湖南大學
HUNAN UNIVERSITY

工程优化期末大作业

姓名：岳靖凯

学号：S2302W0227

专业：机械

班级：机械 2311 班

任课老师：王琥

2024年 8月 12 日

基于 GA-BP 神经网络在核退役工程估算应用的优化研究

1、问题背景介绍

2015 年 3 月 12 日，我国首个核电厂退役工程技术研发中心正式揭牌成立，填补了国内核电厂退役领域空白。相比于国外，我国核电起步较晚，但是最早的秦山第一核电厂也于 1994 年投入运行，意味着近年来我国未来同样也会面临核退役的局面，因此研究核设施退役时的成本预测问题迫在眉睫。

关于工程项目估算，Oberlender 和 Trost 指出在建设项目中前期成本估算的准确性对于业主和项目管理团队来说是非常重要的[1]。目前国内外研究中一致认同采用相似的已建工程项目的成本资料作为当前工程项目的依据的近似估计是较为可取的方法。随着计算机技术的迅速发展，许多专家学者将模糊数学[2]、灰色理论[3]、专家系统、人工神经网络等现代方法和技术应用于工程项目造价估算中。其中近年兴起的人工神经网络的一个明显特征即是具有通过学习最佳逼近非线性映射的能力[4]。

BP 神经网络是一种模拟人大脑神经网络工作原理的机器学习算法，通过多次训练和学习可以建立问题的预测模型，但由于其实质上是一种局部探索最优解的算法，很容易将局部极值当做全局的最优解，本文针对 BP 神经网络的缺点，引入 GA 算法进行改进，并运用收集的相关样本数据进行实验，验证该方法的可行性。

2、网络模型

2.1 BP 神经网络模型

BP 神经网络具有良好的自适应性和自主学习能力等，其结构和训练方法相对简单，典型的 BP 神经网络结构主要包含三层：输入层、隐含层和输出层。其结构如图 1 所示：

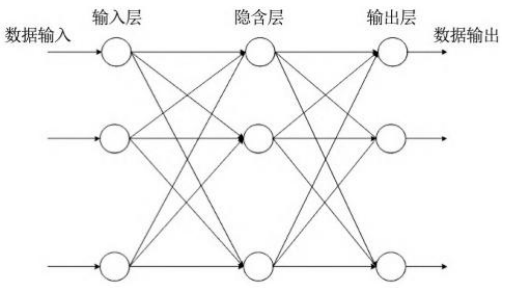

图 1 BP 神经网络结构图

输入层用于与外界相联系，将数据输入到网络中，隐含层通过激活函数等将数据进行处理，若经处理的数据与预测值相比误差较小，则通过输出层输出；若两者误差相对较大，则需通过调节各连接层的权值和阈值，将数据返回到输入层重新进行输入，如此循环往复，直至满足误差要求，数据从输出层输出。这一过程也被称作 BP 神经网络的误差反向传播。而在实际应用过程中，由于 BP 神经网络的设计结构问题以及梯度下降算法的特点，其存在容易陷入局部极值、计算速度慢、网络不稳定等缺点。

输入层用于与外界相联系，将数据输入到网络中，隐含层通过激活函数等将数据进行处理，若经处理的数据与预测值相比误差较小，则通过输出层输出；若两者误差相对较大，则需通过调节各连接层的权值和阈值，将数据返回到输入层重新进行输入，如此循环往复，直至满足误差要求，数据从输出层输出。这一过程也被称作 BP 神经网络的误差反向传播。而在实际应用过程中，由于 BP 神经网络的设计结构问题以及梯度下降算法的特点，其存在容易陷入局部极值、计算速度慢、网络不稳定等缺点。

2.2 GA-BP 神经网络

遗传算法（Genetic Algorithm，GA）是由 Holland 系统阐述，并在此基础上进一步提出了模式理论，由此标志着遗传算法正式诞生。遗传算法的产生主要受生物进化论“选优淘劣，适者生存”的启迪。遗传算法的初始种群由编码产生，初始种群经过适应度值计算、是否满足收敛准则等操作进行是否为最优解的判断，若不是，再进行选择、交叉、变异等操作来产生新的种群，新种群又进行重新判断，直至寻求到最优解，由此可看出遗传算法可以进行全局内的最优解搜寻，这一特点正好弥补了 BP 神经网络的不足。遗传算法用于改进 BP 神经网络的要点在于利用遗传算法的三个重要操作优化 BP 神经网络的初始阈值和权重。具体的优化流程如图 2 所示。

