
# 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 | 华南理工大学 |
| 参赛队号 | 21105610069 |
| 队员姓名 | 1.梁允舜 |
| | 2.黎杰 |
| | 3.黄泳熙 |

# 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：**

在空气质量预报过程中，建立高效精确的二次预报模型对提高预报准确度具有重要意义。本文在计算 AQI 值并判断首要污染物、依据空气质量情况对气象条件进行分类过程中发现，针对不同污染物建立二次预报模型，可达到提高预报准确度的目的。另外，对于以 $O_3$ 为主的污染物，考虑空间相关性的协同预报模型预测效果更好。以监测点 A 为例：

针对问题 1，根据题目所给方法，得出 2020 年 8 月 25 日到 8 月 28 日该监测点的 AQI 分别为 60，46，109，138，首要污染物分别为 $O_3$，无，$O_3$，$O_3$。

针对问题 2，首先进行数据缺失值和异常值处理，对污染物浓度采用分段归一化方法得到扩散条件评价指数，$SO_2$ 和 CO 因其评价指数均小于 1 予以排除；对于气象指标，提出伪独热码方式处理风向，在保留数据连续性的同时消除其有序性，其余指标则作 Z-score 归一化处理；然后采用 SVD 主成分分析以及 Spearman 相关性分析进行气象指标的降维和筛选，得到与 $NO_2$、颗粒物（$PM_{10}$ 和 $PM_{2.5}$）、$O_3$ 主成分，这些主成分不具备时序相关性，故采用两步聚类法完成对气象主成分进行分类，并结合污染物扩散条件评价指数频率分布得到不同类型天气的特征，其中具有代表性的是：当晨间为东北风向时，所有污染物扩散条件均能达到有利，且风速越大，扩散条件越好。

针对问题 3，进行数据预处理后，先采用基于随机森林平均不纯度减少的重要度评价方法对特征进行排序，后将一次预报数据输入前馈神经网络（FNN）、随机森林（RF）、支持向量回归（SVR）以及极端梯度提升树（XGBoost）进行自适应特征选择及建模，其中 FNN 预测效果较好：以 $PM_{2.5}$ 为例，FNN 偏差为 1.92，其余方法偏差则在 2.60 以上；但 $O_3$ 偏差均明显偏高（如 FNN 偏差为 17.14），考虑到 $O_3$ 为二次污染物，受外界影响易产生噪声，故提出 CEEMDAN_IPSOSE_SVR 建模方法：即先使用 CEEMDAN 进行降噪，后采用集成支持向量回归思想进行建模。得到模型偏差为 13.53，优于 FNN（17.14）。最后，针对偏差较高的 $NO_2$、$PM_{10}$、$PM_{2.5}$、$O_3$，考虑其具有时序性，通过 Lasso 回归将原有模型与基于逐小时实测数据的 LSTM_FC 模型进行组合。组合模型预测效果明显提升：以 $PM_{2.5}$ 为例，该模型偏差较 FNN 和 LSTM 分别下降 6.3%和 9.1%，故以其作为最终空气质量二次建模预报模型。

针对问题四，使用 Pearson 相关分析发现各污染物空间相关性均较高，故以一次预报数据为输入建立可捕捉非欧空间内节点信息的图卷积网络（GCN）模型，其中偏差较高的四项污染物（$NO_2$、$PM_{10}$、$PM_{2.5}$、$O_3$）采用可兼顾时间及空间相关性的时空图卷积模型（GRU_GCN_FC）对原有模型进行修正，得到最终空气质量预报模型。相较于问题 3 所建模型，本题模型在 $NO_2$、$O_3$、$SO_2$ 以及 CO 上的预测效果均有所提升，偏差分别下降 17.4%、12.6%、3.6%及 1.8%，$PM_{2.5}$ 及 $PM_{10}$ 预测效果则有所下降，偏差上升 12.8%、1.0%，其原

因与污染物本身特性有关。

**关键词**：主成分分析；两步聚类；CEEMDAN_IPSOSE_SVR；组合模型；图卷积网络

# 目 录

# 1 问题重述

## 1.1 问题背景

随着时代的进步和经济社会的发展，空气质量已经成为当前社会关注的焦点之一。我国日趋明显的空气污染问题会直接影响到居民的身体健康、城市的生产发展，解决污染问题刻不容缓。现有污染防治实践表明，建立空气质量预报模型，提前获知可能发生的大气污染过程并采取相应控制措施，是减少大气污染危害，提高空气质量水平的有效方法之一。目前常搭建 WRF-CMAQ 模型对空气质量进行预报，该模型具有相对坚实的物化基础，但受到污染源清单不确定性高、模拟大气环境难度大等的影响，所得一次预报结果精度往往较低。因此，需要参考污染物浓度一次预报数据、气象一次预报数据、污染物浓度实测数据、气象实测数据等预报基础数据进行二次建模，以探索更精确高效的空气质量预测模型。

## 1.2 问题描述

基于上述研究背景，题目提供了监测点长期空气质量预报基础数据，本文需要解决以下四个问题：

问题 1：空气质量情况可用 AQI 进行描述。通过对时段内污染物浓度数据进行处理，可计算相应 AQI 值并判断首要污染物，从而了解该时段空气质量及其变化情况。现根据题目要求，以监测点 A 2020 年 8 月 25 日到 8 月 28 日的实测数据为例，计算每日实测 AQI 值并判断相应首要污染物。

问题 2：空气质量在很大程度上受到气象因素的影响。在其他因素保持恒定的情况下，不同气象条件会影响污染物的浓度，进而作用于空气质量[1]。要求分析附件 1 中数据，根据其中污染物浓度与气象数据之间的关系，将气象条件划分为不同类型，并阐述各类型所具备的特征。

问题 3：由于一次模型预报结果精度不高，现使用附件 1、2 中的数据，在一次预报模型结果基础上，建立一个同时适用于互不相关的监测点 A、B、C 的二次预报数学模型，用以预测未来三天 6 种常规污染物单日浓度值，达到减小 AQI 预报值相对误差，提高首要污染物预测准确程度的效果。并使用该模型预测 A、B、C 点在 2021 年 7 月 13 日至 7 月 15 日 6 种污染物的单日浓度值，计算相应的 AQI 和首要污染物。

问题 4：相邻区域污染物浓度往往具有一定相关性。而有研究表明，考虑区域相关性的污染联防联控政策存在减排效果好坏不一的现象[2]，故开展区域协同预报可能会提升空气质量预报的准确度，也可能会降低预报准确度。现使用附件 1、3 中的数据，建立适用于相邻区域内监测点 A、A1、A2、A3 的协同预报模型，尽可能降低 AQI 预报值的相对误差及提高首要污染物预测的准确程度，并使用该模型预测 A、A1、A2、A3 在 2021 年 7 月 13 日至 7 月 15 日 6 种污染物的单日浓度值，计算相应的 AQI 和首要污染物。同时讨论相比问题三模型预报准确度是否得到提升，说明理由。

## 2 模型假设

建模过程中，假设污染物浓度变化不与污染源排放、地形地貌等因素相关，仅受气象条件影响。

## 3 符号说明

| 符号 | 注释 |
|---|---|
| cosd(·) | 按角度单位(°)计算余弦值 |
| day_1 | 预报第一天 |
| day_2 | 预报第二天 |
| day_3 | 预报第三天 |
| FC | 全连接层 |
| ⊙ | Hadamard 乘积 |
| r2_score | 可决系数得分 |
| K | 粒子数目 |
| T | 总迭代次数 |
| $\omega_t$ | 惯性权重 |
| c1、c2 | 学习因子 |
| V | 粒子速度 |
| loc | 粒子位置 |
| $loc_d$ | 位置阈值 |
| Solution | 粒子位置存储集 |
| pbest | 粒子最优值 |
| gbest | 种群最优值 |
| R2_score | SVR 集适应度 |
| A | 邻接矩阵 |

# 4　问题分析与求解

## 4.1 问题 1 求解

### 4.1.1 求解方法

根据题意，要计算出监测点 A 2020 年 8 月 25 日到 8 月 28 日逐日实测 AQI 值，并确定相应首要污染物。求解方法如下：

**计算 AQI：**由题目附录项“空气质量指数、空气质量等级及首要污染物”可知，要计算 AQI 值，首先需计算 6 项常规污染物各自的空气质量分指数 IAQI（式(1)）：

$$\mathrm{IAQI_P} = \frac{\mathrm{IAQI_{Hi}} - \mathrm{IAQI_{Lo}}}{\mathrm{BP_{Hi}} - \mathrm{BP_{Lo}}} \cdot (\mathrm{C_P} - \mathrm{BP_{Lo}}) + \mathrm{IAQI_{Lo}} \tag{1}$$

式中，$\mathrm{IAQI_P}$ 为污染物 P 的空气质量分指数，结果进位取整数；$\mathrm{C_P}$ 为污染物 P 的质量浓度值；$\mathrm{BP_{Hi}}$、$\mathrm{BP_{Lo}}$ 分别代表与 $\mathrm{C_P}$ 相近的污染物浓度限值的高位值与低位值；$\mathrm{IAQI_{Hi}}$、$\mathrm{IAQI_{Lo}}$ 分别代表与 $\mathrm{BP_{Hi}}$、$\mathrm{BP_{Lo}}$ 对应的空气质量分指数。

各项常规污染物的浓度限值及对应的空气质量分指数级别参考题目中《表 1 空气质量分指数（IAQI）及对应的污染物项目浓度限值》（下称《限值》）数值。

后得到空气质量指数 AQI：

$$\mathrm{AQI} = \max\{\mathrm{IAQI_{SO_2}}, \mathrm{IAQI_{NO_2}}, \mathrm{IAQI_{PM_{10}}}, \mathrm{IAQI_{PM_{2.5}}}, \mathrm{IAQI_{O_3}}, \mathrm{IAQI_{CO}}\} \tag{2}$$

**判断首要污染物：**当 AQI 小于或等于 50 时，当日无首要污染物；当 AQI 大于 50 时，IAQI 最大的污染物为首要污染物，如 IAQI 最大的污染物为两项或以上时，并列为首要污染物。

### 4.1.2 求解过程

根据求解方法对相应时间段内污染物浓度数据进行处理。为保证结果准确性，提高操作便捷性，团队采用 MATLAB 平台完成求解操作。编程思路如下：按样表格式（以本题为例，如表 1）输入监测点需处理日期及当日 6 种常规污染物浓度值，程序依次计算出各项污染物的 IAQI 值，IAQI 取最大得到当日 AQI，如 AQI 大于 50，则将其对应污染物（首要污染物）赋值给 primary_pollutant 项，最终 AQI 值与首要污染物将以 Excel 表形式输出。若存在污染物浓度值超出 IAQI=500 对应限值（臭氧最大 8 小时滑动平均浓度值超过 800 μg/m$^3$）的情况，程序将提示数据有误，输出 AQI 值为-1，首要污染物为 UNKNOWN。具体编程内容见附件。

表 1　计算表格模板

| TIME | $SO_2$ | $NO_2$ | $PM_{10}$ | $PM_{2.5}$ | $O_3$ | CO |
|---|---|---|---|---|---|---|
| 2020/8/25 | 8 | 12 | 27 | 11 | 112 | 0.5 |
| 2020/8/26 | 7 | 16 | 24 | 10 | 92 | 0.5 |
| 2020/8/27 | 7 | 31 | 37 | 23 | 169 | 0.6 |
| 2020/8/28 | 8 | 30 | 47 | 33 | 201 | 0.7 |

### 4.1.3 求解结果

表 2 为所得 2020 年 8 月 25 日到 8 月 28 日监测点 A 每日实测 AQI 值及相应首要污染物种类。其中 8 月 26 日 AQI 小于 50，无首要污染物。

表 2　AQI 及首要污染物计算结果表

| 监测日期 | 地点 | AQI 计算 | |
|---|---|---|---|
| | | AQI | 首要污染物 |
| 2020/8/25 | 监测点 A | 60 | $O_3$ |
| 2020/8/26 | 监测点 A | 46 | 无 |
| 2020/8/27 | 监测点 A | 109 | $O_3$ |
| 2020/8/28 | 监测点 A | 138 | $O_3$ |

### 4.2 问题 2 建模与求解

#### 4.2.1 问题分析

问题 2 要求根据对污染物浓度的影响程度对气象条件进行分类，并描述每类气象的特征和对污染物的影响。初步判断本题为分类问题。通览数据，分类对象应为气象信息，但最终目标是根据这些分类判断不同类型气象对空气质量的影响情况。由于相关气象特征存在量纲不同、种类较多、数据密度不均等现象，因此，需有针对性地对不同气象数据采用预处理办法、主成分分析以及指定聚类所用的样本间似然函数。

#### 4.2.2 数据预处理

首先，对监测点 A 的预报基础数据进行分析，数据情况如下：

① 逐小时污染物浓度与气象一次预报数据：

包含近地 2 米温度、地表温度、比湿、湿度、近地 10 米风速、近地 10 米风向等 15 种气象指标和 6 种常规污染物共 21 种特征。

由于一次预报模型会预测包括运行日期在内的未来三天各特征逐小时的模拟值，本题中数据模型运行日期为 2020 年 7 月 23 日至 2021 年 7 月 13 日，相应的预测值时间跨度为 2020 年 7 月 23 日 0:00 至 2021 年 7 月 15 日 23:00。

② 逐小时污染物浓度与气象实测数据：

包含 6 项常规污染物监测浓度及温度、湿度、气压、风速、风向共 11 种特征，时间跨度为 2019 年 4 月 16 日 0:00 至 2021 年 7 月 13 日 7:00。

③ 逐日污染物浓度实测数据：

即 6 种常规污染物的逐日监测数据，时间跨度为 2019 年 4 月 16 日至 2021 年 7 月 13 日。

本文使用逐小时气象数据（包括一次预报数据及实测数据）作为模型输入集，逐日实测数据为标签集。数据分配上，取训练集与测试集的比例为 9:1。

**(1) 异常值处理**

对于输入集：使用 SPSS 排查异常指标值大于 2 的个案。其中，实测气象数据未发现异常值；一次预报气象数据标记异常个案 10 个，均为雨量数据。经分析其雨量均在 100 mm 以内，认为合理予以保留。

对于标签集：由于标签值中仅为 6 种常规污染物逐日实测数据，本文结合《限值》内容，对各项指标出现的极端高值进行逐一排查，发现 2021 年 4 月 14 日 $PM_{2.5}$ 值为 465 $\mu g/m^3$，而 $PM_{10}$ 仅为 34 $\mu g/m^3$，与《限值》对这两种污染物的定义存在矛盾，故舍弃当日所有数据。

**(2) 缺失值处理**

对于输入集：如存在整日数据缺失则成对地排除样本，即在标签集中舍弃对应日期数据；其他情况则忽略缺失，如在用均值函数（mean）时对于缺失处理办法的选项为“omitnan”（忽略缺失值）。

对于标签集缺失：进行分段归一化处理时跳过缺失样本，并且在后续回归、相关性分析时排除这些样本。

**(3) 污染物浓度分段归一化处理**

为更直观地通过污染物浓度判断气象影响的污染物扩散条件，根据《空气污染气象学教程》[3]，将污染物扩散条件划分为“有利”、“一般”及“不利”等类别，分别对应空气

质量等级中的“优”、“良”、“轻度污染”等。现通过分段 min-max 标准化，将每项污染物浓度近似转换为污染物扩散条件评价指数，使得扩散条件为“有利”、“一般”、“不利”等的指标分别落在[0,1]、[1,2]、[2,3]等区间上，对于某一污染物的日浓度值 $C_i$，视其所在污染物浓度等级上限值及下限值分别为 max 与 min 值，作 min-max 归一化，并加以等级序数。即对于某项污染物浓度 $C_P$，标准化后的评价指数 $F_P$ 为：

$$F_P = \frac{C_P - BP_{Lo}}{BP_{Hi} - BP_{Lo}} + LEVEL_{Lo}$$

式中$LEVEL_{Lo}$为其下限值对应等级序数。以臭氧为例，标准化函数图像如图 1。

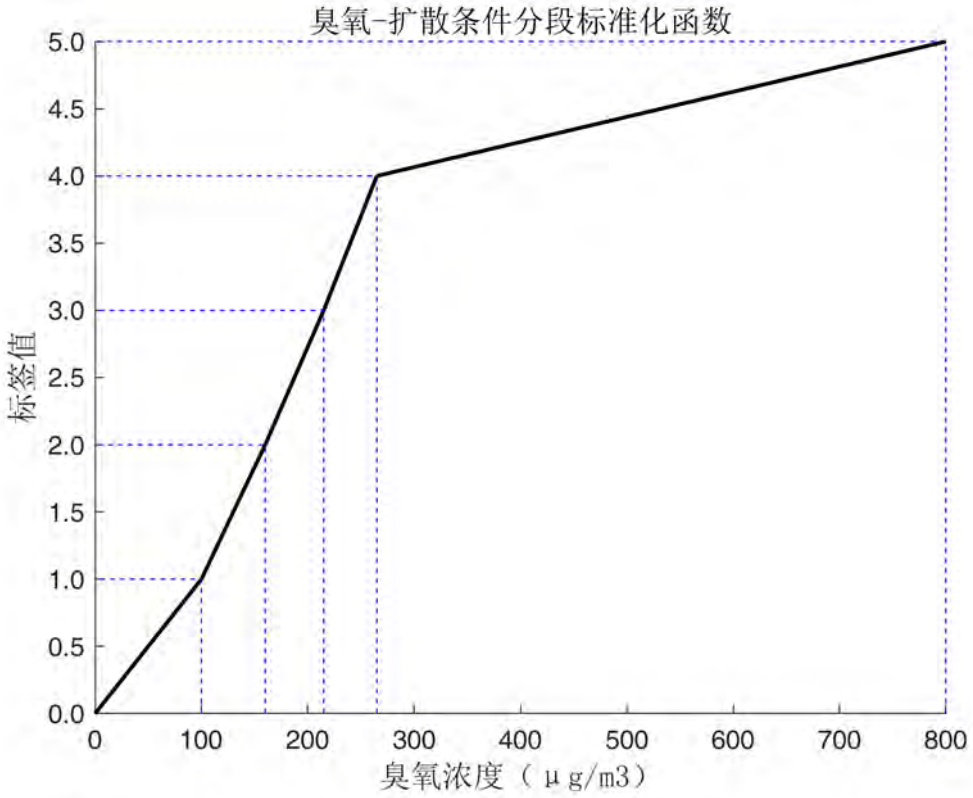



图 1　臭氧扩散条件分段标准化函数

该过程中注意到 $SO_2$、CO 日扩散条件均为“有利”，不具有判断价值。因此，本题内后续讨论将聚焦于 $NO_2$、$PM_{10}$、$PM_{2.5}$、$O_3$ 的扩散条件上。

**(4) 风向指标伪独热码处理**

审视气象数据过程中，团队认为风向在以角度数记录时，直接分析与其他数据的相关性存在漏洞：风向在实际中不应具有大小顺序。例如，在其他要素不变的情况下，风向为 1°（北偏东 1°）的情形与风向为 359°（北偏西 1°）的情形是极为相似的，但在数值上二者相差甚远。

因此，参考独热码[4]思想、国家标准文件《地面气象观测规范 风向和风速（GB/T 35227-2017）》[5]（下称《风向风速规范》）以及污染物排放点源的烟流扩散形状[6]，团队拟将原风向角度数映射为 9 个范围为[0,1]的连续变量，对应的风向如下：

表 3　风向角度映射变量的对应风向

| 变量序号 | +1 风向 | | | -1 风向 | | |
|---|---|---|---|---|---|---|
| | 风向方位 | 风向符号 | 风向度数(°) | 风向方位 | 风向符号 | 风向度数(°) |
| 1 | 北 | N | 360.0（0.0） | 南 | S | 180.0 |
| 2 | 北东北 | NNE | 22.5 | 南西南 | SSW | 202.5 |
| 3 | 东北 | NE | 45.0 | 西南 | SW | 225.0 |
| 4 | 东东北 | ENE | 67.5 | 西西南 | WSW | 247.5 |
| 5 | 东 | E | 90 | 西 | W | 270.0 |
| 6 | 东东南 | ESE | 112.5 | 西西北 | WNW | 292.5 |
| 7 | 东南 | SE | 135.0 | 西北 | NW | 315.0 |
| 8 | 南东南 | SSE | 157.5 | 北西北 | NNW | 337.5 |
| 9 | （静风） | C | | | | |

前 8 项变量聚焦于每小时风向与“+1 风向”的夹角。以第三项（对应 NE-SW）为例，为使得在“+1 风向”（东北方）附近的风向接近 1、“-1 风向”（西南方）附近的风向接近-1、两者均不接近的风向接近 0，现设定任意风向$D$映射至第$i$ $(i = 1,2,\cdots,8)$个新变量$dir^{(i)}$的公式如式 (3)。

$$dir^{(i)} = cosd^{9}\left(D - D^{(i)}_{standard}\right) \tag{3}$$

式中，$D^{(i)}_{standard}$为第$i$个变量的+1 风向角度数。

以第三项变量（$dir^{(3)}$，对应方向 NE-SW）为例，转换函数图像如图 2，当输入风向角度为 50°，计算转换后的变量值为：

$$dir^{(3)} = cosd^{9}(50 - 45) \approx 0.999988$$

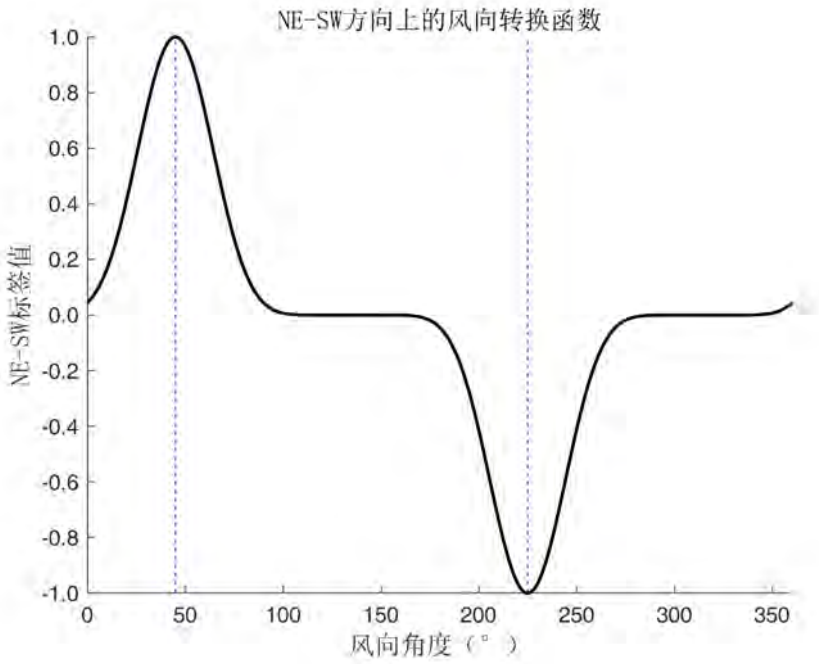



图 2　NE-SW 方向上的风向转换函数

根据《风向风速规范》，静风状态（风速≤0.2 m/s）视为不具有任何风向，前 8 个变量归零，设置第 9 变量为 1；风速＞0.2 m/s 时，保持前 8 项数值不变，第 9 项变量为 0。

