改进的 RBF 神经网络

改进的地方：用 mapminmax 归一化处理方式替代了径向基函数，使得模型和程序更为简便。

1、算法特点：

- RBF 神经网络是一种性能优良的前馈型神经网络，RBF 网络可以任意精度逼近任意的非线性函数或数据，且具有全局逼近能力，从根本上解决了 BP 网络的局部最优问题，而且拓扑结构紧凑，结构参数可实现分离学习，收敛速度快。并且和模糊逻辑能够实现很好的互补，提高神经网络的学习泛化能力。
- 在隐层节点足够多的情况下，经过充分学习，可以用任意精度逼近任意非线性函数，而且具有最优泛函数逼近能力。
- RBF 神经网络具有较强的输入和输出映射功能，并且理论证明在前向网络中 RBF 网络是完成映射功能的最优网络。
- 预测效果好，程序参数易修改。

2、步骤：

①数据径向处理

采用 mapminmax 函数对数据进行归一化处理替代原有的径向基函数的作用，归一化方式为[-1,1]规范化映射。用方程式表示为：

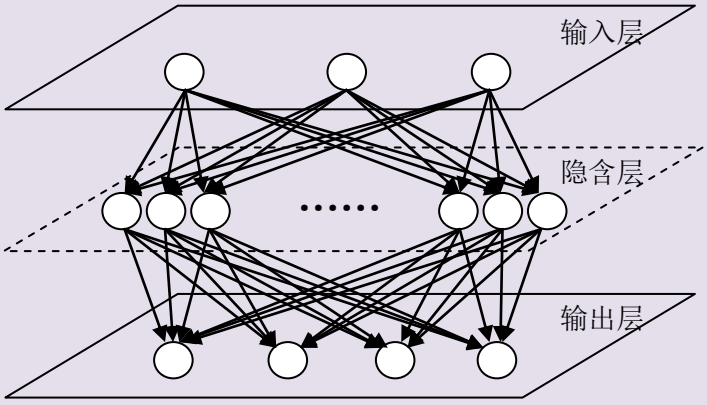
$$y = (ymax-ymin)*(x-xmin)/(xmax-xmin) + ymin$$

采用同种方式对另一组数据按照先前得处理方式进行同样的处理，用公式表示为：

[redacted]

②构建网络

构建 RBF 神经网络结构。网络分为三层：输入层，隐含层、输出层，其中输入层只起到传递信息得作用，建立的径向基神经网络的神经元的个数与输入个数相等。



设置径向基函数目标误差为 1e-6，径向基函数扩展速度值为神经元个数，对数据进行训练。

③检验

进行仿真测试，预测出一组数据并与训练测试数据做对比检验。

④预测

3、程序部分

样本训练数据	样本结果数据（用于检验并预测的数据）
0	0.0001
0.0001	0.0002
0.0002	0.0006
0.0006	0.0009
0.0009	0.001
0.001	0.0006
0.0006	-0.0005
-0.0005	-0.002
-0.002	-0.0035
-0.0035	-0.0044
-0.0044	-0.0043
-0.0043	-0.0031
-0.0031	-0.0012
-0.0012	0.0013
0.0013	0.004
0.004	0.007
0.007	0.01004
0.01004	0.0133
0.0133	0.0166
0.0166	0.0199
0.0199	0.0232
0.0232	0.0265
0.0265	0.0296
0.0296	0.0327
0.0327	0.0356
0.0356	0.0383
0.0383	0.0409
0.0409	0.0434
0.0434	0.0456
0.0456	0.0477
0.0477	0.0496
0.0496	0.0514
0.0514	0.0529
0.0529	0.0543
0.0543	0.0556
0.0556	0.0567
0.0567	0.0576
0.0576	0.0585
0.0585	0.0592
0.0592	0.0597
0.0597	0.0602
0.0602	0.0606
0.0606	0.0609
0.0609	0.0611