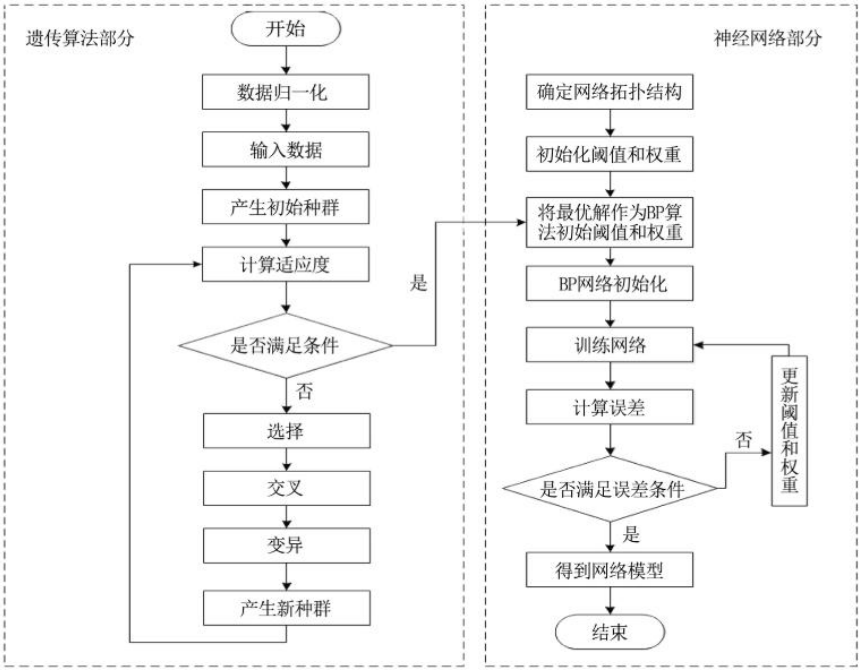

图 2 GA-BP 神经网络结构图

3、工程特征指标的选取和处理

3.1 工程特征指标的选取

由于退役项目人员类别的特殊性以及复杂多样性，影响退役人工成本的因素有很多，但并不是每个因素都对退役人工成本预测有着非常重要的影响，所以在建立预测模型之前，需选取特定的影响因素作为模型的输入量，经过模型处理后得到输出结果。而当选取过多的因素作为模型输入量时，会降低预测模型计算速度。所以在选取影响因素方面需遵循适度原则，筛选出对退役人工成本有着较大影响或者能够反映项目特征的因素，即特征指标。在选取特征指标时，需综合考虑各方面因素，筛选出重要程度较高的因素。

本文结合已国内外退役成功案例的人员相关数据，选取了认为对退役人工成本有较大影响的 7 个特征指标（人员类别），包括没有受过专门培训的辅助工人（LBR）、受过专门培训的熟练工人（SKW）、受过专门培训的技术员且是中学以上教育学历（TCN）、具有行政办公技能的行政人员且是中学以上教育学历（ADM）、核工程师且是大学以上教育学历（ENG）、核工程师且是大学以上教育学历，并在该学科须域有大约 10 年的经验（SEN）、管理层人员且长期在该领域（MNG）。本文选取 12 组住宅数据作为基础样本，其中 10 个为训练样本，2 个为测试样本，模型输入量为经处理后的 7 个人员类别的人数，如表 1 所示。假设退役工程持续五年，模型输出量为每个项目的退役人工成本。

表 1 退役人工成本原数据

样本	LBR（人）	SKW（人）	TCN（人）	ADM（人）	ENG（人）	SEN（人）	MNG（人）	人工成本（万元）
1	83	91	84	42	33	21	5	5600
2	87	93	89	48	37	28	8	6300
3	85	91	83	46	35	27	4	5750
4	86	93	86	49	35	22	6	5880
5	91	85	83	48	42	23	6	6000
6	85	94	85	42	38	26	9	6200
7	84	102	82	38	40	36	9	6620
8	74	119	71	32	31	42	7	6300
9	82	87	78	35	45	42	5	6400
10	105	83	92	26	44	16	2	5600
11	91	72	86	54	42	32	6	6200
12	84	96	82	48	36	28	7	6110

3.2 数据预处理

输入数据的预处理是对 BP 神经网络进行训练前期一个非常重要的工作，若未对数据进行规范化处理，训练速度将十分缓慢，网络也难以收敛，得不出正确的结果[5]。对于定量的因素，应进行归一化处理，并使其属于[-1,1]，如图表 2 所示。样本数据经过处理后才能够被模型所用，并发挥模型的性能。

定量因素变量。在输入到 BP 神经网络前应归一化，本文采用方法是 MATLAB 语言中的 mapminmax 函数所使用的方法，实现代码如:［GYS，PS］＝ mapminmax (YS)，其中变量YS 和GYS 分别表示原始数据和归一化后的数据，PS 归一化的方式记录载体。

