚类模型的架构图

图注意自动编码器：该自动编码器以属性值和图结构为输入，通过最小化重构损失来学习。

自训练聚类模块：自训练模块根据学习到的表示进行聚类，反过来根据当前聚类结果对进行操作。

深度注意力嵌入聚类模型的思想是将属性值与隐藏表示中的图结构相结合，并通过关

注每个节点的邻居节点来学习高维数据中潜在的模式。分配一个节点的邻居节点的重要性的最直接策略是将它的表示与所有邻居平等地连接在一起。然而，为了衡量各个邻居的重要性，考虑在图注意策略中给邻居节点赋予不同的权重:

$$z_i^{l+1}=\sigma\left(\sum_{j\in N_i}\alpha_{ij}Wz_j^l\right) \tag{6.24}$$

其中，z_i^{l+1}是第i个节点的输出表示，N_i代表第i个节点的邻居节点数量，α_{ij}是第j个节点对第i个节点的注意力系数，σ是非线性算子。为了得到α_{ij}，需要计算属性向量X和图的拓扑结构距离。

从属性向量X的角度看，注意力系数α_{ij}可以看作x_i和x_j的加权单层前馈神经网络，其中，$\vec{a}\in R^{2m'}$是加权向量：

$$c_{ij}=\vec{a}^T\left[Wx_i\square Wx_j\right] \tag{6.25}$$

从拓扑角度看，邻居节点通过边集来表给定节点。图具有复杂的结构关系，考虑利用编码器中的高阶邻域进行表示。可通过图中的t阶邻居节点得到一个拓扑权重矩阵M:

$$M=\left(B+B^2+\cdots+B^t\right)/t \tag{6.26}$$

B称为过渡矩阵且满足以下条件：

$$B_{ij}=\begin{cases}1/d_{ij},e_{ij}\in E\\0,otherwise\end{cases} \tag{6.27}$$

其中，d_{ij}第i个节点的自由度。因此，M_{ij}表达了第j个节点对第i个节点在t阶上的拓扑相似度。即$M_{ij}>0$等价于第j个节点是第i个节点的邻居节点。t的值可以根据模型的性能进行调节。

为了方便节点之间的注意力系数比较，通常采用 softmax 函数进行归一化处理：

$$\alpha_{ij}=softmax_j\left(c_{ij}\right)=\frac{\exp\left(c_{ij}\right)}{\sum_{r\in N_i}\exp\left(c_{ir}\right)} \tag{6.28}$$

拓扑权重矩阵M和激活函数δ代入可得：

$$\alpha_{ij}=\frac{\exp\left(\delta M_{ij}\left(\vec{a}^T\left[Wx_i\square Wx_j\right]\right)\right)}{\sum_{r\in N_i}\exp\left(\delta M_{ir}\left(\vec{a}^T\left[Wx_i\square Wx_r\right]\right)\right)} \tag{6.29}$$

设$x_i=z_i^0$作为属性向量的初始值，并将两个图注意层合并：

$$z_i^{(1)}=\sigma\left(\sum_{j\in N_i}\alpha_{ij}W^{(0)}x_j\right) \tag{6.30}$$

$$z_i^{(2)} = \sigma\left(\sum_{j\in N_i}\alpha_{ij}W^{(1)}z_j\right) \tag{6.31}$$

通过这种方式，编码器将图结构和节点属性都编码为一个隐表示，即 $z_i = z_i^{(2)}$ 。

解码器的作用是重构图结构，重构属性值，或重构两者。由于图结构和节点属性表达为隐式信息，所以选择采用相对简单的内积解码器来预测节点之间的链接，这样既高效又灵活:

$$\hat{A}_{ij} = sigmoid\left(z_i^{\top}z_j\right) \tag{6.32}$$

其中，$\hat{A}$ 是重构的近邻矩阵。

通过测量 $\hat{A}$ 与 A 之间的差来最小化重构误差：

$$L_r = \sum_{n}^{i=1} loss\left(A_{i,j}, \hat{A}_{ij}\right) \tag{6.33}$$

除了优化重构误差外，将隐藏的信息输入到一个自优化的聚类模块中，使以下目标最小化:

$$L_c = KL(P\square Q) = \sum_i\sum_u p_{iu}\log\frac{p_{iu}}{q_{iu}} \tag{6.34}$$

其中，q_{iu} 代表节点输出 z_i 与聚类中心 μ_u 的相似度。考虑采用一个 Student-t 分布来测量它，这样便可以处理不同规模的集群，并且计算方便：

$$q_{iu} = \frac{\left(1+\left\|z_i-\mu_u\right\|^2\right)^{-1}}{\sum_k\left(1+\left\|z_i-\mu_k\right\|^2\right)^{-1}} \tag{6.35}$$

这可以视作每个节点的可能的聚类。另外， p_{iu} 是满足如下定义的目标分布：

$$p_{iu} = \frac{q_{iu}^2/\sum_i q_{iu}}{\sum_k\left(q_{ik}^2/\sum_i q_{ik}\right)} \tag{6.36}$$

靠近聚类中心的节点可被认为在 Q 中是值得信赖的。聚类损失函数驱动当前分布 Q 接近目标分布 P 优化。即将聚类损失降至最小，以帮助自动编码器利用数据本身的特性来学习，并尽可能地分散初始节点以获得更好的聚类性能。

为联合优化自编码器性能和聚类性能，并定义总目标函数为:

$$L = L_r + \gamma L_c \tag{6.37}$$

其中，L_r 和 L_c 分别是图的重构损失和聚类损失，γ 用于调节两者之间的平衡。对于节点 v_i 的最优的聚类标签可以用如下公示表达：

$$s_i = \arg\max_u q_{iu} \tag{6.38}$$

深度注意力嵌入聚类模型用伪代码的方式进行了如下总结：

Algorithm 1 Deep Attentional Embedded Graph Clustering

Require：

Graph G with n nodes; Number of clusters k; Number of iterations $Iter$; Target distribution update interval T; Clustering Coefficient γ.

Ensure:

Final clustering results

Update the autoencoder by minimizing Eq.(6.33) to get the autoencoder hidden embedding Z;

Compute the initial cluster centers u based on Z;

for l= 0 to $Iter$ -1 **do**

Calculate soft assignment distribution Q with Z and μ according to Eq.(6.35);

if l%T== 0 **then**

Calculate target distribution P with Q by Eq.(6.36);

end if

Calculate clustering loss L_c according to Eq.(6.34);

Update the whole framework by minimizing Eq.(6.37);

end for

Get the clustering results with final Q by Eq.(6.38)

6.4.3 气象条件聚类效果分析

与监督式学习算法不同，聚类算法没有预先设置标签，聚类的性能不能用准确度或召回等指标衡量。从聚类的目的出发，为了衡量气象条件的聚类结果在类内的聚合程度和类间的分离成都，本文采用 Sihouette 轮廓系数，Calinski Harabasz 指数（CH 分数），Davis Bouldin 指数（DBI）和簇内相似度，簇间相似度构建综合评判模型用于聚类性能分析。这些评价指标的计算公式分别如公式（6.39）到公式（6.41）所示：

Sihouette 轮廓系数：

$$s_1(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{6.39}$$

其中，$b(i)$和$a(i)$分别是不同类点和同类点之前的平均欧氏距离。$s_1(i)$接近 1，则说明样本合理；$s_1(i)$接近-1，说明样本更应该到另外的簇；若$s_1(i)$等于 0，则说明样本出于类的边界上。

Calinski Harabasz 指数是通过类内方差和类间方差来进行聚类评分。对于聚类模型来说，希望聚类结果为相同类别之间的数据距离越近越好，而不同类别之间的数据距离越远越好。因此，对于 k 聚类，Calinski-Harabaz 的分数 S 被定义为组间离散与组内离散的比率，

该分值越大说明聚类效果越好。

$$s_2 = \frac{SS_B}{k-1} / \frac{SS_W}{N-k} \tag{6.40}$$

其中，N 是全部数据的数目，SS_B 是类间方差，SS_W 是类内方差。

Davis Bouldin 指数：

$$\begin{aligned} S_3(i) &= \frac{1}{N}\sum_{N}^{i=1}\max\left(\frac{S_i + S_i}{M_{ij}}\right) \\ M_{ij} &= \left\{\sum_{N}^{k=1}\left|a_{ki} - a_{kj}\right|^p\right\}^{1/p} \\ S_i &= \left\{\frac{1}{T_i}\sum_{T_i}^{j=1}\left|X_j - A_i\right|^q\right\}^{1/q} \end{aligned} \tag{6.41}$$

其中，X_j 代表第 i 类中第 j 个数据点；A_i 表示类中心；T_i 代表第 i 类中数据总数；q 是范数，当 q 取 2 时，即为欧氏距离；a_{ki} 表示第 i 类中第 k 个属性的值；M_{ij} 是第 i 类与第 j 类的中心距离。

经过前文所述的聚类数目计算模型，已经将合适的聚类数目缩小到 6，7，8，9 和 10 类。在此基础上，进一步确定最优类别并进行潜在模式分析，如下图所示。

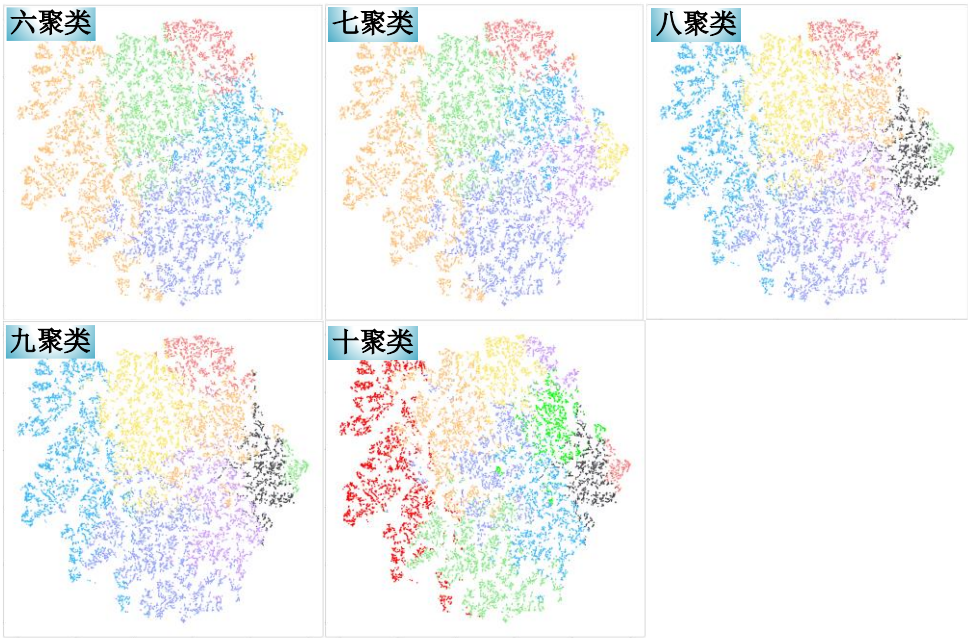


图 5 气象条件聚类效果分析

表 6 各聚类数目的性能指标分析

类别数目	簇内相似度	簇间相似度	轮廓系数	CH 分数	DBI
六聚类	8132.47	6.65	0.22	6810.61	1.28
七聚类	6782.31	6.95	0.22	6369.27	1.33
八聚类	7735.03	6.43	0.20	6733.71	1.28
九聚类	4944.37	7.33	0.19	5585.45	1.34
十聚类	4393.78	7.42	0.19	5251.63	1.33