**(5) 其他气象指标标准化**

对于其他气象指标，采取 Z-score 标准化方法对气象数据进行处理。该方法可以将原始数据变换到在均值为 0，方差为 1 的范围内，从而使模型更容易收敛到最优解。方法公式如下：

$$X' = \frac{X - uI}{\sigma} \tag{4}$$

式中，样本集 $X=\{x_1,x_2,\cdots\cdots,x_n\}^T$ 为一 n 维列向量；u 为样本的平均值；I 为与 X 同维的单位列向量；$\sigma$是训练样本的标准偏差，即

$$\sigma = \sqrt{\frac{(x_1-u)^2+\cdots+(x_n-u)^2}{N}}$$

### 4.2.3 数据关联性分析

考虑到标签值为逐日数据，而对应输入集变量的有 24 小时×36 个指标=864 项，直接使用聚类、回归可能会出现维度爆炸等状况；同时，各项指标对全天污染物浓度影响不一，直接进行相关性分析容易导致相关性指数较小。因此，团队拟先对输入集每项气象内部 24 小时值采用奇异值分解算法（SVD）进行主成分分析，并结合相关性分析筛选出相关性较高的输入集主成分，其中，提取小时值主成分后，由于经预处理处理后 9 风向指标具有一定的离散型，故使用 Spearman 相关性分析标签集中四种变化较大的污染物扩散条件指数与各气象成分的相关性（结果见表 4）。然后进行跨气象要素的主成分分析，提取互相相关的气象因素，最终将输入集的维度降至个位数级别。

考虑到时序相关性可能对建模存在较大影响，团队将每日的扩散条件指数与前一天对应气象主成分指标作相关性分析。其结果如图 3 所示。由图可知，所有指标的相关性绝对值均未能超过 0.15，可以确定以日值为样本的数据集不具有时序性。

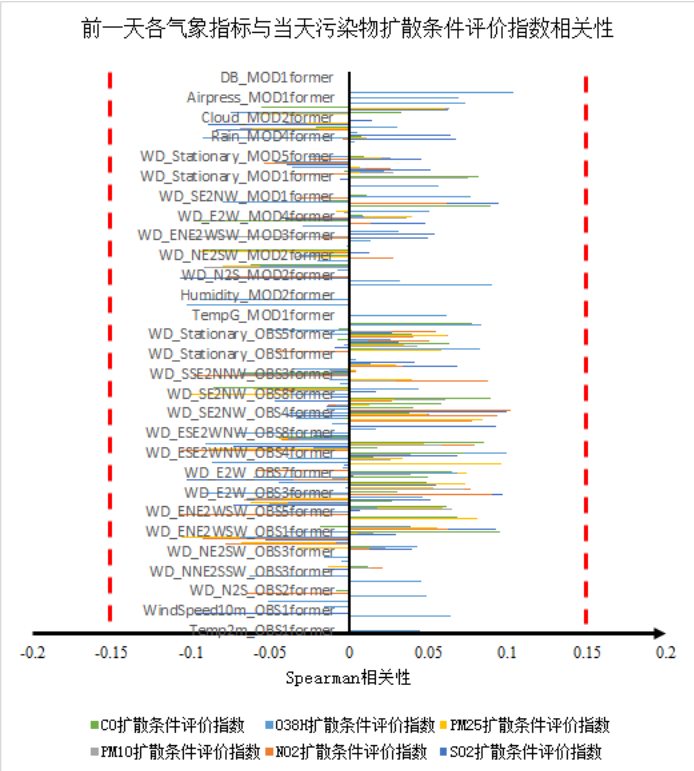



图 3　前一天各气象指标与当天污染物扩散条件评价指数相关性

表 4　Spearman 相关性较高的部分主成分结果及系数条形图

| 气象指标 \ 污染物扩散条件 | | $NO_2$ 扩散条件 | $PM_{10}$ 扩散条件 | $PM_{2.5}$ 扩散条件 | $O_3$ 扩散条件 | 0-23时系数条形图 |
|---|---|---|---|---|---|---|
| 实测气象 | 日平均近地温度 | -0.55609 | -0.53167 | -0.56694 | 0.134254 | |
| | 晨间湿度 | -0.20498 | -0.54592 | -0.4743 | -0.38395 | |
| | 午后湿度 | -0.17329 | | | 0.477208 | |
| | 晨间大气压 | 0.518436 | 0.566378 | 0.567939 | | |
| | 日间近地风速 | -0.49 | -0.38238 | -0.46058 | -0.28726 | |
| | 晚间近地风速 | -0.42249 | -0.30333 | -0.27323 | 0.108588 | |
| | 南风向 | 0.439827 | 0.407955 | 0.430123 | | |
| | 凌晨北东北风向 | 0.502293 | 0.389011 | 0.400662 | | |
| | 日间北东北风向 | -0.27808 | -0.31855 | -0.32959 | -0.21214 | |
| | 晨间西南风向 | 0.428985 | 0.178339 | 0.185726 | | |
| | 晨间东北风向 | -0.24792 | -0.3593 | -0.37901 | -0.32233 | |
| | 凌晨东东北风向 | 0.308255 | | | | |
| | 晨间东东北-西西南风向 | -0.18561 | -0.26808 | -0.28794 | -0.27821 | |
| | 凌晨南东南-北西北风向 | -0.36737 | -0.34517 | -0.37806 | 0.142523 | |
| 一次预报气象 | 日平均近地温度 | -0.5325 | -0.55903 | -0.58774 | | |
| | 日平均地表温度 | -0.52741 | -0.52317 | -0.54555 | 0.129804 | |
| | 晨间至傍晚地表温度 | 0.352725 | 0.552499 | 0.556172 | 0.260572 | |
| | 日平均比湿 | -0.4899 | -0.64186 | -0.64134 | -0.1185 | |
| | 湿度的昼夜差异 | -0.17612 | -0.39635 | -0.34784 | -0.16128 | |
| | 日平均湿度 | -0.11734 | -0.42221 | -0.37667 | -0.36532 | |
| | 平均近地风速 | -0.39447 | -0.34458 | -0.42277 | -0.37856 | |
| | 北风向 | 0.454733 | 0.34804 | 0.370195 | | |
| | 北东北风向 | 0.468901 | 0.364551 | 0.379123 | | |
| | 晨间东北风向 | 0.504581 | 0.380676 | 0.367343 | | |
| | 晨间西南风向 | -0.12117 | -0.15871 | -0.20676 | -0.25338 | |
| | 晨间东东北风向 | 0.528798 | 0.372073 | 0.351986 | | |
| | 午后东东北-西西南风向 | -0.11751 | -0.13957 | -0.19247 | -0.24806 | |
| | 凌晨东风向 | 0.444808 | 0.25908 | 0.249718 | | |
| | 白天雨量 | -0.29555 | -0.56249 | -0.51868 | -0.28058 | |
| | 白天云量 | 0.167138 | -0.10501 | | -0.382 | |
| | 午后边界层高度 | -0.19176 | | | 0.424283 | |
| | 日平均边界层高度 | -0.56473 | -0.3429 | -0.40009 | 0.116277 | |
| | 晚间至凌晨大气压 | 0.515375 | 0.579369 | 0.584796 | | |
| | 日平均感热通量 | -0.47112 | -0.25969 | -0.30507 | 0.187704 | |
| | 日平均潜热通量 | -0.53042 | -0.39057 | -0.4382 | 0.222279 | |
| | 日平均长波辐射 | -0.44719 | -0.60654 | -0.61225 | -0.14646 | |
| | 日平均短波辐射 | -0.33684 | -0.11915 | -0.14246 | 0.368858 | |
| | 日平均地表太阳能 | -0.33681 | -0.1191 | -0.14241 | 0.368852 | |

注：表中空白项为显著性>0.05 相关性结果，不具有参考价值。

根据表 4 信息，可以总结出影响各污染物的主要气象指标。注意到与 $PM_{10}$ 和 $PM_{2.5}$ 扩散条件相关的气象指标大体一致，且 2 种污染物定义和性质相似，故将两者合称为“颗粒物”加以讨论。

- **$NO_2$**

  正相关实测指标：　晨间大气压、凌晨偏南至西南风向
  正相关一次预报指标：　傍晚至晨间地表温度、晨间南风、晨间北东北至偏东风、晚间至凌晨大气压
  负相关实测指标：　日平均近地温度、日间及晚间近地风速、午后南西南风、晨间东北风、凌晨南东南方
  负相关一次预报指标：　日平均近地与地表温度、日平均比湿、日平均近地风速、日均边界层高度、日均感热与潜热通量及长短波辐射、日均地表太阳能

- **颗粒物**

  正相关实测指标：　晨间大气压、凌晨南至南西南风
  正相关一次预报指标：　傍晚至晨间地表温度、晨间南风、凌晨北东北至东东北风、晚间至凌晨大气压
  负相关实测指标：　日均近地温度、晨间湿度、全天近地风速、午后南西南风、晨间东北风、凌晨南东南风
  负相关一次预报指标：　日平均近地与地表温度、日均比湿、湿度昼夜差异及平均湿度、日均近地风速、白天雨量、日平均边界层高度、日均长波辐射

- **$O_3$**

  正相关实测指标：　午后湿度
  正相关一次预报指标：　日均边界层高度、日均短波辐射与地表太阳能
  负相关实测指标：　晨间湿度、日间近地风速、午后南西南风、晨间东北风、晨间东东北-西西南风
  负相关一次预报指标：　日平均湿度、日均近地风速、晨间西南风向

之后，针对不同污染物标签集中高相关性的要素进行第二次主成分提取，得到相应主成分系数矩阵。忽略较小系数，可得到影响各污染物的主要气象因素如下：

- **$NO_2$**

由表 5 可知，影响 $NO_2$ 主要气象因素有：

1. 日均温度-晨间气压-东北风方向联合因素
2. 日均光照相关因素-日均云雨联合因素
3. 日间风速-日间东北与西南风联合因素
4. 日间风速-边界层高度联合因素
5. 风速-东东北风联合因素

表 5　$NO_2$ 跨指标成分系数矩阵

| | 成份 | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| 日平均近地温度 | .089 | -.041 | .011 | .029 | .104 |
| 晨间大气压 | -.086 | .059 | .026 | .011 | -.044 |
| 日间近地风速 | -.009 | .015 | -.183 | .309 | -.191 |
| 晚间近地风速 | -.045 | -.017 | -.004 | .022 | .116 |
| 凌晨南风向 | -.069 | .035 | -.128 | .127 | .205 |
| 凌晨南西南风向 | -.066 | .001 | -.012 | .197 | .270 |
| 晨间西南风向 | -.047 | -.071 | .203 | .159 | .123 |
| 凌晨东东北风向 | -.034 | -.085 | .243 | .127 | -.073 |
| 晨间东东北-西西南风向 | .015 | -.037 | .176 | .072 | -.423 |
| 日平均近地温度 | .089 | -.050 | .004 | .048 | .131 |
| 日平均地表温度 | .092 | -.026 | .025 | .032 | .134 |
| 傍晚至晨间地表温度 | -.054 | .143 | .081 | -.090 | -.035 |
| 日平均比湿 | .086 | -.077 | .003 | .010 | .025 |
| 日平均近地风速 | .001 | .002 | -.129 | .324 | -.266 |
| 晨间南风向 | -.073 | .013 | -.058 | .105 | .089 |
| 晨间东东北风向 | -.043 | -.062 | .184 | .117 | .191 |
| 凌晨东风向 | -.020 | -.063 | .255 | .061 | .033 |
| 白天雨量 | .021 | -.109 | .001 | -.006 | -.180 |
| 白天云量 | -.016 | -.142 | -.059 | .076 | .145 |
| 午后边界层高度 | .030 | .135 | .024 | .014 | .253 |
| 日平均边界层高度 | .057 | .083 | -.043 | .198 | .077 |
| 晚间至凌晨大气压 | -.086 | .062 | .029 | .003 | -.042 |
| 日平均感热通量 | .052 | .130 | .087 | .131 | -.107 |
| 日平均潜热通量 | .078 | .083 | .065 | .114 | .053 |
| 日平均长波辐射 | .076 | -.101 | -.035 | .068 | .125 |
| 日平均短波辐射 | .048 | .146 | .122 | .049 | .022 |
| 日平均地表太阳能 | .048 | .146 | .122 | .049 | .022 |

- **颗粒物**

由表 6 可知，影响颗粒物主要气象因素有：

1. 日均温度-晨间气压-偏东风联合因素
2. 日间风速-西南风及边界层高度联合因素
3. 日间风速-雨量-午后边界层高度联合因素
4. 风向单独因素
5. 晚间风速-晨间北东北至东东北风单独因素

表 6　颗粒物跨指标成分系数矩阵

| | 成份 | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| 日平均近地温度 | .092 | .031 | .005 | .179 | .003 |
| 晨间湿度 | .063 | .105 | -.177 | .003 | -.133 |
| 晨间大气压 | -.092 | -.042 | .029 | -.100 | .122 |
| 日间近地风速 | -.019 | .255 | .258 | .043 | -.093 |
| 晚间近地风速 | .049 | -.045 | .159 | -.066 | -.155 |
| 凌晨南风向 | -.081 | .068 | .109 | .155 | -.231 |
| 凌晨南西南风向 | -.073 | .092 | .012 | .257 | .080 |
| 午后南西南风向 | .009 | .255 | .085 | -.193 | .232 |
| 晨间东北风向 | .022 | .244 | -.017 | -.219 | .215 |
| 凌晨南东南风向 | .066 | .004 | -.057 | -.068 | .404 |
| 日平均近地温度 | .092 | .049 | -.004 | .197 | -.033 |
| 日平均地表温度 | .093 | .012 | .023 | .190 | -.011 |
| 日平均近地风速 | -.005 | .272 | .221 | -.017 | .033 |
| 晨间南风向 | -.082 | .053 | .028 | .109 | -.272 |
| 凌晨北东北风向 | -.087 | .079 | -.016 | .168 | -.094 |
| 晨间东北风向 | -.070 | .083 | -.145 | .279 | .199 |
| 晨间东东北风向 | -.042 | .055 | -.250 | .254 | .389 |
| 白天雨量 | .029 | .126 | -.253 | -.026 | -.274 |
| 午后边界层高度 | .017 | -.162 | .301 | .215 | .198 |
| 晚间至凌晨大气压 | -.092 | -.050 | .032 | -.105 | .140 |
| 日平均潜热通量 | .069 | -.032 | .225 | .179 | .150 |
| 日平均长波辐射 | .083 | .117 | -.082 | .167 | -.125 |

- **$O_3$**

由表 7 可知，影响 $O_3$ 浓度变化主要气象因素有：

1. 晨间湿度-日间地表温度-晨间东北风联合因素
2. 傍晚湿度-气压-风速联合因素
3. 傍晚湿度-晨间东北至东东北风联合因素
4. 傍晚湿度-晨间气压-日间北东北风联合因素

表 7　$O_3$ 跨指标成分系数矩阵

| | 成份 | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 晨间湿度 | .211 | -.191 | -.072 | -.008 |
| 午后湿度 | -.006 | -.227 | .351 | .469 |
| 晨间大气压 | -.179 | .203 | .072 | -.203 |
| 日间近地风速 | .058 | .241 | -.108 | -.300 |
| 午后南西南风向 | .151 | .159 | .271 | -.327 |
| 晨间东北风向 | .202 | .132 | .424 | -.028 |
| 晨间东东北-西西南风向 | .165 | .120 | .384 | .180 |
| 傍晚至晨间地表温度 | -.234 | .111 | .244 | .057 |
| 日平均湿度 | .220 | -.098 | -.268 | -.204 |
| 晨间西南风向 | .077 | .261 | -.250 | .372 |
| 午后东东北-西西南风向 | .076 | .237 | -.235 | .542 |

### 4.2.4 气象分类

通过主成分分析进行数据降维后，气象数据仍至少含 4 个维度，可视化性低，因而较难初步确定分类个数，故采用两步聚类（Two Step Clustering）算法对气象进行分类。

两步聚类是一种较为广泛使用的系统聚类（层次聚类）算法，可以套用不同的系统分类判断准则及距离函数。由于监测点 A 的日值数据量仅处在百级，量级较小，在第一步的预聚类时，拟采用赤池信息准则（AIC）评估分类个数变化时对应的分类效果，以减少过拟合情况的发生。同时，因考虑到样本同时具有数值型变量和分类型变量，在评估点与点之间的距离时，使用了对数似然距离，以 CF 树[7]存储这些样本点的节点信息及相互距离。

在预聚类生成的 CF 树基础上，得到使 AIC 准则达到极小值的分类个数，以系统聚类法完成聚类并输出各样本的类别，同时得到各个类的质心及方差（见附录 质心表），以便于输入新的数据时,可以根据与质心的对数似然距离进行归类。

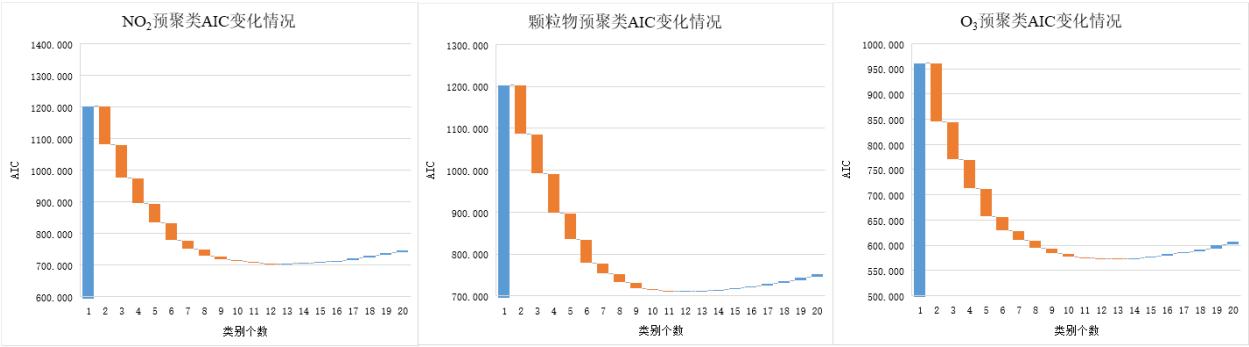



图 4　各主成分组预聚类 AIC 变化瀑布图

各主成分组在预聚类时 AIC 随类别个数变化情况如图 4，系统自动选取 AIC 变化趋于平稳的转折点，作为正式分类的类别数，$NO_2$、颗粒物、$O_3$ 分别为 6 类、9 类、5 类，具体分类获得质心与数据分布如下：

**(1) $NO_2$：**

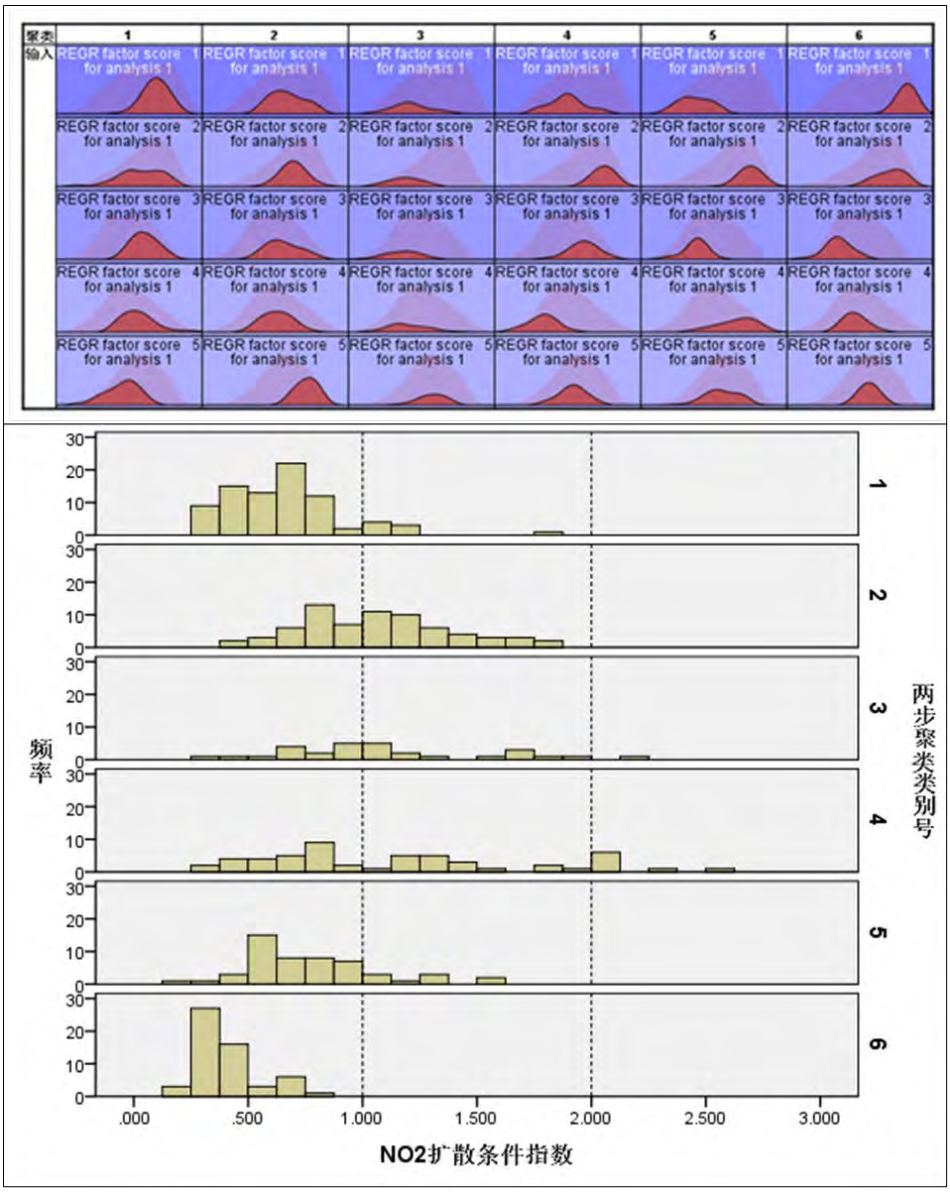



图 5　气象主成分分布特征（上）及各类气象下 $NO_2$ 污染扩散条件参数直方图（下）

由图 5 中气象主成分分布特征可知，可以看出类别 1 和 6 的温度-气压-偏东风联合因素质心均为正值，其余类别均为负；同时，注意到参数直方图中，类别 2、3、4 存在明显的超标风险，其中以类别 4 为最高，类别 3 次之，类别 2、3、4 在质心变化中最明显的为第三特征值，均值呈递增趋势。综上得到两项推论：