表 2 退役人工成本归一化后数据

样本	LBR（个）	SKW（个）	TCN（个）	ADM（个）	ENG（个）	SEN（个）	MNG（个）	人工成本（万元）
1	-0.4193	0.5555	0.2381	0.3913	-0.7142	0.6153	-0.1428	-1
2	-0.1612	0.4444	0.7142	0.9130	-0.1428	0.0769	0.7142	0.3725
3	-0.2903	0.5555	0.1428	0.7391	-0.4285	0.1538	-0.4285	-0.7058
4	-0.2258	0.4444	0.4285	1	-0.4285	0.5384	0.1428	-0.4509
5	0.09677	0.8888	0.1428	0.9130	0.5714	0.4615	0.1428	-0.2156
6	-0.2903	0.3888	0.3333	0.3913	0	0.2307	1	0.1764
7	-0.3548	0.0555	0.0476	0.04347	0.2857	0.5384	1	1
8	-1	1	-1	-0.4782	-1	1	0.4285	0.3725
9	-0.4838	0.7777	0.3333	-0.2173	1	1	-0.1428	0.5686
10	1	-1	1	-1	0.8571	-1	-1	-0.9215
11	-0.4285	0.1538	0.1612	0.7142	0.9130	0.0476	0.04347	0.4285
12	-0.7142	0.9130	0.4615	-0.6153	-0.1428	0.1428	0.9130	0.1428

4、GA-BP 神经网络成本预测模型的建立

4.1 网络结构设计

本案例将设计两组预测网络，第一个是基于 MATLAB 自带工具箱里的 BP 神经网络进行训练以及预测退役人工成本费用。第二个是利用 GA 遗传算法改进 BP 神经网络的权值和阈值，改进后将适应度最优的权值和阈值赋予 BP 神经网络，再进行训练以及预测。

第一组预测选取的模型由 MATLAB 软件编码实现，也可通过软件自带的工具箱进行实现。本案例所建模型采用常见的三层网络结构模式，模型的输入量为上文按照人员类别指标的选取的 7 个特征因素，即输入节点数为 7，用 X1～X7 表示；模型的输出量即为预测结果，选择退役人工成本作为输出量，用 Y 表示，即输出节点为 1；隐含层神经元个数由 kolmogorov 定理求解，经过不断尝试，当 a 的取值为 3，得到隐含层节点数为 6 时，所建立的模型训练效果最好，所以最终确定 BP 神经网络的结构为 7-6-1，模型结构如图 3 所示。

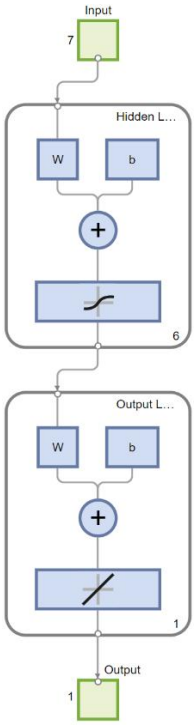

图 3 预测模型结构

使用 GA 优化 BP 神经网络时要解决的首要问题是 GA 的编码问题，GA 有不同的编码方式，不同的编码方式会导致选择、交叉、变异等算子的不同，本文采用的是实数编码，遗传算法的编码长度 l 为 61[6],具体公式如下：

$$l = \mathrm{n1} \times \mathrm{n2} + \mathrm{n2} \times \mathrm{m} + \mathrm{n2} + \mathrm{m}$$

其中，n1 为输入层神经元个数；m 为输出层神经元个数：n2 为隐含层神经元个数。其他相关参数设置如表 3 所示。

表 3 GA-BP 神经网络参数设置

参数名称	参数设置	参数名称	参数设置
训练函数	trainlm	训练次数	200
隐含层传递函数	tansig	初始种群	40
输出层传递函数	purelin	交叉概率	0.4
学习速率	0.1	变异概率	0.05
目标误差	0.00001	迭代次数	500

4.2 神经网络模型的训练与预测

4.2.1 BP 模型的训练

在 MATLAB 软件中导入样本数据，将上文中提及的前 10 组数据设置为网络的训练样本，后 2 组设置为预测样本，设置好网络结构和网络参数，通过代码构建 BP 神经网络，对网络进行训练，训练效果可通过绘制的回归线进行查验，其中，R 表示相关系数，R 值越靠近数值 1，则说明训练的网络对样本数据的拟合程度越好；实线表示理想回归直线；虚线表示最优回归直线，本文训练样本的拟合程度如图 4 所示，训练数据的 R 值为 0.99829，虚线与实线几乎重合，说明训练后的神经网络具有较好的线性拟合能力。