6.4.4 综合评价指标函数

由于我们采用了 5 个指标多维度多角度评估聚类效果，过多评估指标会导致确定最优聚类方案时比较困难，因此，我们采用了一种主因素突出型指标综合评价函数，通过计算各种聚类方案的函数值，选取函数值最大的方案作为最优聚类方案。

设 $W=(w_1,...,w_n)$ 为正规化权向量，其中 $0\leqslant w_i\leqslant 1(i=1,...,n)$，即将评价指标归一化处理后代入计算，$\forall X=(x_1,...,x_n)'\in Z$，则综合评价值 y 可以用如下式子计算。

$$y=f(W,X)=\bigvee_{i=1}^{n}\{x_i\wedge w_i\} \tag{6.42}$$

6.5 分类结果及分析

6.5.1 气象条件分类结果及特点

通过主因素突出型指标综合评价函数计算，我们算得当聚类类别数为 8 的时候，函数评价值 y 最大，因此，我们选取八聚类方案为最终方案。

由于各种气象条件是比较复杂的，即使同一类气象条件也千差万别，因此我们通过可视化每一类的类中心气象条件展示该类气象条件的特点，八聚类聚类中心如下图所示。

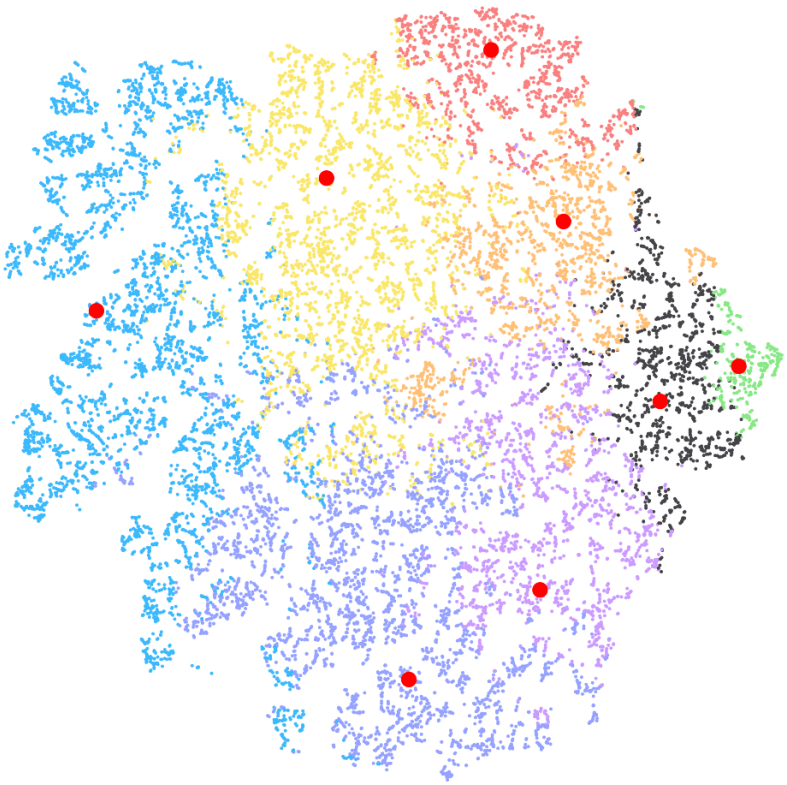


图 6 八聚类情况下的聚类中心

采用欧氏距离相加后求均值的方法得到八聚类情况下的聚类中心，将这些气象条件代入前文气象条件指标级别分类表示得到如下的八类气象条件：

表 7 典型气象条件

分类	AQI	空气质量	首要污染物	温度	风速	湿度	气压	风向
1	39	优	无	炎热	轻风	潮湿	正常	东北
2	128	轻度污染	O3	热	轻风	适宜	正常	东
3	57	良	PM10	温凉	和风	干燥	正常	北
4	42	良	无	酷热	轻风	潮湿	低	西
5	121	轻度污染	NO2	温暖	软风	潮湿	正常	东北
6	20	优	无	暑热	轻风	非常潮湿	低	东南
7	64	良	NO2	炎热	轻风	非常潮湿	正常	西北
8	99	良	NO2	炎热	软风	适宜	正常	西北

6.5.2 分类结果分析

通过观察，我们可以看出部分类别的气象条件是复合我们生活常识认知的。例如第三类气象条件，这一类气象条件，该类气象条件中心点气象状况为温度为温凉，由于该地常年天气温度较高，温凉在当地已属于低温，和风也属于当地的较大分风级，天气比较干燥，风向为北，我们大致可以猜测该气象条件属于冬天，而在这种气象条件下首要污染物为PM10，是雾霾的主要成分，符合冬天雾霾严重的现实情况。再比如第六类气象条件，该类气象条件中心点气象状况为温度为暑热，非常潮湿，风向东南，由此可推断该时刻应该是当地的夏天，夏天的天气质量通常比较好，与现实相符。综上，通过举例验证，该气象分类方案时可靠准确的。

7. 问题三的建模与求解

7.1 问题三的解析

由于空气质量与污染源分布，环境中的反应条件和人为因素具有强耦合关系，准确获取空气质量的可靠预测十分困难。其中，区域级别的自然污染源预测更加具有挑战性。目前的解决办法可以划分为两类 ①基于知识驱动的方法。基于污染源，即排放清单的研究，这类方法常根据社会、经济、人均能源消耗、能源结构和参与到排放中的具体对象相关。因此，在研究过程中需要关于污染形成机制的先验知识。目前，常用的知识驱动方法主要集中在变分同化方法，集合平滑方法和卡尔曼滤波方法等方面；②基于模式驱动的预测或反演方法，通常有启发式算法和人工神经网络等，具有强大的非线性逼近能力。同时，此类方法解释性欠缺，常被用于模型修正。综上所述，考虑分别建立知识驱动和模式驱动的预测模型，利用模式驱动的预测模型对知识驱动的模型进行修正。

问题三的目标是利用基于实测数据的二次建模来预测 A、B、C 三个监测点在 2021 年 7 月 13 日至 7 月 15 日 6 种常规污染物的单日浓度值，计算相应的 AQI 和首要污染物。该问题的解决主要可以分为以下 3 个步骤：

步骤一：建立大气过程动力学—卡尔曼滤波模型预测

子步骤 1：查阅相关气象知识并根据生活常识建立大气过程动力学模型

子步骤 2：在大气过程动力学模型中输入污染物排放清单和污染物浓度数据获得污染物浓度变化和污染源的映射关系

子步骤 3：结合时间序列，获得关于时间的污染物浓度观测表达式

子步骤 4：结合关于时间的污染物浓度观测表达式建立卡尔曼滤波模型

子步骤 5：将污染物浓度、一次数据预报输入卡尔曼滤波模型获得题目指定时间的空气条件状况

步骤二：建立NARX神经网络模型预测

子步骤1：设计神经网络结构（输入层节点数、滞后阶数、隐含层数、隐含层节点数）

子步骤2：选择合适算法、选择训练集对神经网络进行训练

子步骤3：选择测试集对神经网络训练效果进行测试，确定损失函数并进行数值计算

子步骤4：将一次预报数据、实测污染物浓度作为输入，预测题目指定时间的空气条件状况

步骤三：通过分析神经网络模型预测性能参数，对大气过程动力学—卡尔曼滤波模型预测结果进行修正，获得最终预测结果

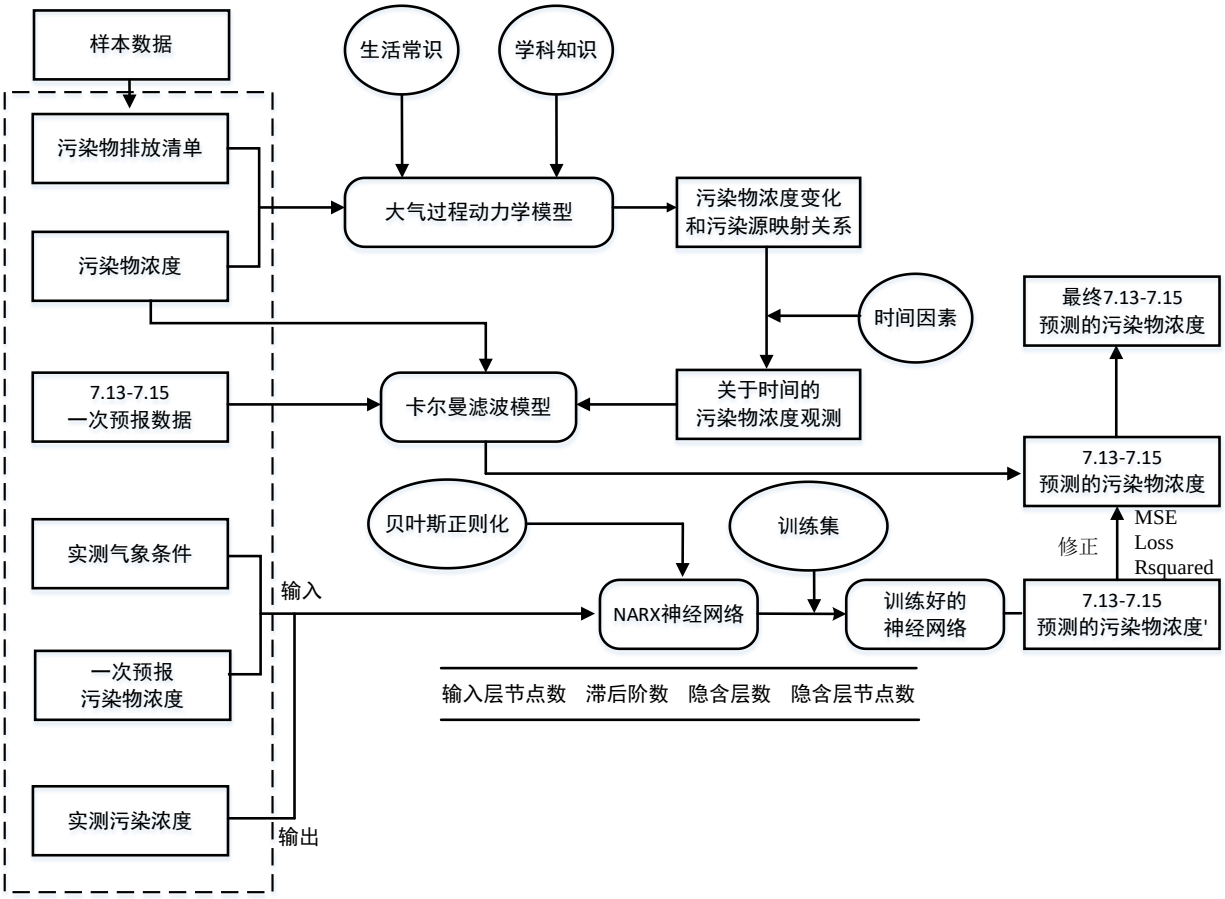


图 7 问题三的处理流程图

7.2 大气过程动力学模型

空气质量受到大气过程动力学影响，具有很强的时变性，非线性和敏感性，建立一个通用的空气质量预测模型是非常困难的。出于简化问题目的考虑，对已知但存在扰动的污染源排放清单建立大气过程动力学模型。其目的是得到观测点的污染物（模型输出）浓度变化和污染源（模型输入）的映射关系。基于所构建的大气过程动力学模型，可以对已知存在，但不具备良好观测的污染源清单进行改进和预测。考虑如下的大气过程动力学模型：

$$\begin{cases} \boldsymbol{C}\left(x,t_{k+1}\right)=M_k\boldsymbol{C}\left(x,t_k\right)+\boldsymbol{Q}\left(x,t_{k+1}\right)+\boldsymbol{q}\left(x,t_{k+1}\right) \\ \boldsymbol{Q}\left(x,t_{k+1}\right)=\boldsymbol{Q}^b\left(x,t_{k+1}\right)+\boldsymbol{Q}'\left(x,t_{k+1}\right) \end{cases} \tag{7.1}$$

