1. Restatement of the Problem

Air pollution control is of great significance to the ecological environment and the health and safety of the people. In actual production and life, many air pollutants are produced. By establishing air pollution prediction models, it is possible to predict air pollution conditions in advance, allowing relevant departments to take action early to reduce the harm caused by air pollution [1]. However, the current WRF-CMAQ forecast model is limited by actual meteorological conditions, types of pollutant emissions, and unclear mechanisms of secondary pollutant generation, leading to inaccurate forecast results. Therefore, it is necessary to combine actual data from air quality monitoring stations for secondary modeling to optimize the CMAQ model's forecast results. [2].

The requirements are to establish a model based on the given monitoring station data, relevant technical standards, and actual conditions to study the following issues, while also paying attention to handling cases where data is missing or abnormal.

Problem 1. Based on the relevant calculation formulas and rules, combined with the actual measured data, calculate the Air Quality Index (AQI) and primary pollutant information for monitoring point A from August 25 to August 28, 2020.

Problem 2. Assume that the pollutant emission status in a certain area has not changed. The AQI value in the area will vary due to different meteorological conditions. For example, when meteorological conditions are favorable for pollutant dispersion or deposition, the local AQI will decrease. Use relevant data to classify the meteorological conditions based on the variation patterns of pollutant concentrations and describe their characteristics.

Problem 3. Establish a secondary forecast mathematical model that is applicable to three monitoring points to predict the pollutant values for each day over the next three days. The model should aim to minimize the maximum error of AQI forecast values and achieve the highest possible accuracy for predicting the primary pollutant.

Problem 4. In practice, the pollutant concentration values in neighboring areas often have a certain correlation. Collaborative forecasting across multiple regions can improve the accuracy of predictions. The requirement is to establish a collaborative forecast model that includes four monitoring points. The model should aim to minimize the maximum relative error of AQI forecast values and achieve the highest possible accuracy for predicting the primary pollutant.

2. Problem Analysis

Problem 1: Based on the calculation formulas and relevant rules provided in the appendix, write a MATLAB program. Then, substitute the corresponding data into the calculations to obtain the results.

Problem 2: Due to the presence of missing values (NA) and abnormal data in the actual measured data from the monitoring stations, it is necessary to preprocess the data matrix for smoothing and filling. For missing values (NA), use the lookup replacement method, replacing the missing values with adjacent data. The correlation between data from two consecutive days is relatively high, so the impact on the model prediction results is minimal after replacement. For abnormal data, use the box plot method for interpolation. After preprocessing the data, calculate the corresponding AQI data, and import it into the SOM neural network model for clustering, assuming four categories. Then, classify the meteorological data from the initial forecast. Combining both classifications will yield the classification of meteorological conditions.

Problem 3: The task of the secondary forecast correction model is to correct the error between the initial forecast results and the actual values. Based on the error of the initial forecast, a neural network model is used to obtain the error variation pattern, which is then applied to the initial forecast to reduce the error of the initial forecast.

Problem 4: To establish a regional collaborative model for the four monitoring stations, the data from the four stations can be fitted. The fitted data will replace the forecast data and be input into the secondary forecast correction model from Problem 3 to test whether the fitted model can reduce errors.

3. Model Assumptions

1. Assume that the pollutant emission status remains unchanged, meaning only weather factors affect the AQI.

2. When using MATLAB for calculations, ignore errors in significant figures..

4. Explanation of Symbols

Symbols	Explanation
$IAQI_P$	Air Quality Sub-Index for pollutant P.
C_P	Concentration of pollutant P.
BP_{Hi}, BP_{Lo}	Breakpoint concentrations that are closest to C_P.
$IAQI_{Hi}$, $IAQI_{Lo}$	Corresponding AQI values for BP_{Hi} and BP_{Lo}.

5. Model Establishment and Solution

5.1 Solution for Problem 1

To obtain the Air Quality Index (AQI), we first need to calculate the Air Quality Sub-Index (IAQI) for each pollutant. The calculation formula is as follows:

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

By referring to the tables, we can find the concentration limits for each pollutant and their corresponding IAQI values. After calculating the IAQI values for each pollutant, we select the maximum value, which is the primary pollutant for the day. The calculation rules are as follows:

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

The final calculation results are presented in the following table:

Monitoring date.	Location	AQI calculation	
		AQI	Primary pollutant
2020/8/25	Point A	65	O_3

2020/8/26	Point A	46	O_3
2020/8/27	Point A	108	O_3
2020/8/28	Point A	137	O_3

5.2 Solution for Problem 2

5.2.1 Data preprocessing

Based on the data in Attachment 1, first handle the missing values. For missing values, use the method of value lookup replacement. Specifically, find the NA values and replace them with the preceding data in the same column.