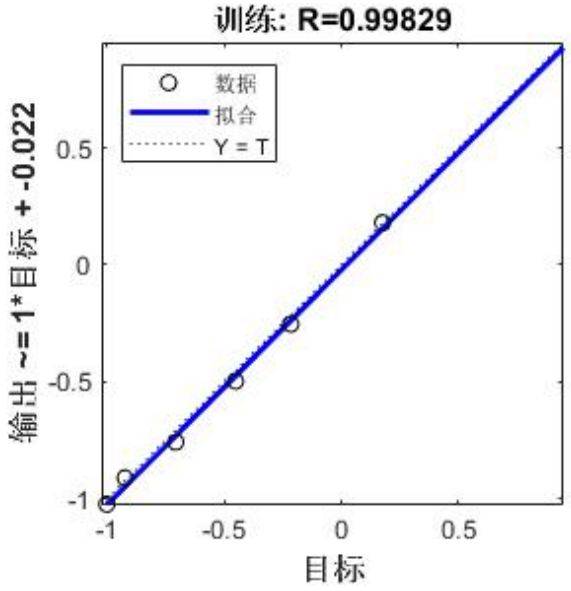

图 4 训练样本拟合程度

设定均方误差 MSE=0.1，收敛过程如图 5 所示，分析后可知网络训练学习效果较好。

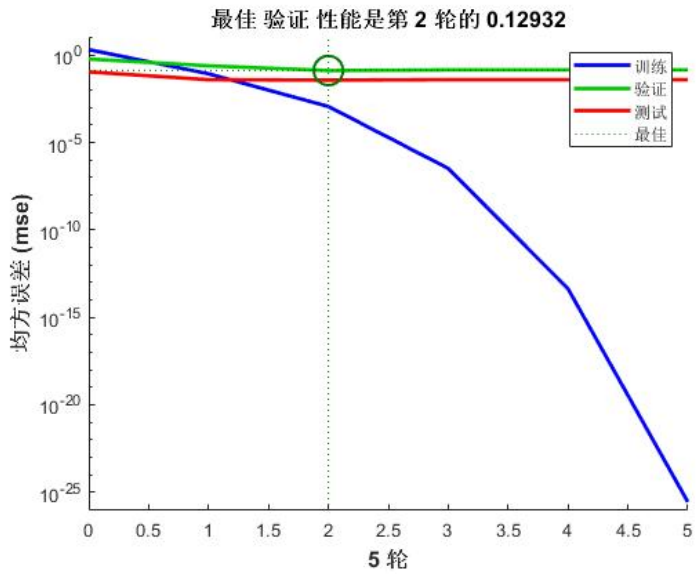

图 5 训练样本拟合程度

4.2.2 GA-BP 模型的训练

将训练好的样本先进行单一的 BP 神经网络造价预测后，得到优化前的退役人工成本预测误差，如图 6 所示，可以得出退役人工成本误差的最大值也仅有 7.3%，而在所有实验当中，最低误差达到了 0.55%。在此基础上编写 GA 代码，求得最优解再次进行网络训练，用 GA 优化后的网络再次进行退役人工成本预测，得到优化后的预测误差仅有 0.27%，如图 7 所示。说明 GA-BP 预测模型在这方面具有更稳定的预测效果，更具可行性。

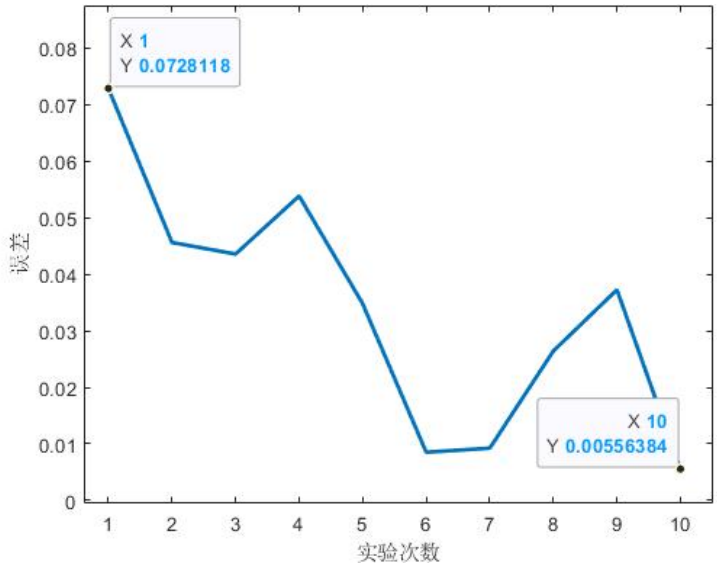

图 6 BP 预测误差程度

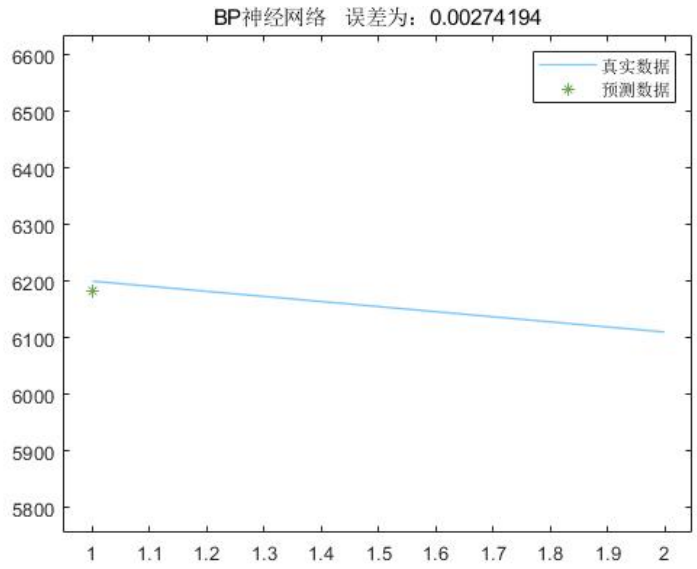

图 7 GA-BP 预测误差程度

5、结语

本文提出了基于 GA-BP 神经网络的退役人工成本预测模型，在特征指标（人员类别）的选取上仅选取 7 个，以此提高模型的学习速率；收集了 12 组样本数据进行实验，验证所构建预测模型的可行性，证明将人工智能算法引入工程估算预测具有可行性。