RBF 完整程序	运行环境：Matlab2011a/2014a

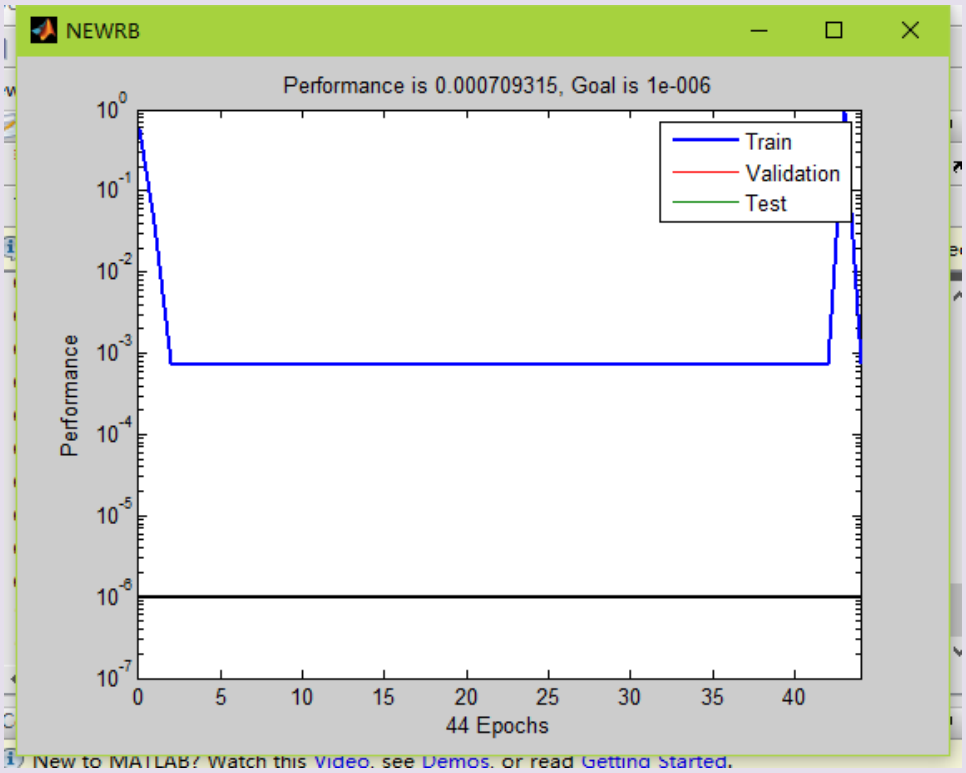
```
clear%清理环境
clc

%%  录入数据，这样录入的方式预测的结果更好些，如果觉得比较繁琐，可以更改录入的方式
AA=[0    0.0001
0.0001    0.0002
0.0002    0.0006
0.0006    0.0009
0.0009    0.001
0.001    0.0006
0.0006    -0.0005
-0.0005    -0.002
-0.002    -0.0035
-0.0035    -0.0044
-0.0044    -0.0043
-0.0043    -0.0031
-0.0031    -0.0012
-0.0012    0.0013
0.0013    0.004
0.004    0.007
0.007    0.01004
0.01004    0.0133
0.0133    0.0166
0.0166    0.0199
0.0199    0.0232
0.0232    0.0265
0.0265    0.0296
0.0296    0.0327
0.0327    0.0356
0.0356    0.0383
0.0383    0.0409
0.0409    0.0434
0.0434    0.0456
0.0456    0.0477
0.0477    0.0496
0.0496    0.0514
0.0514    0.0529
0.0529    0.0543
0.0543    0.0556
0.0556    0.0567
0.0567    0.0576
0.0576    0.0585
```

