人工神经网络及其 matlab 实现

人工神经网络基本理论

1、 人工神经网络模型拓扑结构

人工神经网络是由大量简单的基本元件（神经元）相互连接，通过模拟人的大脑处理问题的方式，进行信息并行处理和非线性转换的复杂网络系统。

优点：多输入、多输出实现了数据的并行处理以及自学能力。

神经网络的拓扑结构包括网络层数、各层神经元数量以及各神经元之间的相互连接方式，三者都根据实际情况再具体确定取值。

2、 常用激励函数

a) 阈值型（一般只存在于 MP 简单分类的情形中）

$$\mathrm{f(x)} = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

b) 线性型（一般只用在输入神经元和输出神经元）

$$\mathrm{f(x)} = \mathrm{x}$$

c) S 型（常用于隐含神经元）

$$\mathrm{f(x)} = \frac{1}{1+e^{-x}} \text{ 或} f(x) = \frac{1-e^{-x}}{1+e^{-x}}$$

3、 常见神经网络理论

a) BP 网络基本数学原理

b) RBF 网络基本数学原理

BP 神经网络的结构设计

1、 鲨鱼闻血腥味道与 BP 神经网络训练

BP 神经网络的训练过程是依靠“负梯度下降“的训练算法，即误差调整的方向总是沿着下降速度最快的方向进行。

2、 透视神经网络的学习步骤

假设由三层 BP 网络，第一层为输入层，第二层为中间层（隐含层），第三层为输出层，每一层由神经元构成，每层神经元之间由权值相互连接。其中，**输入层神经元数量由输入样本的维数决定**，**输出层神经元数量由输出样本的维数决定**，隐含层神经元合理选择。

a) 准备训练网络的样本

由冶炼技术可知：冶炼的“初始温度“会受到出钢时间、钢水净重量、吹止温度、高碳锰铁、低碳锰铁、硅锰铁、硅铁、铝块、增碳剂、中碳锰铁、包龄、运输时间、等待时间这十三个因素影响。这 13 个因子便是输入样本，”开始温度“便是网络的输出样本。（由于开始温度的拿捏在冶炼过程中十分重要）。

b) 确定网络的初始参数

需要说明的是，下表所列的参数都可以根据实际训练情况调整。（另外，数据采用

批量输入，这是因为 matlab 对矩阵的操作是非常快且方便的，而对循环的操作较慢）

例：

参数	数值
最大训练次数	5000
隐含层（中间层）神经元数量	12
网络学习速率	0.035
训练的目标误差	0.65×10E-3
是否添加动量因子	否

c) 初始化网络权值与阈值

在此做以下约定：

①训练步骤是按照数据批量形式进行的；

②阈值写成与权值相似的形式，也就是说，阈值看作样本输入为 1 的随机数；

③数据输入是按照每一行输入，不是每一列，因此要把原始数据进行转置。

初始化网络：给网络的权值和阈值赋予随机数矩阵。因为是十三个输入因子，十二个隐含层（中间层）神经元，则第一层与第二层之间的权值 wij(t)为 12×13 的随机数矩阵。（传递过来的信息不是直接全部交给中间层，而是进行加权后传递给中间层）。我们可以把权值矩阵的大小看为：这一层神经元数量×上一层神经元数量。

神经网络的一个阈值指用来激活神经元而设置的一个临界值。（有多少个神经元就会有多少个阈值），则第二层神经元的阈值为 12×1 的矩阵。

同理，第二层与第三层神经元的权值为 1×12 的一个矩阵，第三层神经元的阈值为一个 1×1 的矩阵。

d) 计算第一层神经元的输入与输出

为了简化理解：假设 X 为输入样本，其数据规模为 13×30 的矩阵。假设第一层神经元中放置的为线性函数（也可以是别的激励函数），所以网络第一层输入和输出都等于实际样本的值，O1=X，为一个 13×30 的矩阵。

e) 计算第二层神经元的输入

对于第二层，神经元的输入 I2 一定来自第一层所有神经元的值与阈值的和。即：I2=wij×x+Bij×ones（都为 1 的矩阵）。这是因为我们把阈值看为输入为 1 的样本。（样本要进行归一化处理）