**推论 1** 在监测点 A 处，若当日平均温度较高且气压较低，并伴有凌晨至日间东北风时，$NO_2$ 扩散条件有利。

**推论 2** 在监测点 A 处，若当晨间风速较弱，且凌晨主导风向为东北至东东南风，较有可能出现 $NO_2$ 扩散条件不利的情形。

**(2) 颗粒物：**

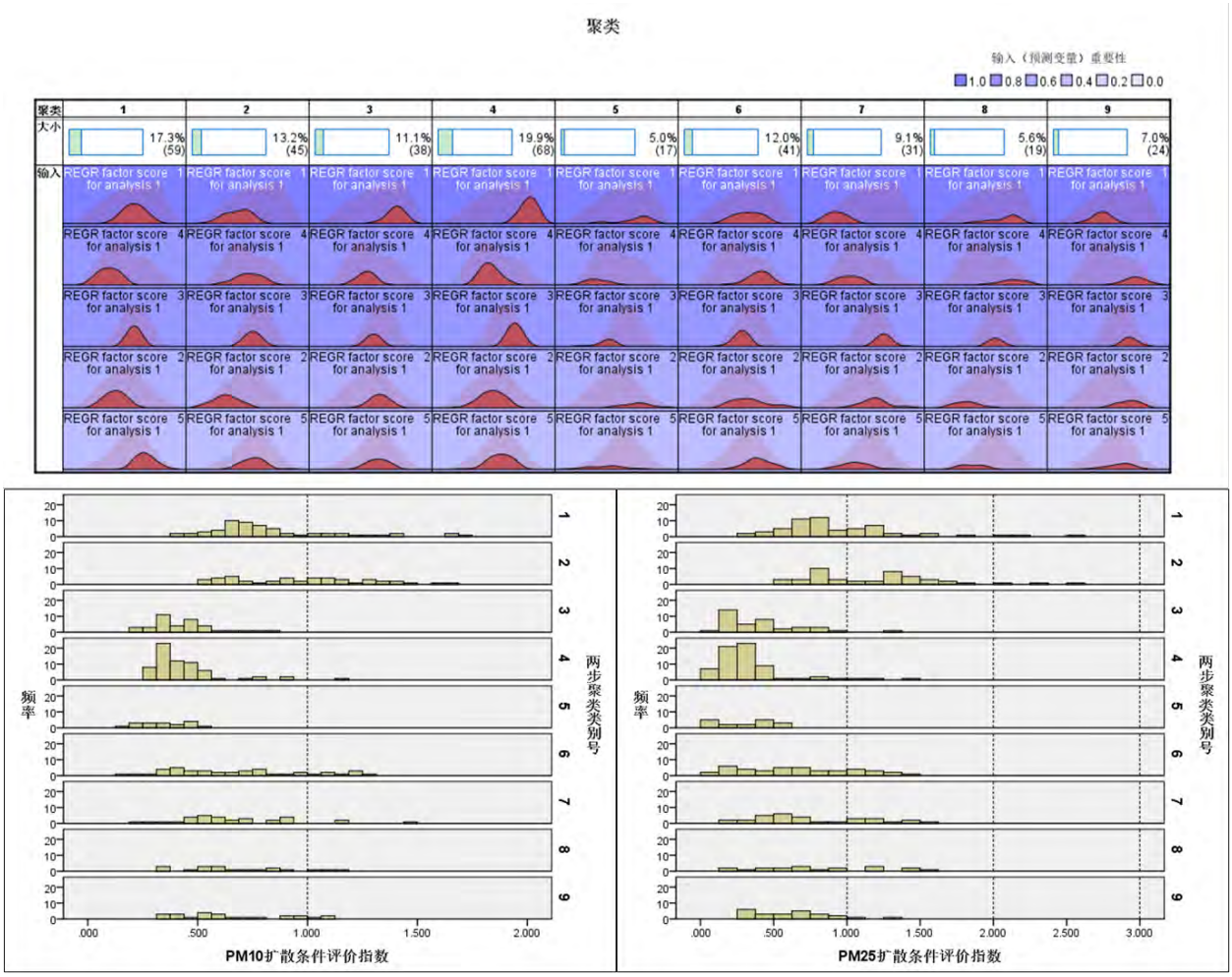



图 6 气象主成分分布特征（上）及各类气象下颗粒物（$PM_{10}$ 和 $PM_{2.5}$）污染扩散条件参数直方图（下）

由图 6 可知，类别 3、4、5、8 的 $PM_{10}$、$PM_{2.5}$ 扩散条件偏于有利，其共同特征为特征 1：日均温度-凌晨气压-偏北至东北风联合因素为 0 附近的值或正值。故提出以下推论：

**推论3** 在监测点A处，日均温度升高、凌晨气压降低，主导风向为偏北至东北风时，颗粒物扩散条件趋向有利。

**(3) $O_3$：**

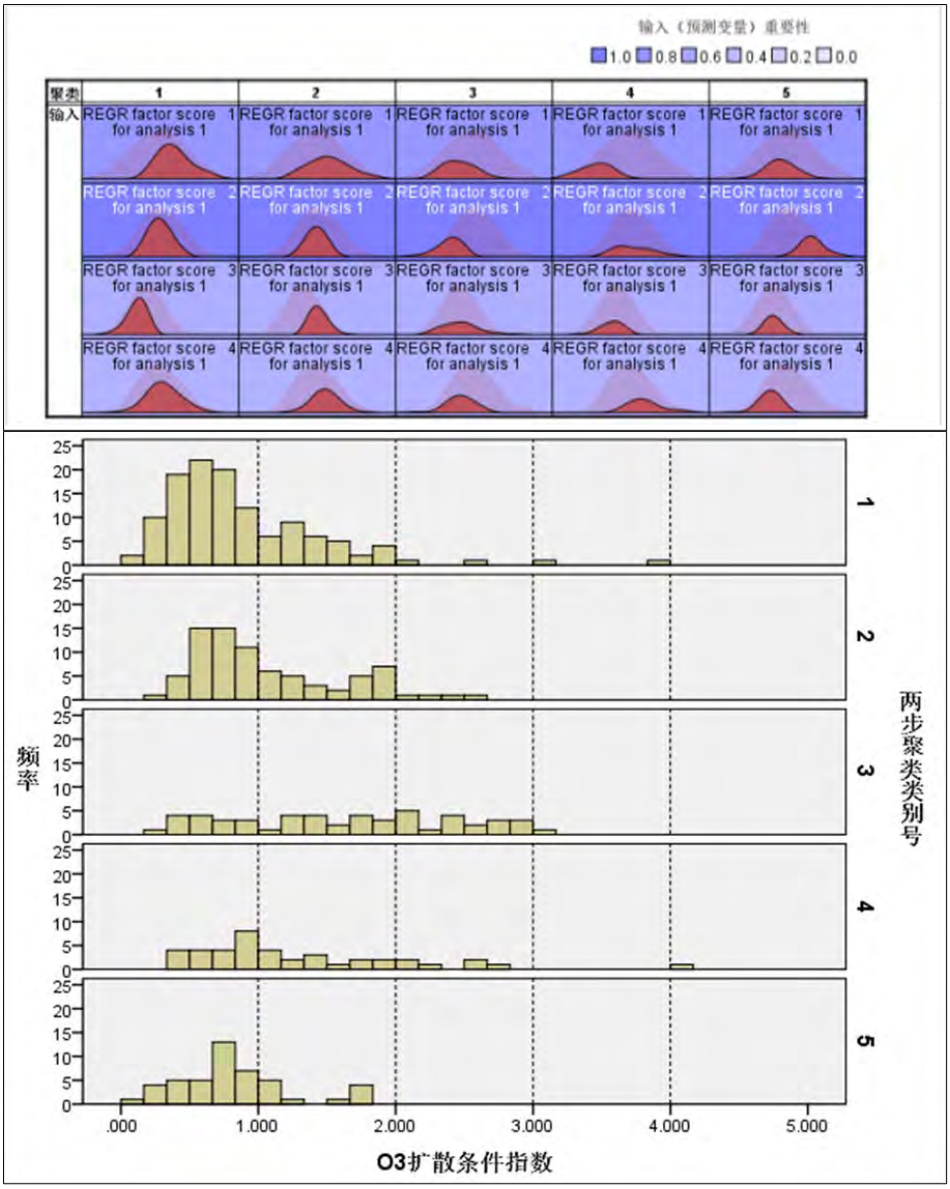



图 7　气象主成分分布特征（上）及各类气象下 $O_3$ 污染扩散条件参数直方图（下）

由图 7 可知，类别 3、4 的气象情况下出现 $O_3$ 扩散条件不利的频率较高，观察其气象特征分布情况可以发现，其特征指标 1、2 均处于较低水平；类别 5 的气象情况出现扩散条件一般的频率较小，同时未出现扩散条件不利情形，其特征指标 2 为较高值水平，故提出以下推论：

**推论 4** 在监测点 A 处，傍晚湿度较小，且日均气压、日均近地风速风速上升时，$O_3$ 扩散条件趋于有利。

**推论 5** 在监测点 A 处，晨间湿度较大，日均地表温度降低，日均气压下降，且晨间主导风向为东北风时，$O_3$ 扩散条件趋于有利。

### 4.2.5 问题 2 小结

综上所述，气象条件分类所用建模方法如图 8 所示：

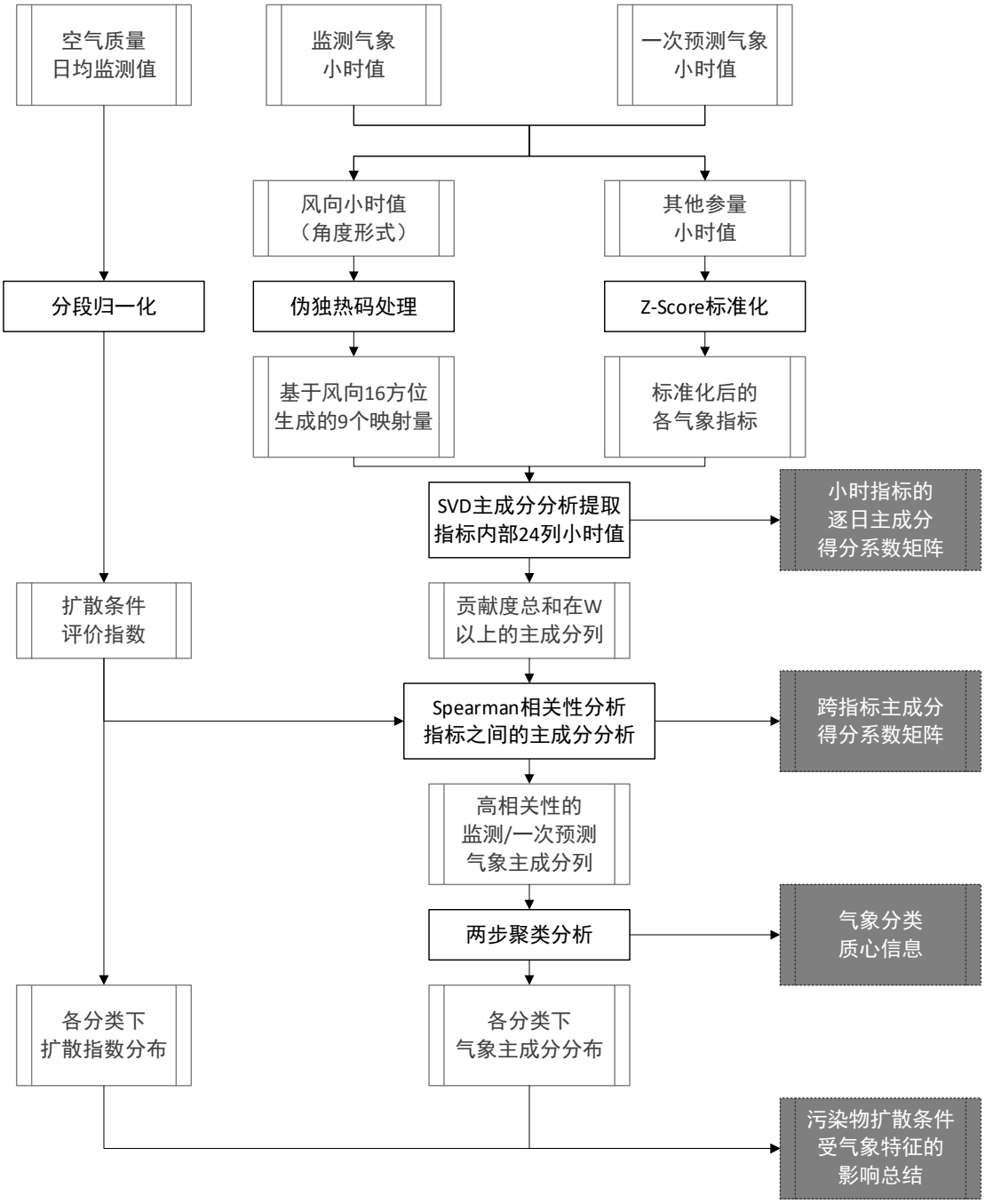
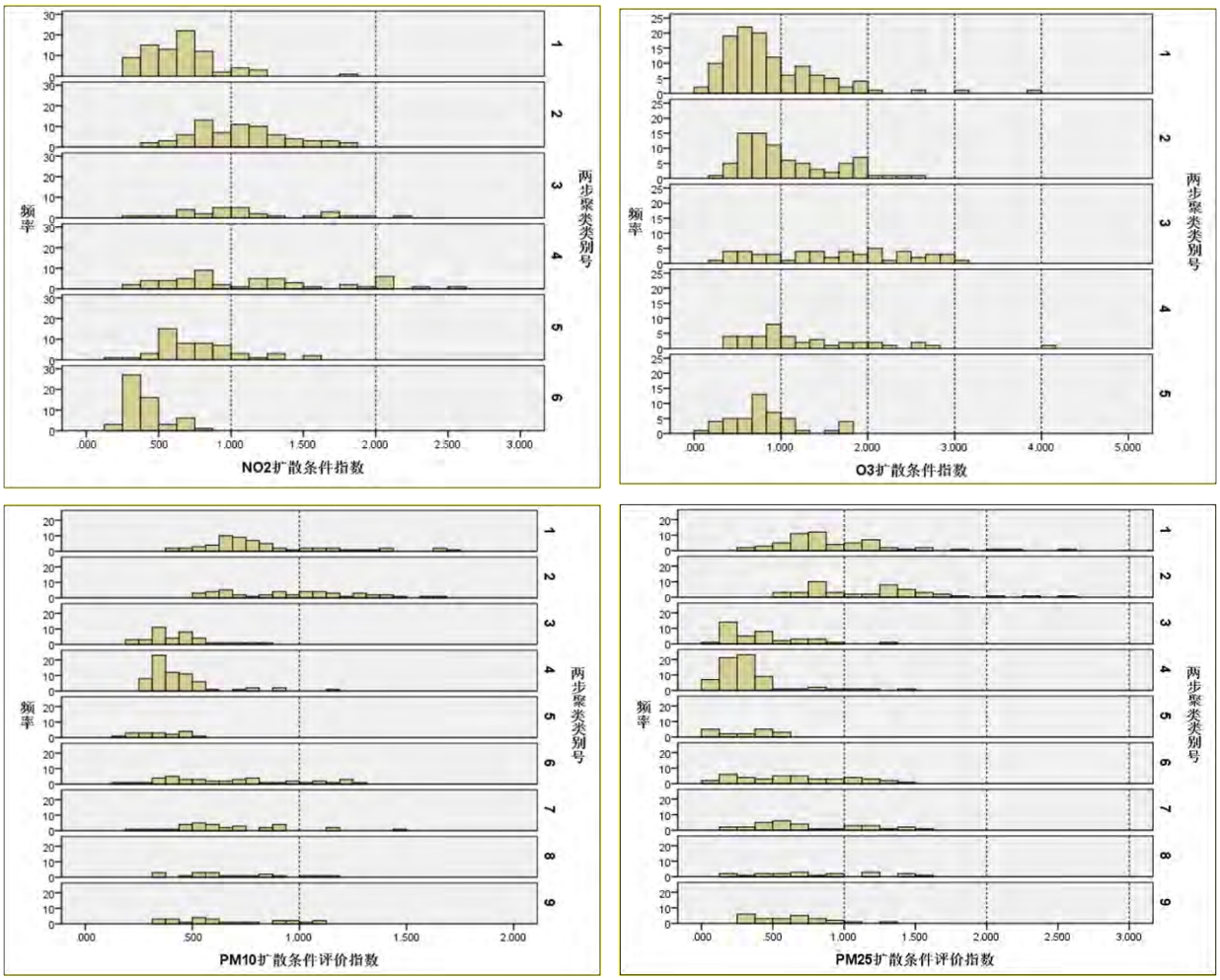



图 8　问题 2 气象条件分类建模方法

使用上诉分类方法，将监测点 A 处 $NO_2$、颗粒物（$PM_{10}$ 和 $PM_{2.5}$）和 $O_3$ 分成 6 类、9 类和 5 类气象，分类情况如下图：




其中，$NO_2$ 浓度受气象影响的变化情形可归纳：①当日平均温度较高且气压较低，并伴有凌晨至日间东北风时，污染物扩散条件有利；②当晨间风速较弱，且凌晨主导风向为东北至东东南风，较有可能出现污染物扩散条件不利的情形；

颗粒物（$PM_{10}$ 和 $PM_{2.5}$）浓度受气象影响的变化情形可归纳：当日均温度升高、凌晨气压降低，主导风向为偏北至东北风时，污染物扩散条件趋向有利。

$O_3$ 浓度受气象影响的变化情形可归纳：①傍晚湿度较小，且日均气压、日均近地风速风速上升时，$O_3$ 扩散条件趋于有利；②晨间湿度较大，日均地表温度降低，日均气压下降，且晨间主导风向为东北风时，$O_3$ 扩散条件趋于有利。

### 4.3 问题 3 建模与求解

#### 4.3.1 问题分析

题目要求建立模型用以预测未来三天 6 种常规污染物单日浓度值，并使相应的 AQI 预测结果误差尽量小，首要污染物预测准确度尽量高。根据 AQI 计算方法可知，若 6 种常规污染物的预测准确度不断提高，相应的 AQI 和首要污染物预测效果也会更好。故在问题 3（及问题 4）建模过程中，团队优先考虑如何提高 6 项常规污染物的预测准确度。

模型选择上，由问题 2 可知，前一日一次预报气象主要成分与当日污染物扩散条件无明显关系（Spearman 相关性均低于 0.2），即一次预报数据时序性相对较低；而大量文献表明实测数据具有较强时序性[8]。基于此，使用前者建模时可不考虑数据时序性的影响，而后者则需选用能处理时序问题的模型。

#### 4.3.2 数据预处理

根据题目要求，所建模型应适用于任意独立监测点，即可从 A、B、C 三个监测点中选择任一点位进行分析与建模，由于问题 4 也涉及到了监测点位 A，故本题以监测点 A 为例开展建模工作。以逐小时一次预报数据和实测数据为输入值，逐日实测数据为标签值，训练集与测试集的比例设置为 9:1（$O_3$ 除外，详见 4.3.5 $O_3$ 建模与评价）。

**(1) 异常值处理**

对监测点各项预报基础数据进行分析后，异常值处理原则为剔除异常高值和存在的负值。其中异常高值剔除方法与问题 2 相同。

**(2) 缺失值处理**

针对数据中原本存在的缺失值和经异常值处理后产生的空值，本文处理方式如下：

对于逐小时污染物浓度与气象一次预报数据，如存在缺失值则直接剔除缺失值所在时段。

对于逐小时污染物浓度与气象实测数据存在的缺失值，采用距缺失值最近的上一小时数据进行代替。以监测点 A 为例，其在 2019 年 4 月 23 日 15 点至 17 点 $PM_{10}$ 监测浓度存在缺失，故使用 2019 年 4 月 23 日 14 点的数据进行填补，具体效果如下图。

| 监测时间 | 地点 | $PM_{10}$监测浓度(μg/m³) |
|---|---|---|
| 2019/4/23 14:00 | 监测点A | 26 |
| 2019/4/23 15:00 | 监测点A | — |
| 2019/4/23 16:00 | 监测点A | — |
| 2019/4/23 17:00 | 监测点A | — |

| 监测时间 | 地点 | $PM_{10}$监测浓度(μg/m³) |
|---|---|---|
| 2019/4/23 14:00 | 监测点A | 26 |
| 2019/4/23 15:00 | 监测点A | 26 |
| 2019/4/23 16:00 | 监测点A | 26 |
| 2019/4/23 17:00 | 监测点A | 26 |

对于逐日污染物浓度实测数据存在的缺失值，先考虑能否使用相应时间内的逐小时污染物浓度数据进行处理以填补缺失，如不能，则用前一日数据进行代替。

**(3) 数据特征缩放**

本题数据特征缩放采用 Z-score 标准化方法（见式 (4)）。

#### 4.3.3 初步建模与评价

##### 4.3.3.1 特征选择

首先尝试结合问题 2 的结论进行特征选择，但使用问题 2 结论需具备当日 0~23 时气象实测数据，而进行空气质量预报不能获得当日 8 时以后的实测数据，如利用气象模拟值对其进行补全，由问题 2 可知气象模拟值与实测值之间相关性较差，补全后数据可参考性

低。综上，本题在进行二次预报建模过程中没有使用问题 2 所提供的特征相关性分析，而是采取随机森林平均不纯度减少的重要度评价方法自适应进行选择特征。其能根据随机森林训练过程自动得出特征重要度排序，因采用集成的思想，结果鲁棒性及稳定性均较强。本题“自适应”体现在结合目标污染物采用的具体模型，根据模型训练效果确定最佳特征个数。该方法具体原理如下：

随机森林[9]决策树在依据特征变量进行分裂生长时，信息的不纯度在不断降低。本题使用均方根误差 RA 作为不纯度评价指标，RA 计算公式为：

$$RA = \frac{\sqrt{(x_1-\bar{x})^2+(x_2-\bar{x})^2+\cdots+(x_{branch}-\bar{x})^2}}{N} \tag{5}$$