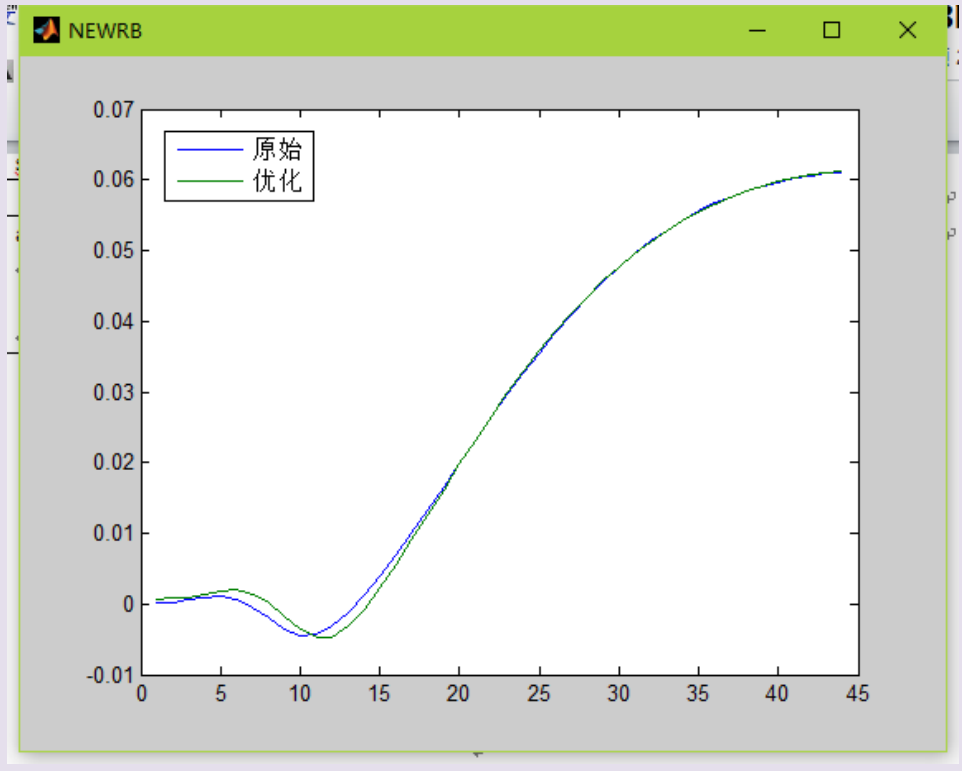
```
0.0585    0.0592
0.0592    0.0597
0.0597    0.0602
0.0602    0.0606
0.0606    0.0609
0.0609    0.0611]';%
[redacted]
[a,b]=size(P_train);%
[redacted]
[redacted]
[redacted]
[Pn_train,inputps] = mapminmax(P_train);%
[Tn_train,outputps] = mapminmax(T_train);
[redacted]
[redacted]
net = [redacted](Pn_train,Tn_train,[redacted]);
%%  仿真测试
Tn_sim_optimized = sim(net,Pn_train);
%  反归一化
T_sim_optimized = mapminmax('reverse',Tn_sim_optimized,outputps);
x0=1:b;%
plot(x0,T_train,x0,T_sim_optimized);%
legend('原始','优化');

%开始预测后面的数据
```

请见完整版

运行结果

```
ans =


    0.061506971165437
```





这样的曲线可能大家不太满意，那么如何让预测的效果更好呢？将训练的数据排列如以下结构：（当然用于预测的数据组也要相应的更改）

训练的样本原始数据					训练的样本结果数据及检验数据
0	0.0001	0.0002	0.0006	0.0009	0.001
0.0001	0.0002	0.0006	0.0009	0.001	0.0006
0.0002	0.0006	0.0009	0.001	0.0006	-0.0005

0.0006	0.0009	0.001	0.0006	-0.0005	-0.002
0.0009	0.001	0.0006	-0.0005	-0.002	-0.0035
0.001	0.0006	-0.0005	-0.002	-0.0035	-0.0044
0.0006	-0.0005	-0.002	-0.0035	-0.0044	-0.0043
-0.0005	-0.002	-0.0035	-0.0044	-0.0043	-0.0031
-0.002	-0.0035	-0.0044	-0.0043	-0.0031	-0.0012
-0.0035	-0.0044	-0.0043	-0.0031	-0.0012	0.0013
-0.0044	-0.0043	-0.0031	-0.0012	0.0013	0.004
-0.0043	-0.0031	-0.0012	0.0013	0.004	0.007
-0.0031	-0.0012	0.0013	0.004	0.007	0.01004
-0.0012	0.0013	0.004	0.007	0.01004	0.0133
0.0013	0.004	0.007	0.01004	0.0133	0.0166
0.004	0.007	0.01004	0.0133	0.0166	0.0199
0.007	0.01004	0.0133	0.0166	0.0199	0.0232
0.01004	0.0133	0.0166	0.0199	0.0232	0.0265
0.0133	0.0166	0.0199	0.0232	0.0265	0.0296
0.0166	0.0199	0.0232	0.0265	0.0296	0.0327
0.0199	0.0232	0.0265	0.0296	0.0327	0.0356
0.0232	0.0265	0.0296	0.0327	0.0356	0.0383
0.0265	0.0296	0.0327	0.0356	0.0383	0.0409
0.0296	0.0327	0.0356	0.0383	0.0409	0.0434
0.0327	0.0356	0.0383	0.0409	0.0434	0.0456
0.0356	0.0383	0.0409	0.0434	0.0456	0.0477
0.0383	0.0409	0.0434	0.0456	0.0477	0.0496
0.0409	0.0434	0.0456	0.0477	0.0496	0.0514
0.0434	0.0456	0.0477	0.0496	0.0514	0.0529
0.0456	0.0477	0.0496	0.0514	0.0529	0.0543
0.0477	0.0496	0.0514	0.0529	0.0543	0.0556
0.0496	0.0514	0.0529	0.0543	0.0556	0.0567
0.0514	0.0529	0.0543	0.0556	0.0567	0.0576
0.0529	0.0543	0.0556	0.0567	0.0576	0.0585
0.0543	0.0556	0.0567	0.0576	0.0585	0.0592
0.0556	0.0567	0.0576	0.0585	0.0592	0.0597
0.0567	0.0576	0.0585	0.0592	0.0597	0.0602
0.0576	0.0585	0.0592	0.0597	0.0602	0.0606
0.0585	0.0592	0.0597	0.0602	0.0606	0.0609
0.0592	0.0597	0.0602	0.0606	0.0609	0.0611