中间层神经元的输入值等于所有与它相连的输入层神经元的输入值*权值+自身的阈值。有多少组样本就会有多少组输入。所以为 12×30 的一个矩阵

f) 计算第二层神经元的输出

假设隐含层（中间层）神经元激励函数为单机 S 型函数，即为$\mathrm{f(x)}=\frac{1}{1+e^{-x}}$

所以，第二层神经元的输出为$\mathrm{O_2}=\frac{1}{1+e^{-I_2}}$。由此可知，第二层输出 O2 是一个 12×30 的数据矩阵。

g) 计算第三层的输入和输出

第三层和第二层的输入道理类似，I3=wjk×O2+Bjk×ones，是一个 1×30 的矩阵。

第三层的输出：通常也将第三层神经元设置为线性函数。即 O3=I3。

h) 计算能量函数 E

计算能量函数的目的是达到一定误差就可以停止训练网络。假设实际输出样本为 Y，则易得

$$\mathrm{E}=\sum(Y-O_3)^2$$

i) 计算第二层和第三层之间权值和阈值调整量（要注意，这里要反复看）

这里是 BP 网络的核心内容，用到鲨鱼追寻血腥味的原理，同样是一个链式偏微分。

$$\Delta w_{jk}=-\eta\frac{\partial E}{\partial w_{jk}}=-\eta\times(\boldsymbol{Y}-\boldsymbol{O}_3)\times\boldsymbol{O}_2$$

说明：O3 是 wjk 的函数，这里复合函数求导加矩阵求导

同理可得：

$$,\Delta\boldsymbol{B}_{jk}=\frac{\partial E}{\partial B_{jk}}=-\eta\times(\boldsymbol{Y}-\boldsymbol{O}_3)\times\boldsymbol{ones}=-16.898\ 3。$$

j) 计算第一层与第二成之间权值和阈值调整量

先计算第二层与第三层，再计算第一层与第二层，体现了误差反向传播的思想。对传递函数求导会有一个特征：$\mathrm{f}'(\mathrm{x})=\mathrm{f}(\mathrm{x})(1-\mathrm{f}(\mathrm{x}))$

$$\Delta w_{ij}=-\eta\frac{\partial E}{\partial w_{ij}}=-\eta\times\boldsymbol{w}_{jk}\times(\boldsymbol{Y}-\boldsymbol{O}_3)\times\boldsymbol{O}_2\times(1-\boldsymbol{O}_2)\times\boldsymbol{X}$$

k) 计算调整之后的权值和阈值

在这里进行简单的加法即可。

$$w_{jk}(t+1)=-\eta\frac{\partial E}{\partial w_{jk}}+w_{jk}(t)=\Delta w_{jk}+w_{jk}(t)$$

$$\boldsymbol{B}_{jk}(t+1)=-\eta\frac{\partial E}{\partial \boldsymbol{B}_{jk}}+\boldsymbol{B}_{jk}(t)=\Delta\boldsymbol{B}_{jk}+\mathrm{B}_{jk}(t)$$

$$w_{ij}(t+1)=-\eta\frac{\partial E}{\partial w_{ij}}+w_{ij}(t)=\Delta w_{ij}+w_{ij}(t)$$