设某特征变量分裂前后的均方根误差分别为 $RA_1$、$RA_2$，则不纯度减少量为$\Delta RA = RA_1 - RA_2$，再考虑随机森林中所有的决策树，则计算出该特征变量对应的均方根误差的平均减少度$\overline{\Delta RA}$（见式 (6)），$\overline{\Delta RA}$即可作为该特征变量重要性的衡量指标，$\overline{\Delta RA}$值越大，特征变量的重要性越高。

$$\overline{\Delta RA} = AVERAGE(\Delta RA_1, \Delta RA_2, \cdots, \Delta RA_{div}) \tag{6}$$

输入监测点 A 已有时段内全部特征（15 种气象指标和 6 种常规污染物）的逐小时模拟值与相应目标污染物逐小时实测值，使用随机森林评价其各项特征的重要度，得到结果如所示：

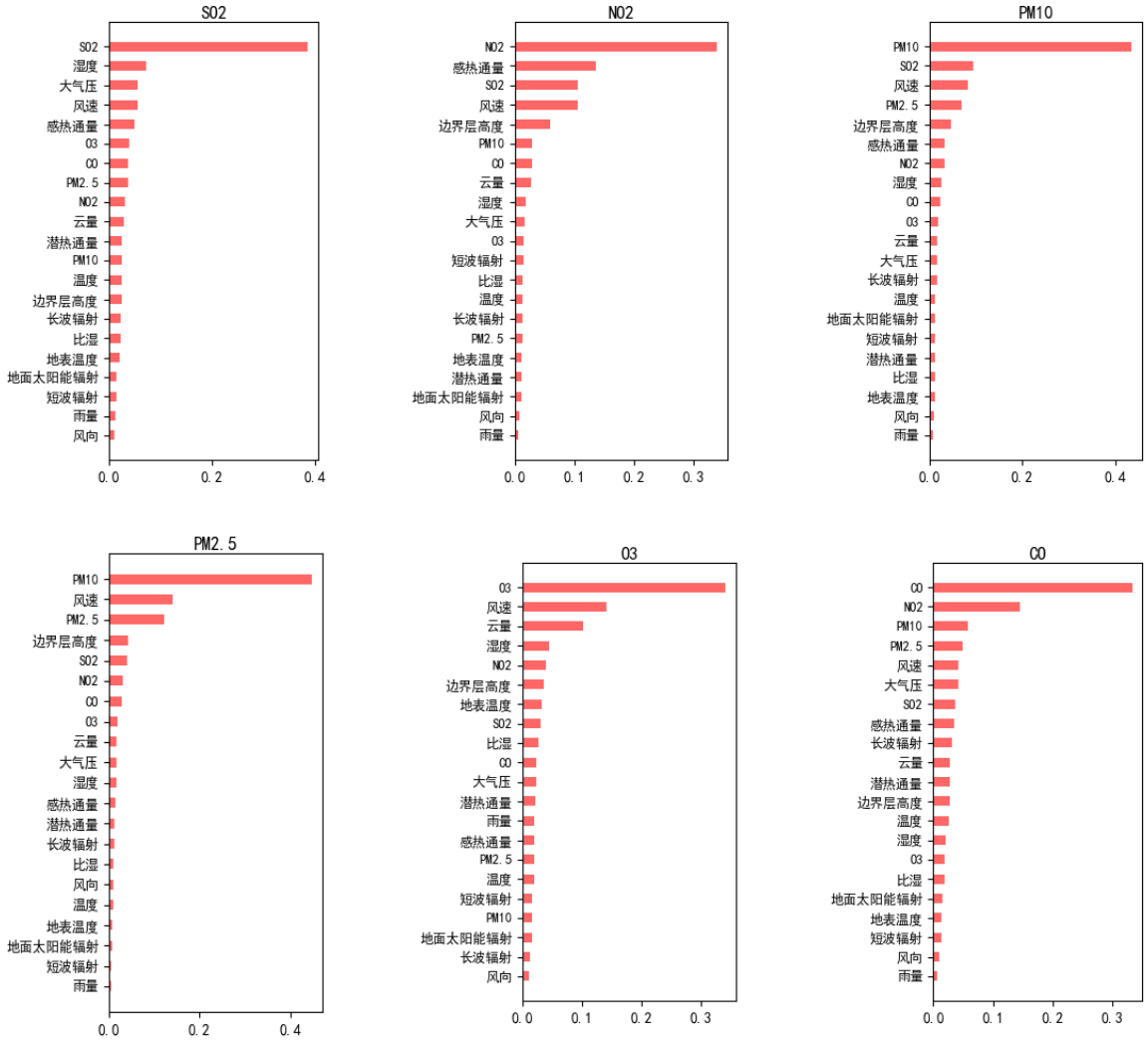



图 9　监测点 A 特征重要度排序情况

特征选择方面，团队考虑到不同的污染物具备不同的特性，会使用不同的预报模型，故本文将结合图 9 中特征重要度排序结果，根据具体模型训练效果选择所需特征。

#### 4.3.3.2 模型建立

当前广泛应用的空气质量预测二次建模方法有前馈神经网络、随机森林、支持向量回归[10]和极端梯度提升树[11]等，因此本文使用这 4 种方法分别进行建模，后进行模型择优等处理。4 种模型介绍如下：

**(1) 前馈神经网络（FNN）**

前馈神经网络是一种简单的梯度下降法，旨在最小化网络计算输出的误差。其训练过程是在权值和阈值之间不断进行调整，以使网络误差减小到预定的最小值或停止在预定的训练步骤。

具体建模时，选择绝对值损失函数（式 (7)）来描述模型经验风险；学习率选择据随污染物不同而变化；epoches 设置随污染物不同而有所变化；优化器选择也随污染物不同而变化，例如 $PM_{10}$ 选择 RMSprop[12]，$SO_2$ 选择 Adam[13]；同时，采取 L2 正则化以及 Dropout[14] 防止过拟合；训练平台选择 pytorch 框架。

$$error = \sum_{k=1}^{m} |z_k - y_k| \tag{7}$$

**(2) 随机森林（RF）**

随机森林作为一种运用广泛的集成模型，在预测问题中获得较多成功应用。本文采用改进后的随机森林算法，其改进思路如下：

使用有放回重复采样方法对样本集进行采样，从而随机生成 k 个训练集，并进一步生成与之相对用的决策树。其中训练第 i 棵决策树时，计算该决策树中所有预测点到实际点距离的方差$\delta^2(j)$，得到第 i 棵决策树的权重为：

$$w_r(i) = \frac{1/\delta^2(i)}{\sum_{j=1}^{T} 1/\delta^2(j)}$$

从而得到该算法预测值$\hat{u}(x)$：

$$\hat{u}(x) = \sum_{i=1}^{k} w_r(i) Z_i$$

式中，$Z_i$为第 i 棵决策树的预测值。

具体建模时，使用随机网格搜索法以及 10 折交叉验证确定森林数量、决策树最大深度、内部节点所需的最小样本数以及叶子节点所需的最小样本数。

**(3) 支持向量回归（SVR）**

支持向量回归是一种经典的机器学习算法，其找到的分割超平面具有很好的鲁棒性，且适用于小样本数据集。SVR 的主要计算流程可描述如下：

给定训练集$x_i \in \mathbb{R}(1 \le i \le m)$、标签值$y_i \in \mathbb{R}(1 \le i \le m)$和$\epsilon$-SVR，解决式 (8)问题：

$$\min_{w,b,\xi_i,\hat{\xi}_i} \{\frac{1}{2} w^T w + C \sum_{i=1}^{m} (\xi_i + \hat{\xi}_i)\} \tag{8}$$

$$s.t.\ \ y_i - w^T \phi(x_i) - b \le \epsilon + \xi_i$$

$$w^T \phi(x_i) + b - y_i \le \epsilon + \hat{\xi}_i$$

$$\xi_i, \hat{\xi}_i \ge 0, i = 1,2,\cdots,m.$$

其对偶问题如式 (9)所示：

$$\max_{\alpha,\hat{\alpha}} \sum_{i=1}^{m} y_i(\hat{\alpha}_i - \alpha_i) - \epsilon(\hat{\alpha}_i + \alpha_i) - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\hat{\alpha}_i - \alpha_i)\left(\hat{\alpha}_j - \alpha_j\right)\phi(x_i^T)\phi\left(x_j^T\right) \quad (9)$$

$$s.t. \quad \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i) = 0$$

其中，$0 \le \alpha_i$，$\hat{\alpha}_i \le C$，$1 \le i \le m$。

上述过程中需满足 KKT 条件。得到 SVR 的解形如：

$$f(x) = \sum_{i=0}^{m}(\hat{\alpha}_i - \alpha_i)\, K(x_i^T, x) + b$$

$K(x_i^T, x)$为核函数，本文采用高斯核函数，即：

$$K(x_i^T, x) = exp\left(\frac{-\|x - x_i\|^2}{2\sigma^2}\right) \quad (10)$$

式中，$\sigma$为可选择的参数。

式 (8)和式 (10)中的参数$C$、$\sigma$较难确定，建模时采用网格搜索法以及 5 折交叉验证确定两者最佳值。其中$C$搜索范围为[1,1000]，$\sigma$搜索范围为[0.1,500]。

**(4) 极端梯度提升树（XGBoost）**

极端梯度提升树是一种集成机器学习算法，其在传统提升树的基础上引入正则化项来控制复杂度，对代价函数做二阶 Talor 展开，并对树结构进行一定变换，使整个代价函数由组成树的数量表示。XGBoost 能自动学习分裂方向，支持各种抽样。

XGBoost 的目标函数为：

$$obj^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j}{H_j + \lambda} + \gamma\mathcal{N}$$

式中$\mathcal{N}$表示数的子节点的个数。

本题使用开源 python 库 xgboost 进行模型搭建，对决策树最大深度、$\gamma$、$\lambda$等参数使用人工调参方法进行模型训练，评价方法采用 5 折交叉验证。

**4.3.3.3 模型评价标准**

本文主要采用可决系数、平均绝对误差和均方根误差三种性能评价指标对测试集预测结果进行评价。其中：

**可决系数$R^2$**表示自变量解释的变异程度占总的变异程度的比例，$R^2$越接近 1，表示该模型的准确度越高，$R^2$有可能为负值。

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i - f(x_i)\right)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

**平均绝对误差 MAE** 常用于判断回归模型的误差。当回归模型的预测误差越小，MAE 就越小，反之则越大。

$$MAE = \frac{1}{N}\sum_{i=1}^{n}|y_i - f(x_i)|$$

**均方根误差 RMSE** 是真实值与预测值差值的平方与观测次数比值的平方根，能够很好地反映出测量的精密度。RMSE 越小，表示模型精密度越高。

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{n}\left(y_i - f(x)\right)^2}$$

#### 4.3.3.4 模型比较

以监测点 A 的逐小时模拟数据作为输入数据，注意到数据中运行时间当天对应产生当天、明天、后天连续三天的模拟值，故本文统一使用当天模拟的当天数值作为输入数值，用以训练上述 4 种模型，并进行相应污染物浓度预测。

图 10 为 4 种常用二次预报模型（FNN、RF、SVR、XGBoost）及 WRF-CMAQ 相应的评价指标结果。其中，由于 $SO_2$ 和 CO 的变异范围不大，且数值值离散程度较高，小幅度误差便会导致 $R^2$ 剧烈变化，故 $SO_2$ 和 CO 的 $R^2$ 不具备评价代表性。基于此，图 10 及后文所有评价指标结果图中均未将 $R^2$ 值列出。

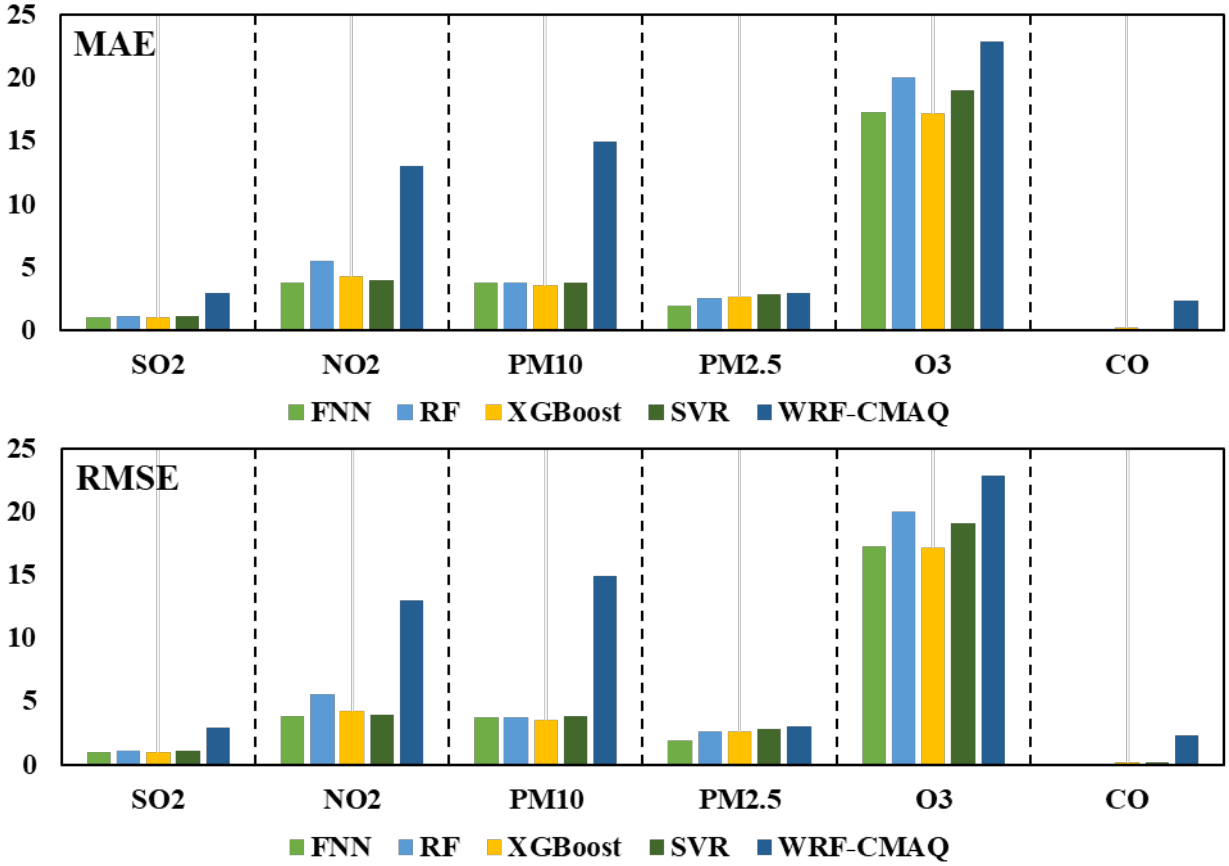



图 10　6 种常规污染物的 FNN、RF、SVR、XGBoost 及 WRF-CMAQ 评价指标结果

由图可知，①相对于 WRF-CMAQ，4 种二次预报模型的预测效果均出现提升；②4 种二次预报模型在不同评价指标下最优项并不完全相同。其中，$SO_2$、$NO_2$、$PM_{2.5}$、CO 的 MAE 和 RMSE 均为 FNN 占优，$O_3$ 的 RMSE 为 FNN 最好，而 MAE 为 XGBoost 更佳，$PM_{10}$ 的 MAE 和 RMSE 最优值分别出现在 SVR 和 XGBoost 上，但 4 种模型预测效果差距不大。总的来说，FNN 在各项评价指标中的表现相对最好；③$O_3$ 的 MAE 和 RMSE 均偏高由题可知，$O_3$ 为二次污染物，是在大气中经过一系列化学及光化学反应生成的，该过程受到一次污染物及气象等的影响，这导致对 $O_3$ 浓度变化的预测难度增高，故团队将在后文对 $O_3$ 进行单独分析和建模。对于除 $O_3$ 外的其它 5 种污染物，团队选择 FNN 作为其二次预报模型。

### 4.3.4 模型组合与评价

考虑到题目要求对 2021.7.13~2021.7.15 三天进行预测，而一次预报数据并未提供 2021.7.14 模拟 2021.7.14 及 2021.7.15 模拟 2021.7.15 的数据，故本文采用 2021.7.13 模拟的 2021.7.14 和 2021.7.15 数据进行替代。三天评价指标结果如图 11 所示。：

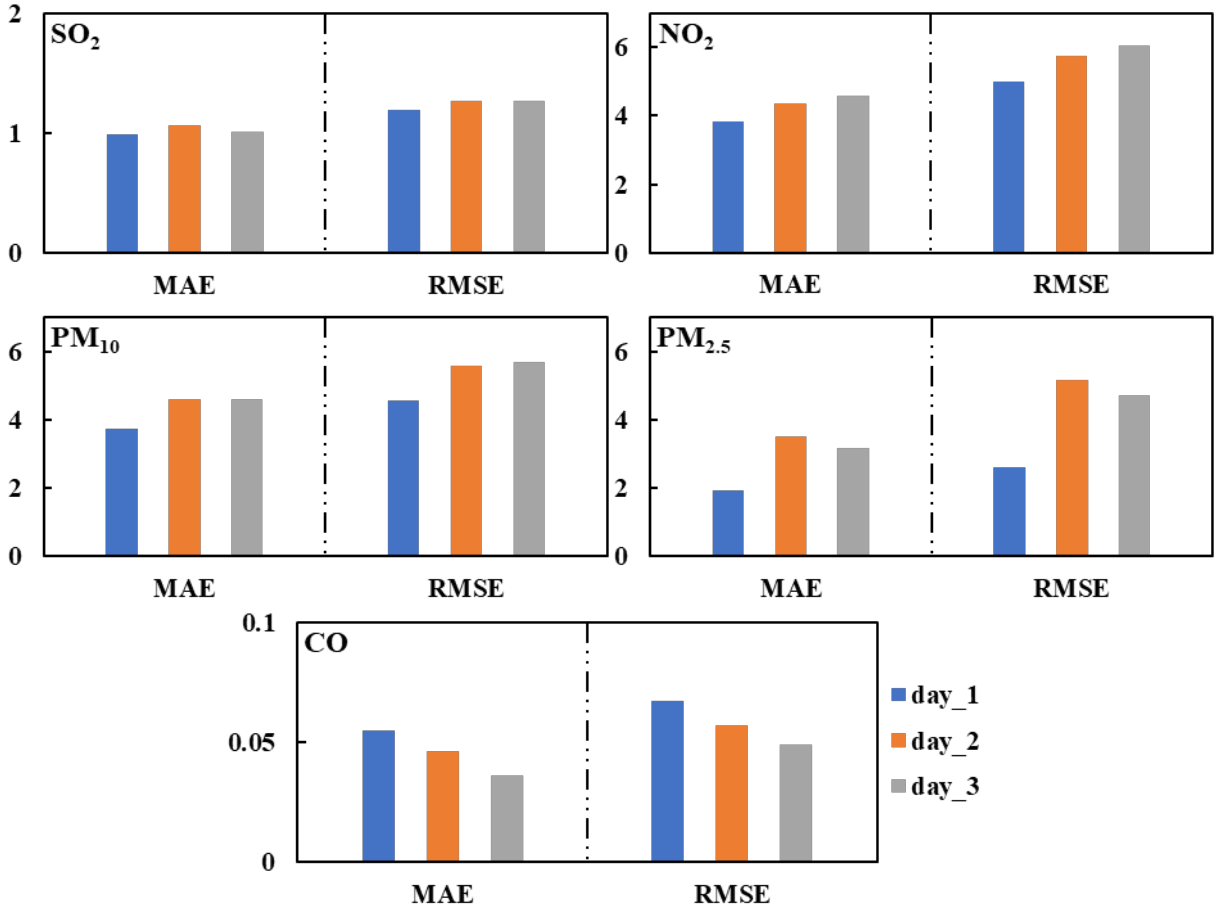



图 11　除 $O_3$ 外 5 种常规污染物 FNN 三天评价指标结果

由图可知，受不确定性累加的影响，对于大部分污染物模型第二、三天预测效果有所下降。为尽可能提高模型预测效果，有文献指出[15]，综合利用不同单项预测模型所得结果，以适当加权平均形式得出的组合预测模型，可以进一步提高预测精度。基于此，团队将污染物浓度逐小时实测数据也用于建模，并考虑组合多模型对污染物浓度进行预测。

但由图 10 和图 11 可知，对于 $SO_2$ 和 CO，FNN 的 MAE 和 RMSE 三天的数值都相对较低，即模型预测结果误差较小。为精简模型，避免计算资源浪费，团队使用 FNN 作为 $SO_2$ 和 CO 的最终二次预报模型，对于 $NO_2$、$PM_{10}$ 和 $PM_{2.5}$ 则进行进一步修正。

#### 4.3.4.1 LSTM_FC 建模

考虑到污染物浓度的逐小时实测数据具有较强时序性，所用模型应对此特性加以利用。长短期记忆网络(LSTM)[16]作为循环神经网络的一个变体，有效解决了简单循环神经网络的梯度爆炸或消失问题，在时间序列问题的处理上取得较好表现。故选择 LSTM 作为逐小时实测数据所用模型，此外，为应对 LSTM 输出局限性，即灵活控制预报时长，在其后添加一层全连接层（FC）。

LSTM_FC 的工作原理可以用以下公式表示：

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = o_t \odot tanh(c_t)$$

式中，$i_t$,$f_t$,$o_t$分别为输入门、遗忘门、输出门；$W_*$,$U_*$,$b_*$为可学习参数，其中$*\in\{i,f,o,c\}$；$h_t$，$c_t$为 t 时刻的隐状态和记忆单元；$h_{t-1}$,$c_{t-1}$为上一时刻的隐状态和记忆单元；$\tilde{c}_t$是通过非线性函数得到的候选状态；$x_t$为 t 时刻的训练集输入；$\sigma(\cdot)$为 sigma 函数。

选择 LSTM 输出的最后一个$h_t$（即最近时刻）作为 LSTM 拟合值，将其输入到全连接层（FC）进行线性变换，从而得到 LSTM_FC 的最终预报值。

与 FNN 类似，选择绝对值损失函数来描述模型经验风险，学习率选择据随污染物不同而变化；epoches 设置随污染物不同而有所变化；优化器选择 Adam 或 RMSprop；选择 L2 正则化以及 Dropout 防止过拟合；训练平台选择 pytorch 框架。

LSTM_FC 输入数据选用当天 7:00 前的 24 个小时实测值（即前一天 8:00 至预报当天 7:00 的逐小时实测值）。另外需要注意的是，由于模型输入的是污染物逐小时实测数据，对其进行特征选择的过程中，其所用特征为逐小时实测数据中除目标污染物外其它 5 项常规污染物及实测值中气象指标共 10 种特征。利用随机森林得到监测点 A 逐小时实测值的各特征重要度排序结果如图所示：