这样这样构建的数据组方式预测效果如何呢，首先我们先来改一下上述数据黄底纹的地方

```
clear%清理环境
clc
%%  录入数据，这样录入的方式预测的结果更好些，如果觉得比较繁琐，可以更改录入的方式
AA=[0   0.0001   0.0002   0.0006   0.0009   0.001
```

```
0.0001  0.0002  0.0006  0.0009  0.001   0.0006
0.0002  0.0006  0.0009  0.001   0.0006  -0.0005
0.0006  0.0009  0.001   0.0006  -0.0005 -0.002
0.0009  0.001   0.0006  -0.0005 -0.002  -0.0035
0.001   0.0006  -0.0005 -0.002  -0.0035 -0.0044
0.0006  -0.0005 -0.002  -0.0035 -0.0044 -0.0043
-0.0005 -0.002  -0.0035 -0.0044 -0.0043 -0.0031
-0.002  -0.0035 -0.0044 -0.0043 -0.0031 -0.0012
-0.0035 -0.0044 -0.0043 -0.0031 -0.0012 0.0013
-0.0044 -0.0043 -0.0031 -0.0012 0.0013  0.004
-0.0043 -0.0031 -0.0012 0.0013  0.004   0.007
-0.0031 -0.0012 0.0013  0.004   0.007   0.01004
-0.0012 0.0013  0.004   0.007   0.01004 0.0133
0.0013  0.004   0.007   0.01004 0.0133  0.0166
0.004   0.007   0.01004 0.0133  0.0166  0.0199
0.007   0.01004 0.0133  0.0166  0.0199  0.0232
0.01004 0.0133  0.0166  0.0199  0.0232  0.0265
0.0133  0.0166  0.0199  0.0232  0.0265  0.0296
0.0166  0.0199  0.0232  0.0265  0.0296  0.0327
0.0199  0.0232  0.0265  0.0296  0.0327  0.0356
0.0232  0.0265  0.0296  0.0327  0.0356  0.0383
0.0265  0.0296  0.0327  0.0356  0.0383  0.0409
0.0296  0.0327  0.0356  0.0383  0.0409  0.0434
0.0327  0.0356  0.0383  0.0409  0.0434  0.0456
0.0356  0.0383  0.0409  0.0434  0.0456  0.0477
0.0383  0.0409  0.0434  0.0456  0.0477  0.0496
0.0409  0.0434  0.0456  0.0477  0.0496  0.0514
0.0434  0.0456  0.0477  0.0496  0.0514  0.0529
0.0456  0.0477  0.0496  0.0514  0.0529  0.0543
0.0477  0.0496  0.0514  0.0529  0.0543  0.0556
0.0496  0.0514  0.0529  0.0543  0.0556  0.0567
0.0514  0.0529  0.0543  0.0556  0.0567  0.0576
0.0529  0.0543  0.0556  0.0567  0.0576  0.0585
0.0543  0.0556  0.0567  0.0576  0.0585  0.0592
0.0556  0.0567  0.0576  0.0585  0.0592  0.0597
0.0567  0.0576  0.0585  0.0592  0.0597  0.0602
0.0576  0.0585  0.0592  0.0597  0.0602  0.0606
0.0585  0.0592  0.0597  0.0602  0.0606  0.0609
0.0592  0.0597  0.0602  0.0606  0.0609  0.0611]';%
[redacted]
[a,b]=size(P_train);%
[redacted]
[Pn_train,inputps] = mapminmax(P_train);%
[Tn_train,outputps] = mapminmax(T_train);
```