Next, handle outliers. For outliers, use the box plot method for smoothing. A simple box plot consists of five parts: the minimum, median, maximum, and two quartiles. The first quartile, Q1, also known as the lower quartile, is the 25th percentile of the sorted data. The median, F, also known as the second quartile (Q2), is the 50th percentile. The third quartile, Q3, also known as the upper quartile, is the 75th percentile. The interquartile range (IQR) is calculated as Q3-Q1.

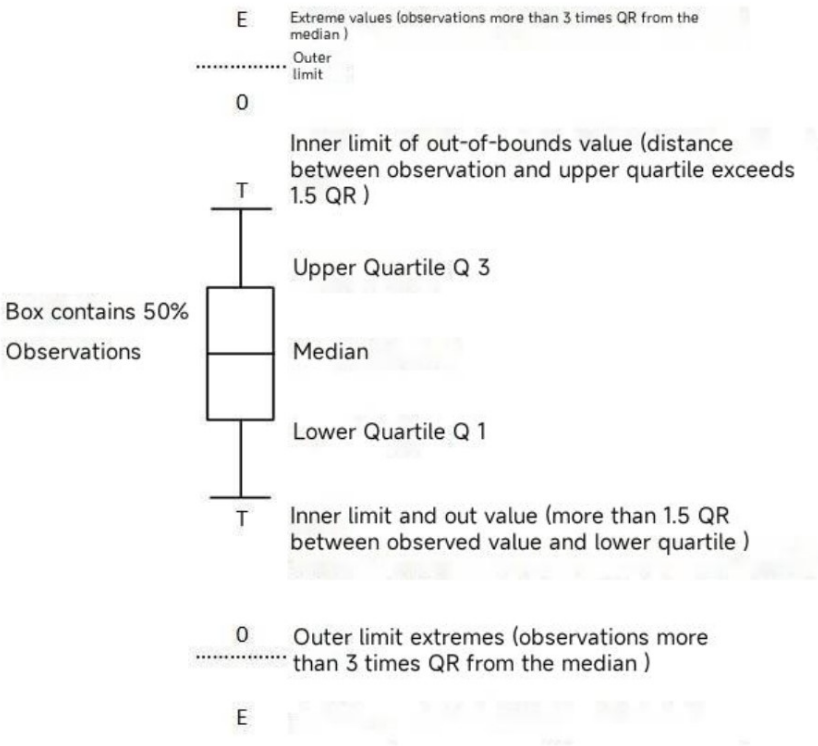

Figure 1: Simple Box Plot

5.2.2 SOM Self-Organizing Neural Network Clustering

Self-Organizing Map (SOM) neural networks can perform unsupervised clustering of data. Essentially, a SOM is a type of neural network with an input layer and a hidden layer, where each node in the hidden layer represents a class. During the training of the neural network, competitive learning is employed. Each input sample finds the most matching node in the hidden layer, referred to as its activation node. Subsequently, the parameters of the activation node are updated using the stochastic gradient descent method. Additionally, nodes close to the activation node are also updated appropriately based on their distance from the activation node.

A notable feature of the SOM model is that the nodes in the hidden layer have a topological relationship. The two-dimensional topology forms a plane.

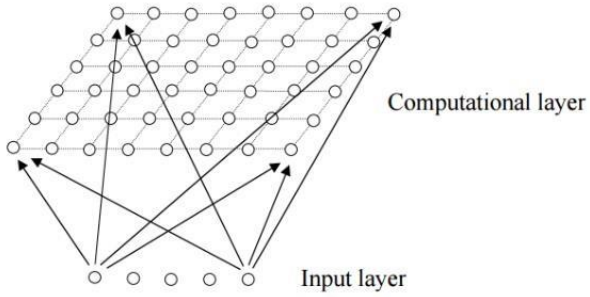

Figure 2: Two-Dimensional Topological Relationship.

SOM can discretize inputs of any dimension into a two-dimensional discrete space. The nodes in the computation layer are fully connected to the nodes in the input layer [4].

The following are the calculation steps:

1. Initialization: Randomly initialize the parameters of each node, with the number of parameters for each node matching the input dimension.

2. Finding Matching Nodes: For each input data, find the most matching node. Assuming the input is D-dimensional data, i.e.,
   $X = \{x_i, i = 1, \ldots, D\}$, the discrimination function can be the Euclidean distance:

$$d_j(x) = \sum_{i=1}^{D} (x_i - w_{ji})^2$$

3. Updating Activation Nodes: After finding the activation node I(x), also update the nodes adjacent to it. Let Sij represent the distance between nodes i and j. For nodes near I(x), assign an update weight:

$$T_{j,I(x)} = \exp(-S_{j,I(x)}^2/2\sigma^2)$$

In simple terms, the closer the nodes are to each other, the smaller the update degree.