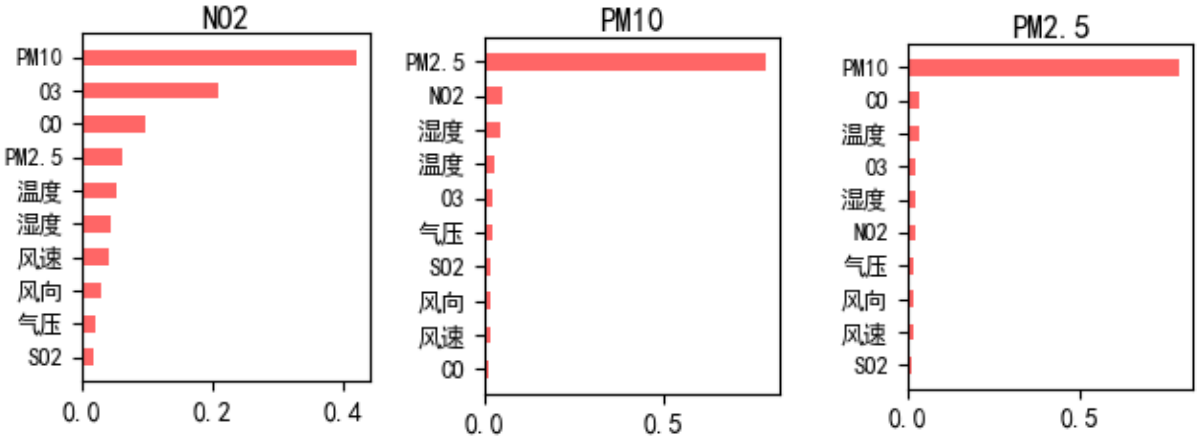



图 12　监测点 A 特征重要度排序情况（实测数据）

将训练集和测试集数据输入到 LSTM_FC 模型中，完成模型训练并预测相应浓度，其中 LSTM_FC 评价结果如所示。与 FNN 情况相似，LSTM_FC 使用历史 24h 数据预测未来三天浓度值，后两天预测效果会出现大幅度下降，故本文使用 LSTM_FC 第一天预测值对 FNN 未来三天的预测值进行修正。其预报效果如下图所示，各污染物预测效果整体较好，因此可用于对一次预报数据所建模型进行修正。

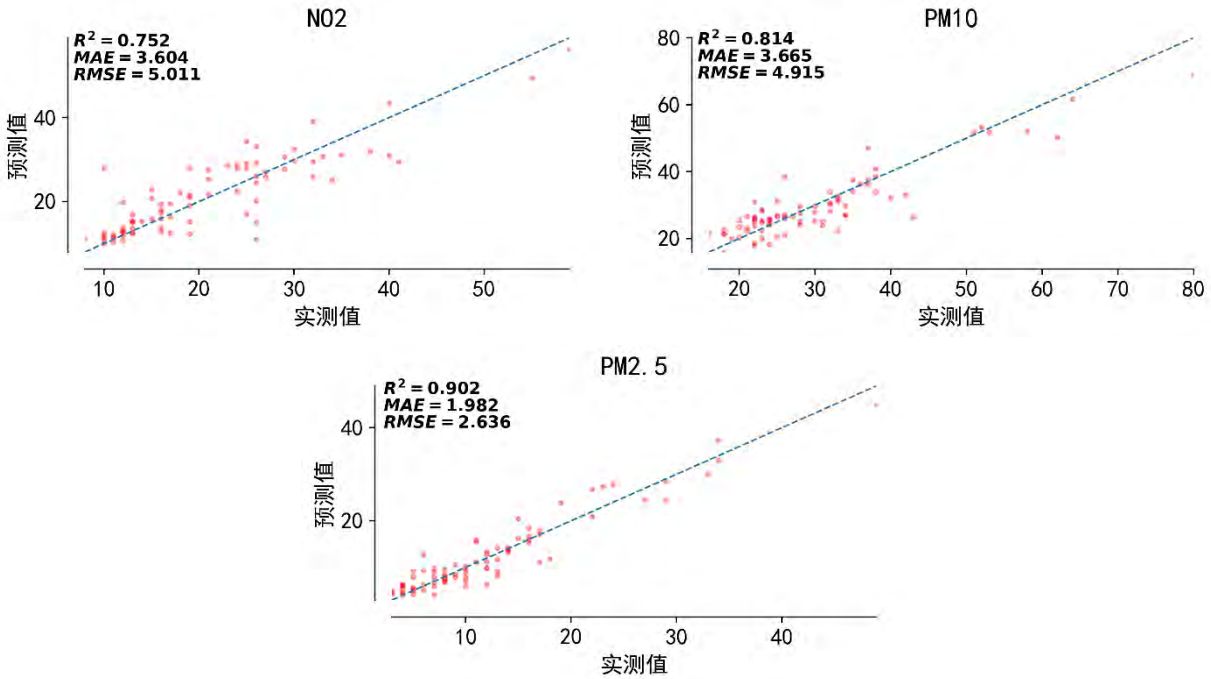



图 13　$NO_2$、$PM_{10}$ 和 $PM_{2.5}$ LSTM_FC 评价指标结果（以预报第一天为例）

#### 4.3.4.2 模型组合方法

考虑到本文选用的两种单模型 FNN 和 LSTM 学习能力均较强，若选择复杂度较高的回归模型，可能会加剧组合模型的过拟合情况。故使用 Lasso 回归[17]进行模型组合。

Lasso 作为一种复杂度相对较低的回归模型，其通过系数收缩及将某些系数的权重降为 0 来降低方差，过程中虽会使偏差稍有增加，但会减少过拟合现象的出现、提高模型可解释性，且结果也更为稳定。Lasso 模型的经验风险如下：

$$J(w) = \frac{1}{n}\|Z - W^T X\|^2 + \lambda\|W\|_1$$

式中，$\lambda$为超参数，L1 范数即向量中各元素绝对值之和。

#### 4.3.4.3 模型组合评价

经过分析和测试，本文最终使用 FNN 和 LSTM_FC 的 Lasso 回归组合模型对 $NO_2$、$PM_{10}$ 和 $PM_{2.5}$ 进行预测。两种模型预报值通过 Lasso 回归进行组合预测。该组合模型预测其评价结果见图 14。

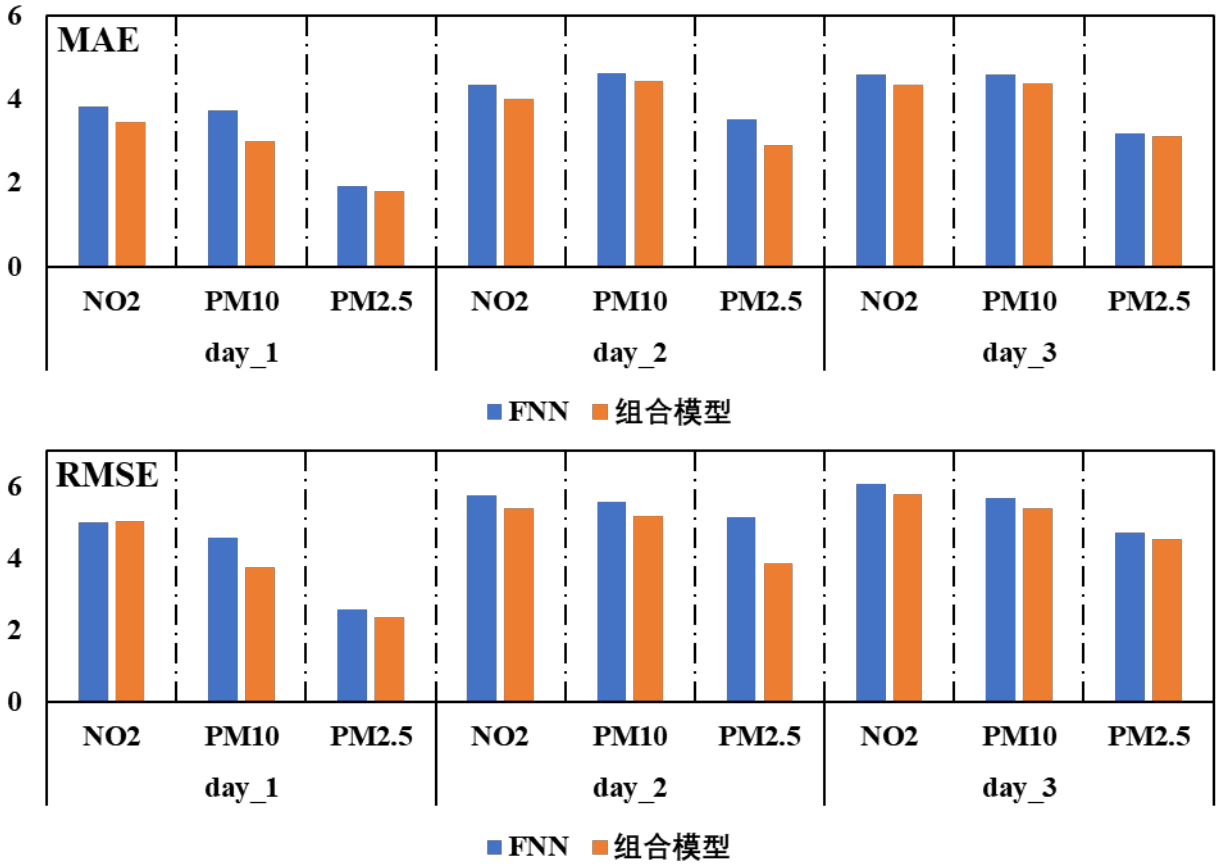



图 14 $NO_2$、$PM_{10}$ 和 $PM_{2.5}$ 的组合模型及 FNN 三天评价指标结果

由图可知，对于 MAE 及 RMSE，组合模型数值整体优于 FNN。故本文采用其作为 $NO_2$、$PM_{10}$ 和 $PM_{2.5}$ 的最终二次预报模型。

### 4.3.5 $O_3$ 建模与评价

由于 $O_3$ 本身的特殊性，其预测难度相对更高，需要进行更为精细的处理。经过查阅文献和多次实验，团队提出了 CEEMDAN_IPSOSE_SVR 方法用于 $O_3$ 浓度的预测。其中，CEEMDAN 即自适应噪声完备集合经验模态分解，是由 Torres M E.等[18]提出的一种新型信号分解算法，可用于对数据进行降噪；SVR 作为 $O_3$ 预报模型的基学习器；IPSOSE 是指使用改进的粒子群算法进行寻优和集成。具体介绍如下：

#### 4.3.5.1 CEEMDAN 降噪

在模型训练之前，考虑到实际工作中站点监测仪器及其周边存在周期性异常，会导致包括 O3 在内的污染物监测数据存在一定的噪音，从而影响建模质量，故团队决定采用 CEEMDAN 对 O3 日最大八小时滑动平均监测数据进行降噪。

首先向 $O_3$ 监测数据添加 100 次高斯白噪音，再使用经验模态分解（EMD）将加噪序列分解为多个子序列并求平均，不断迭代获得最后分解后的子序列，而噪声通常存在于某个或某些子序列中（如图 15 所示），因此本文使用排列熵的方法将存在噪声的子序列挑选出来，再使用软阈值法对其进行降噪，具体降噪流程见附录 CEEMDAN 降噪流程。

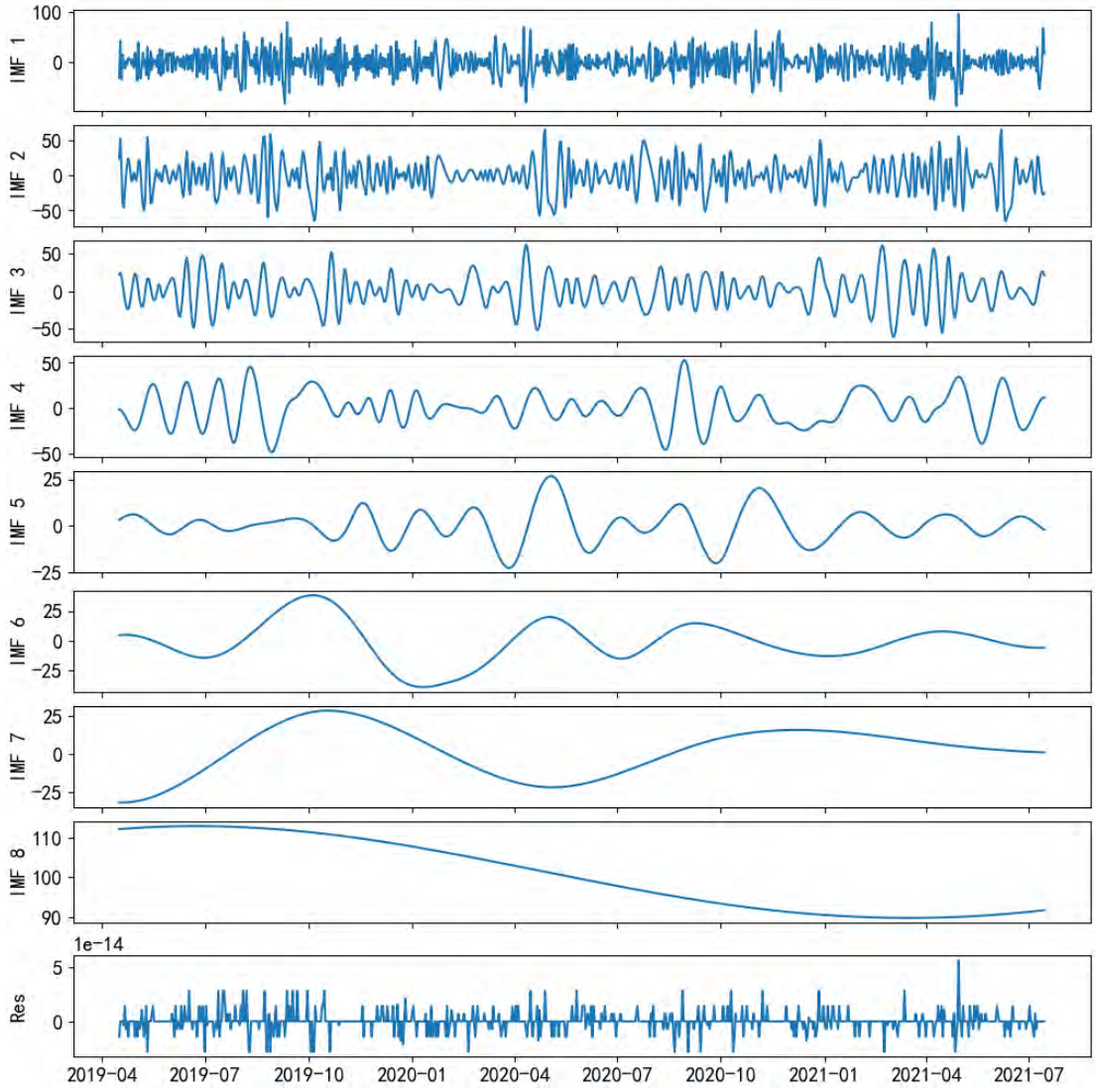



图 15　CEEMDAN 分解结果

经过上述处理，得到降噪后的 $O_3$ 监测数据，降噪前后数据对比如图 16 所示：

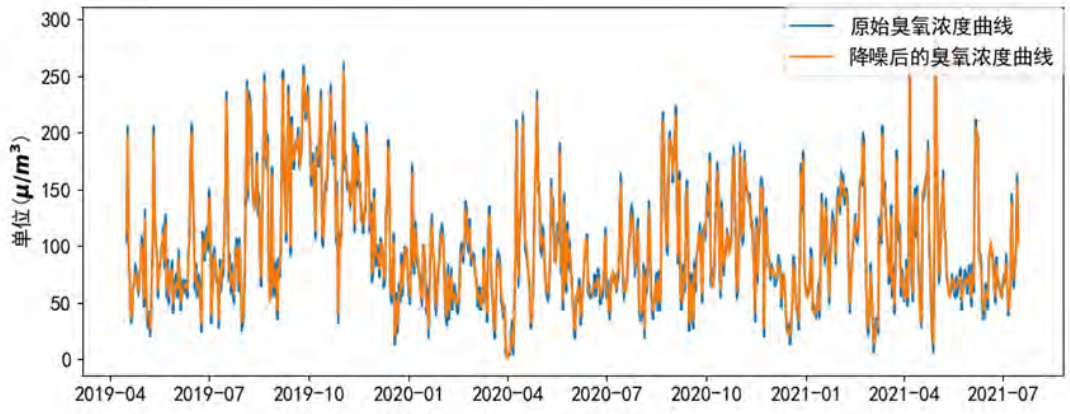



图 16　$O_3$ 日最大八小时滑动平均监测数据降噪前后对比

#### 4.3.5.2 基学习器选择

数据降噪后开始选择基学习器。由前文可知，在常用方法中 FNN 的预测效果相对更好。但由于 $O_3$ 本身的特殊性和测试数据的具体情况等，团队考虑使用支持向量回归（SVR）作为预报基学习器，具体原因如下：①SVR 针对少量的数据样本就能实现最优解，在一定程度上解决了 FNN 的局部极值问题；②SVR 通过空间映射能有效减少“维数灾难”的发生；③SVR 采用结构风险最小化使其具有很好的泛化性能。基于上述优点，本文在预测 $O_3$ 浓度时，最终选择 SVR 作为基学习器。

#### 4.3.5.3 IPSOSE 建模

对选择好的 SVR 使用集成思想进行进一步处理。周志华等[19]提出的选择性集成技术证明了通过选择部分个体学习器来构建集成学习器，其效果要优于使用单个学习器或多个学习器直接集成。此方法要求从训练的所有学习器中选择出差异性大，泛化能力强的个体加以集成，以获得更好的性能。基于此，团队首先采用 Bootstrap 方法从训练集内抽取不同子集以训练得到多个基学习器 $SVR_i$，IPSOSE 算法则使用改进的粒子群算法 IPSO 对这些 $SVR_i$ 进行筛选和集成寻优，得到最终集成模型 $SVR^*$。

IPSOSE 具体建模过程描述如下：

**步骤 1**：首先将 $O_3$ 训练集与测试集的比例确定为 8:2，采用 Bootstrap 方法从训练集中生成 m 个训练子集，其中每个子集的样本数为原训练集样本数的 50%。用 m 个子集训练 SVR，得到 m 个互不相同的基学习器 $SVR_i$ 及各自对应的可决系数得分 $r2\_score_i$。本文取 m=40。即一组 SVR 集为：

| $SVR_1$ | $SVR_2$ | …… | $SVR_{40}$ |
|---|---|---|---|

m=40

**步骤 2**：进行 IPSOSE 相关系数的设置及初始化。

IPSOSE 算法的相关参数主要包括粒子数目 K、总迭代次数 T、惯性权重$\omega_t$、学习因子$c1$和$c2$、粒子速度 $V_k^i(t)$、粒子位置 $loc_k^i(t)$、阈值 $loc_d$ 等。

本文设置粒子数目 K=20；总迭代次数 T=20，即迭代次数$t \in \{1,2,\ldots,20\}$；惯性权重$\omega$会随迭代过程进行更新，本文设置其初始值$\omega_0 = 0.9$；学习因子$c1$和$c2$均设置为 0.2；粒子 k 的位置初始值 $loc_k^i(0)$是随机生成的介于 0~1 之间的数值，存储在 $Solution_k(0)$中；阈值 $loc_d$ 取 0.5；如图，每个粒子 k 均具备一行 $Solution_k(0)$，其中 $loc_k^i$ 与 $SVR_i$ 一一对应。

| $loc_k^1(0)$ | $loc_k^2(0)$ | …… | $loc_k^{40}(0)$ |
|---|---|---|---|

m=40

**步骤 3**：由步骤 2 可知，取 K=20，即存在：

| | | | |
|---|---|---|---|
| $SVR_1$ | $SVR_2$ | …… | $SVR_{40}$ |
| …… | …… | …… | …… |
| $SVR_1$ | $SVR_2$ | …… | $SVR_{40}$ |
| …… | …… | …… | …… |
| $SVR_1$ | $SVR_2$ | …… | $SVR_{40}$ |

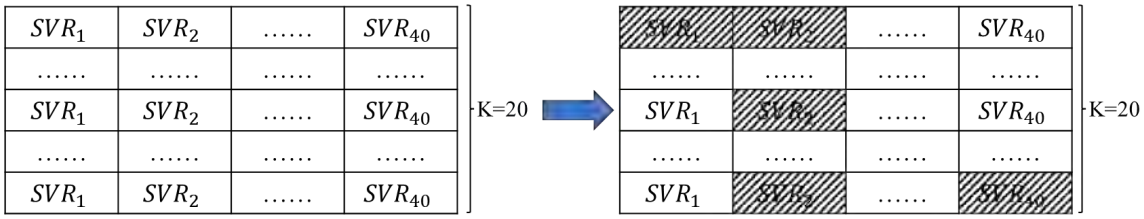
K=20

对应的 Solution 如下图所示：

| | | | |
|---|---|---|---|
| $loc_1^1$ | $loc_1^2$ | …… | $loc_1^{40}$ |
| …… | …… | …… | …… |
| $loc_k^1$ | $loc_k^2$ | …… | $loc_k^{40}$ |
| …… | …… | …… | …… |
| $loc_{20}^1$ | $loc_{20}^2$ | …… | $loc_{20}^{40}$ |

K=20

判断各行初始 $Solution_k(0)$中 $loc_k^i(0)$与阈值 $loc_d$ 的关系：当 $loc_k^i(0) \geqslant loc_d$ 时，将 $loc_k^i(0)$ 赋值为 1，当 $loc_k^i(0) < loc_d$ 时，则 $loc_k^i(0)$赋值为 0。如 $loc_k^i(0)=1$，则保留该行中对应的基学习器 $SVR_i$，如 $loc_k^i(0)=0$，则舍去该基学习器 $SVR_i$，由此完成每一行基学习器的取舍。




注：图中有填充部分代表该处 SVRi 被保留，未填充则为舍去。

**步骤 4**：将测试集输入步骤 3 生成的 K 组经过筛选的 SVR 集中。如第 k 组 SVR 集保留 $SVR_i$ 个数为 b，则测试集将遍历这 b 个 $SVR_i$，得到对应的预测值 $z_j$（其中 j=1,2,⋯,b），利用加权平均得到该组最终预测值 $z^0{}_{best}(k)$：

$$z^0{}_{best}(k)=\frac{1}{b}\left(\sum_{j=1}^{b}\frac{r2\_score_j}{\sum_{i=1}^{m}r2\_score_i}*z_j\right)$$