将 GA 优化前后的预测数据与原始值进行了误差对比，结果表明，两种模型都具有一定的预测效果，但经 GA 优化后的模型预测数据的误差更小，精确度更高，说明优化后的模型稳定性更好，预测效果更好，可信度更高。可在核设施退役时提供较为精确的耗费人工成本评估基础和退役工程成本优化的规划导向，也能使得此预测模型在核设施建立初期就能发挥前瞻性、战略性的关键技术作用。

参考文献

[1] PIETROFORTE R, STEFANI T P. ASCE Journal of Construction Engineering and Management: Review of the Years 1983–2000[J/OL]. Journal of Construction Engineering and Management, 2004, 130(3): 440-448. DOI:10.1061/(ASCE)0733-9364(2004)130:3(440).

[2] 张利荣. 多基元模糊算法在工程估价中的应用[J]. 施工技术, 2010, 39(6): 64-66.

[3] 孙涛. 灰色系统预测理论在建筑工程造价中的应用[D/OL]. 西北工业大学, 2006[2022-10-20].

[4] 刘洋. 人工神经网络技术研究[J]. 农业网络信息, 2013(9): 29-31.

[5] 罗定贵, 王学军, 郭青. 基于 MATLAB 实现的 ANN 方法在地下水质评价中的应用[J/OL]. 北京大学学报(自然科学版), 2004(2): 296-302. DOI:10.13209/j.0479-8023.2004.039.

[6] 谢金豪, 刘文昌. 基于 GA-BP 神经网络的建筑工程造价预测研究[J/OL]. 建筑经济, 2022, 43(S1): 235-240. DOI:10.14181/j.cnki.1002-851x.2022S10235.

MATLAB 程序

1 初始数据（shuju_data）：

	1	2	3	4	5	6	7	8	9
1	83	91	84	42	33	21	5	5600	
2	87	93	89	48	37	28	8	6300	
3	85	91	83	46	35	27	4	5750	
4	86	93	86	49	35	22	6	5880	
5	91	85	83	48	42	23	6	6000	
6	85	94	85	42	38	26	9	6200	
7	84	102	82	38	40	36	9	6620	
8	74	119	71	32	31	42	7	6300	
9	82	87	78	35	45	42	5	6400	
10	105	83	92	26	44	16	2	5640	
11	91	72	86	54	42	32	6	6200	
12	84	96	82	48	36	28	7	6110	
13									

2 子函数

因为需要在选择函数当中考虑权值和阈值的的最佳适应度，需要用到 MATLAB 工具箱中的神经网络训练，因此没有使用谢菲尔德大学的 matlab 遗传算法工具箱，自行编程选择（采用轮盘赌），交叉（采用 PMX 算子），变异等函数。

2.1 select（选择函数）

```
function [new_chrom,new_fitness]=select(input_chrom,fitness_group,group_num)
% 用轮盘赌在原来的函数里选择
% fitness_group        种群信息
% group_num             种群规模
% newgroup              选择后的新种群
%求适应度值倒数
fitness1=10./fitness_group; %individuals.fitness 为个体适应度值
%个体选择概率
sumfitness=sum(fitness1);
sumf=fitness1./sumfitness;
%采用轮盘赌法选择新个体
index=[];
for i=1:1000     %group_num 为种群数
    pick=rand;
    while pick==0
        pick=rand;
    end
  for j=1:group_num
        pick=pick-sumf(j);
        if pick<0
            index=[index j];
            break;
        end
  end
  if length(index) == group_num
        break;
 end
```

```
end
%新种群
new_chrom=input_chrom(index,:);
new_fitness=fitness_group(index);
end
```