4. Updating Node Parameters: Update according to the gradient descent method:

$$\Delta w_{ji} = \eta(t) \cdot T_{j,I(x)}(t) \cdot (x_i - w_{ji})$$

Iterate until convergence.

5.2.3 Calculation Results

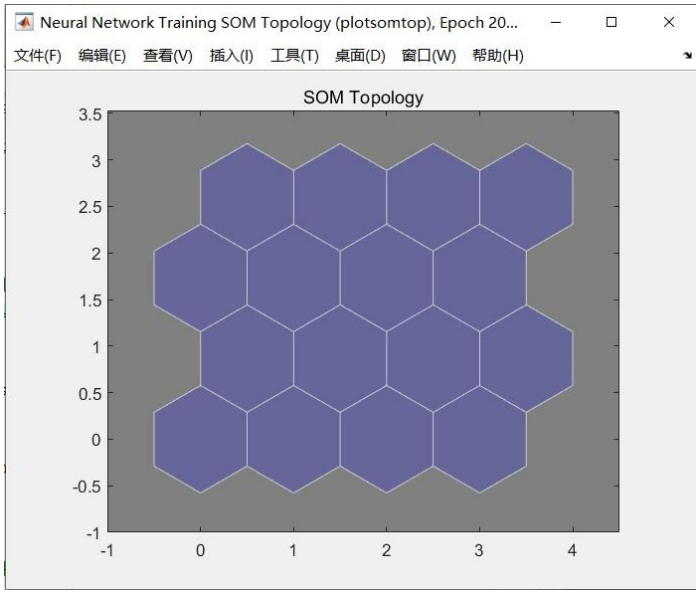

Figure 3: 16 computation layer nodes

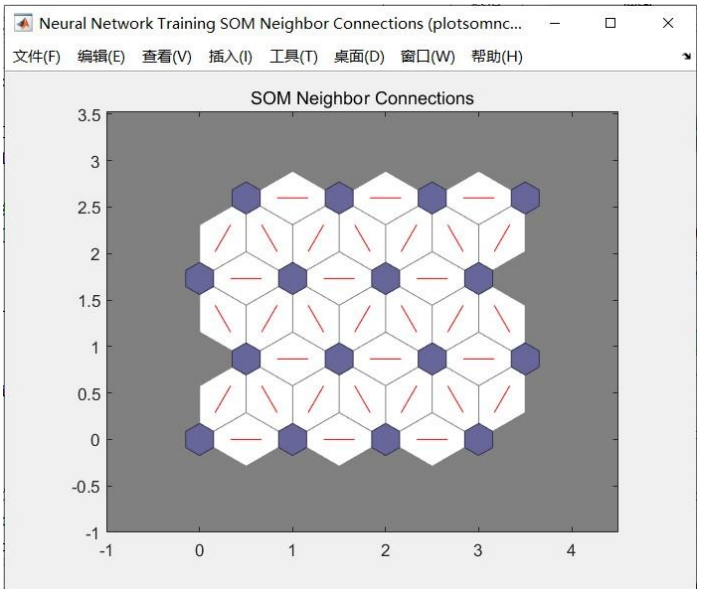

Figure 4: Relationships between nodes.

The correlation between nodes is divided by color; the lighter the color, the higher the correlation; the darker the color, the lower the correlation. As shown in the figure, the lower left is node 1, and the upper right is node 16. For example, the light color between node 1 and node 2 indicates that these two nodes are more similar and likely belong to the same category. Checking the computed classification table, node 1 is in category 2, and node 2 is also in category 2. Similarly, the dark color between node 6 and node 11 indicates they are in different categories, with node 6 in category 1 and node 11 in category 2.