计算 K 组 SVR 集各自的最终预测值 $z^0{}_{best}(1)$、$z^0{}_{best}(2)$、……、$z^0{}_{best}(K)$，再计算每组 SVR 集的适应度 $R2\_score^0(k)$，其中

$$R2\_score^0(k)=1-\frac{\sum_{l=1}^{L}(y_l-z^0{}_{best}(k)_l)^2}{\sum_{l=1}^{L}(y_l-\bar{y})}$$

式中，L 为测试集时间序列（逐日）的长度，$y_i$ 为第 l 天污染物浓度的实测值，$\bar{y}$为实测值的平均值，$z^0{}_{best}(k)_l$ 为第 l 天的浓度预测值。R2_score 越大，预测效果越好。由此得 $R2\_score^0(best)$，其对应的 SVR 集即为初始的最优集成模型 $SVR^*(0)$。

**步骤 5**：更新粒子群算法的相关参数，开始迭代。

当迭代次数为 1，即 t=1 时，先让每个粒子 k 都通过遍历所有基学习器 $SVR_i$，并在遍历的过程中，将新生成的粒子位置 $loc_k^i(t)$保留在 $Solution_k(t)$中，判断各行 $Solution_k(t)$中 $loc_k^i(t)$ 与阈值 $loc_d$ 的关系，具体过程与步骤 3 相同。

更新粒子群算法的相关参数：

$$\omega_t=0.9-(0.9-0.4)*t/T$$

$$V_k{}^i(t)=\omega_{t-1}*V_k{}^i(t-1)+c1*r1*\left(pbest_k(t)-loc_k{}^i(t-1)\right)+c2*r2$$

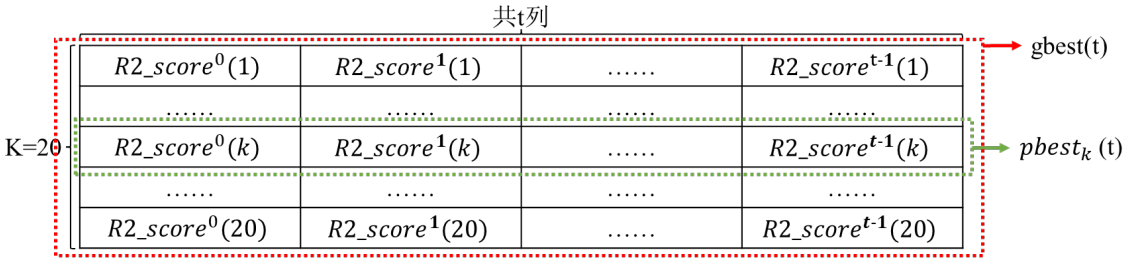
$$*\left(gbest(t)-loc_k{}^i(t-1)\right)$$

$$loc_k{}^i(t)=loc_k{}^i(t-1)+V_k{}^i(t)$$

式中，惯性权重$\omega_t$具有平衡局部搜索和全局搜索的能力，本文设置其更新方式为线性

递减；$pbest_k(t)$是粒子最优值，指每个粒子 k 前 t-1 个适应度中的最优值对应的 $loc^i$，gbest(t)是种群最优值，指所有粒子前 t-1 个适应度最优值对应的 $loc_k^i$，如图：

- 第t次迭代




注：$pbest_k(t)$为 $\max_{i=0,\ldots,t-1} R2_score^i(k)$对应的位置$loc^i$；gbest(t)为 $\max_{\substack{i=0,\ldots,t-1\\k=1,\ldots,20}} R2_score^i(k)$对应的位置$loc_k{}^i$

其中，t=1 时 $pbest_k(1)$即粒子 k 的 $R2\_score^0(k)$，gbest(1)即 $R2\_score^0(best)$。

将测试集输入筛选过的 K 组 SVR 集中，得到各组对应的最终预测值 $z^t_{best}(k)$及其对应的 $R2\_score^t(k)$，比较得出 $R2\_score^t(best)$。如 $R2_score^t(best)$优于 $R2\_score^{t-1}(best)$，则最优集成模型 $SVR^*(t)$为 $R2_score^t(best)$对应的 SVR 集，否则 $SVR^*(t)$仍为 $R2_score^{t-1}(best)$对应的 SVR 集。

**步骤 6**：继续迭代至迭代次数 t 达到总迭代次数。如果在迭代次数为 $t^*$后 R2_score 保持在最高值处，则认为模型训练完成，可输出所得到的最优集成模型 $SVR^*$；如果 R2_score 仍保持持续上升趋势，则需增加总迭代次数 T 并重复步骤 2~5，直至模型完成训练。

#### 4.3.5.4 $O_3$ 模型评价

经过分析，本文使用 CEEMDAN_IPSOSE_SVR 作为 $O_3$ 二次预测预测模型。又考虑到组合模型可以提高预报准确度，故同样采用经降噪处理后得到的 CEEMDAN_LSTM_FC 模型进行组合，得到各模型评价指标结果如下：

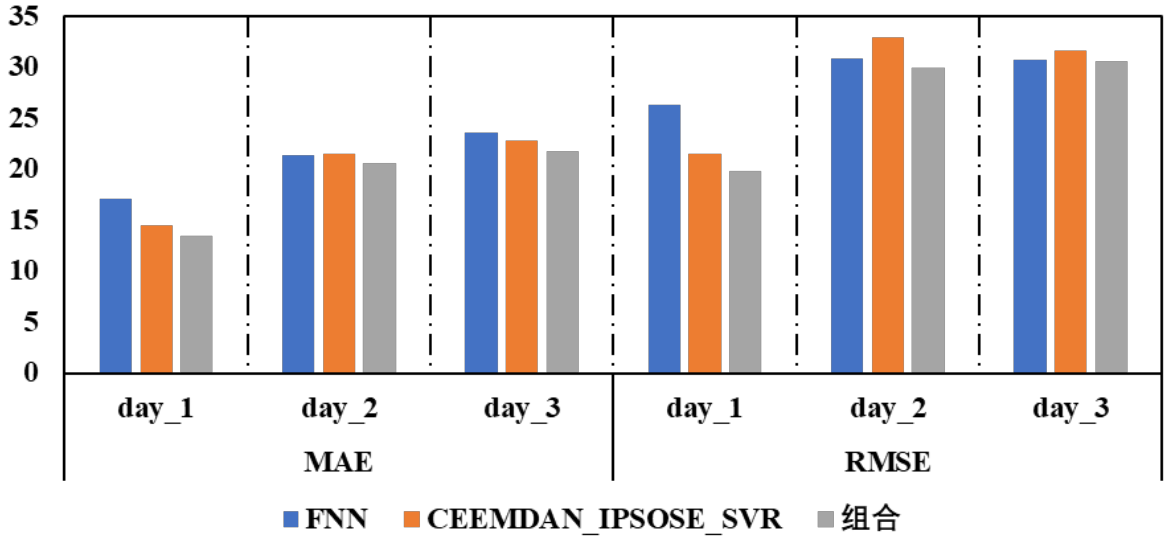



图 17　$O_3$ FNN、CEEMDAN_IPSOSE_SVR 及组合模型三天评价指标结果

由图可知，对于 $O_3$，组合模型的预测效果在三种模型中表现最好，故本文用其作为 $O_3$ 最终二次预报模型。

### 4.3.6 问题 3 小结

综上所述，6 种常规污染物所用建模方法如图 18 所示：

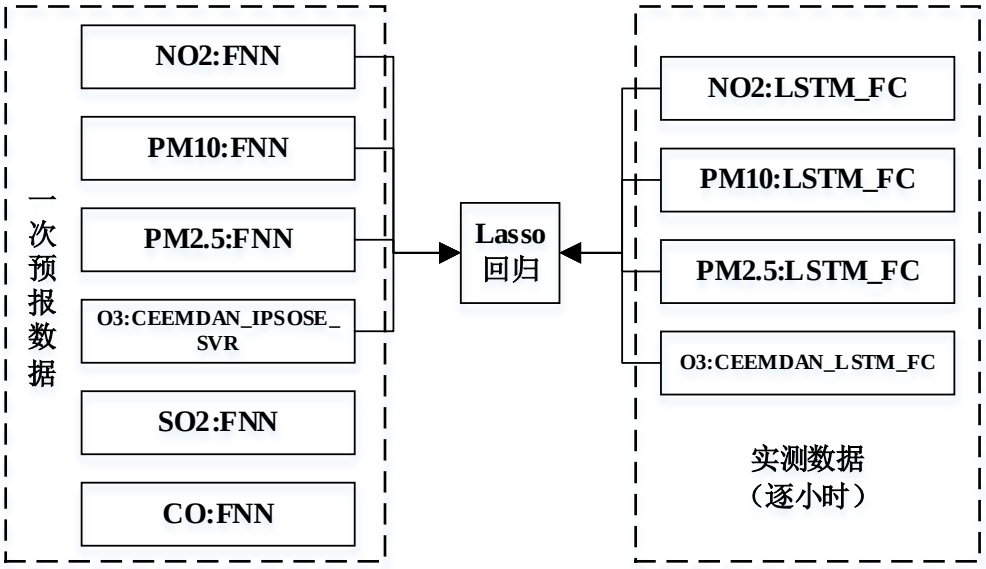



图 18　问题 3 各项常规污染物建模方法

将监测点 A、B、C 在 2021 年 7 月 13 日至 7 月 15 日相应数值输入模型，得到 6 种常规污染物单日浓度预测值，并计算相应的 AQI 和首要污染物。结果如表 8 所示：

表 8　问题 3 污染物浓度及 AQI 预测结果表

| 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 | 5.6 | 13.09 | 24.48 | 5.62 | 93.55 | 0.36 | 47 | 无 |
| 2021/7/14 | 监测点 A | 6.01 | 14.67 | 25.39 | 8.53 | 140 | 0.36 | 83 | $O_3$ |
| 2021/7/15 | 监测点 A | 5.98 | 16.35 | 24.35 | 7.35 | 114.33 | 0.37 | 62 | $O_3$ |
| 2021/7/13 | 监测点 B | 6.48 | 10.16 | 15.95 | 4.7 | 78.83 | 0.4 | 39 | 无 |
| 2021/7/14 | 监测点 B | 6.24 | 10.3 | 16.79 | 5.54 | 75.32 | 0.38 | 38 | 无 |
| 2021/7/15 | 监测点 B | 6.44 | 8.51 | 15.98 | 5.97 | 56.52 | 0.41 | 28 | 无 |
| 2021/7/13 | 监测点 C | 8.27 | 14.12 | 35.12 | 16.47 | 128.44 | 0.56 | 74 | $O_3$ |
| 2021/7/14 | 监测点 C | 8.02 | 13.82 | 38.52 | 18.21 | 129.8 | 0.55 | 75 | $O_3$ |
| 2021/7/15 | 监测点 C | 8.65 | 18.54 | 41.25 | 23.21 | 115.5 | 0.67 | 63 | $O_3$ |

### 4.4 问题 4 建模与求解

#### 4.4.1 问题分析及数据处理

问题 4 要求建立相邻区域内监测点 A、A1、A2、A3 的协同预报模型，即需在建模过程中对空间相关性加以考虑。为对 4 个监测站点各污染物之间可能存在的空间相关性产生更加科学直观地认识，团队首先采用 Pearson 检验方法，检验结果如图 19 所示：

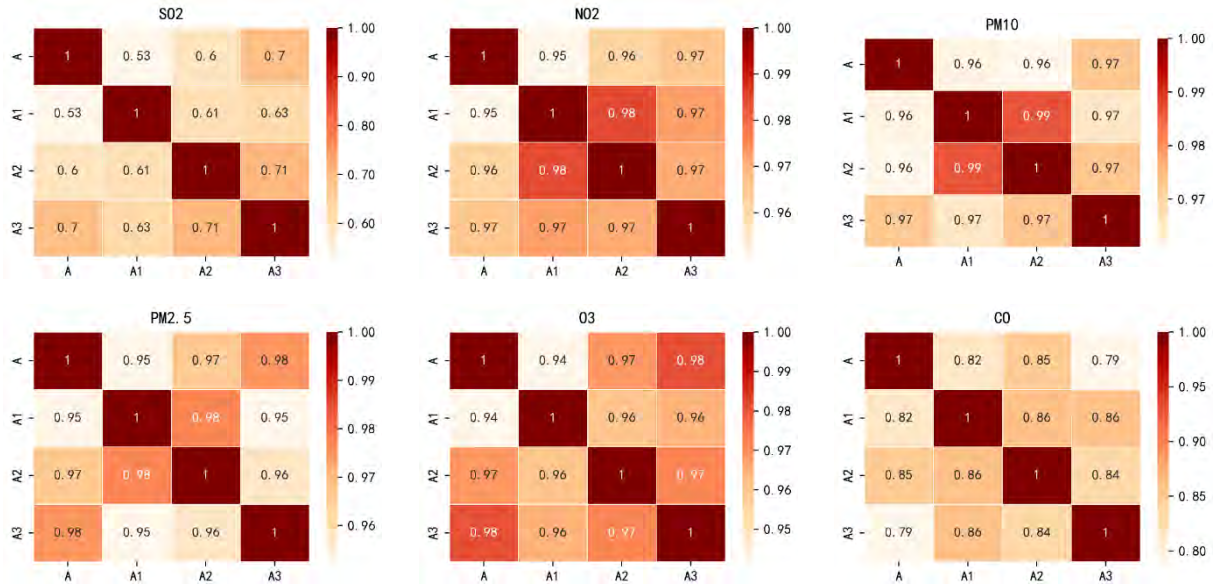



图 19　Pearson 检验结果

由图可知，监测点间 6 种常规污染物均具有一定程度的空间相关性，其中 $PM_{10}$、$PM_{2.5}$、$NO_2$、$O_3$ 空间相关性相对较强，CO 相关性一般，$SO_2$ 的空间相关性表现最差，但总的来说，可考虑空间相关性进行建模。

团队首先考虑使用卷积神经网络（CNN）[20]进行建模。经分析发现，尽管传统的 CNN 可以获得局部空间特征，但仅适用于如图像、规则网格等欧几里德空间；而题目中 A、A1、A2、A3 站点分布较为散乱，是典型的非欧空间，这意味着 CNN 模型不能反映各监测点所构成网络复杂的拓扑结构，也不能准确地捕捉空间依赖。此时，将 CNN 推广到可处理任意图结构数据的图卷积网络（GCN）[21]得到关注。

**数据预处理**方面，问题 4 异常值处理、缺失值处理及特征缩放过程与问题 3 一致。

**特征选择**方面，同样采用随机森林平均不纯度减少的重要度评价方法自适应方法，根据模型自适应选择所需特征数量。

#### 4.4.2 GCN 建模与评价

##### 4.4.2.1 GCN 建模

假设空气质量监测站点组成的网络为$G$，即使用$G = (V, \mathcal{F})$来描述空气质量监测站点网络的拓扑结构。将监测点视为一个节点 v，则$V = (v_1, \ldots, v_N)$代表一组节点，其中 N 是监测站点的个数。建立映射：

$$f: V \times V \to A, f(v_i, v_j) = a_{ij}$$

映射 f 中，邻接矩阵 A 表示 V 中节点的相互联系情况：

$$A = (a_{ij})_{N \times \mathrm{N}} \in \mathrm{R}^{N \times N}$$

A 中仅包含 0 和 1 的元素：若元素为 0，表示节点之间没有联系，若元素为 1，则节点之间存在联系。因此，考虑空间关系的空气质量预报可视为在网络拓扑结构 G 和输入矩阵 X 的前提下学习映射函数 f，去预测未来时刻的空气质量信息：

$$[X_{t+1}, \dots, X_{t+T}] = f(G; X_t)$$

式中，T 是需要预测的时间序列的长度，G 代表使用 GCN 网络。在给定邻接矩阵$A$和特征矩阵$X$的情况下，GCN 模型构造了傅里叶域的滤波器。该滤波器作用于图的节点，通过其一阶邻域捕获节点之间的空间特征，然后通过叠加多个卷积层来建立 GCN 模型，该模型可以表示为：

$$H^{l+1} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{l}W^{l}\right)$$

式中，$\tilde{A}$是具有附加自连接的矩阵：

$$\tilde{A} = A + I_N$$

其中$I_N$是单位矩阵；$\tilde{D}$是度矩阵；$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$表示对$\tilde{A}$进行标准化；$H^{l}$是第$l$层的输出；$W^{l}$是第$l$层的参数集；$\sigma(\cdot)$表示非线性模型的 sigma 函数。

在本题中，由于数据量较少，选择简单的单层 GCN 模型即可获得空间相关性，其可表示为：

$$f(X, A) = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}XW_0\right)$$

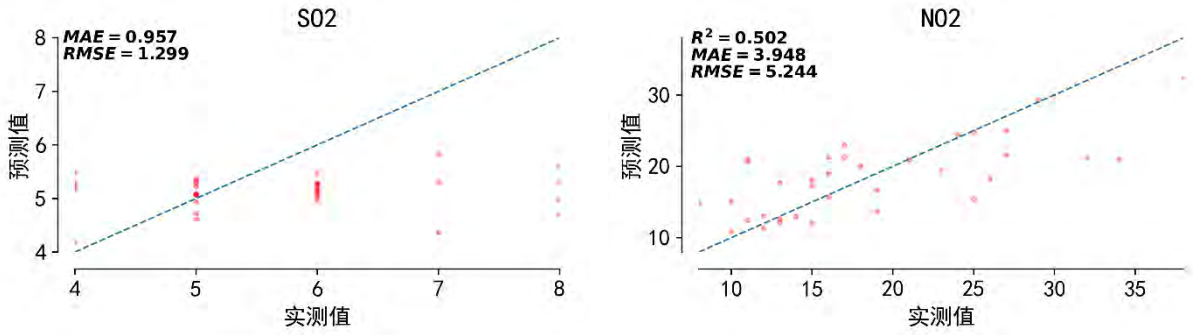
其中，$W_0 \in R^{P\times H}$表示从输入到隐藏层的权重矩阵，P 表示特征矩阵的数量，H 表示隐藏单元数。$f(X, A) \in R^{N\times T}$表示预测长度为$T$的输出，若输出长度与预测长度不一致，可采取全连接层 FC 将其调整为一致。

具体建模时，选用绝对值损失函数（式 (7)）描述模型经验风险，学习率、隐藏层选择及 epoches 设置均随污染物不同而变化；优化器统一选择为 Adam，采取 L2 正则化、Dropout 以及 Dropedge[22]防止过拟合。训练平台选择 pytorch 框架。针对 Dropedge 作如下说明：

GCN 训练效果不佳主要有过拟合和过平滑两个原因，而 DropEdge 增强了输入数据的随机性和多样性，并且依据不同污染选择不同概率去掉原始图中固定比例的边，以使节点连接更加稀疏，从而缓解了 GCN 过拟合和过平滑现象。

#### 4.4.2.2 GCN 评价

使用 GCN 模型从预报基础数据中学习空间特征。对于 $O_3$，同样采用 CEEMDAN 对其进行降噪后再进行建模训练。6 种常规污染物模型评价结果如下：



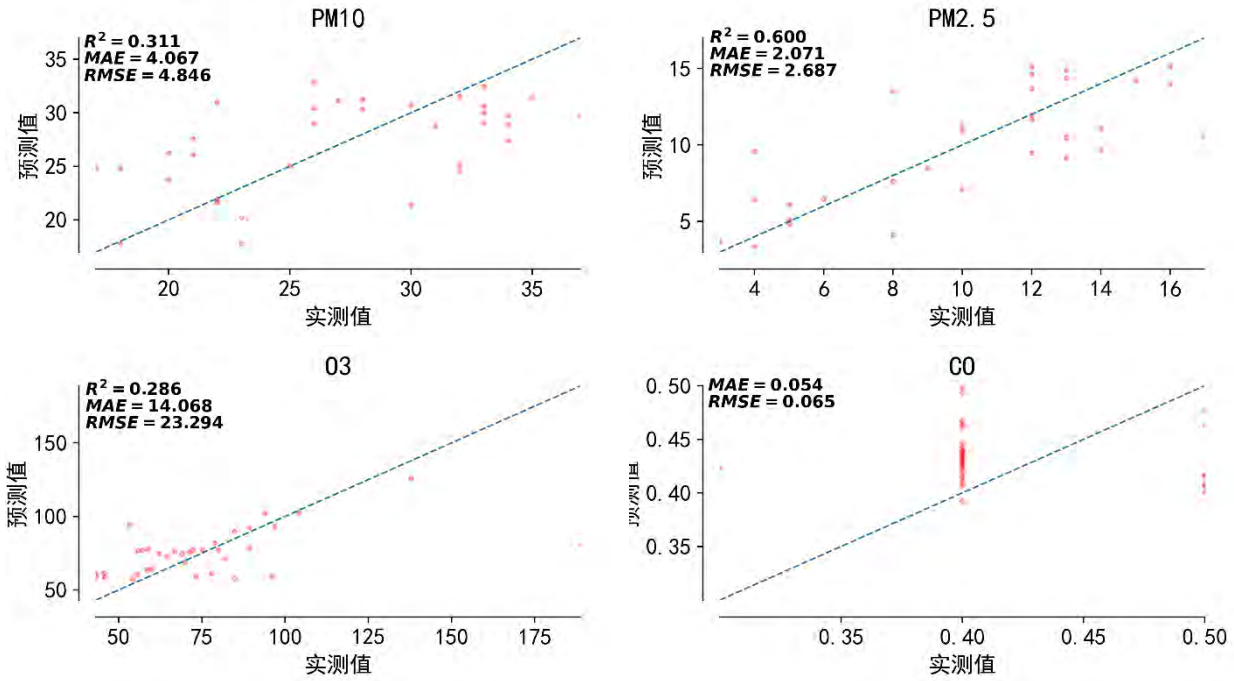



图 20　6 种常规污染物 GCN 评价指标结果（以预报第一天为例）

考虑结合监测数据使用组合模型进一步提高预报精度。但由图可知 CO 和 $SO_2$ 偏差足够小，为节约计算开销，不再进行监测值的修正。

### 4.4.3 GRU_GCN_FC 建模与评价

#### 4.4.3.1 GRU_GCN_FC 建模

由于 GCN 只考虑了空间相关性进行二次预报建模，获取时间相关性是空气质量预测中的另一个关键问题。LSTM 模型和 GRU[23]模型均可用于处理时间序列数据，两者基本原理大致相同，即使用门控机制来记忆尽可能多的长期信息。然而，由于 LSTM 结构相对复杂，具有较长的训练时间，而 GRU 无需引入额外的记忆单元，其结构相对简单、使用参数较少、训练速度更快。因此，本题我们选择 GRU 模型从监测数据中获取时间依赖性。如下式所示：

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$
$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$