[redacted]

net = [redacted](Pn_train,Tn_train,[redacted]);%

%% 仿真测试

Tn_sim_optimized = sim(net,Pn_train);

% 反归一化

T_sim_optimized = mapminmax('reverse',Tn_sim_optimized,outputps);

x0=1:b;%

plot(x0,T_train,x0,T_sim_optimized);%

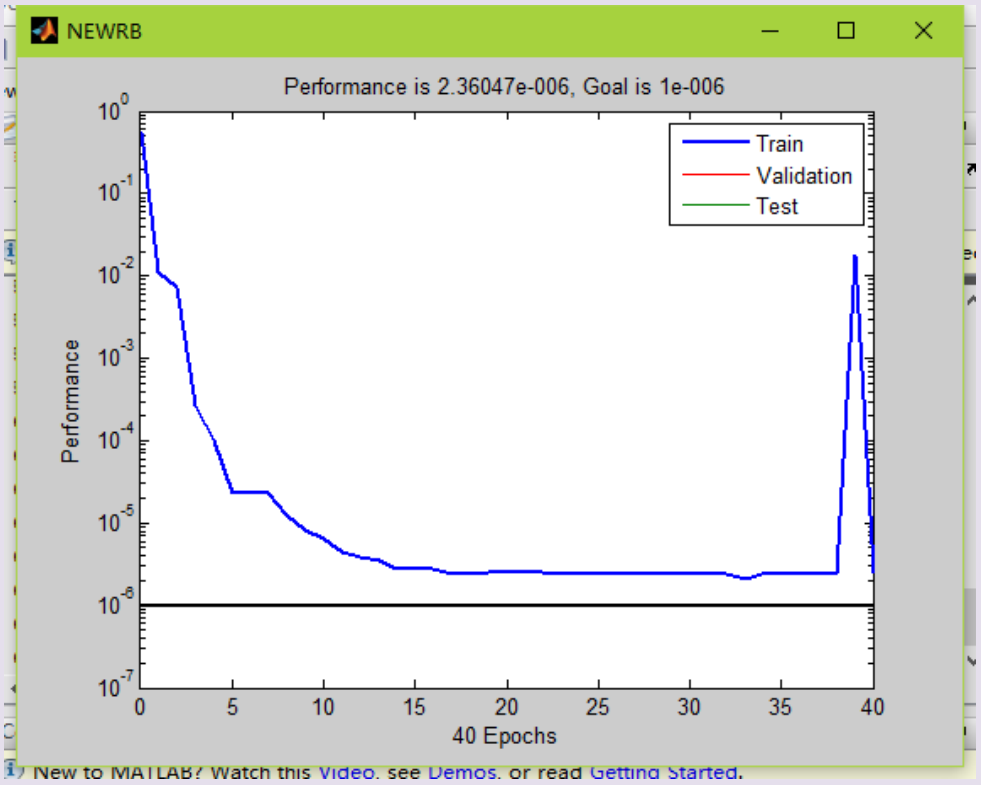
legend('原始','优化');