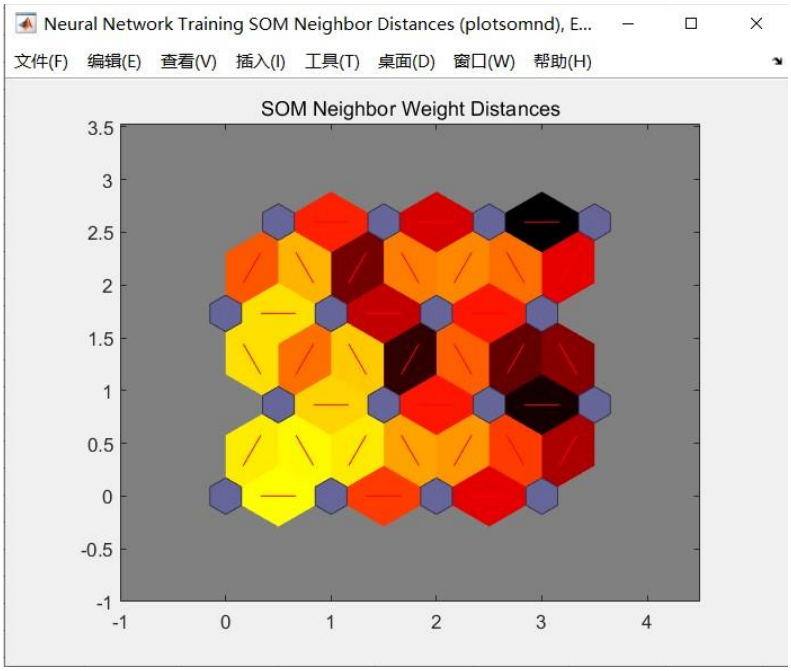

Figure 5: Correlation between nodes

71	2	2
72	1	3
73	1	6
74	2	2
75	2	1
76	1	6
77	2	2

Figure 6: Partial classification result table

Comparing the classification results with the actual measured data at the monitoring points, it can be seen that higher wind speeds and humidity facilitate pollutant dispersion, resulting in lower AQI values.

5.3 Problem Three's Solution

5.3.1 BP Neural Network Structure Optimized by Genetic Algorithm

The network structure is as follows:

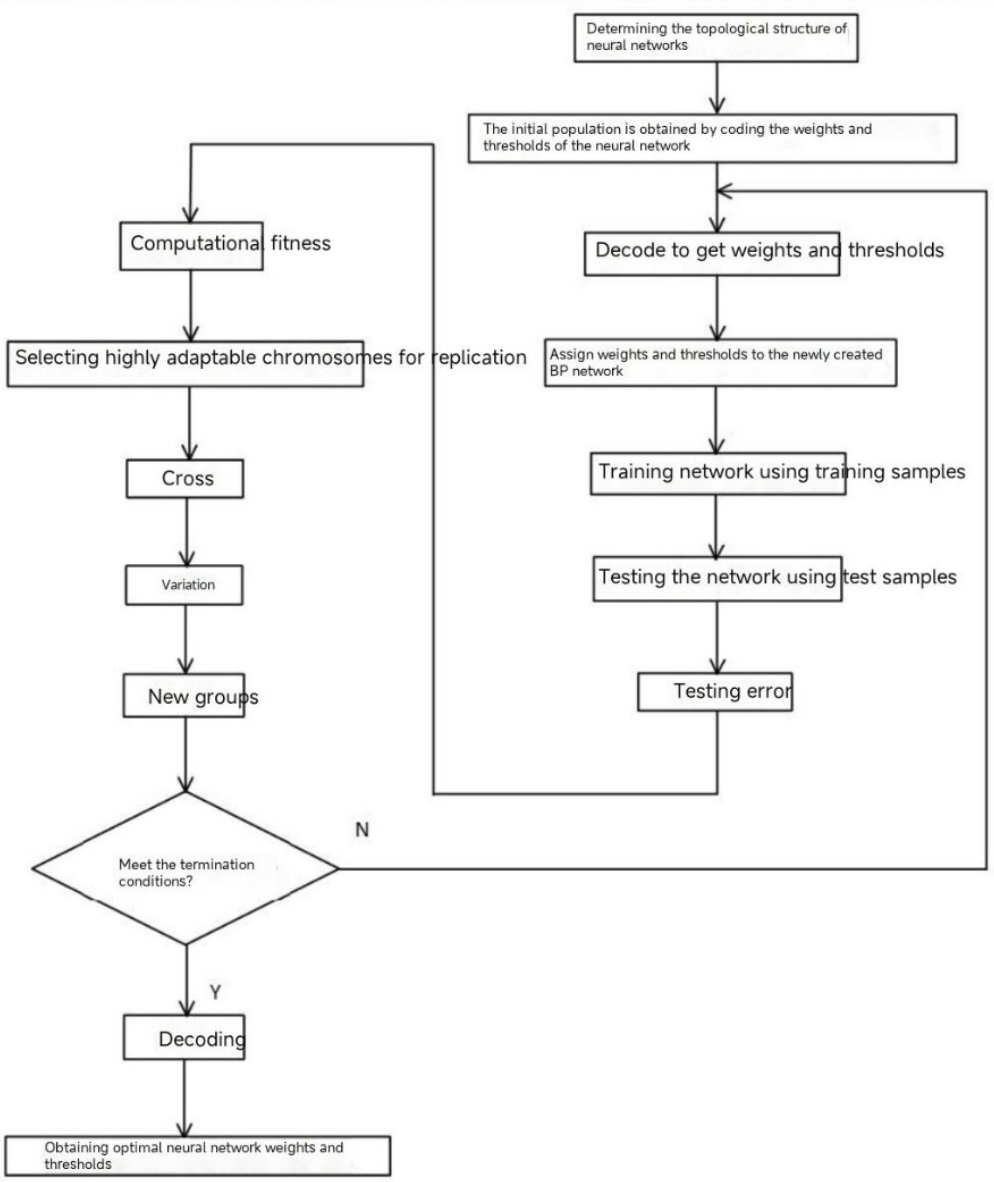

Figure 7: BP neural network structure diagram optimized by genetic algorithm

5.3.2 Implementation of the Neural Network Algorithm

The implementation of the BP neural network algorithm mainly includes the following steps [5]:

1. Network Creation

   There are two important guidelines for determining the structure of a BP network:

   (1) For general pattern recognition problems, a three-layer network can solve the problem effectively.