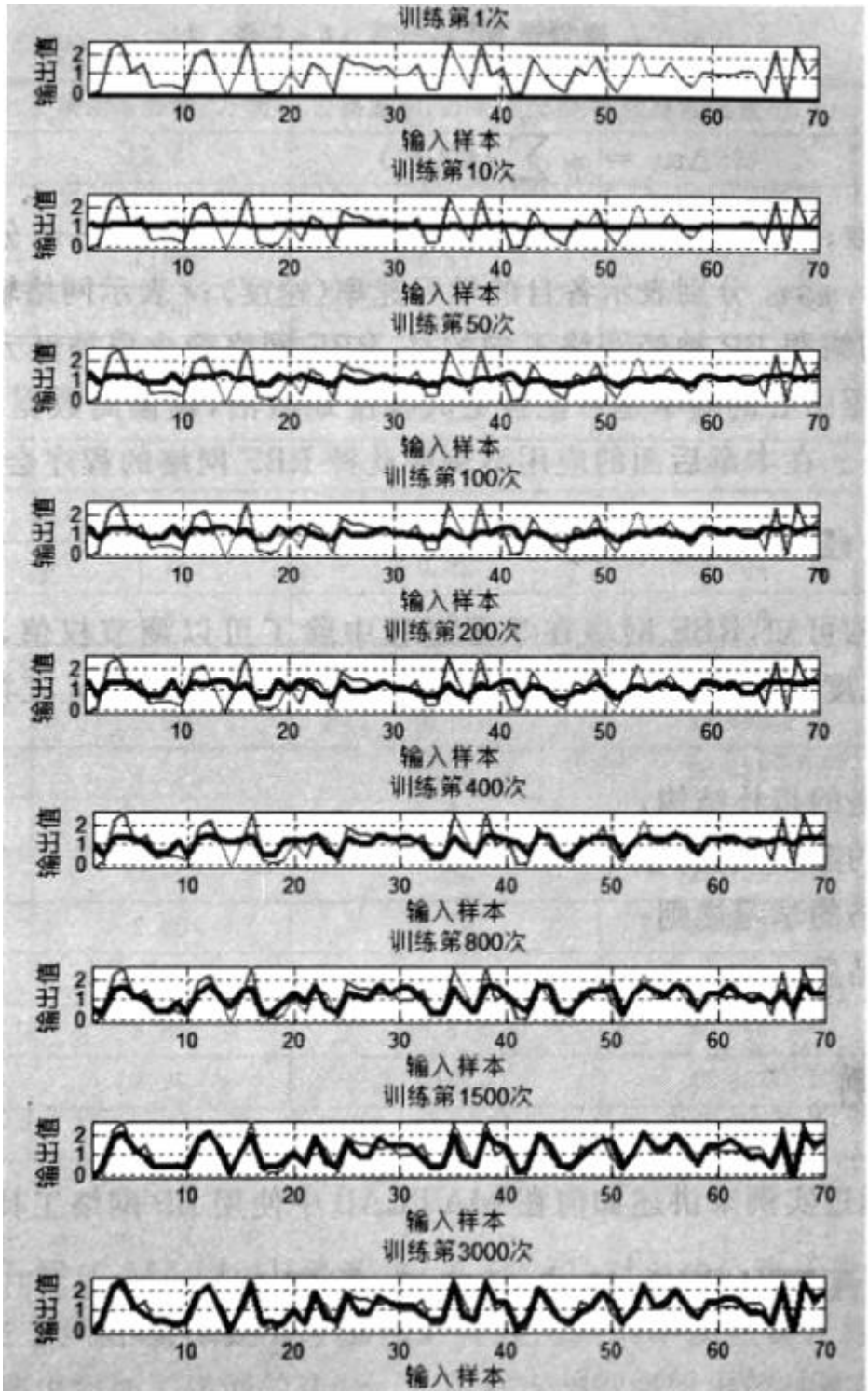
$$\boldsymbol{B}_{ij}(t+1)=-\eta\frac{\partial E}{\partial \boldsymbol{B}_{ij}}+\boldsymbol{B}_{ij}(t)=\Delta\boldsymbol{B}_{ij}+\boldsymbol{B}_{ij}(t)$$

l) 网络输出的值需要还原

由于样本提前进行过归一化，那么现在将 O3 还原为原始数据的量级。

3、 BP 神经网络的动态拟合过程

神经网络的本质是拟合的过程。可以由图得知。



RBF 神经网络的结构设计

程序设计实例

1、 附加函数学习：

a) premnmx 函数

i. 用处：把数据处理为[-1,1]之间的数

ii. 算法：

Nowarr[i]=2*(arr[i]-min)/(max-min)-1

iii. 标准用法：

[pn,minp,maxp,tn,mint,maxt]=premnmx(p,t)

说明：pn：p 的归一化结果

tn：t 的归一化结果

minp：p 每一行的最小值

其他类似。

b) Logsig 函数

logsig(n)=$\frac{1}{1+e^{-x}}$，很简单

c) Repmat 函数

i. 用处：复制一个矩阵多次，使之构成另一个大矩阵。

ii. 用法：B=repmat(A,m,n)

这样，B 就等于{A,A,A,A

A,…} (m 行 n 列)

d) Sumsqr 函数 用处：计算矩阵或向量的平方和。Sumsqr（A）。很 easy。每个元素的平方的和。

e) postmnmx 函数 用处：将被归一化的数据进行反归一化（还原）

用法：[P]=postmnmx(PN,pmin,pmax)

