小波神经网络

BP 神经网络主要存在易陷入局部最小值、收敛速度过慢（网络不收敛）、网络结构无法确定、对于训练样本要求过高（数据波动不能太大）等问题。

小波神经网络是以 BP 神经网络为模型基础，把小波基函数作为神经网隐藏层的激活函数的一种新型神经网络，它与 BP 神经网络最大的不同就是他们的隐藏层的激活函数不一样。因此小波神经网络是一种有反馈的前馈式神经网络，小波变换可以使网络具有很快的学习速度同时也避免了数据陷入局部极小的缺陷，使其具有了很广泛的应用领域。

例如一束普通白光具有不同频率的波形，想要采集这束白光的信息，需要对不同频率的波形进行分析。小波分析是时间与空间的局部分析，它可以通过伸缩平移对信号进行多段细分，分别细分为高频处和第频处，小波分析可以通过平移和伸缩变换逼近任意函数使其具有了良好的精度和容错性，而且在维空间有选择信号方向的能力，从而使结果更加精确且具有较高的容错性。。

目前两者能通过两种方式相互结合：第一“松散型”结合方式，先对数据信号做小波分析处理，再把处理完的信号输入神经网络中；第二“紧致型”结合方式，用小波基函数代替神经网络隐含层众多激活函数，这种模型同时具有了神经网络和小波变换的优点。

一般的“紧致型”小波神经网络处理数据的能力更强。

1.初始化相关参数

令 $q(1,2,\ldots,n)$ 为输入信号样本的个数，第 p 个样本的输入值为 x_i^p，y_k^p 为第 p 个样本的网络输出值，z_k^p 为样本的输出目标值。输入层与隐含层、隐含层与输出层之间的连接权值为 w_{ij}、w_{kj}。设伸缩因子为 a_i，平移因子为 b_i。且按照下式进行平移与伸缩。

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a_i}}\psi(\frac{t-b_i}{a_i})$$

2.建立小波基函数

在 Hilbert 向量空间中选取一个母小波函数（基本小波函数）$\varphi(t)$，使其满足 $\int_R \frac{|\overline{\varphi}(w)^2|}{w}dw < +\infty$（$\overline{\varphi}(w)$ 为 $\varphi(t)$ 的 Fourier 变换）。通过 $\varphi(t)$ 的伸缩和平移变换产生小波函数基：

$$\varphi_{a,b}(x) = \frac{1}{\sqrt{a}}\varphi(\frac{x-b}{a})$$

其中 a、b 分别为伸缩尺度因子和平移因子。

目前小编只验证了一个小波基函数如下，数据的预测主要靠小波基函数，**本文的程序处理波动数据较一般的神经网络好一点，**对于其他类型的数据就自己去找相应的小波基函数，小波基函数也可通过计算可得，网上也有相应的算法。

```
y = exp(-(t.^2)/2) * cos(1.75*t);
```

从而建立小波神经网络激活函数（小波基函数是小波函数的一部分）：

$$y_k = \sum_{j=1}^{J} w_{kj}^{*} \varphi \left(\frac{\sum_{i=1}^{I} w_{ji} x_i - b_j}{a_j} \right)$$

其中，$x_i (i = 1,2 \cdots I)$ 为输入层第 i 个神经元输入信号；

$y_k (k = 1,2 \cdots k)$ 为输出层第 k 个神经元的输出信号；

w_{ji} 为隐含层神经元 j 和输出层神经元 i 之间的权值；

w_{kj}^{*} 为输入层神经元 j 和隐含层结点 k 之间的权值；

a_j 为第 j 个隐含层神经元的伸缩尺度因子；

b_j 为第 j 个隐含层神经元的平移因子；

3.预测网络输出、计算误差

向已经初始化后的网络中输入一个训练样本 $(P_k, T_k), k \in \{1, 2, \cdots, N\}$，其中 N 为训练样本个数，P_k 为样本输入信号，T_k 为网络输出目标值，$P_k \in R^n, T_k \in R^m$。