(2) In a three-layer network, there is an approximate relationship between the number of neurons in the hidden layer (n2) and the number of neurons in the input layer (n1):

$$n_2 = 2 \times n_1 + 1$$

The transfer function for the neurons in the hidden layer uses the S-type tangent function tansig(), and the transfer function for the output layer neurons uses the S-type logarithmic function logsig(). This is because the output mode is 0-1, which perfectly meets the network's output requirements.

2. Network Training and Testing

Network training is a process of continuously adjusting weights and thresholds to minimize the output error. The training function trainlm() can use the Levenberg-Marquardt algorithm for network training. After training, the network also needs to be tested.

5.3.3 Implementation of Genetic Algorithm

Genetic algorithm optimization for the BP neural network involves using genetic algorithms to optimize the initial weights and thresholds of the BP neural network, enabling better sample prediction. The main steps include population initialization, fitness function, selection operator, crossover operator, and mutation operator.

1. Population Initialization

Individuals are encoded using binary coding. Each individual is a binary string, consisting of weights connecting the input layer and hidden layer, hidden layer thresholds, weights connecting the hidden layer and output layer, and output layer thresholds. Each weight and threshold is encoded using M bits of binary code. The encoding of all weights and thresholds together constitutes the encoding of an individual.

2. Fitness Function

The goal is to minimize the residuals between the predicted and expected values. The norm of the error matrix between the predicted and expected sample values is used as the output of the objective function.

3. Selection Operator

The selection operator uses random traversal sampling.

4. Crossover Operator

   The crossover operator uses single-point crossover.

5. Mutation Operator

   The mutation operator randomly selects mutant genes with a certain probability.

6. Model Evaluation and Application

Advantages:

1. The model clearly reflects the variations in pollutant concentrations under different meteorological conditions, and the data classification is intuitive.

2. The algorithm has significant potential for broader application.

Disadvantages:

1. The accuracy and predictive stability of the neural network model need improvement.

2. The model does not make full use of meteorological data.

7. References

[1]姚文强. 基于数值预报的空气质量预测模型的研究[D].浙江理工大学,2017.

[2]李昊. 基于机器学习的兰州市空气质量预报方法研究[D].兰州大学,2017.

[3]景华,陈世平.融合 SOM 和多目标寻优的 Web 服务组合优化方法[J].小型微型计算机系统,2021,42(09):1810-1817.

[4]Cai Ying,Wang Xu,Xiong LiRan,Ahmed Syed Hassan. Difference Analysis of Regional Economic Development Based on the SOM Neural Network with the Hybrid Genetic Algorithm[J]. Computational Intelligence and Neuroscience,2021,2021:

[5]武晨晨,苗霁,祝佳楠,张文惠.遗传算法优化 BP 神经网络的电能质量预测预警研究[J].电工电气,2021(09):18-22.