其中，$\boldsymbol{C}$ 是污染浓度的向量；表达各监测点的各类污染物浓度；x 是监测点位置；t_{k+1} 是观测时间的次态；M_k 是污染的化学过程引导的污染传送模式；$\boldsymbol{q}$ 是污染传送模式的误差，代表如观测误差，人为因素和其他非污染传送过程导致的干扰，可用一个随机变量表达；$\boldsymbol{Q}^b$ 是污染源相关的排放清单；$\boldsymbol{Q}'$ 污染源相关的排放清单的误差，也用一个随机变量表达。

在 $t_0,t_1,\cdots,t_k$ 时刻对污染物浓度进行观测，

$$\boldsymbol{y}\left(t_k\right)=\boldsymbol{H}_k\left[\boldsymbol{C}\left(x,t_k\right)\right]+\boldsymbol{r}_k \tag{7.2}$$

其中，$\boldsymbol{r}$表示观测误差，可用一个随机变量表达。由于观测的物理位置，采样设备可用性和被采样物种存在不完全性，考虑用$\boldsymbol{H}_k$表达观测模式。

7.3 卡尔曼滤波的预测模型

卡尔曼滤波是一种基于贝叶斯理论的估计方法，它能从具有噪声的系统的动力学过程中提取出真实的信息。它是一种闭环算法，最突出的特点是引入了自我矫正的机制，利于容忍实际工况下的测量误差和随机噪声，适合复杂的控制质量预测问题。

上述的大气过程动力学模型是简化的线性模型，但巧妙地是，由于在预测方程和观测方程中引入了服从噪声（一般服从正态分布），使得集合卡尔曼滤波具有解决非线性问题的能力。因此，考虑将建立的大气过程动力学模型转化为集合卡尔曼滤波的表达。

假定上述的大气过程动力学模型服从一阶Markov过程，即C_k仅与C_{k-1}相关，而与$C_{k-2},...,C_0$无关，则可以通过进一步增加一些假设条件将原问题转化为卡尔曼滤波问题求解。首先确定初值的先验分布，假定为正态分布形式：

$$f\left(\boldsymbol{C}_0\right) \propto \exp\left\{-\frac{1}{2}\iint\left[\boldsymbol{C}_0(x)-\boldsymbol{C}_0^f(x)\right]^{\mathrm{T}}\boldsymbol{W}_{C_0}\left(x,x'\right)\left[\boldsymbol{C}_0\left(x'\right)-\boldsymbol{C}_0^f\left(x'\right)\right]\mathrm{d}x\mathrm{d}x'\right\}\square\,\boldsymbol{w} \tag{7.3}$$

如果考虑时间相关性，则可用如下的一阶Markov过程得到污染源的先验概率分布：

$$\boldsymbol{Q}_k = \alpha\boldsymbol{Q}_{k-1} + \sqrt{1-\alpha^2}\boldsymbol{w} \tag{7.4}$$

其中，α>0，用于表达观测的时间尺度，考虑对逐日和逐小时气象数据加以区分。对于观测模式，观测误差的先验分布定义为：

$$f\left(\boldsymbol{y}_k|\ \boldsymbol{C}_k\right) \propto \exp\left\{-\frac{1}{2}\left[\boldsymbol{y}_k-\boldsymbol{H}_k\left(\boldsymbol{C}_k\right)\right]^{\mathrm{T}}\boldsymbol{W}_{\circ}\left[\boldsymbol{y}_k-\boldsymbol{H}_k\left(\boldsymbol{C}_k\right)\right]\right\} \tag{7.5}$$

基于各先验分布的正态分布假设，则$f\left(\boldsymbol{C}_k,\boldsymbol{Q}_k|\ \boldsymbol{C}_{k-1}^{+}\right)$也服从正态分布，其中+上标表示满足后验概率分布的随机变量，即

$$f\left(\boldsymbol{C}_k,\boldsymbol{Q}_k|\ \boldsymbol{C}_{k-1}^{+}\right) \propto \exp(-\frac{1}{2}\theta(x)) \tag{7.6}$$

其中，

$$\theta(x)\square\iint\begin{bmatrix}\boldsymbol{C}_k(x)-\boldsymbol{C}_k^f(x)\\ \boldsymbol{Q}_k(x)-\boldsymbol{Q}_k^b(x)\end{bmatrix}^T\begin{bmatrix}\boldsymbol{W}_{ck}\left(x,x'\right) & \boldsymbol{W}_{QCk}^T\left(x,x'\right)\\ \boldsymbol{W}_{QCk}\left(x,x'\right) & \boldsymbol{W}_{Qk}\left(x,x'\right)\end{bmatrix}\begin{bmatrix}\boldsymbol{C}_k(x')-\boldsymbol{C}_k^f(x')\\ Q_k(x')-\boldsymbol{Q}_k^b(x')\end{bmatrix}dxdx' \tag{7.7}$$

从而，

$$\begin{aligned} &f\left(\boldsymbol{C}_k^{+},\boldsymbol{Q}_k^{+}\right)=f\left(\boldsymbol{C}_k,\boldsymbol{Q}_k|\ \boldsymbol{C}_{k-1}^{+}\right)f\left(\boldsymbol{y}_k|\ \boldsymbol{C}_k\right)\propto\\ &\exp\left\{\theta(x)-\frac{1}{2}\left[\boldsymbol{y}_k-\boldsymbol{H}_k\left(\boldsymbol{C}_k\right)\right]^{\mathrm{T}}\boldsymbol{W}_o\left[\boldsymbol{y}_k-\boldsymbol{H}_k\left(\boldsymbol{C}_k\right)\right]\right\}\end{aligned} \tag{7.8}$$

如果记观测的估计误差为：

$$J\left(\boldsymbol{C}_k,\boldsymbol{Q}_k\right)=\theta(x)+\left[\boldsymbol{y}_k-\boldsymbol{H}_k\left(\boldsymbol{C}_k\right)\right]^{\mathrm{T}}\boldsymbol{W}_o\left[\boldsymbol{y}_k-\boldsymbol{H}_k\left(\boldsymbol{C}_k\right)\right] \tag{7.9}$$

那么观测的最优估计可表达为：

$$\left(\boldsymbol{C}_k^a,\boldsymbol{Q}_k^a\right)=\arg\min\left\{J\left(\boldsymbol{C}_k,\boldsymbol{Q}_k\right)\right\} \tag{7.10}$$

由此，可以得到满足后验分布 $f\left(\boldsymbol{C}_k^+,\boldsymbol{Q}_k^+\right)$ 的一组样本用于空气质量的预测：

$$\begin{bmatrix}\boldsymbol{C}_1^{(j)+}\\ \boldsymbol{Q}_1^{(j)+}\end{bmatrix}=\begin{bmatrix}\boldsymbol{C}_1^{(j)}\\ \boldsymbol{Q}_1^{(j)}\end{bmatrix}+\boldsymbol{K}\left[\boldsymbol{y}_1^{(j)}-H_1\left(\boldsymbol{C}_1^{(j)}\right)\right] \tag{7.11}$$

其中，$\boldsymbol{y}_1^{(j)}$ 是一组气象实测数据的观测值初值，在此基础上进行迭代，直到计算出气象数据的预测值 $\boldsymbol{C}_k^a$ 和 Q_k^a，则在 t_{k+1} 时刻的污染物可以用下面的公式预测：

$$\boldsymbol{Q}_{k+1}^{(j)}=\alpha\boldsymbol{Q}_k^{(j)+}+\sqrt{1-\alpha^2}\boldsymbol{w}_w^{(j)} \tag{7.12}$$

7.5 基于神经网络的预测模型

7.5.1 NARX 动态神经网络

NARX（Nonlinear Auto-Regressive eXogenous）是一种非线性自回归模型，其最大优点是可以把复杂系统描述为线性参数模型，另外 NARX 模型的结构可以通过一些回归算法进行确定。NARX 模型可以用以下数学表达式描述：

$$y(t)=f\left(y(t-1),...,y(t-n_y),x(t-n_x)\right)+\xi(t)=f\left(x(t)\right)+\xi(t) \tag{7.13}$$

其中，$x(t)$ 代表模型系统的输入，$y(t)$ 代表模型系统的输出，t 是时间，其取值范围为 $[1,N]$（N 为训练数据容量大小），(t) 代表外部误差，n_x 和 n_y 分别代表系统的输入和输出里最大的延迟项。