根据式 4-1 计算网络输出；根据 $E_k^p = \frac{1}{2} \sum_{k=1}^{K} (t_k^p - y_k^p)^2$ 计算预测误差

4.网络权值修正

根据 E_k^p 的情况利用最速下降法究竟小波神经网络的参数 w_{ji}、w_{kj}^{*}、a_j 和 b_j 的变化情况：

令 $x = \dfrac{\sum_{i=1}^{I} w_{ji} x_i^p - b_j}{a_j}$

$$\nabla w_{kj}^{*}=\frac{\partial E_k^p}{\partial w_{kj}^{*}}=-(t_k^p-y_k^p)\psi(x)$$

$$\nabla w_{ji}=\frac{\partial E_k^p}{\partial w_{ji}}=-\sum_{k=1}^{K}(t_k^p-y_k^p)w_{ji}\frac{\partial\psi(x)}{\partial x}\frac{x_i^p}{a_j}$$

$$\nabla b_j=\frac{\partial E_k^p}{\partial b_j}=\sum_{k=1}^{K}(t_k^p-y_k^p)w_{kj}\frac{\partial\psi(x)}{\partial x}\frac{1}{a_j}$$

$$\nabla a_j=\frac{\partial E_k^p}{\partial a_j}=\sum_{k=1}^{K}(t_k^p-y_k^p)w_{kj}\frac{\partial\psi(x)}{\partial x}\frac{x_i^p}{a_j^2}$$

5.优化收敛速度

为了加快网络的收敛速度，引入一个网络参数动量因子 α，因此权向量的迭代公式为：

$$w_{kj}^{*}(t+1)=w_{kj}^{*}(t)-\eta\nabla w_{kj}^{*}+\alpha\Delta w_{kj}^{*}$$
$$w_{ji}(t+1)=w_{ji}(t)-\eta\nabla w_{ji}+\alpha\Delta w_{ji}$$
$$b_j(t+1)=b_j(t)-\eta\nabla b_j+\alpha\Delta b_j$$
$$a_j(t+1)=a_j(t)-\eta\nabla a_j+\alpha\Delta a_j$$

6.判断训练是否结束

完成上述所有过程，查看系统的误差 $E_k^p=\frac{1}{2}\sum_{k=1}^{K}(t_k^p-y_k^p)^2$，看系统误差是否小于所要求的误差最大值，若已经小于规定的误差最大值，则网络训练结束；当然如果网络训练的次数已经达到规定的训练的最大次数，则网络训练结束；否则将网络训练次数更新为 $t=t+1$；重复上述过程。