8. Appendix

```
clear
clc
A=xlsread('C:\Users\WR\Desktop\A1\A1\附件1 监测点A空气质量预报基础数据.xlsx',3);
Cp=zeros(100,2);
for i=1:100
    Cp(i,1)=A(i,5);
end

BP=[0,100,160,215,265,800];
IAQI=[0,50,100,150,200,300];
for i=1:100
    for j=1:5
     if Cp(i,1)>BP(j)&&Cp(i,1)<=BP(j+1)
      BPhi=BP(j+1);
      BPlo=BP(j);
      IAQIhi=IAQI(j+1);
      IAQIlo=IAQI(j);
      IAQIp=(IAQIhi-IAQIlo)/(BPhi-BPlo)*(Cp(i,1)-BPlo)+IAQIlo;
      Cp(i,2)=round(IAQIp);
      break
     end
    end
end
```

```
%AS2为使用excel导入的table数据，程序开始运行前需要手动将表A的数据导入到Matlab中
table_1=AS2(1:819,3:end);    %数据预处理，使用相邻元素取代NA值
data=table2array(table_1);
new_data=findNaN(data);
new_data_1=new_data;

%异常值处理
%采用箱线法
[m,n]=size(new_data);
w1=round(m/4);  %第一四分位位置，
%m1=m/2%中位数位置，
w3=round(3*m/4);%第三四分位位置
```

```
    %将所有异常值处理，最后new_data更新为一个处理好异常值的新矩阵
for j=1:n
    b11=new_data(:,j);
    [a1,b1]=sort(b11);%[a,b]=sort(x);是从小到大排列，a是排序后结果，b是a结果中各元素的原始位置。
    q11=a1(w1,1);    %第一四分位数
    q13=a1(w3,1);    %第三四分位数
    qr1=q13-q11;       %四分位距
    sx1=q13+1.5*qr1;   %上限
    xx1=q11-1.5*qr1;   %下限
    ycz1=[];%正常值矩阵
    ycz2=[];%异常值矩阵
    s1=1;
    s2=1;
    for i=1:m
        if b11(i,1)<=sx1&&b11(i,1)>=xx1   %求正常值矩阵
            ycz1(s1,1)=b11(i,1);
            s1=s1+1;
        else
            ycz2(s2,1)=b11(i,1);    %求异常值举证
            ycz2(s2,2)=i;
            s2=s2+1;
        end
    end
    average_1=round(mean(ycz1,1));  %四舍五入取整

    for i=1:m       %使用正常值部分的均值将异常值替换
        if b11(i,1)>sx1||b11(i,1)<xx1
            b11(i,1)=average_1;
        end
    end
    new_data(i,j)=b11(i,1);
    j=j+1;
end
%以上就是对监测点A空气质量预报基础数据的预处理代码


Cp=zeros(819,2);          %求取臭氧的空气质量指标IAQI
for i=1:819
    Cp(i,1)=new_data_1(i,5);
end
```

```
55 —    BP=[0,100,160,215,265,800];
56 —    IAQI=[0,50,100,150,200,300];
57 —  for i=1:819
58 —      for j=1:5
59 —       if Cp(i,1)>BP(j)&&Cp(i,1)<=BP(j+1)
60 —        BPhi=BP(j+1);
61 —        BPlo=BP(j);
62 —        IAQIhi=IAQI(j+1);
63 —        IAQIlo=IAQI(j);
64 —        IAQIp=(IAQIhi-IAQIlo)/(BPhi-BPlo)*(Cp(i,1)-BPlo)+IAQIlo;
65 —        Cp(i,2)=round(IAQIp);
66 —        break
67 —       end
68 —      end
69 —  end                          %Cp为臭氧为首要污染条件下的IAQI数据
```

```
71      %根据求取的IAQI将空气气象状况标注分类，分类等级再考虑分为几类，说明这几类大概的气象条件
72 —    classes=Cp(:,2);          %将气象条件分类
73 —    classes_min=min(classes);
74 —    classes_max=max(classes);
75 —    classes_q=(classes_max-classes_min)/4;
76 —  for i=1:m
77 —      if classes(i,1)<classes_min+round(classes_q)
78 —          classes(i,1)=1;
79 —      elseif classes(i,1)<classes_min+round(classes_q*2)
80 —          classes(i,1)=2;
81 —      elseif classes(i,1)<classes_min+round(classes_q*3)
82 —          classes(i,1)=3;
83 —      else
84 —          classes(i,1)=4;
85 —      end
86 —  end
```

```
classes=classes';


%数据归一化并进行矩阵转置
new_data=mapminmax(new_data);