NARX 神经网络是带有外部变量的 NARX 模型，该神经网络主要由输入层、隐含层、输出层以及输入延迟和输出延迟组成，其结构示意如图 8：

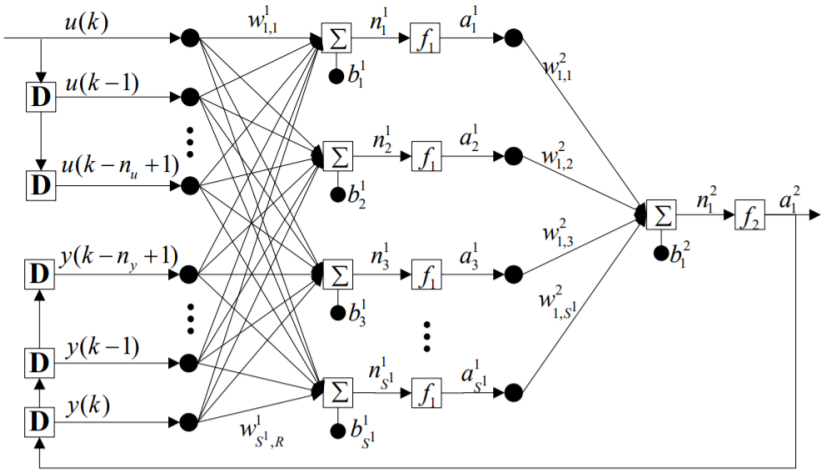


图 8　NARX 神经网络结构图

其中，n_u 表示非线性系统的输入阶数，n_y 表示非线性系统的输出阶数，$R=n_u+n_y$ 表示两层神经网络的输入个数。$w^1_{i,j}$ 表示网络隐藏层中第 i 个神经元和输入向量中第 j 个元素之间的连接权值，S^1 表示网络隐藏层中神经元的个数，b^1_i 表示网络隐藏层中第 i 个神经元的偏置值，n^1_i 表示网络隐藏层中第 i 个神经元的净输入，$f_1(x)$ 表示网络隐藏层神经元的传输函数，使用 Sigmoid 函数，a^1_i 表示网络隐藏层中第 i 个神经元的输出值。$w^2_{i,j}$ 表示网络的输出神经元和隐藏层中第 i 个神经元之间的连接权值，b^2_i 表示网络输出神经元的偏置值，n^2_i 表示网络输出神经元的净输入，$f_2(x)$ 表示网络输出神经元的传输函数，使用纯线性函数，a^2_i 既表示网络输出神经元的输出值，又表示网络在 $k+1$ 时刻的预测输出值 $y_m(k+1)$。

假设 n_p 表示预测的时间区域，则可以通过 S 遍历 n_p 中的任意一个元素得到一个从 k 时刻开始 n_p 维的预测输出向量 $y_m(k+1)$，如下所示：

$$y_m(k+1)=\left[y_m(k+1),y_m(k+2),...,y_m(k+n_p)\right]^T \tag{7.14}$$

由此可知，对于本题的预测，需要选择合适的 n_p 维输入向量 $u_m(k)$，这样输出向量 $y_m(k+1)$ 才更加接近于实际情况。其中，$u_m(k)$ 和 $y_m(k+1)$ 的形式如公式（7.15）和公式（7.16）所示：

$$u_m(k)=\left[u(k),u(k+1),...,u(k+n_p-1)\right]^T \tag{7.15}$$

$$y_r(k+1)=\left[y_r(k+1),y_r(k+2),...,y_r(k+n_p)\right]^T \tag{7.16}$$

7.5.2 NARX 神经网络结构设计

NARX神经网络结构设计的主要任务在于确定其输入层节点数、动态时滞、隐含层数及隐含层节点数。

① NARX神经网络输入层节点数的确定

由前文论述可知，本文用一个预测周期3天的同一时刻6中空气污染物浓度的一次预报值和5种气象条件实测值合并作为一个预测周期的输入，假设训练集中总共有N个预测周期，那么该神经网络的输入就是一个（N,3,11）的元组，因此本文设定神经网络输入层节点数为3。

② NARX神经网络滞后阶数的确定

由于某日某时刻预测的气象数据需要根据前三日同时刻的气象数据来预测，所根据的气象数据和预测的气象数据最小时间差为1天，因此本文设定神经网络的滞后阶数为1。

③ NARX神经网络隐含层数的确定

神经网络隐含层数太多太少都不好，虽然隐含层数量越多意味着神经网络的处理能力越强，但是神经网络隐含层数量越多同样可能造成反传误差的现象，这样的神经网络结构会变得更加复杂。多数研究一般将神经网络的隐含层数量设置为1，因此本文设定NARX神经网络隐含层数位1。

④ NARX神经网络隐含层节点数的确定

目前学界没有对于神经网络隐含层节点数确定的详细设计步骤，因此隐含层节点数都是通过多次实验进行均方误差（MSE）、平均绝对百分误差（MAPE）等参数比较后进行确定的，本文设置6、12、18、24四种节点数实验方案，通过实验（详细内容见附录）筛选，设定NARX神经网络隐含层节点数为12。

最终，我们设计了一个输入层节点数为3，滞后阶数为1，隐含层数为1，隐含层节点数为12的NARX动态神经网络。

7.5.3 NARX 神经网络训练与测试

① 训练集与测试集的选择

由于，题目所给数据在 2020-8-2-9:00 和 2020-11-21-9:00 时出现整行缺失且无法填充修复，因此本文将 2020-8-2-10:00 到 2020-11-21-8:00 的气象数据（大约占比 31.66%）作为测试集，其他气象数据作为训练集。

② 神经网络训练算法的选择

在神经网络模型结构设计的基础上，本文选择以贝叶斯正则化算法为 NARX 神经网络训练算法。贝叶斯正则化算法在很大程度上避免了对少量的、有噪音的或冗余的数据集的大量推测，避免了神经网络系统模型的过度训练和过度拟合问题。贝叶斯正则化过程如下。x 是 NARX 神经网络权重向量，C 是训练集，α 和 β 是函数参数，M 代表神经网络模型，对于贝叶斯分析公式（7.17）：

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)} \tag{7.17}$$

其中，$P(A|B)$ 是已知在事件 B 发生的条件下 A 发生的概率，也称后验证概率，即 $P(x|C,\alpha,\beta,M)$，$P(B|A)$ 是已知在事件 A 发生的条件下 B 发生的概率，即 $P(C|x,\beta,M)$，$P(A)$ 是事件 A 发生的概率，也称先验证概率，即 $P(x|\alpha,M)$，$P(B)$ 是事件 B 发生的概

率，也称边缘概率，即 $P(C|\alpha,\beta,M)$，可得公式（7.18）：

$$P(x|C,\alpha,\beta,M)=\frac{P(C|x,\beta,M)P(x|\alpha,M)}{P(C|\alpha,\beta,M)} \tag{7.18}$$

设 σ_ε^2 代表网络输入中的每一个元素的方差，$N=Q\times S^M$，Q 是训练集目标中数据数量，S^M 是训练集中每一个元素变化的敏感度，t_q 表示神经网络的期望输出，a_q 表示神经网络的实际输出。结合

$$\beta=\frac{1}{2\sigma_\varepsilon^2} \tag{7.19}$$

$$Z_c(\beta)=(2\pi\sigma_\varepsilon^2)^{\frac{N}{2}}=(\frac{\pi}{\beta})^{\frac{N}{2}} \tag{7.20}$$

$$E_c=\sum_{q=1}^{Q}(t_q-a_q)^T(t_q-a_q) \tag{7.21}$$

$$P(C|x,\beta,M)=\frac{e^{-\beta E_c}}{Z_c(\beta)} \tag{7.22}$$

再设 $\alpha=1/2\sigma_\omega^2$ 代表神经网络输入数据中每一个数据元素的权值的方差，$Z_W(\alpha)=(2\pi\sigma_\omega^2)^{n/2}=(\pi/2)^{n/2}$，$n$ 是 NARX 神经网络中权值的数量，$E_W=\sum_{i=2}^{n}x_i^2$，E_W 是权值的平方和。得到以下式子（7.23）：

$$P(x|C,\alpha,\beta,M)=\frac{\frac{e^{-\beta E_c}}{Z_c(\beta)}\frac{e^{-\alpha E_W}}{Z_W(\alpha)}}{\text{归一化项}}=\frac{e^{-F(x)}}{Z_F(\alpha,\beta)} \tag{7.23}$$

其中 $F(x)=\alpha E_W+\beta E_c$，$Z_F(\alpha,\beta)=Z_c(\beta)Z_W(\alpha)$，$Z_F(\alpha,\beta)$ 是关于 α 和 β 的函数，即 $Z_F(\alpha,\beta)$ 与 x 无关。

所以，只需最大化 $P(C|x,\beta,M)$，即可实现最小正则化 $F(x)=\alpha E_W+\beta E_c$，通过估计参数 α 和 β 再带入计算即可找到可行解。

③ 神经网络的训练

设计了一个输入层节点数为 3，滞后阶数为 1，隐含层数为 1，隐含层节点数为 12 的 NARX 动态神经网络，如图 9 所示：

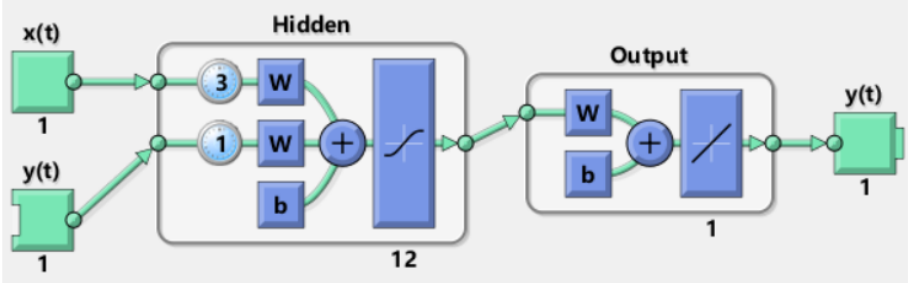


图 9 MATLAB 中设定的 NARX 神经网络结构图

其中，输入的 $x(t)$ 和 $y(t)$ 分别是关于污染物浓度的一次预报数据和关于气象条件的实测数据，输出的 $y(t)$ 是关于污染物浓度的实测数据。

④ 损失函数

该模型选用均方误差损失函数，对训练集或数据集测试样本，利用如下式子计算其均方误差损失函数值：

$$MSE = \sum_{i=1}^{n} (y_i - y_i)^2 \qquad (7.24)$$

其中，y_i 代表神经网络的预测值，y_i 是真实的样本标签。训练神经网络的时候利用 Adam 优化器进行优化。

利用上述NARX神经网络模型和算法，可以根据一次预报数据和实测数据学习天气状况变化特征，从而较为准确地预测出后续的天气情况。

7.6 问题三的结果分析

7.6.1 卡尔曼滤波模型的预测性能评估

表 7 卡尔曼滤波方法计算的均方误差和 R^2 的值

MSE	R^2
5.84	0.94

回归评价指标均方误差指标 MSE 和 R^2 可以很好地评价模型拟合效果，卡尔曼滤波模型方法计算后的 R^2 为 0.94，说明该模型的预测性能非常好。

7.6.2 神经网络模型的预测性能评估

① 损失函数值

训练神经网络的时候利用 Adam 优化器进行优化，训练集和验证集损失函数对比如图，训练集合验证集损失函数值见下表

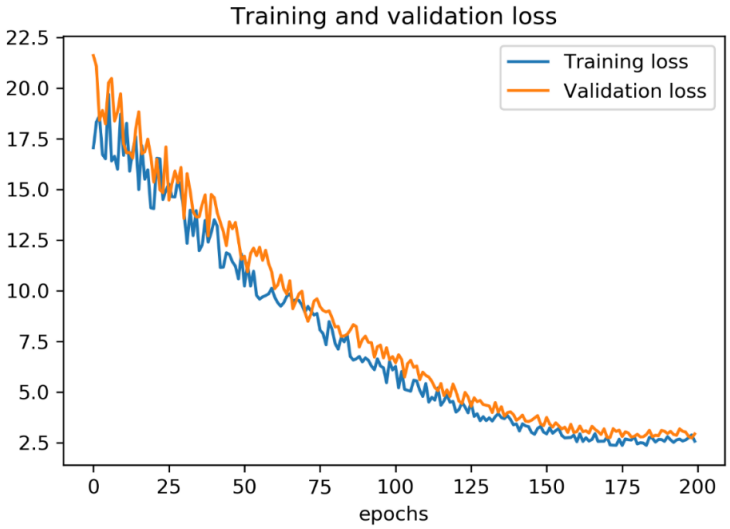


图 10 神经网络模型训练集和验证集损失函数对比图

表 8 神经网络模型训练集和验证集损失函数数值表

$L_{Training}$	$L_{Validation}$
2.58	2.94

② 正确率

训练集和验证集准确率对比如图 5.23，训练集合验证集准确率数值见表 5.8:

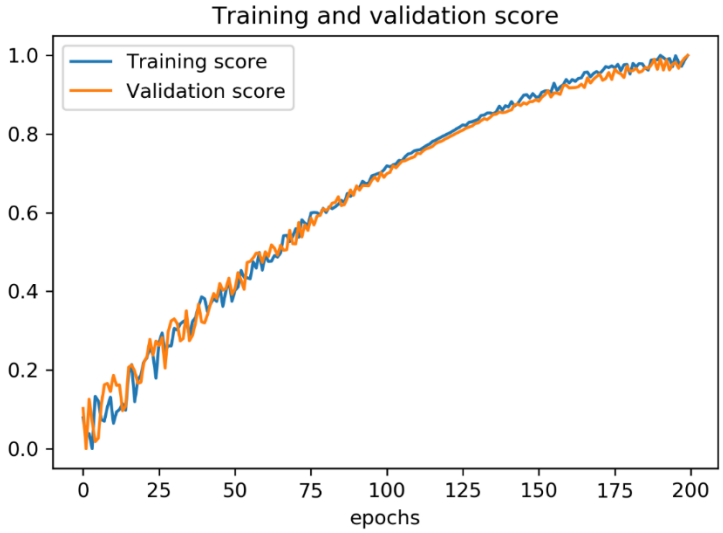


图 11 神经网络模型训练集和验证集准确率对比图

表 9 神经网络模型训练集和验证集准确率数值表

$score_{Training}$	$score_{Validation}$
0.95	0.93

通过损失函数值和预测准确率可知，NARX 动态神经网络有着很好的处理非线性复杂问题的能力，且最终的预测结果非常好。

7.6.3 综合模型的预测性能评估

我们需要借由，神经网络模型强大的拟合能力对卡尔曼滤波模型进行结果上的修正，在这里我们设置一个修正结果生产函数 $f(\cdot)$，令

$$f(C)=\alpha A+\beta B \tag{7.25}$$

其中，A代表卡尔曼滤波模型输出结果，$A=[a_{SO_2},a_{NO_2},a_{PM10},a_{PM2.5},a_{O_3},a_{CO}]$，$B$代表 NARX 神经网络模型的输出结果，$B=[b_{SO_2},b_{NO_2},b_{PM10},b_{PM2.5},b_{O_3},b_{CO}]$，$\alpha$ 和 β 是权值，在模型预测中可以由程序反馈得到。

表 10 组合预测模型的均方误差和 R^2

MSE	R^2
5.20	0.98

通过对 MSE 和 R^2 的计算，我们能够看出我们对初始的模型一步步的进行修正，且精度也在一步步的上升。

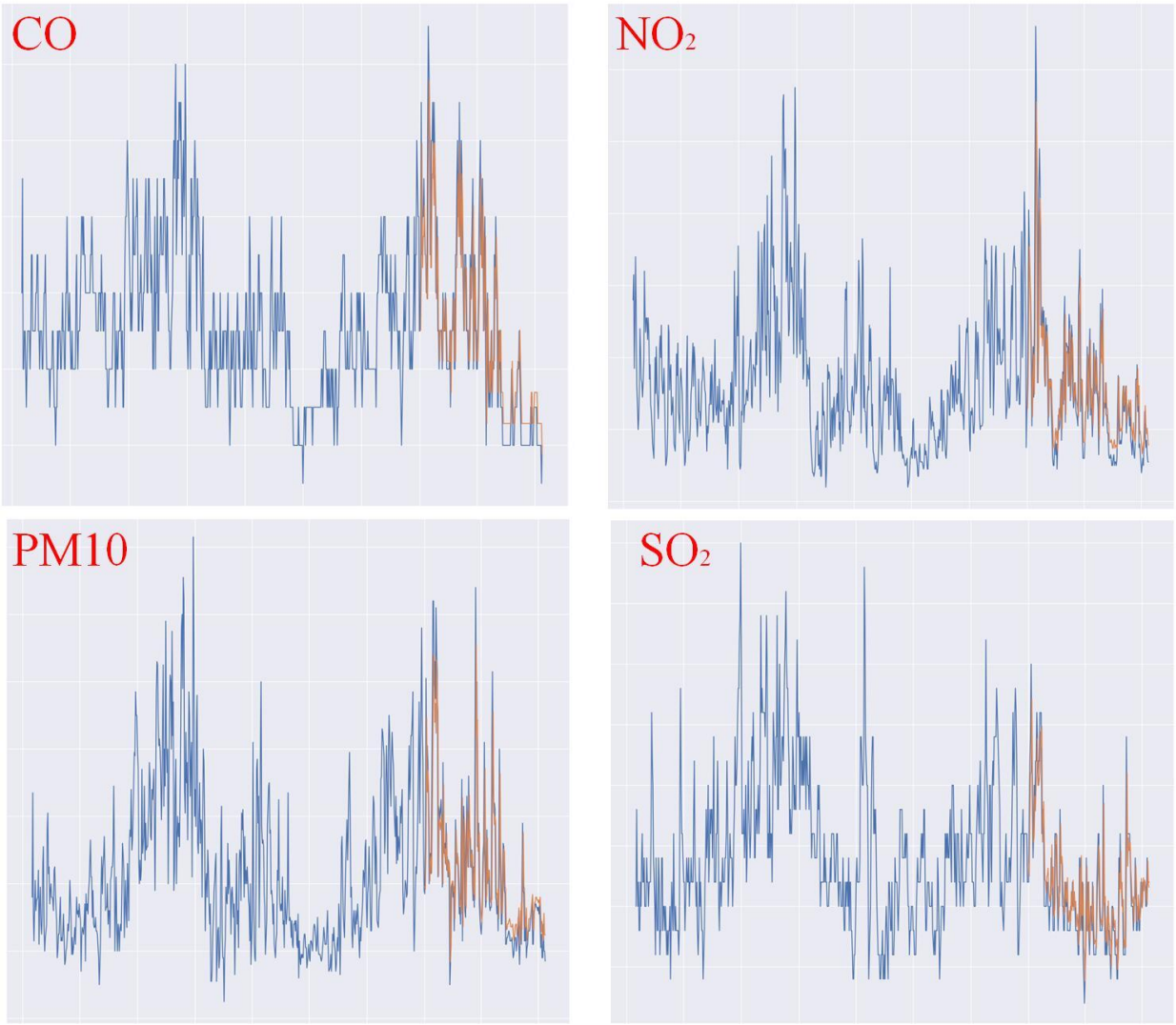
7.6.4 综合模型最终预测结果

我们利用最终的模型进行数据拟合，得到 A，B，C 三地的 MSE 如表 12 所示。

表 11 监测点 A、B 和 C 数据拟合的各类污染物均方误差

地点\污染物 MSE	SO_2	NO_2	PM_{10}	$PM_{2.5}$	O_3	CO
A	1.11	8.59	10.14	12.70	18.05	0.10
B	0.40	2.52	3.97	2.45	7.66	0.03
C	1.05	3.14	5.64	4.23	11.74	0.04

以 A 地为例，拟合的结果如图 12 所示。



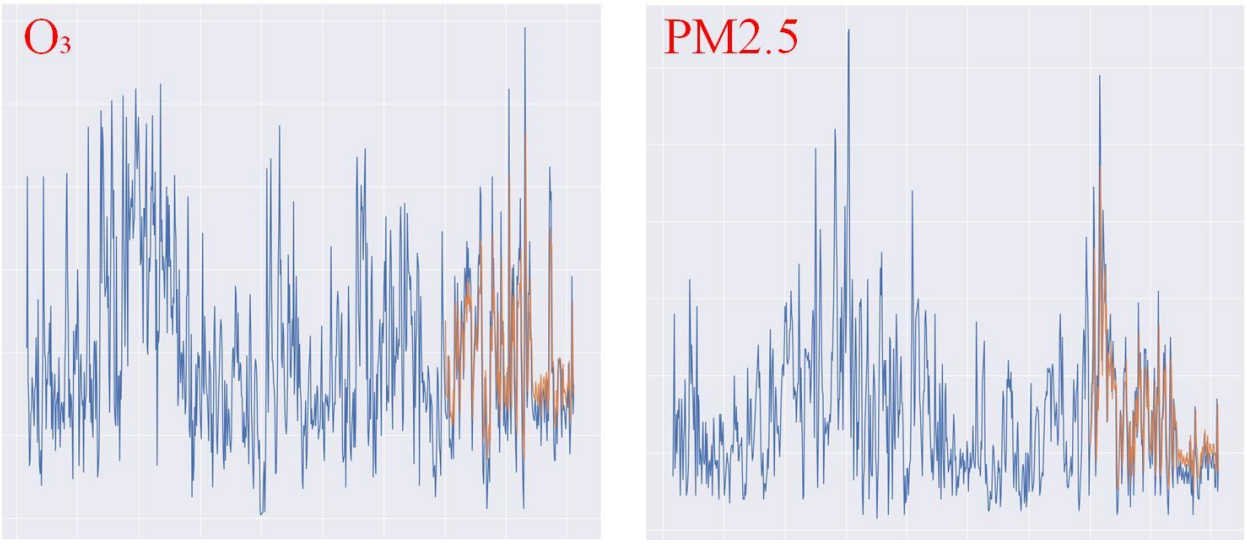


图 12 监测点 A 拟合结果图

表 12 AQI 和首要污染物 和污染物浓度

预报日期	地点	二次模型日值预测							
		SO_2 ($\mu g/m^3$)	NO_2 ($\mu g/m^3$)	PM_{10} ($\mu g/m^3$)	$PM_{2.5}$ ($\mu g/m^3$)	O_3 最大八小时滑动平均 ($\mu g/m^3$)	CO (mg/m^3)	AQI	首要污染物
2021/7/13	监测点 A	6.21	15.62	22.41	16.7	86.66	0.46	44	无
2021/7/14	监测点 A	6.04	19.26	26.73	21.28	90.49	0.5	46	无
2021/7/15	监测点 A	6.19	22.13	30.19	23.07	93.08	0.54	47	无
2021/7/13	监测点 B	4.97	8.60	18.92	4.93	46.12	46.98	24	无
2021/7/14	监测点 B	4.99	8.66	19.57	5.73	50.01	0.41	26	无
2021/7/15	监测点 B	4.98	8.65	19.28	5.60	51.27	0.38	26	无
2021/7/13	监测点 C	8.15	18.34	34.33	16.46	105.69	0.57	55	O_3
2021/7/14	监测点 C	7.64	17.77	33.66	15.46	97.74	0.54	49	无
2021/7/15	监测点 C	7.41	17.78	33.92	16.58	96.66	0.54	49	无