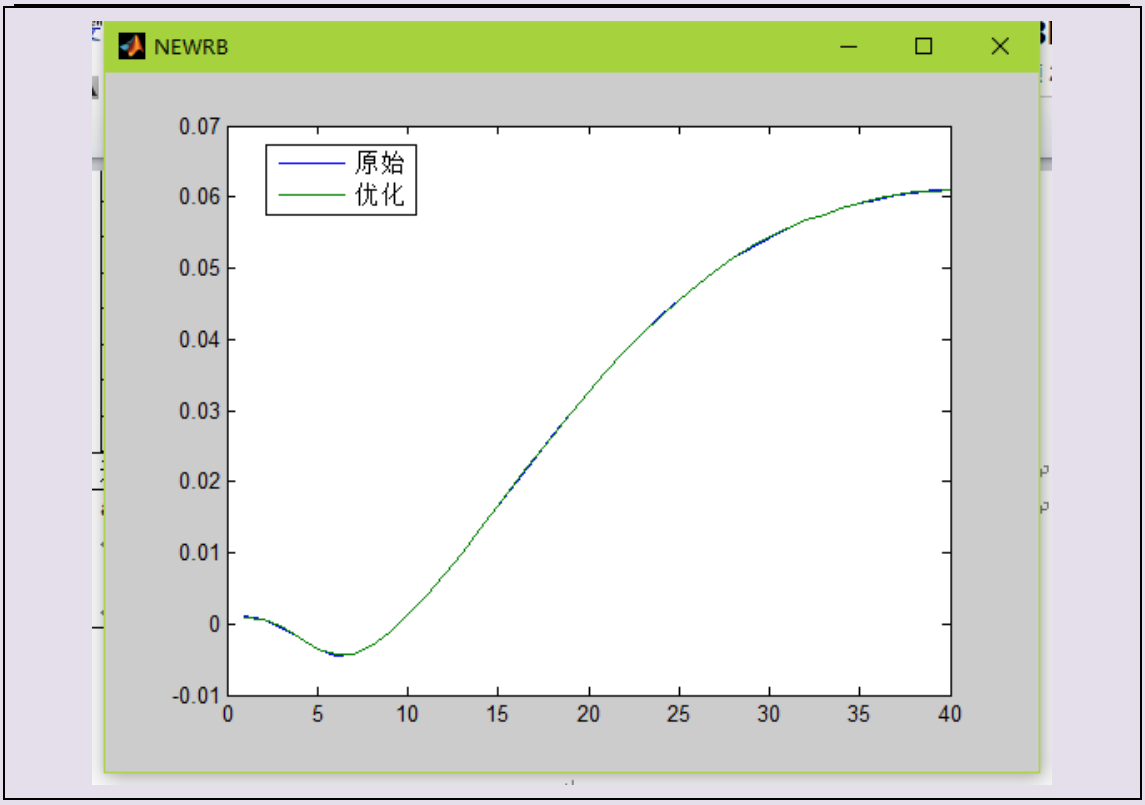
%开始预测后面的数据

请见完整版

ans =

0.061335402570816





其实从两种程序的误差训练图就可以看出哪一种更好了，但是这是为什么呢？我们首先来看下第一种程序中，每个数据矩阵的归一化结果吧：

Pn_train <1x44 double>

	1	2	3	4	5	6	7	8	9	10
1	-0.8652	-0.8622	-0.8591	-0.8469	-0.8377	-0.8346	-0.8469	-0.8806	-0.9265	-0.9724

Tn_train <1x44 double>

	1	2	3	4	5	6	7	8	9	10
1	-0.8626	-0.8595	-0.8473	-0.8382	-0.8351	-0.8473	-0.8809	-0.9267	-0.9725	-1

第二种：

Pn_train <5x40 double>

	1	2	3	4	5	6	7	8	9	10
1	-0.8616	-0.8585	-0.8553	-0.8428	-0.8333	-0.8302	-0.8428	-0.8774	-0.9245	-0.9717
2	-0.8596	-0.8565	-0.8440	-0.8346	-0.8315	-0.8440	-0.8783	-0.9251	-0.9719	-1
3	-0.8576	-0.8452	-0.8359	-0.8328	-0.8452	-0.8793	-0.9257	-0.9721	-1	-0.9969
4	-0.8462	-0.8369	-0.8338	-0.8462	-0.8800	-0.9262	-0.9723	-1	-0.9969	-0.9600
5	-0.8377	-0.8346	-0.8469	-0.8806	-0.9265	-0.9724	-1	-0.9969	-0.9602	-0.9020

Tn_train <1x40 double>

	1	2	3	4	5	6	7	8	9	10
1	-0.8351	-0.8473	-0.8809	-0.9267	-0.9725	-1	-0.9969	-0.9603	-0.9023	-0.8260

第二种的 Pn_train 矩阵中，斜对着的数据之差很小，误差可忽略。小编认为哈，神经网络训练的数据越多，那么能提取到的数据特征就越多，尤其是这样的数据组构建方式，最后能预测出的效果就越好。

以上所述的程序是单个预测的数据，如果题目要求预测长期的数据我想各位小伙伴一定觉得很麻烦，下面小编为大家改写了可预测长期结果的程序。并且大家只需要更改蓝底纹的部分。