new_data=new_data';
%训练集——800个样本
P_train=new_data(:,1:700);
T_train=classes(:,1:700);
%测试集——19个样本
P_test=new_data(:,701:end);
T_test=classes(:,701:end);

%SOFM神经网络(一种自组织的神经网络)的创建、训练及仿真测试
net=newsom(P_train,[4 4]);

%设置训练参数
net.trainParam.epochs=200;

%训练网络
net=train(net,P_train);
```

```
%仿真测试

%训练集测试        %数据量有800个，太大了暂时无需测试。只需对测试集中的数据测试，看分类效果即可。
t_sim_sofm_1=sim(net,P_train);
T_sim_sofm_1=vec2ind(t_sim_sofm_1);

%测试集测试
t_sim_sofm_2=sim(net,P_test);
T_sim_sofm_2=vec2ind(t_sim_sofm_2);

%sofm神经网络对测试集分类结果
result_sofm_1=[T_train' T_sim_sofm_1']
result_sofm_2=[T_test' T_sim_sofm_2']
```

```
%% 加载神经网络的训练样本 测试样本每列一个样本 输入P 输出T，T是标签
%样本数据就是前面问题描述中列出的数据
%epochs是计算时根据输出误差返回调整神经元权值和阀值的次数
load data
% 初始隐层神经元个数
hiddennum=31;
% 输入向量的最大值和最小值
threshold=[0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1];
inputnum=size(P,1); % 输入层神经元个数
outputnum=size(T,1); % 输出层神经元个数
w1num=inputnumhiddennum; % 输入层到隐层的权值个数
w2num=outputnumhiddennum;% 隐层到输出层的权值个数
N=w1num+hiddennum+w2num+outputnum; %待优化的变量的个数
```

```
%% 定义遗传算法参数
NIND=40; %个体数目
MAXGEN=50; %最大遗传代数
PRECI=10; %变量的二进制位数
GGAP=0.95; %代沟
px=0.7; %交叉概率
pm=0.01; %变异概率
trace=zeros(N+1,MAXGEN); %寻优结果的初始值

FieldD=[repmat(PRECI,1,N);repmat([-0.5;0.5],1,N);repmat([1;0;1;1],1,N)]; %区域描述器
Chrom=crtbp(NIND,PRECI*N); %初始种群
```

```
%% 优化
gen=0; %代计数器
X=bs2rv(Chrom,FieldD); %计算初始种群的十进制转换
ObjV=Objfun(X,P,T,hiddennum,P_test,T_test); %计算目标函数值
while gen<MAXGEN
fprintf('%d\n',gen)
FitnV=ranking(ObjV); %分配适应度值
SelCh=select('sus',Chrom,FitnV,GGAP); %选择，随机遍历抽样
SelCh=recombin('xovsp',SelCh,px); %重组，单点交叉
SelCh=mut(SelCh,pm); %变异
X=bs2rv(SelCh,FieldD); %子代个体的十进制转换
ObjVSel=Objfun(X,P,T,hiddennum,P_test,T_test); %计算子代的目标函数值
[Chrom,ObjV]=reins(Chrom,SelCh,1,1,ObjV,ObjVSel); %重插入子代到父代，得到新种群，注意插入后新种群与老种群的规模
%代沟只是说选择子种群的时候是选择95%的个体作为待插入的子种群
%1，父代chrome和子代selch中的子种群个数都是1，1，基于适应度的选择，子代代替父代中适应度最小的个体
X=bs2rv(Chrom,FieldD);%插入完成后，重新计算个体的十进制值
gen=gen+1; %代计数器增加
%获取每代的最优解及其序号，Y为最优解,I为个体的序号
[Y,I]=min(ObjV);%Objv是目标函数值，也就是预测误差的范数
trace(1:N,gen)=X(I,:); %记下每代个体的最优值，即各个权重值
trace(end,gen)=Y; %记下每代目标函数的最优值，即预测误差的范数
end
```

```
%% 画进化图
figure(1);
plot(1:MAXGEN,trace(end,:));
grid on
xlabel('遗传代数')
ylabel('误差的变化')
title('进化过程')
bestX=trace(1:end-1,end);%注意这里仅是记录下了最优的初始权重，训练得到的最终的网络的权值并未记录下来
bestErr=trace(end,end);
fprintf(['最优初始权值和阈值:\nX=',num2str(bestX'),'\n最小误差err=',num2str(bestErr),'\n'])
```

```
%% 比较优化前后的训练&测试
callbackfun

子函数：
function Obj=Objfun(X,P,T,hiddennum,P_test,T_test)
%% 用来分别求解种群中各个个体的目标值
%% 输入
% X：所有个体的初始权值和阈值
% P：训练样本输入
% T：训练样本输出
% hiddennum：隐含层神经元数
% P_test:测试样本输入
% T_test:测试样本期望输出
%% 输出
% Obj：所有个体的预测样本的预测误差的范数
%这个函数的目的就是用种群中所有个体所代表的神经网络的初始权重值去进行网络的训练，训练次数是1000次，然
%后得出所有个体作为初始权重训练网络1000次所得出的预测误差，也就是这里的obj，返回到原函数中，迭代maxgen=50次
%记录下每一代的最优权重值和最优目标值(最小误差值)
[M,N]=size(X);
Obj=zeros(M,1);
for i=1:M%M是40，即有40个个体，每个个体就是一次初始权重，在BPfun中用每个个体作为初始值去进行了1000次的训练