2.2 cross（交叉函数）

```
function new_chrom=Cross(cross_pro,lenchrom,input_chrom,group_num,limit)
%随机选择两个染色体位置交叉
% cross_pro                    交叉概率
% lenchrom                     染色体的长度，即所有参数的数量
% input_chrom                  染色体群,经过选择遗传下来的表现比较好的
% group_num                    种群规模
% new_chrom                    交叉后的染色体
 for i=1:group_num
%每一轮 for 循环中，可能会进行一次交叉操作，染色体是随机选择的，交叉位置也是随机选择的，
%但该轮 for 循环中是否进行交叉操作则由交叉概率决定（continue 控制）
    pick=rand(1,2);                % 随机选择两个染色体进行交叉
     while prod(pick)==0           %连乘
          pick=rand(1,2);
     end
    index=ceil(pick.*group_num);   % 交叉概率决定是否进行交叉
    pick=rand;
     while pick==0
          pick=rand;
     end
     if pick>cross_pro
          continue;
     end
          % 随机选择交叉位
          pick=rand;
          while pick==0
               pick=rand;
          end
          flag=0;
        while flag==0
          pos=ceil(pick*length(lenchrom));
%随机选择进行交叉的位置，即选择第几个变量进行交叉，注意：两个染色体交叉的位置相同
          pick=rand; %交叉开始
          v1=input_chrom(index(1),pos);
          v2=input_chrom(index(2),pos);
          input_chrom(index(1),pos)=pick*v2+(1-pick)*v1;
          input_chrom(index(2),pos)=pick*v1+(1-pick)*v2; %交叉结束

        %判断交叉后的两条染色体可不可行
```

```
        limit1=mean(limit);
        f11=isempty(find(input_chrom(index(1),:)>limit1(2)));
        f12=isempty(find(input_chrom(index(1),:)<limit1(1)));
        if f11*f12==0
            flag1=0;
        else
            flag1=1;
        end

        f21=isempty(find(input_chrom(index(2),:)>limit1(2)));
        f22=isempty(find(input_chrom(index(2),:)<limit1(1)));
        if f21*f22==0
            flag2=0;
        else
            flag2=1;
        end

         if    flag1*flag2==0
              flag=0;
         else
              flag=1;
         end     %如果两个染色体不是都可行，则重新交叉
        end
 end
 new_chrom=input_chrom;
end
```

2.3 fitness（适应度函数）

```
function fitness_value=fitness1(input_chrom,input_num,hidden_num,output_num,input_data,output_data)
%该函数用来计算适应度值
%input_chrom        输入种群
%input_num          输入层的节点数，即数据特征数量
%output_num         隐含层节点数,隐藏层神经元的个数
%input_data          训练输入数据
%output_data        训练输出数据
%fitness_value      个体适应度值
w1=input_chrom(1:input_num*hidden_num);     %输入和隐藏层之间的权重参数
B1=input_chrom(input_num*hidden_num+1:input_num*hidden_num+hidden_num); %隐藏层神经元的偏置
w2=input_chrom(input_num*hidden_num+hidden_num+1:input_num*hidden_num+...
    hidden_num+hidden_num*output_num);   %隐藏层和输出层之间的偏置
B2=input_chrom(input_num*hidden_num+hidden_num+hidden_num*output_num+1:input_num*hidden_num+...
    hidden_num+hidden_num*output_num+output_num); %输出层神经元的偏置
% %网络权值赋值
net=newff(input_data,output_data,hidden_num,{'tansig','purelin'});
%网络进化参数
```

```
net.trainParam.epochs=20;
net.trainParam.lr=0.1;
net.trainParam.goal=0.00001;
net.trainParam.show=100;
net.trainParam.showWindow=0;
%网络权值赋值
net.iw{1,1}=reshape(w1,hidden_num,input_num); %信息见 csdn %重组
net.lw{2,1}=reshape(w2,output_num,hidden_num);
net.b{1}=reshape(B1,hidden_num,1);
net.b{2}=reshape(B2,output_num,1);

%网络训练
net=train(net,input_data,output_data);
pre=sim(net,input_data);
error=sum(sum(abs(pre-output_data)));
fitness_value=error; %误差即为适应度
end
```

2.4 mutation（变异函数）

```
function new_chrom=Mutation(mutation_pro,lenchrom,input_chrom,group_num,num,iter_num,limit)
%  本函数完成变异操作
% mutation_pro                变异概率
% lenchrom                      染色体长度
% input_chrom     输入交叉过后的染色体
% group_num                   种群规模
% iter_num                  最大迭代次数
% limit                       每个个体的上限和下限
% num                         当前迭代次数
% new_chrom                 变异后的染色体
for i=1:group_num     %每一轮 for 循环中，可能会进行一次变异操作，染色体是随机选择的，变异位置也是随机选择的，
    %但该轮 for 循环中是否进行变异操作则由变异概率决定（continue 控制）
    %  随机选择一个染色体进行变异
    pick=rand;
    while pick==0
        pick=rand;
    end
    index=ceil(pick*group_num);
    %  变异概率决定该轮循环是否进行变异
    pick=rand;
    if pick>mutation_pro
        continue;
    end
    flag=0;
    while flag==0
```

```
        % 变异位置
        pick=rand;
        while pick==0
            pick=rand;
        end
        pos=ceil(pick*sum(lenchrom));   %随机选择了染色体变异的位置，即选择了第 pos 个变量进行变异
        pick=rand; %变异开始
        fg=(pick*(1-num/iter_num))^2;
        if pick>0.5
            input_chrom(index,pos)=input_chrom(index,pos)+(limit(pos,2)-input_chrom(index,pos))*fg;
        else
            input_chrom(index,pos)=input_chrom(index,pos)-(input_chrom(index,pos)-limit(pos,1))*fg;
        end    %变异结束

        limit1=mean(limit);
        f1=isempty(find(input_chrom(index,:)>limit1(2)));
        f2=isempty(find(input_chrom(index,:)<limit1(1)));
        if f1*f2==0
            flag=0;
        else
            flag=1;
        end
    end
end
new_chrom=input_chrom;
```