8. 问题四的建模与求解

8.1 问题四的解析

由于污染物容易扩散传播，所以我们考虑利用周围环境的情况来提高对当前环境的预测准确率。我们在问题三的模型基础上新增了区域协同预测机制。将一次预报数据进行了修正，使其能够提高最终结果的准确性。采用区域网格的划分来得到各个监测点之间的拓扑关系，然后利用大气流动力学理论构建了拓扑关系的权重初始系数，然后采用图卷积神经网络调整，得到最终的结果。本问题可以归纳为以下四个步骤：

步骤一：使用迭代的方法划分网格，构建拓扑结构，划分出节点和边；

步骤二：利用大气流动力学理论，考虑了站点距离，站点的风速，站点间的方向，站点的风向等多种因素，对拓扑结构的边进行赋值；

步骤三：由于人工建模的局限性，考虑不全面性，我们采用 step2 的值作为初始值，构建图卷积神经网络，输入是一次预报污染物浓度和拓扑结构，输出是修正后的一次预报污染物浓度，将其输出作为问题 3 的输入，进行反向传播，修正模型参数以及拓扑结构参数；

步骤四：训练好模型后，进行预测。

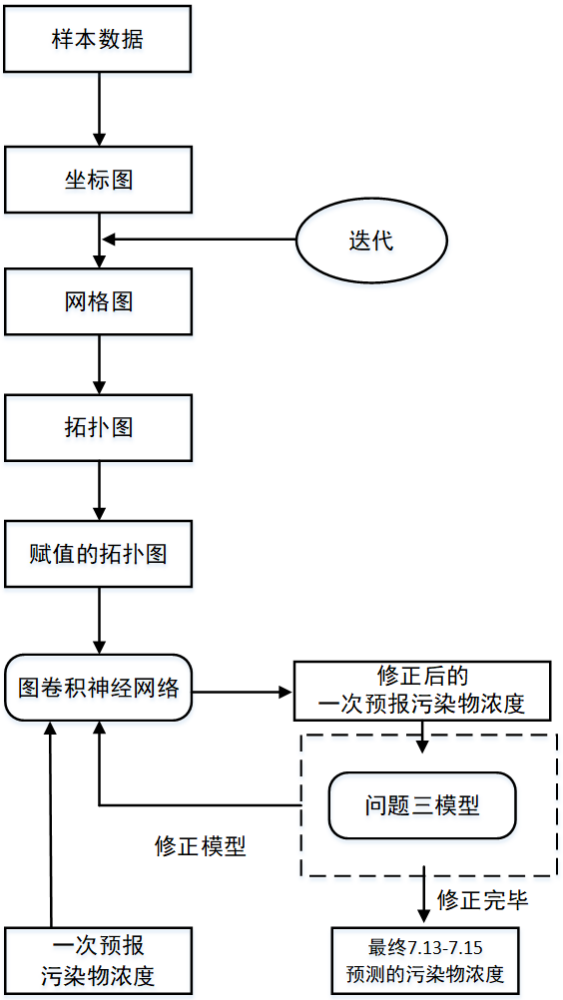


图 13 问题四的处理流程图

8.2 区域协同预测模型

8.2.1 拓扑结构定义

采用拓扑结构来描述各个区域之间的相互作用，包括区域距离等特征属性，通过对研究区域进行网格的划分，根据风向风速，建立区域空气质量污染物传播拓扑结构特征的表征方法。

网格的划分采用迭代的方法，其过程如下。

（1） 迭代变量：规定 N 为划分正方形网格的边长份数

（2） $P\{P_1,P_2,...,P_i,...\}$是监测点集合，$S\{S_1,S_2,...,S_j,...\}$是网格的集合，若$i<j$，则P_i属于S_j，P中的其他点不属于S_j。当满足迭代终止条件，即N最大时，若P中有点也属于S_j，则令$N=N+1$继续迭代。

（3） 当N能够区分监测点的划分正方形网格的边长份数。

监测站点位置由经纬度坐标标识，与网格为一一对应的关系，划分的网格大小能够确保每个监测点只属于一个网格且保证此网格为最大的网格划分，如下图所示。

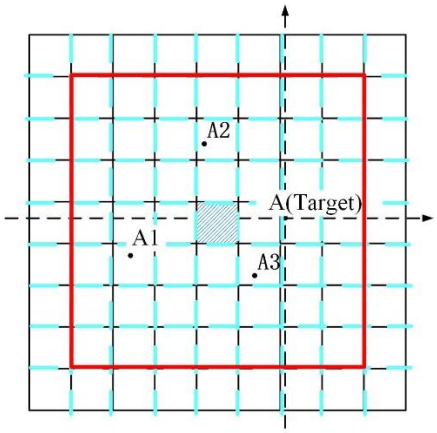


图 14 网格划分及其影响区域

通过对区域网格的划分，赋予边的权重来分析网络关联影响，从而建立区域拓扑结构。

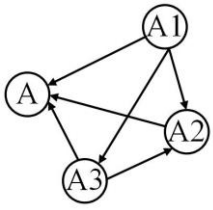


图 15 拓扑结构表征示意图

8.2.2 网络边的定义

两个站点的距离越近，污染物浓度差越大，污染物越容易扩散传播。污染物的传播方向主要是顺着风向传播。t时刻站点i和站点j的传播系数为

$$tc_{ij}(t)=\gamma\frac{R_{ij}(t)\,|\,\Delta AQI_{ij}(t)\,|}{|\,S_{ij}\,|}\pm\varepsilon(t) \tag{8.1}$$

S_{ij} 表示站点 i 和站点 j 的水平距离，$\Delta AQI_{ij}(t)$ 表示站点 i 和站点 j 在 t 时刻的 AQI 差值。$R_{ij}(t)$ 表示站点 i 和站点 j 在 t 时刻的风力系数，γ 表示矫正系数，取值是[0,1]。$\varepsilon(t)$ 是一个波动调整值。依据大气流动力学理论表示如下：

$$R_{ij}(t)=\begin{cases}0 & if\ \ \frac{\pi}{2}\le\theta_{ij}(t)<\pi\\ |\,F_i\,|\cos\theta_{ij}(t) & if\ \ 0\le\theta_{ij}(t)<\frac{\pi}{2}\end{cases} \tag{8.2}$$

其中，θ_{ij} 是 t 时刻站点 i 到站点 j 的实际方向和站点 i 的风向 F_i 的夹角。

8.2.3 模型构建

由于边的权重受很多因素的影响，所以我们将上述模型的结果作为边的权重的初始值，然后利用图卷积网络来自学习其相互之间的影响关系。图卷积神经网络将卷积运算从传统数据推广到图数据，其核心思想是学习一个函数映射 $f(\square)$，通过该映射图中的节点 v_i 可以聚合它自己的特征 x_i 与它的邻居特征 x_j，来生成节点 v_i 的新表示，如下图所示。

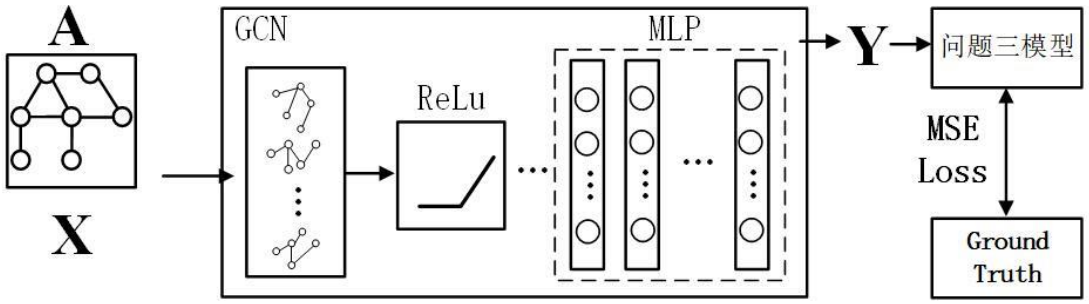


图 16 区域协同预测模型框架图

图中的 X 代表的是污染物一次预报的浓度，A 代表的是数据之间的关系，Y 代表的是利用拓扑结构修正后的一次预报的浓度。

8.3 问题四的结果及分析

采用问题三的 MSE 损失对该模型进行训练，相当于是对问题三模型的数据进行了一个预处理。在训练了 150 个 epoch 后，模型收敛。我们将自学习到的拓扑结构进行了可视化，如图 19 所示。

图 17 各监测点之间相互影响结果图

如图所示，每一个监测点与自身都是相关性最强的，影响程度与两个地点的距离是有关的，而且 A 对 A1 的影响与 A1 对 A 的影响是不一样的，这说明了的确与风向等这种单向的因素是有关系的，这个结果同时也证明了我们一开始初始化的模型的正确性。

表 13 区域协同预测模型对监测点 A、A1、A2 和 A3 数据拟合的各类污染物均方误差

地点\污染物 MSE	SO_2	NO_2	PM_{10}	$PM_{2.5}$	O_3	CO
A	0.41	3.89	4.53	7.09	13.26	0.03
A1	0.67	3.94	5.52	3.34	13.20	0.04
A2	0.59	3.96	5.46	3.06	14.04	0.03
A3	0.69	3.98	5.89	4.12	13.33	0.04

通过对该模型对 A 的预测与问题 3 模型中对 A 的预测的 MSE 进行对比，可以得出，区域协同预测模型的结果更好，各类污染物的 MSE 相较之前降低了 42.37%。说明污染物的确是很容易扩散的，临近地区的数据确实能帮助我们提高数据预测的准确性。最终，在问题三基础上加入了区域协同影响预测的结果如表 14 所示：

表 14 区域协同预测模型的预测结果

预报日期	地点	二次模型日值预测							
		SO_2 ($\mu g/m^3$)	NO_2 ($\mu g/m^3$)	PM_{10} ($\mu g/m^3$)	$PM_{2.5}$ ($\mu g/m^3$)	O_3 最大八小时滑动平均 ($\mu g/m^3$)	CO (mg/m^3)	AQI	首要污染物
2021/7/13	监测点 A	5.71	11.72	20.19	7.09	77.11	0.38	39	无
2021/7/14	监测点 A	5.56	12.11	20.70	6.52	69.18	0.36	35	无
2021/7/15	监测点 A	5.48	12.79	21.98	6.47	71.50	0.36	36	无
2021/7/13	监测点 A1	6.90	15.65	28.01	10.02	92.86	0.43	47	无
2021/7/14	监测点 A1	6.76	17.26	29.46	10.84	87.72	0.43	44	无
2021/7/15	监测点 A1	6.64	18.85	29.46	11.05	91.22	0.43	46	无
2021/7/13	监测点 A2	5.16	18.46	27.10	8.70	89.87	0.51	45	无
2021/7/14	监测点 A2	4.92	17.61	29.11	9.10	90.51	0.49	46	无
2021/7/15	监测点 A2	4.83	16.74	31.11	8.88	93.67	0.48	47	无
2021/7/13	监测点 A3	3.78	9.65	17.12	6.78	83.95	0.33	42	无
2021/7/14	监测点 A3	3.80	9.92	16.19	7.58	78.62	0.33	40	无
2021/7/15	监测点 A3	3.95	10.04	17.19	8.00	85.04	0.35	43	无