式中，$\tilde{h}_t$表示当前时刻的候选状态；$r_t$为重置门，用来控制候选状态$\tilde{h}_t$的计算是否依赖上一时刻的状态$h_{t-1}$：当$r_t = 0$时，候选状态

$$\tilde{h}_t = \tanh(W_h x_t + b_h)$$

只和当前输入$x_t$相关，和历史状态无关，当$r_t = 1$时，候选状态

$$\tilde{h}_t = \tanh(W_h x_t + U_h h_{t-1} + b_h)$$

和当前输入$x_t$以及历史状态$h_{t-1}$相关；$z_t$为更新门，当$z_t = 0$时，当前状态$h_t$和前一时刻的状态$h_{t-1}$之间为非线性函数关系，当$z_t = 1$时，$h_t$和$h_{t-1}$之间为线性函数关系；$W_*, U_*, b_*$为可学习参数，其中$*\in \{b, r, z\}$；$\sigma(\cdot)$为 sigma 函数；$x_t$表示时刻$t$的特征信息。

GRU 以前一时刻的隐藏状态和当前特征信息作为输入，获取当前时刻的空气质量信息，在捕获当前时刻特征信息的同时，模型仍然记录了历史特征信息的变化趋势，并具有获取时间依赖性的能力。因此本题使用 GRU 和 GCN 的融合模型 GRU_GCN_FC 进行时空

关系建模。

假设网络 G 中在时间 t 时节点的特征矩阵$\mathrm{X}_t \in \mathrm{R}^{N\times P}$，其中 P 表示节点属性特征的数量，考虑时空关系的污染物浓度预测问题可被视为在网络拓扑结构 G 和特征矩阵 X 的前提下学习映射函数 f，空气质量信息预测可表示如下：

$$[X_{t+1},\dots,X_{t+T}] = f\big((G;X_{t-Ts}),\dots,(G;X_{t-1})(G;X_t)\big)$$

其中$Ts$是历史时间序列的长度。

GRU_GCN_FC 具体描述见图 21：上侧为 GRU_GCN_FC 预测过程；下侧为 GRU_GCN cell 组件具体构造，其中 GCN 表示图卷积过程；FC 表示全连接层，该层选择 GRU_GCN 输出的目标站点的最后一个$\mathrm{out}_t$作为 GRU_GCN 拟合值，将其输入到全连接层进行线性变换，从而得到最终预报值。

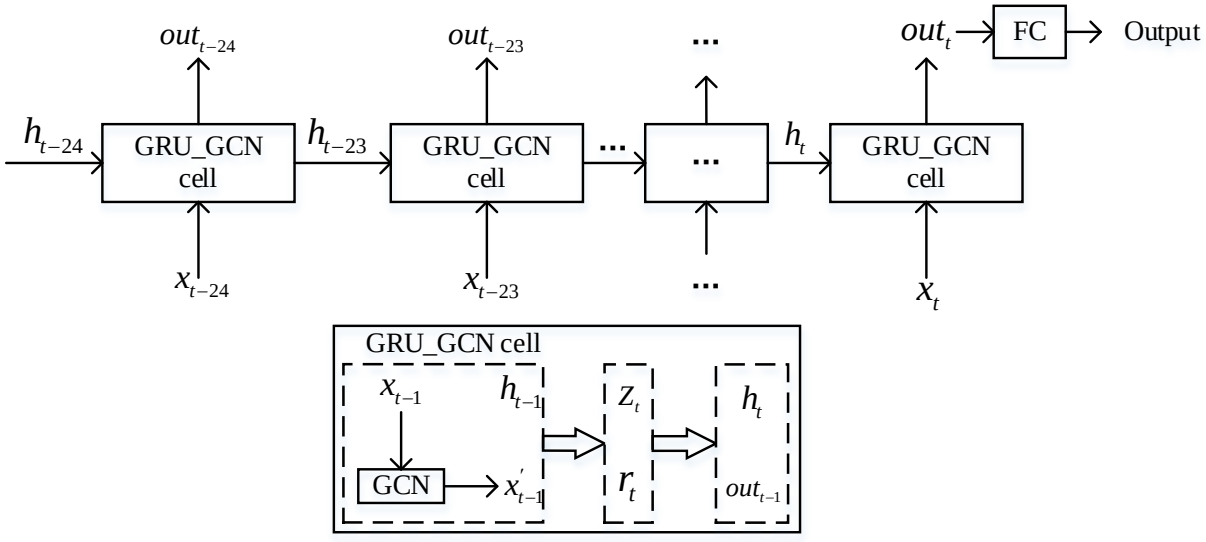



图 21　GRU_GCN_FC 预测过程（上）及 GRU_GCN cell 组件构造

### 4.4.3.2 GRU_GCN_FC 评价

对于 $NO_2$、$PM_{10}$、$PM_{2.5}$、$O_3$，GRU_GCN_FC 模型评价结果如图 22 所示。其中，考虑到 $O_3$ 的特殊性，仍对其进行 CEEMDAN 降噪处理。

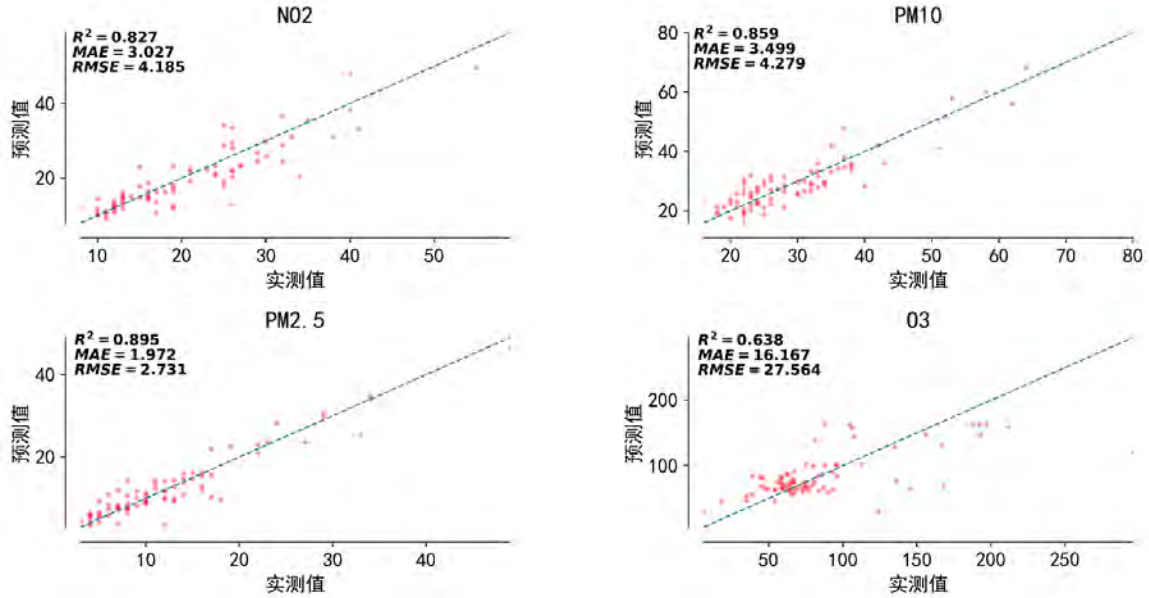



图 22　$NO_2$、$PM_{10}$、$PM_{2.5}$、$O_3$ GRU_GCN_FC 模型评价指标结果（以预报第一天为例）

### 4.4.4 组合模型评价与讨论

综上所述，将 GCN 模型与 GRU_GCN_FC 模型使用 Lasso 回归得到组合模型，用以预测 $NO_2$、$PM_{10}$、$PM_{2.5}$、$O_3$ 未来三天的污染物浓度值；对于 $SO_2$ 和 CO，则仍使用 GCN 进行预测。

以下简称问题 4 所用模型为 Q4 模型，问题 3 所用模型为 Q3 模型。将 6 种常规污染物 Q4 模型与 Q3 模型预测效果进行比较，结果如图 23 所示：

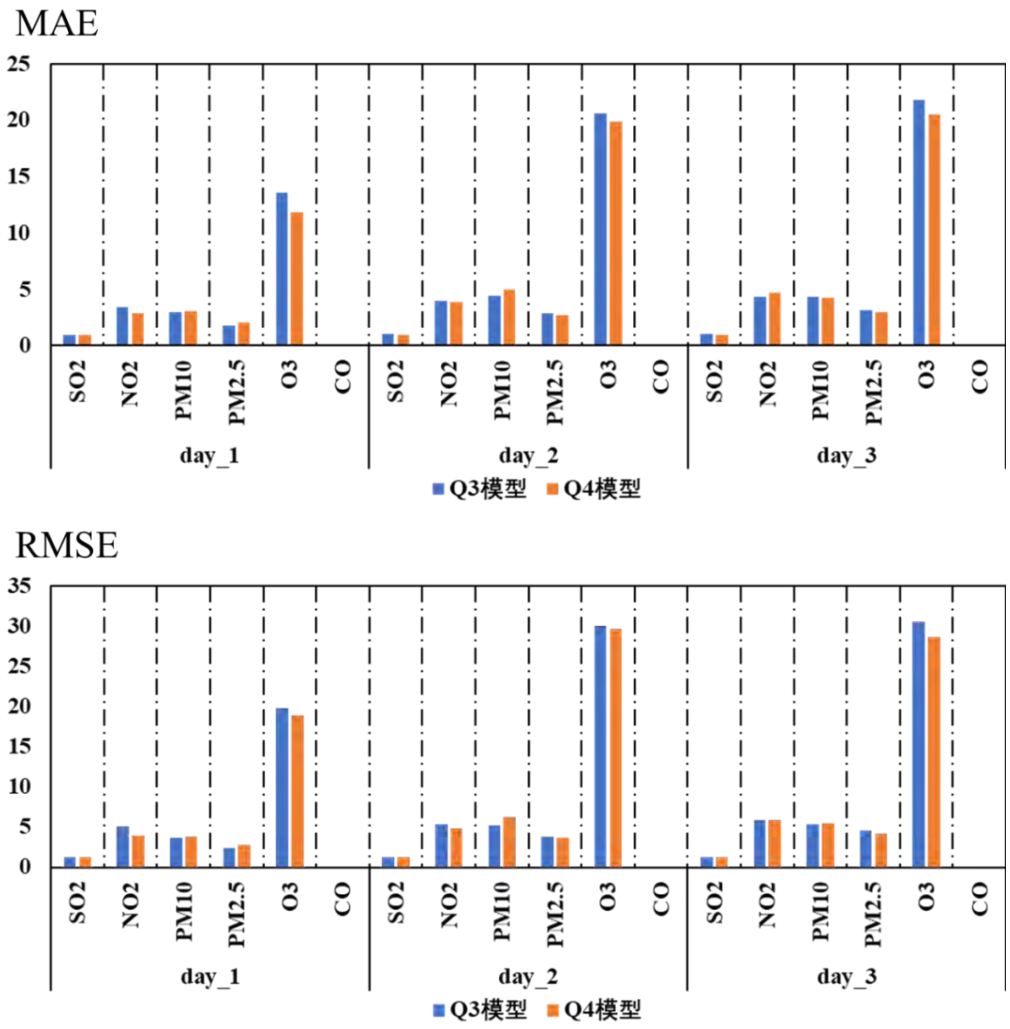



图 23　6 种常规污染物 Q3 模型及 Q4 模型三天评价指标结果

由图可知，对于 $O_3$，Q4 模型 MAE 和 RMSE 值均低于 Q3 模型，即 Q4 模型预测未来三天污染物浓度效果优于 Q3 模型；对于 $SO_2$，除第一天的 Q3 模型 RMSE 更优外，其他数值 Q4 模型均低于 Q3 模型；对于 $NO_2$，第一、二天的 Q4 模型 MAE 和 RMSE 低于 Q3 模型，第三天则是 Q3 模型占优；对于 $PM_{2.5}$，第二、三天的 Q4 模型占优；对于其它 2 种污染物，CO Q4 模型仅在第一天占优；$PM_{10}$ 则仅第三天的 MAE 是 Q4 模型低，其余数值 Q3 模型效果更好。

综上所述，对 $O_3$ 进行预测时，考虑空间相关性有利于提高模型准确性；对 $SO_2$、$NO_2$ 和 $PM_{2.5}$，考虑空间相关性对提高模型准确性有一定助益；而对于 CO 和 $PM_{10}$，空间相关性对提高模型准确性无明显助益。总的来说，$O_3$ 相比其他污染物预测效果提升更明显，这与 $O_3$ 作为一种区域传输污染物相关。

### 4.4.5 问题 4 小结

问题 4 中，6 种常规污染物关于空间相关性建模方法如下：

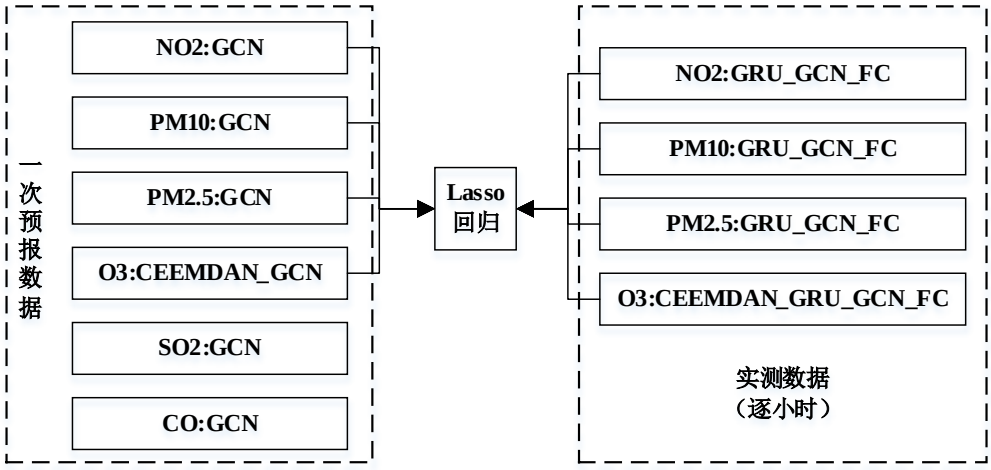



图 24　问题 4 各项常规污染物建模方法

与问题 3 的模型相比，协同预报模型对 $O_3$ 进行预报时准确度有明显提高，其原因与 $O_3$ 具有较强区域传输性有关。

将监测点 A、A1、A2、A3 在 2021 年 7 月 13 日至 7 月 15 日相应数值输入模型，得到 6 种常规污染物单日浓度预测值，并计算相应的 AQI 和首要污染物。结果如表 9 所示：

表 9　问题 4 污染物浓度及 AQI 预测结果表

| 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 | 5.74 | 12.79 | 24.7 | 4.52 | 98.89 | 0.42 | 49 | 无 |
| 2021/7/14 | 监测点 A | 5.83 | 14.24 | 26.06 | 8 | 140.1 | 0.42 | 83 | $O_3$ |
| 2021/7/15 | 监测点 A | 5.59 | 17.1 | 24.72 | 7.46 | 110.23 | 0.49 | 59 | $O_3$ |
| 2021/7/13 | 监测点 A1 | 6.57 | 13.66 | 25 | 6.46 | 92.78 | 0.48 | 46 | 无 |
| 2021/7/14 | 监测点 A1 | 6.58 | 15.56 | 27.72 | 8.96 | 114.41 | 0.38 | 62 | $O_3$ |
| 2021/7/15 | 监测点 A1 | 6.59 | 17.16 | 29.51 | 10.97 | 108.24 | 0.46 | 57 | $O_3$ |
| 2021/7/13 | 监测点 A2 | 4.69 | 13.82 | 21.23 | 7.57 | 99.69 | 0.43 | 50 | 无 |
| 2021/7/14 | 监测点 A2 | 4.64 | 14.9 | 22.15 | 9.25 | 138 | 0.49 | 82 | $O_3$ |
| 2021/7/15 | 监测点 A2 | 4.61 | 16.89 | 23.44 | 9.47 | 119.04 | 0.49 | 66 | $O_3$ |
| 2021/7/13 | 监测点 A3 | 4.01 | 11.27 | 14.1 | 4.65 | 98.66 | 0.39 | 49 | 无 |
| 2021/7/14 | 监测点 A3 | 4.01 | 13.53 | 21.01 | 6.71 | 125.15 | 0.39 | 71 | $O_3$ |
| 2021/7/15 | 监测点 A3 | 4.01 | 14.14 | 14.18 | 6.65 | 107.29 | 0.39 | 56 | $O_3$ |

# 5 模型评价

## 5.1 模型优点与创新

- **数据预处理方面：**

问题 2 结合风向监测规范内容提出以伪独热码方式处理风向，在保留数据连续性的同时消除其有序性，既具创新性，又符合统计规范，同时为后续分析提供便利。

- **模型建立方面：**

(1) 问题 3、问题 4 模型首先对经预处理后的多项输入指标采用随机森林平均不纯度减少进行重要度排序，并根据目标污染物的不同自适应选择相应输入指标，有效提升了模型泛化能力。

(2) 问题 3 和问题 4 中分别提出了回归集成模型 FNN+LSTM_FC+Lasso 和 GCN+GRU_GCN_FC+Lasso，两种模型均能将一次预报数据及实测数据中蕴含的有用信息很好地结合起来进行预报，预测效果相比单模型均有所提升；

(3) 针对预测难度较高的 $O_3$，选用更为适应小样本集和鲁棒性较高的支持向量回归作为基学习器并对目标污染物进行 CEEMDAN 降噪，创新性使用 IPSOSE 算法完成基学习器集的择优与集成，从而大幅提升 $O_3$ 预测效果；

(4) 问题 4 中采用图卷积网络 GCN 对空间关系进行聚合。GCN 能较好适应非欧空间关系图，相比传统空间回归模型及卷积神经网络优势更大。

## 5.2 模型缺点及改进

- **数据预处理方面：**

问题 2 在其他气象指标的标准化上存在改进空间：如：大多时刻雨量均为 0，直接使用 Z-score 标准化或 min-max 归一化，前者会导致大量雨量数据处于负值范围，后者在数据集样本存在某个极端暴雨天时，所有雨量数据仍会接近 0。在后续研究中可思考新的标准化思路。

- **特征选择方面：**

问题 2 数据降维过程中，可尝试先参考更多专业文献及气象单位对气象指标的提取结果，如国家气象科学数据中心 • 中国气象数据网（http://data.cma.cn/data/online/t/1）提供日平均气温、日最高气温、日最低气温，在已知此三项气温的前提下再使用主成分分析等方法挖掘更多信息，从而减少成分系数的误差，增加模型可解释性。

- **模型建立方面：**

(1) 问题 3 和问题 4 均存在模型训练过程不稳定现象，且超参数较多，模型学习效果在较好的调参技术下仍会有所提升；

(2) 受时间限制，降噪技术仅用在对 $O_3$ 的预测上，而实际 6 种污染物均可能存在噪音，对其余 5 种污染物进行降噪可进一步提升预测效果；

(3) 集成学习过程中，使用 Bootstrap 从样本中选择子集训练基学习器 SVR，从而增加 $SVR_i$ 之间的差异，但此差异并没有达到最大，若使用 Bootstrap 选择样本子集的同时辅以抽取不同特征子集，则可进一步增大 $SVR_i$ 的差异。

# 参考文献


[1] 陈雪. 2013-2019 年兰州市城市环境空气质量变化趋势研究 [J]. 兰州大学, 2021.

[2] 张达标. 广西大气污染特征及污染天气分型研究 [J]. 南宁师范大学, 2020.

[3] 蒋维楣. 空气污染气象学教程 [M]. 北京: 气象出版社, 5-12. 2004.

[4] 帅气滴点 C. 独热编码（One-Hot Encoding）介绍及实现 [Z]. CSDN. 2018

[5] 地面气象观测规范 风向与风速 [Z]//中华人民共和国国家质量监督检验检疫总局, 中国国家标准化管理委员会. 2017

[6] 蒋维楣, 孙鉴泞, 曹文俊, et al. 空气污染气象学教程 [M]. 2 ed. 北京: 气象出版社, 53-64. 2004.

[7] 韦相. 基于密度的改进 BIRCH 聚类算法 [J]. Ji suan ji gong cheng yu ying yong, 2013, 49(10): 201-5.

[8] PAK U, MA J, RYU U, et al. Deep learning-based PM2.5 prediction considering the spatiotemporal correlations: A case study of Beijing, China [J]. Sci Total Environ, 2020, 699: 133561.

[9] BREIMAN. Random Forests [J]. MACH LEARN, 2001.

[10] VAPNIK V. Measuring the VC-Dimension of a Learning Machine [J]. Neural Computation, 1994.

[11] CHEN T. XGBoost: A Scalable Tree Boosting System [J]. the 22nd ACM SIGKDD International Conference, 2016.

[12] T T. Divide the gradient by a running average of its recent magnitude [J]. Lecture 65-rmsprop, 2012.

[13] KINGMA D P. ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION [J]. International Conference on Learning Representations, 2014.

[14] HINTON G E. Improving neural networks by preventing co-adaptation of feature detectors [J]. Computer Science, 2012.

[15] WINKLER R. The Combination of Forecasts [J]. Journal of the Royal Statal Society Series A (General), 1983.

[16] HOCHREITER S. Long Short-Term Memory [J]. Neural Computation, 1997.

[17] TIBSHIRANI R. Regression shrinkage and selection via the lasso_ a retrospective [J]. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2011.

[18] TORRES M ı E. A COMPLETE ENSEMBLE EMPIRICAL MODE DECOMPOSITION WITH ADAPTIVE NOISE [J]. IEEE International Conference on Acoustics, 2011.

[19] GANG W. Genetic_Algorithm_Based_Selective_Ensemble_with_Multiset_Representation [J]. International Conference on Artificial Intelligence & Computational Intelligence, 2010.

[20] LECUN. Backpropagation Applied to Handwritten Zip CodeLecun [J]. 1989.

[21] KIPF T N. SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS [J]. International Conference on Learning Representations, 2016.

[22] RONG Y. DROPEDGE: TOWARDS DEEP GRAPH CONVOLUTIONAL NETWORKS ON NODE CLASSIFICATION [J]. 2019.

[23] CHUNG J. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling [J]. Eprint Arxiv, 2014.