特别注意：

1、该算法预测的结果主要是通过规范映射实现的，也就是【-1，1】，归一化的值预测，然后反归一化复原数值，这样解决了一般神经网络每次运行的结果都不相同的问题，但也存在一定的缺陷，我想，大家在运行自己的神经网络程序时会发现当预测的数据量很多时，可能运行出来的结果会出现值相同、偏差很大的现象，小编认为网络构建只针对目前训练的数据而已，并且提取的数据变化特征也针对于训练的数据，所以建议大家分成小段预测，然后将预测的数据带入训练数据中重新构建网络，本文程序也一样，当然小编也尝试了加循环重新构建网络，但系统会报错，因此只能将循环放在预测部分，方便大家的操作，经个人研究发现该程序预测后续数据个数最好不超过训练数据个数的 1/4，然后重新构建数据矩阵和网络再预测这样预测出的结果更好一点。

2、此外误差设置也是一个影响预测结果的因素，并不是误差设置的越小越好，有些时候一些类型的数据会出现反常情况，因此大家在调试本文程序和其他神经网络时需注意下。

3、该程序不适合齿状波动的数据类型，针对该类型的数据预测，大家可查看小编整理的小波神经网络。

```
clear%清理环境
clc

%%  录入数据，这样录入的方式预测的结果更好些，如果觉得比较繁琐，可以更改录入的方式
AA=[0 0.0001 0.0002 0.0006 0.0009 0.001
0.0001 0.0002 0.0006 0.0009 0.001 0.0006
0.0002 0.0006 0.0009 0.001 0.0006 -0.0005
0.0006 0.0009 0.001 0.0006 -0.0005 -0.002
0.0009 0.001 0.0006 -0.0005 -0.002 -0.0035
0.001 0.0006 -0.0005 -0.002 -0.0035 -0.0044
0.0006 -0.0005 -0.002 -0.0035 -0.0044 -0.0043
-0.0005 -0.002 -0.0035 -0.0044 -0.0043 -0.0031
-0.002 -0.0035 -0.0044 -0.0043 -0.0031 -0.0012
-0.0035 -0.0044 -0.0043 -0.0031 -0.0012 0.0013
-0.0044 -0.0043 -0.0031 -0.0012 0.0013 0.004
-0.0043 -0.0031 -0.0012 0.0013 0.004 0.007
-0.0031 -0.0012 0.0013 0.004 0.007 0.01004
-0.0012 0.0013 0.004 0.007 0.01004 0.0133
0.0013 0.004 0.007 0.01004 0.0133 0.0166
0.004 0.007 0.01004 0.0133 0.0166 0.0199
0.007 0.01004 0.0133 0.0166 0.0199 0.0232
0.01004 0.0133 0.0166 0.0199 0.0232 0.0265
```

```
0.0133  0.0166  0.0199  0.0232  0.0265  0.0296
0.0166  0.0199  0.0232  0.0265  0.0296  0.0327
0.0199  0.0232  0.0265  0.0296  0.0327  0.0356
0.0232  0.0265  0.0296  0.0327  0.0356  0.0383
0.0265  0.0296  0.0327  0.0356  0.0383  0.0409
0.0296  0.0327  0.0356  0.0383  0.0409  0.0434
0.0327  0.0356  0.0383  0.0409  0.0434  0.0456
0.0356  0.0383  0.0409  0.0434  0.0456  0.0477
0.0383  0.0409  0.0434  0.0456  0.0477  0.0496
0.0409  0.0434  0.0456  0.0477  0.0496  0.0514
0.0434  0.0456  0.0477  0.0496  0.0514  0.0529
0.0456  0.0477  0.0496  0.0514  0.0529  0.0543
0.0477  0.0496  0.0514  0.0529  0.0543  0.0556
0.0496  0.0514  0.0529  0.0543  0.0556  0.0567
0.0514  0.0529  0.0543  0.0556  0.0567  0.0576
0.0529  0.0543  0.0556  0.0567  0.0576  0.0585
0.0543  0.0556  0.0567  0.0576  0.0585  0.0592
0.0556  0.0567  0.0576  0.0585  0.0592  0.0597
0.0567  0.0576  0.0585  0.0592  0.0597  0.0602
0.0576  0.0585  0.0592  0.0597  0.0602  0.0606
0.0585  0.0592  0.0597  0.0602  0.0606  0.0609
0.0592  0.0597  0.0602  0.0606  0.0609  0.0611]';%
n=5%要预测的数据个数
[redacted]
[redacted]
[a,b]=size(P_train);%
[redacted]
[Pn_train,inputps] = mapminmax(P_train);%
[Tn_train,outputps] = mapminmax(T_train);
[redacted]
[redacted]
net = [redacted](Pn_train,Tn_train,[redacted]);%
%% 仿真测试
Tn_sim_optimized = sim(net,Pn_train);
%  反归一化
T_sim_optimized = mapminmax('reverse',Tn_sim_optimized,outputps);
x0=1:b;%
plot(x0,T_train,x0,T_sim_optimized);%
legend('原始','优化');
for i=1:n

%开始预测后面的数据
请见完整版
```

```
j1 =

   0.061335402570816   0.061446341596694   0.061499780975433   0.061579752646537   0.061649636112304
```