9. 模型的评价、改进与推广

9.1 模型的优点

① 本文将数据预处理过程模型化，增加了问题建模的鲁棒性，可拓展性强。

② 本文将天气条件分类转换成对于气象条件成因相同概率的衡量，即将数据集（气象条件）的合并转换成对于数据集的生成概率（气象条件成因）计算，利用核函数和隐变量进行变换，然后合并生成概率值等大范围的数据集。

③ 本文先用监督学习方法进行初分类，获得了大致气象条件分类的分类数量，之后在该分类数量附近做多种聚类方案，避免盲目聚类，提升了模型的可靠性。

④ 本文建立了以知识驱动的大气过程动力学—卡尔曼滤波模型和以数据驱动的神经网络模型，并用神经网络模型修正气过程动力学—卡尔曼滤波模型结果，综合的大模型由知识、数据双驱动，提升模型效果、增强可解释性。

9.2 模型的缺点

① 本文采用的深度学习模型，虽然结合了可解释性强的知识模型，但也存在着一些不可解释的情况。

② 本文采用的模型没有实现完全端到端的处理。

9.3 模型的改进

对一次预测的气象条件和实测的气象条件之间的转换关系进行建模，能够帮助提升模型预测性能，考虑地形地势的影响，加入气象机理相关，社会行为和人为因素等的约束条件，可以让模型结果更准确。

9.4 模型的推广

① 问题三预测模型所采用的卡尔曼滤波-神经网络修正的预测模式具有普遍意义。

② 问题四的模型也可以用于化学领域，对于放射性，有害性气体的协同预测。

③ 本文提出了一种知识驱动和数据驱动相结合的深度学习方法，为其他深度学习领域的研究者提供了新的思路。

④ 本文提出了一种新的知识迁移的方法，可以被推广到其他深度学习领域，例如自然语言处理。

参考文献

[1]赵秀玲，中国城市典型空气污染物时空分布特征与影响因素研究，中国科学技术大学学位论文，2021。
[2]周璐，长沙市空气质量与气象条件的关系研究，兰州大学学位论文，2018。
[3]杨雪玲，兰州市重污染天气过程环流形势与气象条件研究，兰州大学学位论文，2018。
[4]宁贵财，四川盆地西北部城市群冬季大气污染气象成因及其数值模拟研究，兰州大学学位论文，2018。
[5]赵阳.，南昌市大气 PM2.5 污染特征与来源解析研究，华侨大学学位论文，2017。
[6]丁玄，吴链，唐笑男，李元波，长沙市区空气污染气象条件预报及应用检验，科技创新与应用，2017(18)：299-300，2017。
[7]张红，典型沿江城市空气污染物特征及与气象条件的耦合关系研究，中国科学技术大学学位论文，2017。
[8]王明莹，上海市大气 PM2.5 的时空分布特征及其相关影响因素分析，上海交通大学学位论文，2017。
[9]付桂琴，张杏敏，尤凤春，田亚芹，李二杰，气象条件对石家庄 PM2.5 浓度的影响分析，干旱气象，34(02)：349-355，2016。
[10]赵敬国，王式功，张天宇，余世旺，胡钰玲，朱哲，尚可政，兰州市大气重污染气象成因分析，环境科学学报，35(05)：1547-1555，2015。
[11]江瑶，苏州地区 PM2.5 和 PM10 时空变化特征及其影响因子分析，南京信息工程大学学位论文，2014。
[12]陈斌，我国北方重点城市 PM10 污染特征及气象成因研究，兰州大学学位论文，2014。
[13]周国兵，重庆市主城区气象条件对空气污染影响分析及数值模拟研究，兰州大学学位论文，2014。
[14]周甘霖，兰州市空气污染特征及其与气象条件关系研究，兰州大学学位论文，2012。
[15]杨凌霄，济南市大气 PM2.5 污染特征、来源解析及其对能见度的影响，山东大学学位论文，2008。
[16]郑美琴，日照市区环境空气污染与气象条件关系的研究，中国海洋大学学位论文，2004。
[17]李新令，西安城市气候年变化特征及其与 PM10 污染特征的相关分析，西安建筑科技大学学位论文，2003。

附录

问题 1 代码

```
import numpy as np
import pandas as pd
from pandas.io.parsers import read_csv
def cal(C_p, index):
    xz=np.array([[0,50,100,150,200,300,400,500],[0,50,150,475,800,1600,2100,2620],[0,40,80,180,280,565,
750,940],[0,50,150,250,350,420,500,600],[0,35,75,115,150,250,350,500],[0,100,160,215,265,800,-1,-
1],[0,2,3,14,24,36,48,60]])
    ii = -1
    if index == 5 and C_p>800:
        return -1
    for i in range(1,xz.shape[1]):
        if C_p >= xz[index,i-1] and C_p <= xz[index,i]:
            ii = i
            break
    if ii == -1:
        return -1
    IAQI_hi = xz[0,i]
    IAQI_lo = xz[0,i-1]
    BP_hi = xz[index,i]
    BP_lo = xz[index,i-1]

    return np.ceil((IAQI_hi-IAQI_lo)/(BP_hi-BP_lo)*(C_p-BP_lo)+IAQI_lo)
data_day = pd.read_csv('data_day.csv')

def cal_kqzl(x):
    r = 0
    kq = ['优','良','轻度污染','中度污染','重度污染','严重污染']
    if x<=50:
        r = 0
    elif x >50 and x <=100:
        r = 1
    elif x >100 and x <=150:
        r = 2
    elif x >150 and x <=200:
        r = 3
    elif x >200 and x <=300:
        r = 4
    elif x >300:
        r = 5
```

```
        return kq[r]

def day_cal(riqi):
        name = ['SO2','NO2','PM10','PM2.5','O3','CO']

        cal_data = data_day[data_day['监测日期']==riqi].values[0]

        result = np.zeros(6)

        for i in range(2,len(cal_data)):
                result[i-2] = cal(cal_data[i],i-1)

        AQI = np.max(result)
        kqzl = cal_kqzl(AQI)
        if AQI<=50:
                print(riqi,f'AQI {AQI}',f'空气质量{kqzl}','无首要污染物')
        elif AQI>50:
                rr = np.where(result==AQI)[0]
                print(riqi,f'AQI {AQI}',f'空气质量{kqzl}','首要污染物是')
                if rr.shape[0]==1:
                        print(name[rr[0]])
                else:
                        for i in range(rr.shape[0]):
                                print(name[rr[i]])
if __name__ == '__main__':
        riqi = ['2020/8/25','2020/8/26','2020/8/27','2020/8/28']
        for i in range(len(riqi)):
                day_cal(riqi[i])
```

问题 2 代码

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pandas import DataFrame
from sklearn import manifold
from sklearn import metrics
np.set_printoptions(suppress=True)

from sklearn.cluster import AffinityPropagation
from sklearn import cluster
# data = pd.read_csv('my.csv')
# data = data.iloc[:,12].values

# x = range(data.shape[0])
# plt.scatter(x, data)
# plt.show()
def norm(data):
    for i in range(data.shape[1]):
        data[:,i] = (data[:,i] - np.mean(data[:,i]))/ np.std(data[:,i])
    return data


data_ = pd.read_csv('data_hour.csv').iloc[:,2:].values
data = norm(pd.read_csv('data_hour.csv').iloc[:,2:].values)
data1 = data[:,:6]
data2 = data[:,6:]

def tsne(data):
    '''t-SNE'''
    tsne = manifold.TSNE(n_components=2, init='pca', random_state=501)
    X_tsne = tsne.fit_transform(data)

    #print("Org data dimension is {}. Embedded data dimension is {}".format(data.shape[-1], X_tsne.shape[-
1]))

    '''嵌入空间可视化'''
    x_min, x_max = X_tsne.min(0), X_tsne.max(0)
    X_norm = (X_tsne - x_min) / (x_max - x_min)  #  归一化
    return X_norm
    plt.figure(figsize=(8, 8))
    plt.scatter(X_norm[:,0],X_norm[:,1])
    plt.xticks([])
```

```
    plt.yticks([])
    plt.show()

# tsne(data)
# tsne(data1)
# tsne(data2)

def cal(data,cen):
    min = np.inf
    r = -1
    for i in range(data.shape[0]):
        if min>np.sum(np.abs(data[i,:]-cen)):
            min = np.sum(np.abs(data[i,:]-cen))
            r = i
    return r

def cal_cen(cen):
    num = 0
    count = 0
    for i in range(cen.shape[0]):
        for j in range(cen.shape[0]):
            num += np.sum(np.abs(cen[i,:]-cen[j,:]))
            count += 1
    return num/count

def cal_da(data,cen):
    return np.sum(np.abs(data-cen))

def kmeansjl(data_set,l):
    X = tsne(data_set)
    m = ['x','o','+','D','v','>','<','s']
    cl                                                                                          =
np.array([[149,162,255],[250,128,128],[255,192,118],[250,231,104],[135,232,133],[60,185,252],[203,155,255
],[67,67,72],[255,0,0],[0,255,0]])
    cl = cl/255
    km = cluster.KMeans(n_clusters=l, random_state=9)
    y_pred = km.fit_predict(data_set)
    cen = km.cluster_centers_
    # print('类间', cal_cen(cen))
    fig = plt.figure(figsize=(32,32))
    plt.scatter(X[:, 0], X[:, 1], c=cl[y_pred])
    # with open("2.txt","w+") as f:
    # num = 0
    for i in range(cen.shape[0]):
```

```
        #         num += cal_da(data_set[y_pred==i],cen[i,:])
        # print('类内',num/cen.shape[0])
            index = cal(data_set,cen[i])
                # f.write(str(index)+':'+str(data_[index,:])+'\n')
            plt.scatter(X[index, 0], X[index, 1], c='red',s=1000,marker='o')
            plt.legend()

        #pj = metrics.calinski_harabasz_score(data_set, y_pred)
        plt.savefig(f'img/result/center.png')