Obj(i)=BPfun(X(i,:),P,T,hiddennum,P_test,T_test);%Obj是一个40*1的向量，每个值对应的是一个个体作为初始权重值
%网络1000次得出来的误差
end
```

```
function err=BPfun(x,P,T,hiddennum,P_test,T_test)
%% 训练&测试BP网络
%% 输入
% x：一个个体的初始权值和阈值
% P：训练样本输入
% T：训练样本输出
% hiddennum：隐含层神经元数
% P_test:测试样本输入
% T_test:测试样本期望输出
%% 输出
% err：预测样本的预测误差的范数
%用每一个个体的初始权值去训练1000次
inputnum=size(P,1); % 输入层神经元个数
outputnum=size(T,1); % 输出层神经元个数
%% 新建BP网络
%神经网络的隐含层神经元的传递函数采用S型正切函数tansing（），输出层神经元的函数采用S型对数函数logsig（）
net=newff(minmax(P),[hiddennum,outputnum],{'tansig','logsig'},'trainlm');
```

```
%% 设置网络参数：训练次数为1000，训练目标为0.01，学习速率为0.1
net.trainParam.epochs=1000;%允许最大训练次数，实际这个网络训练到迭代次数是3时就已经到达要求结束了
net.trainParam.goal=0.01;%训练目标最小误差，应该是mean square error， 均方误差，就是网络输出和目标值
LP.lr=0.1;%学习速率学习率的作用是不断调整权值阈值。w(n+1)=w(n)+LP.lr*(d(n)-y(n))x(n),d(n)是期望的相
%量化的实际响应，x(n)是输入向量，如果d(n)与y(n)相等的话，则w(n+1)=w(n),这里是指输入到隐含层的调整方式
%隐含层到输出层的调整 Iout(j)=1/(1+exp(-I(j)));
%dw2=eIout;db2=e';w2=w2_1+xitedw2';e是错误值
%b2=b2_1+xitedb2';xite是学习率
%对于traingdm等函数建立的BP网络，学习速率一般取0.01-0.1之间。
net.trainParam.show=NaN;
% net.trainParam.showwindow=false; %高版MATLAB
%% BP神经网络初始权值和阈值
w1num=inputnumhiddennum; % 输入层到隐层的权值个数
w2num=outputnumhiddennum;% 隐层到输出层的权值个数
w1=x(1:w1num); %初始输入层到隐层的权值
B1=x(w1num+1:w1num+hiddennum); %初始隐层阈值
w2=x(w1num+hiddennum+1:w1num+hiddennum+w2num); %初始隐层到输出层的阈值
B2=x(w1num+hiddennum+w2num+1:w1num+hiddennum+w2num+outputnum); %输出层阈值
net.iw{1,1}=reshape(w1,hiddennum,inputnum);%输入到隐藏层的权重
net.lw{2,1}=reshape(w2,outputnum,hiddennum);%隐藏到输出层的权重
net.b{1}=reshape(B1,hiddennum,1);
net.b{2}=reshape(B2,outputnum,1);
```

```
%% 训练网络以
net=train(net,P,T);
%% 测试网络
disp(['1、使用随机权值和阈值 '])
disp('测试样本预测结果：')
Y1=sim(net,P_test)%测试样本的网络仿真输出
err1=norm(Y1-T_test); %测试样本的仿真误差
err11=norm(sim(net,P)-T); %训练样本的仿真误差
disp(['测试样本的仿真误差:',num2str(err1)])
disp(['训练样本的仿真误差:',num2str(err11)])

%% 使用遗传算法
%% 使用优化后的权值和阈值，利用遗传算法得出来的最优的初始权重和阈值去进行网络的初始化
inputnum=size(P,1); % 输入层神经元个数
outputnum=size(T,1); % 输出层神经元个数
%% 新建BP网络
net=newff(minmax(P),[hiddennum,outputnum],{'tansig','logsig'},'trainlm');
%% 设置网络参数：训练次数为1000，训练目标为0.01，学习速率为0.1
net.trainParam.epochs=1000;
net.trainParam.goal=0.01;
LP.lr=0.1;
```

```
%% BP神经网络初始权值和阈值
w1num=inputnumhiddennum; % 输入层到隐层的权值个数
w2num=outputnumhiddennum;% 隐层到输出层的权值个数
w1=bestX(1:w1num); %初始输入层到隐层的权值
B1=bestX(w1num+1:w1num+hiddennum); %初始隐层阈值
w2=bestX(w1num+hiddennum+1:w1num+hiddennum+w2num); %初始隐层到输出层的阈值
B2=bestX(w1num+hiddennum+w2num+1:w1num+hiddennum+w2num+outputnum); %输出层阈值
net.iw{1,1}=reshape(w1,hiddennum,inputnum);
net.lw{2,1}=reshape(w2,outputnum,hiddennum);
net.b{1}=reshape(B1,hiddennum,1);
net.b{2}=reshape(B2,outputnum,1);
%% 训练网络以
net=train(net,P,T);
%% 测试网络
disp(['2、使用优化后的权值和阈值'])
disp('测试样本预测结果：')
Y2=sim(net,P_test)%测试样本的仿真输出
err2=norm(Y2-T_test);%测试样本的仿真误差
err21=norm(sim(net,P)-T);%训练样本的仿真误差
disp(['测试样本的仿真误差:',num2str(err2)])
disp(['训练样本的仿真误差:',num2str(err21)])
```