附录

==========有关 mapminmax 的用法详解==========

几个要说明的函数接口:

```
[Y,PS] = mapminmax(X)
[Y,PS] = mapminmax(X,FP)
Y = mapminmax('apply',X,PS)
X = mapminmax('reverse',Y,PS)
```

用实例来讲解,测试数据 x1 = [1 2 4], x2 = [5 2 3];

```
>> [y,ps] = mapminmax(x1)
y =
    -1.0000    -0.3333    1.0000
ps =
name: 'mapminmax'
    xrows: 1
    xmax: 4
    xmin: 1
    xrange: 3
    yrows: 1
    ymax: 1
    ymin: -1
    yrange: 2
```

其中 y 是对进行某种规范化后得到的数据,这种规范化的映射记录在结构体 ps 中.让我们来看一下这个规范化的映射到底是怎样的?

```
y = (ymax-ymin)*(x-xmin)/(xmax-xmin) + ymin;
```

[关于此算法的一个问题.算法的假设是每一行的元素都不想相同,那如果都相同怎么办?实现的办法是,如果有一行的元素都相同比如 xt = [1 1 1],此时 xmax = xmin = 1,把此时的变换变为 y = ymin,matlab 内部就是这么解决的.否则该除以 0 了,没有意义!]

也就是说对 x1 = [1 2 4]采用这个映射 f: 2*(x-xmin)/(xmax-xmin)+(-1),

就可以得到 y = [-1.0000 -0.3333 1.0000]
我们来看一下是不是: 对于 x1 而言 xmin = 1,xmax = 4;
则:
y(1) = 2*(1 - 1)/(4-1)+(-1) = -1;
y(2) = 2*(2 - 1)/(4-1)+(-1) = -1/3 = -0.3333;
y(3) = 2*(4-1)/(4-1)+(-1) = 1;
看来的确就是这个映射来实现的.
对于上面 algorithm 中的映射函数 其中 ymin,和 ymax 是参数,可以自己设定,默认为-1,1;
比如:

```
>>[y,ps] = mapminmax(x1);
>> ps.ymin = 0;
>> [y,ps] = mapminmax(x1,ps)
y =
         0    0.3333    1.0000
```

```
ps =
name: 'mapminmax'
    xrows: 1
    xmax: 4
    xmin: 1
    xrange: 3
    yrows: 1
    ymax: 1
    ymin: 0
    yrange: 1
```

则此时的映射函数为: f: 1*(x-xmin)/(xmax-xmin)+(0)。
如果我对 x1 = [1 2 4]采用了某种规范化的方式, 现在我要对 x2 = [5 2 3]采用同样的规范化方式[同样的映射],如下可办到:

```
>> [y1,ps] = mapminmax(x1);
>> y2 = mapminmax('apply',x2,ps)
y2 =
    1.6667   -0.3333    0.3333
```

即对 x1 采用的规范化映射为: f: 2*(x-1)/(4-1)+(-1),(记录在 ps 中),对 x2 也要采取这个映射.
x2 = [5,2,3],用这个映射我们来算一下.
y2(1) = 2(5-1)/(4-1)+(-1) = 5/3 = 1+2/3 = 1.66667
y2(2) = 2(2-1)/(4-1)+(-1) = -1/3 = -0.3333
y2(3) = 2(3-1)/(4-1)+(-1) = 1/3 = 0.3333

X = mapminmax('reverse',Y,PS)的作用就是进行反归一化,讲归一化的数据反归一化再得到原来的数据:

```
>> [y1,ps] = mapminmax(x1);
>> xtt = mapminmax('reverse',y1,ps)
xtt =
     1     2     4
```

此时又得到了原来的 x1(xtt = x1);