        #y_pred = cluster.Birch(threshold=0.01, n_clusters=l).fit_predict(data_set)

        pj = round(metrics.calinski_harabasz_score(data_set, y_pred),2)
        SC = round(metrics.silhouette_score(data_set, y_pred, metric='euclidean'),2)
        DBI = round(metrics.davies_bouldin_score(data_set, y_pred),2)
        fig = plt.figure(figsize=(32,32))
        plt.scatter(X[:, 0], X[:, 1], c=cl[y_pred])
        plt.savefig(f'img/result/kmeans_浓度_{l}_{pj}_{SC}_{DBI}.png')

        y_pred = cluster.AgglomerativeClustering(n_clusters=l).fit_predict(data_set)

        pj = metrics.calinski_harabasz_score(data_set, y_pred)
        plt.scatter(X[:, 0], X[:, 1], c=y_pred)
        plt.title(f'{l}_{pj}')
        plt.savefig(f'img/AgglomerativeClustering/气象条件_{l}.png')

def jl(data_set):
        X = tsne(data_set)
        y_pred = cluster.DBSCAN(eps=0.30, min_samples=9).fit_predict(data_set)
        clusters = np.unique(y_pred)
        l = len(clusters)
        pj = metrics.calinski_harabasz_score(data_set, y_pred)
        plt.scatter(X[:, 0], X[:, 1], c=y_pred)
        plt.title(f'{l}_{pj}')
        plt.savefig(f'img/DBSCAN/气象条件_{l}.png')

        y_pred = cluster.MeanShift().fit_predict(data_set)
        l = len(clusters)
        pj = metrics.calinski_harabasz_score(data_set, y_pred)
        plt.scatter(X[:, 0], X[:, 1], c=y_pred)
        plt.title(f'{l}_{pj}')
        plt.savefig(f'img/MeanShift/气象条件_{l}.png')
        y_pred = cluster.OPTICS(eps=0.8, min_samples=10).fit_predict(data_set)
        l = len(clusters)
```

```
    pj = metrics.calinski_harabasz_score(data_set, y_pred)
    plt.scatter(X[:, 0], X[:, 1], c=y_pred)
    plt.title(f'{l}_{pj}')
    plt.savefig(f'img/OPTICS/气象条件_{l}.png')
for i in range(6,11):
    kmeansjl(data1,i)

jl(data2)

kmeansjl(data1,8)
```

问题 3 代码

```
import pandas as pd
import math
import random
from sklearn.model_selection import train_test_split

input =pd.read_excel("数据集/in.xlsx")
output= pd.read_excel("数据集/out.xlsx")
x_train, x_test, y_train, y_test = train_test_split(input,output,test_size=0.2, random_state=0)#导入数据
random.seed(0)
#在数区间 a ~ b 中，随机生成一个 float 数
def rand(a, b):
    return (b - a) * random.random() + a# random.random() 生成一个 0~1 的浮点数
#创建一个指定大小的 矩阵，并用 fill 去填充
def make_matrix(m, n, fill = 0.0 ):
    mat = []
    for i in range(m):# 对行进行循环
        mat.append([fill] * n)#创建每行的列元素
    return mat
#定义 sigmoid 函数，及它的导数
def sigmoid(x):
    return 1.0 / (1.0 + math.exp(-x))
def sigmoid_derivate(x):
    return x * (1 - x)
#定义 BPNeuralNetwork 类，使用三个列表维护输入层，隐含层，输出层神经元
class BPNeuralNetwork:
    def __init__(self):
        self.input_n = 0
        self.hidden_n = 0
        self.output_n = 0
        self.input_cells = []
        self.hidden_cells = []
        self.output_cells = []
        self.input_weights = []
        self.output_weights = []
        self.input_correction = []
        self.output_correction = []
#定义 setup 的方法初始化神经网络
    # ni ,nh ,no ->各层神经元的个数
    def setup(self, ni, nh, no):
        self.input_n = ni + 1 #输入层额外加一个偏置神经元，提供一个可控的输入修正；（或者为每个
隐含层神经元设置一个偏置参数）
        self.hidden_n = nh   #隐藏层神经元个数
```

```
        self.output_n = no    #输出层神经元个数
        #init cells
        #初始化神经元的输出值
        self.input_cells = [1.0] * self.input_n #输入层各神经元的值初始化为 1
        self.hidden_cells = [1.0] * self.hidden_n#隐藏层神经元的值初始化为 1
        self.output_cells = [1.0] * self.output_n#输出层神经元的值初始化为 1
        # init weights
        #初始化神经网络各层权重的值  各层的权重已矩阵的形式存储
        self.input_weights = make_matrix(self.input_n, self.hidden_n) #初始化输入层与隐藏层间的权重
        self.output_weights = make_matrix(self.hidden_n,self.output_n)#初始化隐藏层与输出层间的权重
        #random activate
        #给权重矩阵  随机赋初值
        for i in range(self.input_n):#给输入层及隐藏层间的权重矩阵赋初值
            for h in range(self.hidden_n):
                self.input_weights[i][h] = rand(-0.2, 0.2)
        for h in range(self.hidden_n):#给隐藏层及输出层间的权重赋初值
            for o in range(self.output_n):
                self.output_weights[h][o] = rand(-2.0, 2.0)
        #init correction matrix
        #创建矫正矩阵  此处应该是指各层权重矩阵的矫正矩阵
        self.input_correction = make_matrix(self.input_n, self.hidden_n) #输入矫正矩阵
        self.output_correction = make_matrix(self.hidden_n, self.output_n)#输出矫正矩阵
        #定义 predict 方法进行一次前馈，并返回输出
    def predict(self,inputs):
        #activate input layer
        #激活输入层
        for i in range(self.input_n - 1):
            self.input_cells[i] = inputs[i] #对输入层神经元赋值 （此处不含输入层神经元的偏置）
        #activate hidden layer
        #激活隐藏层
        for j in range(self.hidden_n):#对隐含层神经元进行计算求值
            total = 0.0
            for i in range(self.input_n):#前一层神经元的输出 * 相应权重值  后求和
                total += self.input_cells[i] * self.input_weights[i][j] #
            self.hidden_cells[j] = sigmoid(total)#对每个神经元经过前一层的求和后  计算经过所选激励函数映射后的输出。
        #activate output layer
        #激活输出层
        for k in range(self.output_n):#对输出层各神经元的值进行计算
            total = 0.0
            for j in range(self.hidden_n):#隐藏层的输出经过神经元的加权后求和
                total += self.hidden_cells[j] * self.output_weights[j][k]
            self.output_cells[k] = sigmoid(total)#所得和经过激励函数的映射后的输出
        return self.output_cells[:]
```

```
#定义一次反向传播 和更新权值的过程，并返回最终预测误差
    def back_propagate(self, case, label, learn, correct):
        #feed forward ->前馈
        self.predict(case) #对实例 case 进行前馈预测
        #get output layer error ->获取输出层误差
        output_deltas = [0.0] * self.output_n #初始化输出层更新差值列表
        #计算实际的标签与预测标签的差值进行计算
        for o in range(self.output_n):
            error = label[o] -self.output_cells[o]
            output_deltas[o] = sigmoid_derivate(self.output_cells[o]) * error#？？？
        #get hidden layer error ->获取隐藏层的误差列表
        hidden_deltas = [0.0] * self.hidden_n #初始化隐藏层更新差值列表
        for h in range(self.hidden_n):
            error = 0.0
            for o in range(self.output_n):
                error += output_deltas[o] * self.output_weights[h][o]
            hidden_deltas[h] = sigmoid_derivate(self.hidden_cells[h]) * error
        # update output weights 更新输出层权重
        for h in range(self.hidden_n):
            for o in range(self.output_n):
                change = output_deltas[o] * self.hidden_cells[h]
                self.output_weights[h][o] += learn * change + correct + self.output_correction[h][o]
                self.output_correction[h][o] = change
        #update input weights       更新输入层权重
        for i in range(self.input_n):
            for h in range(self.hidden_n):
                change = hidden_deltas[h] * self.input_cells[i]
                self.input_weights[i][h] += learn * change + correct *self.input_correction[i][h] #correct 为
矫正矩阵的矫正率
                self.input_correction[i][h] = change #更新矫正矩阵
        #get global error 获取全局误差
        error = 0.0
        for o in range(len(label)):
            error += 0.5 * (label[o] - self.output_cells[o]) ** 2
        return error
  #定义 train 方法控制迭代，该方法可以修改最大迭代次数，学习率 矫正率 三个参数
    def train(self, cases, labels, limit = 10000,learn = 0.05 , correct = 0.1):
        for i in range(limit):
            error = 0.0
            for i in range(len(cases)):
                label = labels[i]
                case = cases[i]
                error += self.back_propagate(case, label, learn, correct)
    #test 方法演示如何使用神经网络学习异或逻辑
```

```
    def test(self):
        cases = [
            [0, 0],
            [0, 1],
            [1, 0],
            [1, 1],
                ]
        labels = [[0], [1], [1], [0]]
        self.setup(2, 5, 1) #初始化神经网络
        self.train(cases, labels, 10000, 0.05, 0.1)#神经网络的学习
        for case in cases: #对输入进行分类预测
            print(self.predict(case))

if __name__ == '__main__':
    nn = BPNeuralNetwork()
    nn.test()
```

数据标准化：

```
x_train, x_test, y_train, y_test = train_test_split(Input,output,test_size=0.2, random_state=0)
#数据标准化——匿名函数 max[x]-x/max[x]-min[x]
x_train=x_train.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
y_train=y_train.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
```

问题 4 代码

```
import numpy as np
import scipy.sparse as sp
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.nn.init as init
import torch.optim as optim
import matplotlib.pyplot as plt
from load_cora import *
import sys
sys.path.append("/content/drive/My Drive/nlpdata/cora/")

class GraphConvolution(nn.Module):
    def __init__(self, input_dim, output_dim, use_bias=True):
        """图卷积：L*X*\theta
        Args:
        ----------
            input_dim: int
                节点输入特征的维度
            output_dim: int
                输出特征维度
            use_bias : bool, optional
                是否使用偏置
        """
        super(GraphConvolution, self).__init__()
        self.input_dim = input_dim
        self.output_dim = output_dim
        self.use_bias = use_bias
        self.weight = nn.Parameter(torch.Tensor(input_dim, output_dim))
        if self.use_bias:
            self.bias = nn.Parameter(torch.Tensor(output_dim))
        else:
            self.register_parameter('bias', None)
        self.reset_parameters()

    def reset_parameters(self):
        init.kaiming_uniform_(self.weight)
        if self.use_bias:
            init.zeros_(self.bias)

    def forward(self, adjacency, input_feature):
        """邻接矩阵是稀疏矩阵，因此在计算时使用稀疏矩阵乘法
```

```
        Args:
        -------
            adjacency: torch.sparse.FloatTensor
                邻接矩阵
            input_feature: torch.Tensor
                输入特征
        """
        device = "cuda" if torch.cuda.is_available() else "cpu"
        support = torch.mm(input_feature, self.weight.to(device))
        output = torch.sparse.mm(adjacency, support)
        if self.use_bias:
            output += self.bias.to(device)
        return output

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

### 模型定义
class GcnNet(nn.Module):
    """
    定义一个包含两层 GraphConvolution 的模型
    """
    def __init__(self, input_dim=1433):
        super(GcnNet, self).__init__()
        self.gcn1 = GraphConvolution(input_dim, 16)
        self.gcn2 = GraphConvolution(16, 7)

    def forward(self, adjacency, feature):
        h = F.relu(self.gcn1(adjacency, feature))
        logits = self.gcn2(adjacency, h)
        return logits
```