3 运用 MATLAB 工具箱的 BP 网络神经

```
clc;clear;close all;
load('shuju_data.mat')
[m,n]=size(data);
train_num=10;     %前 10 个为训练集
%下面是训练集
x_train_data=data(1:train_num,1:n-1);
y_train_data=data(1:train_num,n);
%下面是测试集
x_test_data=data(train_num+1:end,1:n-1);
y_test_data=data(train_num+1:end,n);

x_train_data=x_train_data';
y_train_data=y_train_data';
x_test_data=x_test_data';
[x_train_regular,x_train_maxmin] = mapminmax(x_train_data);        %归一化
[y_train_regular,y_train_maxmin] = mapminmax(y_train_data);
%创建网络
%%调用形式
```

```
EMS_all=[];                 %运行误差记录
TIME=[];                    %运行时间记录
num_iter_all=10;          %随机运行次数
for NN=1:num_iter_all
t1=clock;
net=newff(x_train_regular,y_train_regular,6,{'tansig','purelin'});
%6：神经元个数、激活函数：tansig
[net,~]=train(net,x_train_regular,y_train_regular);   %进行训练
%将输入数据归一化
x_test_regular = mapminmax('apply',x_test_data,x_train_maxmin);
%将 x_test_data 数据结构按照 x_train_maxmin 的参数方法，进行归一化，x_train_maxmin 这个方法在上面归一化 x_train_data 数据时保留下来了
%放入到网络输出数据
y_test_regular=sim(net,x_test_regular);
%网络输出的预测值，注意，此与 y_test_data 归一化不同，y_test_regular 他是通过神经网络训练出来的数据
%将得到的数据反归一化得到预测数据
BP_predict=mapminmax('reverse',y_test_regular,y_train_maxmin);
% RBF_predict(find(RBF_predict<0))=-0.244;
%%
BP_predict=BP_predict';
errors_nn=sum(abs(BP_predict-y_test_data)./(y_test_data))/length(y_test_data);
t2=clock;
Time_all=etime(t2,t1); %计算
EMS_all=[EMS_all,errors_nn];
TIME=[TIME,Time_all];
end
figure(2)
plot(EMS_all,'LineWidth',2)
xlabel('实验次数')
ylabel('误差')
hold on
figure(3)
color=[111,168,86;128,199,252;112,138,248;184,84,246]/255;
plot(y_test_data,'Color',color(2,:),'LineWidth',1)
hold on
plot(BP_predict,'*','Color',color(1,:))
hold on
titlestr=['MATLAB 自带 BP 神经网络','    误差为：',num2str(errors_nn)];
title(titlestr)
```