2、 矩阵知识

可以见这个网址 https://blog.csdn.net/u014303046/article/details/78200010/

3、 预测公路问题：

1．问题的描述

公路运量主要包括公路客运量和公路货运量两个方面。据研究，某地区的公路运量主要与该地区的人数、机动车数量和公路面积有关，表 7－3 给出了某地区 20 年的公路运量相关数据。

根据相关部门数据，该地区 2010 年和 2011 年的人数分别为 73.39 和 75.55 万人，机动车数量分别为 3.963 5 和 4.097 5 万辆，公路面积分别为 0.988 0 和 1.026 8 万平方公里。请利用 BP 网络预测该地区 2010 年和 2011 年的公路客运量和公路货运量。

问题求解过程：

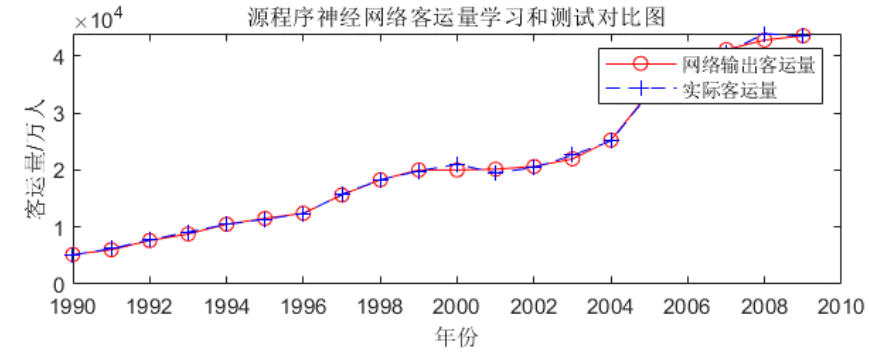
① 原始数据的输入；
② 数据归一化；
③ 网络训练；
④ 对原始数据进行仿真；
⑤ 将原始数据仿真的结果与已知样本进行对比；
⑥ 对新数据进行仿真。