该程序适于对波动程序进行预测，数据录入较繁琐，可以用 Excel 来调整数据然后复制进程序中然后再次预测下一个

程序：针对数据	运行环境：Matlab2011a

```
%% 清空环境变量
clc
clear
%% 网络参数配置
load traffic_flux input output input_test output_test
M=size(input,2); %输入节点个数
N=size(output,2); %输出节点个数
n=6; %隐形节点个数
lr1=0.01; %学习概率
lr2=0.001; %学习概率
maxgen=3000; %迭代次数
%权值初始化
```

```
Wjk=randn(n,M);Wjk_1=Wjk;Wjk_2=Wjk_1;
Wij=randn(N,n);Wij_1=Wij;Wij_2=Wij_1;
a=randn(1,n);a_1=a;a_2=a_1;
b=randn(1,n);b_1=b;b_2=b_1;
%节点初始化
y=zeros(1,N);
net=zeros(1,n);
net_ab=zeros(1,n);
%权值学习增量初始化
d_Wjk=zeros(n,M);
d_Wij=zeros(N,n);
d_a=zeros(1,n);
d_b=zeros(1,n);
%% 输入输出数据归一化
[inputn,inputps]=mapminmax(input');
[outputn,outputps]=mapminmax(output');
inputn=inputn';
outputn=outputn';
%% 网络训练
for i=1:maxgen
    %误差累计
    error(i)=0;
    % 循环训练
    for kk=1:size(input,1)
        x=inputn(kk,:);
        yqw=outputn(kk,:);
        for j=1:n
            for k=1:M
                net(j)=net(j)+Wjk(j,k)*x(k);
                net_ab(j)=(net(j)-b(j))/a(j);
            end
            temp=mymorlet(net_ab(j));
            for k=1:N
                y=y+Wij(k,j)*temp;      %小波函数
            end
        end
        %计算误差和
        error(i)=error(i)+sum(abs(yqw-y));
        %权值调整
        for j=1:n
            %计算 d_Wij
            temp=mymorlet(net_ab(j));
            for k=1:N
                d_Wij(k,j)=d_Wij(k,j)-(yqw(k)-y(k))*temp;
            end
            %计算 d_Wjk
            temp=d_mymorlet(net_ab(j));
            for k=1:M
```

```
                    for l=1:N
                        d_Wjk(j,k)=d_Wjk(j,k)+(yqw(l)-y(l))*Wij(l,j) ;
                    end
                    d_Wjk(j,k)=-d_Wjk(j,k)*temp*x(k)/a(j);
                end
                %计算 d_b
                for k=1:N
                    d_b(j)=d_b(j)+(yqw(k)-y(k))*Wij(k,j);
                end
                d_b(j)=d_b(j)*temp/a(j);
                %计算 d_a
                for k=1:N
                    d_a(j)=d_a(j)+(yqw(k)-y(k))*Wij(k,j);
                end
                d_a(j)=d_a(j)*temp*((net(j)-b(j))/b(j))/a(j);
            end
            %权值参数更新
            Wij=Wij-lr1*d_Wij;
            Wjk=Wjk-lr1*d_Wjk;
            b=b-lr2*d_b;
            a=a-lr2*d_a;
            d_Wjk=zeros(n,M);
            d_Wij=zeros(N,n);
            d_a=zeros(1,n);
            d_b=zeros(1,n);
            y=zeros(1,N);
            net=zeros(1,n);
            net_ab=zeros(1,n);
            Wjk_1=Wjk;Wjk_2=Wjk_1;
            Wij_1=Wij;Wij_2=Wij_1;
            a_1=a;a_2=a_1;
            b_1=b;b_2=b_1;
        end
    end
    %% 网络预测
    %预测输入归一化
    x=mapminmax('apply',input_test',inputps);
    x=x';
    %网络预测
    for i=1:11
        x_test=x(i,:);
        for j=1:1:n
            for k=1:1:M
                net(j)=net(j)+Wjk(j,k)*x_test(k);
                net_ab(j)=(net(j)-b(j))/a(j);
            end
            temp=mymorlet(net_ab(j));
            for k=1:N
                y(k)=y(k)+Wij(k,j)*temp ;
```

```
            end
        end
        yuce(i)=y(k);
        y=zeros(1,N);
        net=zeros(1,n);
        net_ab=zeros(1,n);
end
%预测输出反归一化
ynn=mapminmax('reverse',yuce,outputps);
%% 结果分析
figure(1)
plot(ynn,'r*:')
hold on
plot(output_test,'bo--')
title('XX','fontsize',12)
legend('预测','实际')
xlabel('时间点')
ylabel('值')
l=1:1:maxgen;
figure(2)
plot(l,error(l))%训练误差变化图
```

接下来说一下数据的录入，假如有 14 个数据，预测前需要检验下，下面是第几个数据的标号。

数据标号为 1 到 14

input				output
1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
4	5	6	7	8
5	6	7	8	9

input_test				output_test(时间序列预测段数据)
6	7	8	9	10
7	8	9	10	11
8	9	10	11	12
9	10	11	12	13
10	11	12	13	14