4 GA-BP 网络神经

```
%% 准备数据
clc;clear;close all;
load('shuju_data.mat')   %数据
```

```
%% 导入数据
%设置训练数据和测试数据
[m,n]=size(data);
train_num=10;                    %自变量
x_train_data=data(1:train_num,1:n-1);
y_train_data=data(1:train_num,n);
%测试数据
x_test_data=data(train_num+1:end,1:n-1);
y_test_data=data(train_num+1:end,n);
x_train_data=x_train_data';
y_train_data=y_train_data';
x_test_data=x_test_data';
%% 标准化
[x_train_regular,x_train_maxmin] = mapminmax(x_train_data);
[y_train_regular,y_train_maxmin] = mapminmax(y_train_data);

%% 初始化参数
input_num=size(x_train_data,1);          %输入特征个数
hidden_num=6;                                %隐藏层神经元个数
output_num=size(y_train_data,1);         %输出特征个数

% 遗传算法参数初始化
iter_num=500;                                %总体进化迭代次数
group_num=40;                                %种群规模
cross_pro=0.4;                               %交叉概率
mutation_pro=0.05;                           %变异概率，相对来说比较小

%这个优化的主要思想就是优化网络参数的初始选择，初始选择对于效果好坏是有较大影响的
num_all=input_num*hidden_num+hidden_num+hidden_num*output_num+output_num;
%网络总参数，只含一层隐藏层
lenchrom=ones(1,num_all);                                %染色体总长度
limit=[-1*ones(num_all,1) 1*ones(num_all,1)];            %初始参数给定范围

EMS_all=[];
TIME=[];
num_iter_all=1;
for NN=1:num_iter_all
   t1=clock;
%% 初始化种群
input_data=x_train_regular;
output_data=y_train_regular;
for i=1:group_num
      initial=rand(1,length(lenchrom));          %产生 0-1 的随机数
      initial_chrom(i,:)=limit(:,1)'+(limit(:,2)-limit(:,1))'.*initial;
```

```
    %变成染色体的形式，一行为一条染色体
    %上述产生的是-1~1 之间的随机数，1x61 的矩阵
    fitness_value=fitness(initial_chrom(i,:),input_num,hidden_num,output_num,input_data,output_data);
    fitness_group(i)=fitness_value;
end
[bestfitness,bestindex]=min(fitness_group);
bestchrom=initial_chrom(bestindex,:);          %最好的染色体
avgfitness=sum(fitness_group)/group_num;       %染色体的平均适应度
trace=[avgfitness bestfitness];                % 记录每一代进化中最好的适应度和平均适应度
%% 迭代过程
new_chrom=initial_chrom;
new_fitness=fitness_group;
 for num=1:iter_num
    % 选择
     [new_chrom,new_fitness]=select(new_chrom,new_fitness,group_num);
     avgfitness=sum(new_fitness)/group_num;
    %交叉
     new_chrom=Cross(cross_pro,lenchrom,new_chrom,group_num,limit);
    % 变异
    ew_chrom=Mutation(mutation_pro,lenchrom,new_chrom,group_num,num,iter_num,limit);
    % 计算适应度
    %将处理过后的 chrom 和适应度转入 sgroup 和 new_fitness 里
    for j=1:group_num
        sgroup=new_chrom(j,:); %新群体
        new_fitness(j)=fitness(sgroup,input_num,hidden_num,output_num,input_data,output_data);
    end
    %找到最小和最大适应度的染色体及它们在种群中的位置
    [newbestfitness,newbestindex]=min(new_fitness);
    [worestfitness,worestindex]=max(new_fitness);
    % 代替上一次进化中最好的染色体
    if   bestfitness>newbestfitness
        bestfitness=newbestfitness;
        bestchrom=new_chrom(newbestindex,:);
    end
    new_chrom(worestindex,:)=bestchrom;
    new_fitness(worestindex)=bestfitness;
    %这两步操作只是使得最烂的数据也变成最好的
    avgfitness=sum(new_fitness)/group_num;
    trace=[trace;avgfitness bestfitness];         %记录每一代进化中最好的适应度和平均适应度
 end
figure(1)
[r ,~]=size(trace);
plot([1:r]',trace(:,2),'b--');
title(['适应度曲线  ''终止代数＝' num2str(iter_num)]);
```

```
xlabel('进化代数');ylabel('适应度');
legend('最佳适应度');
%% 把最优初始阀值权值赋予网络预测
% %用遗传算法优化的 BP 网络进行值预测
%net=newff(x_train_regular,y_train_regular,hidden_num,{'tansig','purelin'},'trainlm');
% input_chrom=bestchrom;
w1=bestchrom(1:input_num*hidden_num);     %输入和隐藏层之间的权重参数
B1=bestchrom(input_num*hidden_num+1:input_num*hidden_num+hidden_num);
%隐藏层神经元的偏置
w2=bestchrom(input_num*hidden_num+hidden_num+1:input_num*hidden_num+...
    hidden_num+hidden_num*output_num);    %隐藏层和输出层之间的偏置
B2=bestchrom(input_num*hidden_num+hidden_num+hidden_num*output_num+1:input_num*hidden_num+...
    hidden_num+hidden_num*output_num+output_num); %输出层神经元的偏置
%网络权值赋值
% net.iw{1,1}=reshape(w1,hidden_num,input_num);
% net.lw{2,1}=reshape(w2,output_num,hidden_num);
% net.b{1}=reshape(B1,hidden_num,1);
% net.b{2}=reshape(B2,output_num,1);
% net.trainParam.epochs=200;                %最大迭代次数
% net.trainParam.lr=0.1;                    %学习率
% net.trainParam.goal=0.00001;
% [net,~]=train(net,x_train_regular,y_train_regular);
w1=reshape(w1,hidden_num,input_num);
w2=reshape(w2,output_num,hidden_num);
B1=reshape(B1,hidden_num,1);
B2=reshape(B2,output_num,1);
%将输入数据归一化
x_test_regular = mapminmax('apply',x_test_data,x_train_maxmin);
[~,n1]=size(x_test_regular);
%放入到网络输出数据
A1=tansig(w1*x_test_regular+repmat(B1,1,n1));     %需与 main 函数中激活函数相同
A2=purelin(w2*A1+repmat(B2,1,n1));                %需与 main 函数中激活函数相同
% y_test_regular=sim(net,x_test_regular);
y_test_regular=A2;
%将得到的数据反归一化得到预测数据
GA_BP_predict=mapminmax('reverse',y_test_regular,y_train_maxmin);
errors_nn=sum(abs(GA_BP_predict'-y_test_data)./(y_test_data))/length(y_test_data);
EcRMSE=sqrt(sum((errors_nn).^2)/length(errors_nn));
t2=clock;
Time_all=etime(t2,t1);
EMS_all=[EMS_all,EcRMSE];
TIME=[TIME,Time_all];
end
figure(2)
```

```
plot(EMS_all,'LineWidth',2)
xlabel('实验次数')
ylabel('误差')
hold on
figure(3)
color=[111,168,86;128,199,252;112,138,248;184,84,246]/255;
plot(y_test_data,'Color',color(2,:),'LineWidth',1)
hold on
plot(GA_BP_predict,'*','Color',color(1,:))
hold on
legend('真实数据','预测数据')
disp('相对容量误差为：')
disp(EcRMSE)
titlestr=['BP 神经网络','    误差为：',num2str(min(EcRMSE))];
title(titlestr)
```