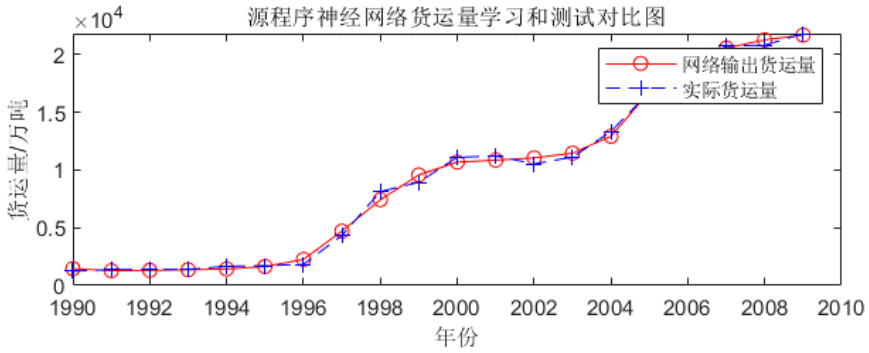
4、代码（边看书上的边写的）

```
 clc;
clear all;
close all;
SamNum=20;                %样本输入数量
TestSamNum=20;            %测试样本数
ForcastSamNum=2;          %预测数量 2
HiddenUnitNum=8;          %中间层神经元数量 8
InDim=3;                  %输入维度 3
OutDim=2;                 %输出维度 2
%读取原始数据
sqALL=xlsread('E:\顾子涵专用文件夹\学习\matlab 学习\matlab 与数学模型\神经网络数据.xlsx',1,'A2:F21');
%人数/万人
sqrs=sqALL(:,2);
%机动车数量/万辆
sqjdcs=sqALL(:,3);
%公路面积/万平方公里
sqglmj=sqALL(:,4);
%公路客运量/万人
glkyl=sqALL(:,5);
%公路货运量/万吨
glhyl=sqALL(:,6);
%转置
sqrs=sqrs';
sqjdcs=sqjdcs';
glkyl=glkyl';
sqglmj=sqglmj';
glhyl=glhyl';


p=[sqrs;sqjdcs;sqglmj]; %输入数据矩阵
t=[glkyl;glhyl];        %目标数据矩阵
%目的，使得输出与目标数据矩阵之间的误差越来越小
%这样，网络训练完毕后，就可以进行对下两年的预测
[SamIn,minp,maxp,tn,mint,maxt]=premnmx(p,t);
%对原始样本初始化
```

```
rand('state',sum(100*clock));
%依据系统时钟种子产生随机数
NoiseVar=0.01;
%噪声强度为 0.01，目的是为了防止过度拟合
Noise=randn(2,SamNum)*NoiseVar;%生成噪声
SamOut=tn+Noise;                %把噪声添加到输出样本上
TestSamIn=SamIn;
TestSamOut=SamOut;      %输入输出样本与测试样本相同
                          %因为样本数量偏少
%设定初始参数
MaxEpochs=50000;     %最多训练次数 50000
lr=0.035;           %学习速率
E0=0.65*10^(-3);     %误差
W1=0.5*rand(HiddenUnitNum,InDim)-0.1;
%维数说明：行数为中间神经元数目，列数为输入维数
%wij 表示：第 j 个输入变量与第 i 个中间神经元之间的权值
B1=0.5*rand(HiddenUnitNum,1)-0.1;
%多少中间层神经元就多少个阈值
W2=0.5*rand(OutDim,HiddenUnitNum)-0.1;
B2=0.5*rand(OutDim,1)-0.1;


ErrHistory=[];   %给中间变量预先占据内存


%开始训练
for i=1:MaxEpochs

    %由于第一层输入和输出相同。则直接搞第二层
    HiddenOut=logsig(W1*SamIn+repmat(B1,1,SamNum));
    %隐含层输出
    %第三层输入
    NetworkOut=W2*HiddenOut+repmat(B2,1,SamNum);
    %第三层输出和输入相同（输出层为线性函数）
    Error=SamOut-NetworkOut;
    SSE=sumsqr(Error);%计算平方和
    ErrHistory=[ErrHistory SSE];
    %计算调整量
    if SSE<E0
        %如果到达误差要求则跳出学习循环
        break;
    end
```

```
    %调整阈值和权值
    %一下代码及其重要，核心关键。
    Delta2=Error;
    Delta1=W2'*Delta2.*HiddenOut.*(1-HiddenOut);
    dW2=Delta2*HiddenOut';
    dB2=Delta2*ones(SamNum,1);
    dW1=Delta1*SamIn';
    dB1=Delta1*ones(SamNum,1);

    W2=W2+lr*dW2;
    B2=B2+lr*dB2;

    W1=W1+lr*dW1;
    B1=B1+lr*dB1;
end


HiddenOut=logsig(W1*SamIn+repmat(B1,1,TestSamNum));
NetworkOut=W2*HiddenOut+repmat(B2,1,TestSamNum);
a=postmnmx(NetworkOut,mint,maxt);
x=1990:1:2009;
newk=a(1,:);
newh=a(2,:);
figure;
subplot(2,1,1);plot(x,newk,'r-o',x,glkyl,'b--+');
legend('网络输出客运量','实际客运量');
xlabel('年份');ylabel('客运量/万人');
title('源程序神经网络客运量学习和测试对比图');
subplot(212);plot(x,newh,'r-o',x,glhyl,'b--+');
legend('网络输出货运量','实际货运量');
xlabel('年份');ylabel('货运量/万吨');
title('源程序神经网络货运量学习和测试对比图');


%下面进行预测
pnew=[73.39 75.55
    3.9635 4.0975
    0.9880 1.0268];
pnewn=tramnmx(pnew,minp,maxp);%利用原始输入数据归一化参数进行归一化
HiddenOut=logsig(W1*pnewn+repmat(B1,1,ForcastSamNum));
anewn=W2*HiddenOut+repmat(B2,1,ForcastSamNum);


%把网络预测得到的数据还原为原始的数量级
anew=postmnmx(anewn,mint,maxt)
```