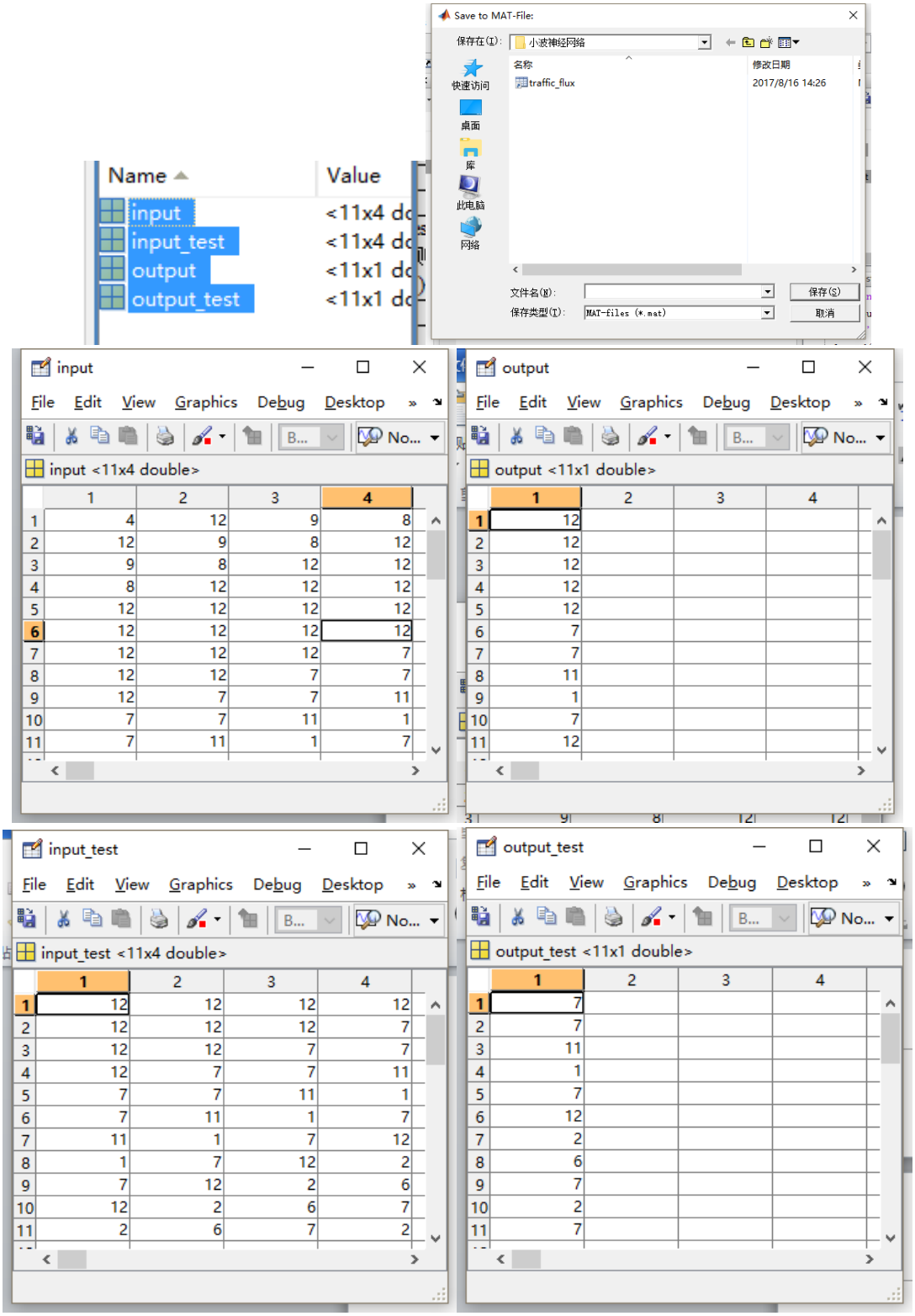
最后运行出的 ynn 预测数据和 output_test 的对比曲线。如果相近且达到上图修正效果，那么采用 ynn 文件数据并继续按标号更替 input_test 和 output_test 中的数据，循环下去，直到运行出较满意的数据和曲线。如果不相近则多次运行，取效果最好的数据和曲线。

举个例子

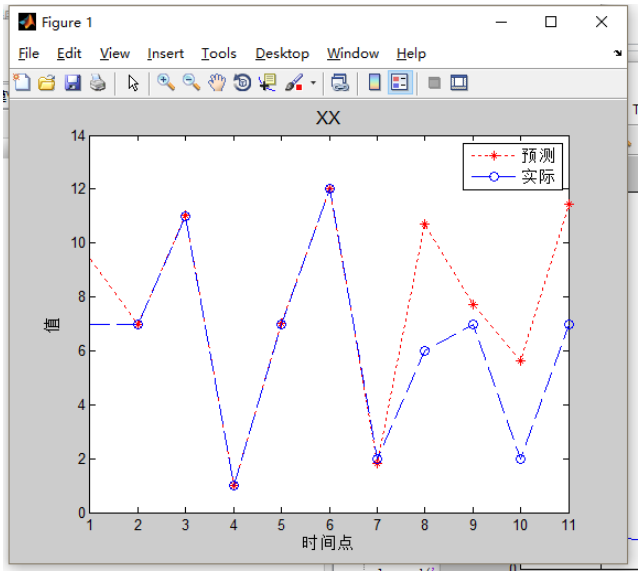
首先录入数据（20 个数据）

input=[数据矩阵]；output=[数据矩阵]；input_test=[数据矩阵]；output_test=[数据矩阵]