# 附 录

- 质心表

$NO_2$ 质心表

| | | REGR factor score 1 for analysis 1 | | REGR factor score 2 for analysis 1 | | REGR factor score 3 for analysis 1 | | REGR factor score 4 for analysis 1 | | REGR factor score 5 for analysis 1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 均值 | 标准差 | 均值 | 标准差 | 均值 | 标准差 | 均值 | 标准差 | 均值 | 标准差 |
| 聚类 | 1 | .5574190 | .45629515 | -.4621346 | 1.00571799 | .7031853 | .57302688 | .4330343 | .74072008 | -.8383371 | .89278352 |
| | 2 | -.0447645 | .64196363 | -.1728589 | .54629907 | .4368506 | .78636139 | .0650650 | .66947254 | 1.2062879 | .55938215 |
| | 3 | -.8958478 | .72851841 | -1.7963383 | .72350470 | -.8884528 | .79355284 | -.6466276 | .93765698 | .0656471 | .70205208 |
| | 4 | -.5590830 | .65617551 | .7066893 | .44990672 | .7081611 | .67341301 | -1.1761164 | .63592812 | -.2353877 | .71353731 |
| | 5 | -1.1588764 | .62112363 | .7208563 | .47808854 | -.8406560 | .50601507 | 1.0549607 | .92813001 | -.0484666 | .91124344 |
| | 6 | 1.3288253 | .28191059 | .4639480 | .65277496 | -1.0053394 | .61211830 | -.2183178 | .49265445 | -.0572752 | .43278151 |
| | 组合 | .0000000 | 1.0000000 | .0000000 | 1.0000000 | .0000000 | 1.0000000 | .0000000 | 1.0000000 | .0000000 | 1.0000000 |

颗粒物质心表

| | | REGR factor score 1 for analysis 1 | | REGR factor score 2 for analysis 1 | | REGR factor score 3 for analysis 1 | | REGR factor score 4 for analysis 1 | | REGR factor score 5 for analysis 1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 均值 | 标准差 | 均值 | 标准差 | 均值 | 标准差 | 均值 | 标准差 | 均值 | 标准差 |
| 聚类 | 1 | -.0369361 | .44676634 | -.4881081 | .51471709 | -.0722296 | .45486661 | -.8733555 | .52337683 | .8844399 | .56602905 |
| | 2 | -.9147502 | .58695429 | -1.2217001 | .69101525 | -.4591819 | .62538720 | .0638572 | .69939027 | -.1325150 | .70170086 |
| | 3 | .7076847 | .33362792 | .4673715 | .46069514 | -.6895555 | .51436819 | -.1749860 | .42155612 | .0086025 | .59578292 |
| | 4 | 1.1864456 | .28256716 | -.0886723 | .61646300 | .9284867 | .53613639 | -.2019924 | .49444089 | -.0598591 | .71283800 |

| | REGR factor score 1 for analysis 1 | | REGR factor score 2 for analysis 1 | | REGR factor score 3 for analysis 1 | | REGR factor score 4 for analysis 1 | | REGR factor score 5 for analysis 1 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 均值 | 标准差 | 均值 | 标准差 | 均值 | 标准差 | 均值 | 标准差 | 均值 | 标准差 |
| 5 | .4257306 | .7206312 | 1.178636 | .7831689 | -1.926287 | .8292789 | -1.092264 | .5647008 | -1.204046 | 1.210095 |
| 6 | -.131021 | .5990329 | .5596694 | .9581435 | -.827327 | .6126483 | 1.135141 | .5994205 | .7046063 | .7897517 |
| 7 | -1.715664 | .4101777 | .7073741 | .7476491 | 1.019157 | .6067611 | -.753313 | .5728946 | -.977324 | .8567329 |
| 8 | .3532782 | .6753908 | -1.055581 | .6368739 | -.082194 | .4909143 | 1.515163 | .6830994 | -1.408391 | .6687207 |
| 9 | -.817481 | .3578751 | 1.132860 | .5399180 | 1.026078 | .5157637 | 1.484652 | .5343890 | .2567110 | .7818284 |
| 组合 | 0.000000 | 1.000000 | .0000000 | 1.000000 | .0000000 | 1.000000 | .0000000 | 1.000000 | .0000000 | 1.000000 |

$O_3$质心表

| | | REGR factor score 1 for analysis 1 | | REGR factor score 2 for analysis 1 | | REGR factor score 3 for analysis 1 | | REGR factor score 4 for analysis 1 | |
|---|---|---|---|---|---|---|---|---|---|
| | | 均值 | 标准差 | 均值 | 标准差 | 均值 | 标准差 | 均值 | 标准差 |
| 聚类 | 1 | .5785790 | .67005899 | -.0608879 | .52068195 | -.5933045 | .62516757 | .1611906 | .86824255 |
| | 2 | .4925879 | .83816246 | -.0168725 | .43686890 | 1.0557345 | .54196357 | .4697468 | .67456243 |
| | 3 | -.6394533 | .65505562 | -1.4031189 | .66657227 | .0718081 | 1.36369291 | -.7747038 | .78697513 |
| | 4 | -1.4842031 | .66389180 | .2985163 | 1.08436698 | -.3960559 | .79272745 | .8095470 | .99119937 |
| | 5 | -.3189833 | .57849694 | 1.4466719 | .59676763 | -.0034301 | .51482598 | -1.0414716 | .52118263 |
| | 组合 | .0000000 | 1.00000000 | .0000000 | 1.00000000 | .0000000 | 1.00000000 | .0000000 | 1.00000000 |

- **CEEMDAN 降噪流程**

步骤 1：以 $O_3$ 全体日最大八小时滑动平均监测浓度值为原始序列 Y，往该序列上添加 I 次相同长度的高斯白噪声$\varepsilon_0 w^i(i=1,\cdots,\mathrm{I})$，使用经验模态分解 EMD 对序列$Y+\varepsilon_0 w^i$进行分解，得到第一个 IMF（$IMF_1$）：

$$IMF_1=\frac{1}{I}\sum_{i=1}^{I}E_1\left(Y+\varepsilon_0 w^i\right)$$

式中，$E_1(\cdot)$表示取 EMD 分解的第 1 个 IMF；$\varepsilon_0$是自适应系数；$w^i$表示第$i$个概率密度函数服从标准高斯分布$(\mathrm{N}(0,1))$的噪声；$I$表示高斯噪声的添加次数，本文$I$设置为 100。

步骤 2：当求出第一个 IMF 分量，即 $IMF_1$ 后，计算出剩余分量$r_1=Y-IMF_1$，再在余量中添加高斯白噪声（同第一步），使用 EMD 对添加噪声后的余量进行分解，得到的

第二个 IMF 分量如下：

$$IMF_2 = \frac{1}{I}\sum_{i=1}^{I} E_1\left(r_1 + \varepsilon_1 E_1\left(w^i\right)\right)$$

步骤 3：重复步骤 2 的操作，在剩余分量$r_k = r_{k-1} - IMF_k(k = 3,\cdots,K)$中添加高斯白噪声，再使用 EMD 进行分解，得到：

$$IMF_{k+1} = \frac{1}{I}\sum_{i=1}^{I} E_1\left(r_k + \varepsilon_k E_k\left(w^i\right)\right)$$

当余量不能再分解时，得到最终余量 R 为：

$$R = Y - \sum_{k=1}^{K} IMF_k$$

最终将原始序列分解为：

$$Y = \sum_{k=1}^{K} IMF_k + R$$

通过该方法能够将原始序列分解为多个子序列，分解后噪声通常存在于某个或某些子序列中（如所示）。现使用排列熵量化噪声的方法将存在噪声的子序列挑选出来，并对其进行降噪。

排列熵是时间序列复杂性的一种度量，常用在从噪声时间序列中发现复杂结构。排列熵的大小可度量时间序列的随机性，排列熵越大，时间序列包含的噪声就越多。排列熵的具体步骤如下所示：

将序列$Y = \{y_1, y_2, \cdots, y_m\}^T$进行重构处理：

$$\mathcal{Y} = \begin{bmatrix} y_1 & y_{1+\tau} & \cdots & y_{1+(d-1)\tau} \\ y_2 & y_{2+\tau} & \cdots & y_{2+(d-1)\tau} \\ \vdots & \vdots & \vdots & \vdots \\ y_m & y_{m+\tau} & \cdots & y_{m+(d-1)\tau} \end{bmatrix}$$

其中$d$是嵌入维数，$\tau$是延迟时间。嵌入延迟可描述复杂结构的时间尺度，嵌入维数越高时，越能检测到时间序列中的复杂模式，但所需的计算时间也会大幅延长。故本文设置$d$为 7，$\tau$为 2。

重构矩阵中的每一行都是一个重构的组件，现对每行的组件元素进行升序排列，即：

$$y_{i+(j_1-1)\tau} \le y_{i+(j_2-1)\tau} \le \cdots \le y_{i+(j_d-1)\tau}$$

式中$j_1, j_2, \cdots, j_d$表示重构组件中列的索引号。因此，矩阵中的每一行均可得到一组索引号序列，记为：

$$\mathrm{S}(j) = (j_1, j_2, \cdots, j_d)$$

式中$j = 1,2,\cdots,m$，其中$m \le d!$。

计算索引号序列的频率分布为$P_1, P_2, \cdots, P_m$，然后按照 Shannon 熵的形式，时间序列 X 的 m 种不同符号序列的排列熵可以定义如下：

$$H_p(d) = -\sum_{j=1}^{m} P_j \ln P_j$$

由 IMF 的排列熵值对 IMFs 进行分类：通过查阅文献和多次实验，设置排列熵的阈值

为 0.7，如 IMF 的排列熵值大于阈值，则认为 IMF 有噪声，否则 IMF 是无噪声。基于此，团队将 IMF 分为有噪和无噪两类。假设 P 为分界点，得到：

$$Y = \sum_{k=1}^{P-1} IMF^k + \sum_{k=P}^{K} IMF^k + R$$

式中$IMF^k(k = 1,2,\cdots,P-1)$是低频无噪 IMF，$IMF^k(k = P,P+1,\cdots,K)$为高频带噪 IMF，R 为最终余量。

现对高频带噪$IMF^k(k = P,P+1,\cdots,K)$进行降噪处理。根据 $O_3$ 本身的特性，本文选择软阈值法对其进行降噪。该方法是对所有高频带噪 IMF 进行阈值化处理，处理过程相对精细。降噪过程中，阈值的选择极为重要：如阈值过小，可能会过滤掉有用信息；如阈值过大，又会导致噪声去除的效果变得不明显。软阈值法设置自适应软阈值$\lambda$，其公式如下：

$$\lambda = \sigma * \sqrt{\frac{2 * \ln m}{\ln(K+1)}}$$

式中，$\sigma$是噪声标准差，即$\sigma = \sqrt{\frac{median(IMF_p)}{0.6745}}$，$median(*)$表示求解中值。

软阈值法用阈值函数$w$量化 IMF 信号值，即使用自适应软阈值$\lambda$与每组 IMF 信号值进行比较：小于阈值的信号值将被置零；而对于比阈值大的信号值，使用信号绝对值与阈值之差替代其原始值。具体公式如下：

$$w_\lambda = \begin{cases} sgn(w)(|w| - \lambda), |w| \geq \lambda \\ 0, |w| < \lambda \end{cases}$$

经过上述处理，得到高频降噪$IMF'^k(k = P,P+1,\cdots,K)$），从而得到 $O_3$ 降噪时间序列。降噪后的序列：

$$Y' = \sum_{K=1}^{P-1} IMF^k + \sum_{K=P}^{K} IMF'^k + R$$

- **CEEMDAN 降噪代码**

```
#重构矩阵
def _embed(x, order, delay):
    N = len(x)
    Y = np.empty((order, N - (order - 1) * delay))
    for i in range(order):
        Y[i] = x[i * delay:i * delay + Y.shape[1]]
    return Y.T
#计算排列熵
def permutation_entropy(time_series, order, delay, normalize=False):
    x = np.array(time_series)
    hashmult = np.power(order, np.arange(order))
    sorted_idx = _embed(x, order=order, delay=delay).argsort(kind='quicksort')
    hashval = (np.multiply(sorted_idx, hashmult)).sum(1)
```

```
    _, c = np.unique(hashval, return_counts=True)
    p = np.true_divide(c, c.sum())
    pe = -np.multiply(p, np.log(p)).sum()
    if normalize:
        pe /= np.log(factorial(order))
    return pe
emd = CEEMDAN(seed=0)
emd.random = np.random.RandomState(30)
emd(S)
imfs, res = emd.get_imfs_and_residue()
plot_imfs(imfs=imfs, residue=res, t=t, include_residue=True)
#各分量排列熵
PE_LIST = []
for i in range(imfs.shape[0]):
    PE_LIST.append(permutation_entropy(imfs[i, :], order=7, delay=1, normalize=True))
#将排列熵大于 0.7 的进行降噪
threshold = 0.04
for i in range(imfs.shape[0]):
    if PE_LIST[i] > 0.7:
        w = imfs[i]
        n = len(w)
        K =imfs.shape[0]
        Standard_deviation = np.sqrt(np.median(np.abs(w))/0.6745)
        threshold = Standard_deviation * np.sqrt(2*np.log(n))/np.log(K+1)
        for j in range(imfs.shape[1]):
            if np.abs(w[j]) >= threshold:
                w[j] = np.sign(w[j])*(np.abs(w[j])-threshold)
            else:
                w[j] = 0
#重构
X = imfs.sum(axis=0)
plot_hunhe(orignal_signal=S, recon_signal=X, t=t)
```

- **IPSOSE_SVR 代码**

```
# 构造数据子集
from random import randrange
from sklearn.model_selection import GridSearchCV
def get_subsample(dataSet, ratio):
    subdataSet = []
    lenSubdata = round(len(dataSet) * ratio)#返回浮点数
    while len(subdataSet) < lenSubdata:
        np.random.seed(30)
        index = randrange(len(dataSet) - 1)
```

```
            subdataSet.append(dataSet[index])
        return subdataSet
SVR_list = []
SVR_score = []
for trainer in range(40):
        bootstrap_train = get_subsample(train_X_Y, ratio=0.5)
        parameters = {'kernel': ['rbf'], 'C': [x for x in np.linspace(1, 1000, num=20)], 'gamma': [x for x in
np.linspace(0.1, 500, num=20)]}
        svc = svm.SVR()
        clf = GridSearchCV(svc, parameters)
        clf.fit(np.array(bootstrap_train)[:, :-1], np.array(bootstrap_train)[:, -1])
        random_forest_model_final = clf.best_estimator_
        SVR_list.append(random_forest_model_final)
        SVR_score.append(clf.best_score_)
# IPSO 参数优化
class PSO(object):
        def __init__(self, population_size, max_steps):
                np.random.seed(30)
                self.w = 0.9    #惯性权重
                self.c1 = self.c2 = 2 #1.4962 学习因子
                self.population_size = population_size    #粒子群数量
                self.dim = 40    #搜索空间的维度
                self.max_steps = max_steps    #迭代次数
                self.x_bound = [0, 1]    #解空间范围
                self.x = np.random.choice([0, 1], size=((self.population_size, self.dim)), replace=True)#x =
np.around(x, decimals=0)
                self.v = np.random.rand(self.population_size, self.dim)    #初始化粒子群速度
                fitness, best_param = self.calculate_fitness(self.x)
                self.p = self.x #个体的最佳位置
                self.param = best_param #记录个体最佳参数
                self.individual_best_fitness = fitness #个体的最优适应度
                self.fitness_score = []
        ###定义目标损失函数
        def calculate_fitness(self, x):
                r2 = []
                for num in range(self.population_size):
                        index1 = np.where(x[num] == 1)
                        temp_core = []
                        for nn in range(index1[0].shape[0]):
                                temp_w = SVR_score[index1[0][nn]] / sum(SVR_score)
                                temp_core.append(SVR_list[index1[0][nn]].predict(testX)*temp_w)
                        sum_one = sum(temp_core)
                        r2.append(metrics.r2_score(testY, sum_one))
```

```
        best_scores_index = np.argmax(r2, axis=0)
        best_scores = np.max(r2)
        best_param = x[best_scores_index]
        return best_scores, best_param
    def evolve(self):
        for step in range(self.max_steps):
            np.random.seed(30)
            r1 = np.random.rand(self.population_size, self.dim)
            np.random.seed(30)
            r2 = np.random.rand(self.population_size, self.dim)
            # 更新速度和权重
            self.w = 0.9-(0.9-0.4)*step/self.max_steps
            self.v = self.w*self.v+self.c1*r1*(self.p-self.x)+self.c2*r2*(self.param-self.x)
            self.x = self.x+self.v
            mask1 = self.x < 0.5
            mask2 = self.x >= 0.5
            self.x[mask1] = 0
            self.x[mask2] = 1
            fitness, best_param = self.calculate_fitness(self.x)
            # 需要更新的个体
            if fitness > self.individual_best_fitness:
                self.p = self.x
                self.param = best_param
                self.individual_best_fitness = fitness
            self.fitness_score.append(self.individual_best_fitness)
        # 绘制迭代次数与适应度误差相关曲线
        plt.rcParams.update({'font.size': 13.5})
        plt.rc('font', family='SimHei', weight='bold')
        x_axis = [i for i in range(self.max_steps)]
        plt.figure(figsize=(10, 4))
        plt.plot(x_axis, self.fitness_score, '-^r')
        plt.xticks(x_axis)
        plt.xlabel('迭代次数')
        plt.ylabel('适应度')
        plt.grid()
        plt.tight_layout()
        plt.savefig('IPSO 寻优过程')
        return self.param, self.individual_best_fitness
my_pso = PSO(20, 20)
local_best_param, best_score = my_pso.evolve()
```

- **GCN 代码**

```
class GraphConvolution(Module):
```

```
    def __init__(self, in_features, out_features, bias=True):
        super(GraphConvolution, 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):
        stdv = 1. / math.sqrt(self.weight.size(1))
        self.weight.data.uniform_(-stdv, stdv)   # 随机化参数
        if self.bias is not None:
            self.bias.data.uniform_(-stdv, stdv)
    def forward(self, input, adj):
        support = torch.matmul(input, self.weight)
        output = torch.matmul(adj, support)
        if self.bias is not None:
            return output + self.bias
        else:
            return output
class GCN(nn.Module):
    def __init__(self, nfeat, nhid, pre):
        super(GCN, self).__init__()
        self.gc1 = GraphConvolution(nfeat, nhid)
        self.gc2 = GraphConvolution(nhid, pre)
        self.dropout = nn.Dropout(p=0.5)
        self.jihuo = nn.Tanh()
    def dropdege(self, adj, level, train):
        if train == False:
            return adj
        if level < 0. or level >= 1:
            raise Exception('Dropout level must be in interval [0, 1[.')
        retain_prob = 1. - level
        sample = np.random.binomial(n=1, p=retain_prob, size=adj.shape)
        adj *= sample
        adj = adj.to(torch.float32)
        return adj
    def forward(self, adj, x, train=True):
        adj = self.dropdege(adj, 0, train)
        x = self.dropout(x)
        x = self.jihuo(self.gc1(x, adj))
```

```
        adj = self.dropdege(adj, 0, train)
        x = self.dropout(x)
        x = self.gc2(x, adj)
        return x
```

- **GRU_GCN_FC 代码**

```
import torch
import torch.nn as nn
class GRU_GCNGraphConvolution(nn.Module):
    def __init__(self, adj, num_gru_units, feat_dim, output_dim, bias: float = 0.0):
        super(GRU_GCNGraphConvolution, self).__init__()
        self._num_gru_units = num_gru_units
        self._output_dim = output_dim
        self._feat_dim = feat_dim
        self._bias_init_value = bias
        self.register_buffer('adj', torch.FloatTensor(adj))
        self.weights = nn.Parameter(torch.FloatTensor(self._num_gru_units + self._feat_dim,
self._output_dim))
        self.biases = nn.Parameter(torch.FloatTensor(self._output_dim))
        self.reset_parameters()
    def reset_parameters(self):
        nn.init.xavier_uniform_(self.weights)
        nn.init.constant_(self.biases, self._bias_init_value)
    def forward(self, inputs, hidden_state):
        batch_size, num_nodes, _ = inputs.shape
        hidden_state = hidden_state.reshape((batch_size, num_nodes, self._num_gru_units))
        concatenation = torch.cat((inputs, hidden_state), dim=2)
        concatenation = concatenation.transpose(0, 1).transpose(1, 2)
        concatenation = concatenation.reshape((num_nodes, (self._num_gru_units + self._feat_dim) *
batch_size))
        a_times_concat = self.adj @ concatenation
        a_times_concat = a_times_concat.reshape((num_nodes, self._num_gru_units + self._feat_dim,
batch_size))
        a_times_concat = a_times_concat.transpose(0, 2).transpose(1, 2)
        a_times_concat = a_times_concat.reshape((batch_size * num_nodes, self._num_gru_units +
self._feat_dim))
        outputs = a_times_concat @ self.weights + self.biases
        outputs = outputs.reshape((batch_size, num_nodes, self._output_dim))
        outputs = outputs.reshape((batch_size, num_nodes * self._output_dim))
        return outputs
class GRU_GCNCell(nn.Module):
    def __init__(self, adj, input_dim, hidden_dim, feat_dim):
        super(GRU_GCNCell, self).__init__()
```

```
        self._input_dim = input_dim
        self._hidden_dim = hidden_dim
        self._feat_dim = feat_dim
        self.register_buffer('adj', torch.FloatTensor(adj))
        self.graph_conv1 = GRU_GCNGraphConvolution(self.adj, self._hidden_dim, self._feat_dim, self._hidden_dim * 2, bias=1.0)
        self.graph_conv2 = GRU_GCNGraphConvolution(self.adj, self._hidden_dim, self._feat_dim, self._hidden_dim)
    def forward(self, inputs, hidden_state):
        concatenation = torch.sigmoid(self.graph_conv1(inputs, hidden_state))
        r, u = torch.chunk(concatenation, chunks=2, dim=1)
        c = torch.tanh(self.graph_conv2(inputs, r * hidden_state))
        new_hidden_state = u * hidden_state + (1.0 - u) * c
        return new_hidden_state, new_hidden_state
class GRU_GCN(nn.Module):
    def __init__(self, adj, hidden_dim, features):
        super(GRU_GCN, self).__init__()
        self._input_dim = adj.shape[0]
        self._hidden_dim = hidden_dim
        self._feat_dim = features
        self.register_buffer('adj', torch.FloatTensor(adj))
        self.GRU_GCN_cell = GRU_GCNCell(self.adj, self._input_dim, self._hidden_dim, self._feat_dim)
        self.regressor = nn.Linear(hidden_dim, 1)
        self.dropout = nn.Dropout(p=0.1)
    def forward(self, inputs):
        batch_size, seq_len, num_nodes, _ = inputs.shape
        inputs = self.dropout(inputs)
        hidden_state = torch.zeros(batch_size, num_nodes * self._hidden_dim).type_as(inputs)
        output = None
        all_out = []
        for i in range(seq_len):
            output, hidden_state = self.GRU_GCN_cell(inputs[:, i, :, :], hidden_state)
            output = output.reshape((batch_size, num_nodes, self._hidden_dim))
            all_out.append(hidden_state)
        output = self.dropout(output)
        predictions = self.regressor(output)
        return predictions
```

（附录仅展示部分代码，其余代码见附件）