5、代码（大部分自己写，核心部分已经进行过分析，与上面程序稍有不同）

```
 clc;
clear all;
close all;


%读取数据
sqALL=xlsread('E:\顾子涵专用文件夹\学习\matlab 学习\matlab 与数学模型\神经网络数据.xlsx',1,'A2:F21');
%人数/万人
sqrs=sqALL(:,2);
%机动车数量/万辆
sqjdcs=sqALL(:,3);
%公路面积/万平方公里
sqglmj=sqALL(:,4);
%公路客运量/万人
glkyl=sqALL(:,5);
%公路货运量/万吨
glhyl=sqALL(:,6);
%转置
sqrs=sqrs';
sqjdcs=sqjdcs';
glkyl=glkyl';
sqglmj=sqglmj';
glhyl=glhyl';
```

```
%神经网络基本数据
SamNum=20;                %样本输入数量
TestSamNum=20;            %测试样本数
ForcastSamNum=2;          %预测数量 2
HiddenUnitNum=8;          %中间层神经元数量 8
InDim=3;                  %输入维度 3
OutDim=2;                 %输出维度 2


%神经网络初始参数设置
MaxTextNum=50000;  %最大训练次数
v_learn=0.035;     %学习速率
E0=0.65E-3;        %最小误差


%输入输出样本进行合体
InputSam=[sqrs;sqjdcs;sqglmj];
OutputSam=[glkyl;glhyl];
%对样本进行归一化
[InputSamN,pmin,pmax,OutputSamN,tmin,tmax]=premnmx(InputSam,OutputSam);


%初始化网络因子
rand('state',sum(100*clock));
%依据系统时钟种子产生随机数
NoiseVar=0.01;
%噪声强度为 0.01，目的是为了防止过度拟合
Noise=randn(2,SamNum)*NoiseVar;%生成噪声


OutputSamN=OutputSamN+Noise;
TestSamIn=InputSamN;
TestSamOut=OutputSamN;       %输入输出样本与测试样本相同
                             %因为样本数量偏少




%初始化权值和阈值
W1=0.5*rand(HiddenUnitNum,InDim)-0.1;
B1=0.5*rand(HiddenUnitNum,1)-0.1;
W2=0.5*rand(OutDim,HiddenUnitNum)-0.1;
B2=0.5*rand(OutDim,1)-0.1;


ErrHistory=[];   %给中间变量预先占据内存
```

```
%开始学习
for i=1:MaxTextNum
    %Input_1=TestSamIn;
    %Output_1=TestSamIn;%线性函数，第一层输出

    %第二层输入输出
    %Input_2=W1*TestSamIn+repmat(B1,1,SamNum);
    Output_2=logsig(W1*InputSamN+repmat(B1,1,SamNum));
    %第三层输入输出
    Input_3=W2*Output_2+repmat(B2,1,SamNum);
    Output_3=Input_3;%线性

    %计算误差
    ERROR=-(Output_3-OutputSamN);
    SSE=sumsqr(ERROR);
    ErrHistory=[ErrHistory SSE];

    if SSE<E0
        break;
    end

    Delta2=ERROR;
    Delta1=W2'*Delta2.*Output_2.*(1-Output_2);
    dW2=Delta2*Output_2';
    dB2=Delta2*ones(SamNum,1);
    dW1=Delta1*InputSamN';
    dB1=Delta1*ones(SamNum,1);



    W2=W2+v_learn*dW2;
    B2=B2+v_learn*dB2;

    W1=W1+v_learn*dW1;
    B1=B1+v_learn*dB1;
end


Output_2=logsig(W1*InputSamN+repmat(B1,1,SamNum));
%第三层输入输出
Input_3=W2*Output_2+repmat(B2,1,SamNum);
Output_3=Input_3;%线性
```

```
a=postmnmx(Output_3,tmin,tmax);
x=1990:1:2009;
newk=a(1,:);
newh=a(2,:);
figure;
subplot(2,1,1);plot(x,newk,'r-o',x,glkyl,'b--+');
legend('网络输出客运量','实际客运量');
xlabel('年份');ylabel('客运量/万人');
title('源程序神经网络客运量学习和测试对比图');
subplot(212);plot(x,newh,'r-o',x,glhyl,'b--+');
legend('网络输出货运量','实际货运量');
xlabel('年份');ylabel('货运量/万吨');
title('源程序神经网络货运量学习和测试对比图');


%下面进行预测
pnew=[73.39 75.55
    3.9635 4.0975
    0.9880 1.0268];
pnewn=tramnmx(pnew,pmin,pmax);%利用原始输入数据归一化参数进行归一化
Output_2=logsig(W1*pnewn+repmat(B1,1,ForcastSamNum));
anewn=W2*Output_2+repmat(B2,1,ForcastSamNum);


%把网络预测得到的数据还原为原始的数量级
anew=postmnmx(anewn,tmin,tmax)
```