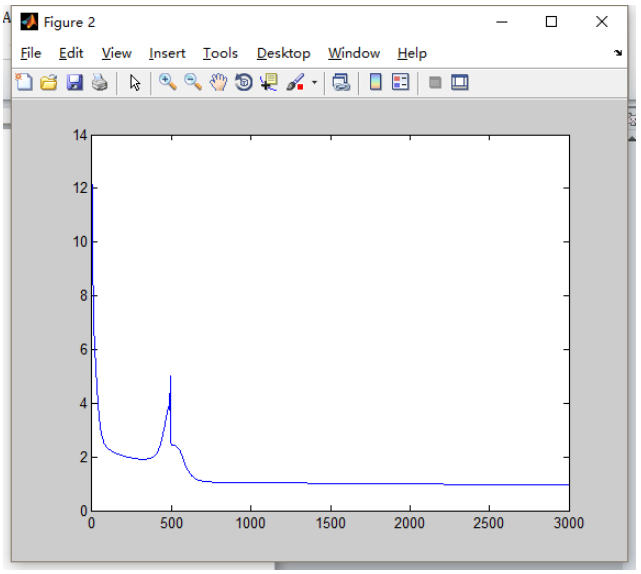
然后选中，点击右键，save as 保存为 traffic_flux，也可以自己命名。



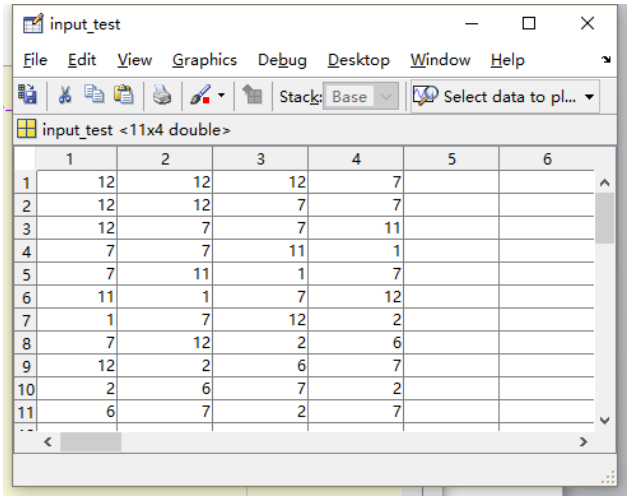
运行出来的 ynn 及对比图如下（每次运行都不一样）：



训练误差



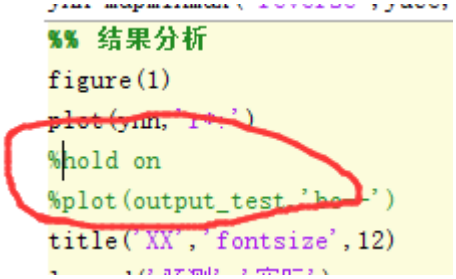
如果要预测下一年，则将 output_test 中的数据转移到 input_test 第 5 列。然后将第 1 列删掉。



	1	2	3	4	5	6
1	12	12	12	7		
2	12	12	7	7		
3	12	7	7	11		
4	7	7	11	1		
5	7	11	1	7		
6	11	1	7	12		
7	1	7	12	2		
8	7	12	2	6		
9	12	2	6	7		
10	2	6	7	2		
11	6	7	2	7		

然后将这部分锁掉，之后逐年预测就改 input，output，input_test 就行了。改动后在 matlab

界面输 save traffic_flux



预测的结果为 ynn 矩阵的最后一个，矩阵其他的不用管，如果预测有问题的话重新运行数据，BP 神经网络情况和小波神经网络类似，总体来看，运行几次后平均每次训练误差小一点，运行速度快一点。还有就是小波训练可以达到规定的训练次数，BP 可能在没有达到训练次数前就输出了，这样预测出来的结果不怎么好。