BP神经网络原理和Matlab设计（含代码）

**1、原理**

       BP（Back Propagation,反向传播）神经网络由一个输入层、一个或多个隐含层和一个输出层构成。相邻之间各神经元进行全连接，而每层各神经元之间无连接。网络按有教师示教（desire signal）的方式进行学习，也就是监督性学习。当一对学习模式提供给网络后，各神经元获得网络的响应输入和连接权值，按减小希望输出与实际输出误差的方向，从输出层经中间层交替，逐层反向修正连接权，回到输入层。此过程反复，直到网络的全局误差趋向给定的极小值，即完成学习过程。**本质**上是一个非线性优化问题，**缺点**是陷入局部极小问题（本意是想得到全局最优）

**2、网络训练以及参数设置**

a、学习速率

      BP神经网络主要采用梯度下降法进行学习，学习速率慢，稳定性高。动量法虽然比梯度法速度快，但均只适用于递增训练。另外学习速率过大会导致训练过程不稳定，学习速率过小会导致过程变长，时间久。因此合适的学习速率是关键。针对快速训练算法，缺省参数值，基本上都是最有效的设置。

b、训练注意事项：

     1）非线性网络比线性网络的误差面更复杂

     2）初始点如果更靠近局部最优点，而不是全局最优点，就不能得到正确的结果。

     3）应重复选取多个初始点进行训练，以保证训练结果的全局最优解。

     4）神经元数目过少会造成网络的不适性，神经元数目大多，会造成网络过适性。

c、具备容错能力的神经网络设计

     如果希望设计的神经网络具备容错能力，则需要用理想数据和带噪信号对网络进行多次训练。

      第一次：无噪声训练

      训练次数最多，目标误差较大，学习速率较大（相当于粗调）

      第二次：有噪声训练

      训练次数少，目标误差最大，学习速率最大

      第三次：无噪声训练

       训练次数较多，目标误差小，学习速率小（相当于精调）。该步骤可使网络在对理想信号进行识别时节省资源。

**3、matlab工具箱函数使用与神经网络设计**

a、前向神经网络创建

1）级联前向网络创建newcf

     net=newcf(P,I,[s1,s2,s3…sn],{TF1,TF2,……TFn},BTF,BLF,PF，IPF,OFF,DFF)

其中P是输入矢量（input vector）

       I是期望信号（desire vector）

       [s1,s2,s3…sn]是隐藏层的神经元个数default=[]

       {TF1,TF2,……TFm}是各层的传递函数。传递函数有tansig、purelin、logsig。隐藏层（hidden layers）之间的默认传递函数是default=“tansig”，隐藏层到输出层的默认传递函数是default=”purelin“

        BTF是反向传播网络的训练方法。可取trainlm、trainbfg、trainrp、traingd，其中默认方法是default=”trainlm“。训练速度依次下降，内存依次减少。也就是说训练速度越快，占用的电脑内存越大。当电脑内存不足时，可采用后三种训练方法

      BLF是权值学习函数，可取learngdm,learngd,参数默认取default=”learngdm“

      PF（performance function）是性能函数，可取mse\msereg，参数默认取default=”mse“

      IPF是输入矩阵处理函数default=”{’fixunknowns‘，’remconstrantrows‘，’mapminmax‘}“

     OFF是输出矩阵处理函数default=”{’remconstantrows‘，’mapminmax‘}“

     DFF是数据划分函数default=”dividerand“

2）前向BP网络创建newff

    net=newff(P,T,S)

          =newff(P,I,S,{TF1,TF2,……TFn},BTF,BLF,PF，IPF,OFF,DFF)

    参数设置可参考newcf

3）输入延迟的前向网络newfftd

**b、常用的激活函数**

purelin:线性函数（linear transfer function）

             f(x)=x

logsig:对数S型转移函数（logarithmic sigmoid transfer function）

           f(x)=1/(1+exp(-x)) 其中0<1

tansig:双曲正切S形（hyperbolic tangent  sigmoid transfer function）

            f(x)=2/(1+exp(-2x))-1     其中-1<1

**c、常用的学习函数**

learngd：基于梯度下降的学习函数。该函数通过神经元的输入和误差、以及权值和阈值的学习效率来计算权值                   阈值的变化率。

learngdm：基于梯度下降动量学习函数

**d、常用的训练函数**

1）基本训练函数train

    [net2,tr]=train(net1,X,T,xi,Ai,EW)

     net1：未训练的旧网络

     X：网络输入

     T：网络期望信号, default=zeros;

     tr:网络实际输出信号

     xi:起始延迟条件（initial delay conditon）

     Ai:initial layer delay conditon,default=zeros;

     EW:error weights

     net2：训练后的神经网络

    [net,tr,Y1,E]=train(net,X,Y)

    E：误差矩阵

2）还有其他的学习函数traingd，traingdm

**e、网络参数设置**

    net.trainparam.goal：神经网络训练的目标误差

    net.trainparam.show：显示中间结果的周期

    net.trainparam.epochs：最大迭代次数

    net.trainparam.lr：学习速率

    net.trainparam.mc：动量因子0.7-0.85

**f、其余相关函数**

1）数据归一化与反归一化

* premnmx预处理数据，可以使数据的最大值和最小值为1，-1

         [pn,minp,maxp,Tn,mint,maxt]=premnmx(P,T)

         P,T分别为原始输入和输出数据

         pn为P归一化的数据

        minp,maxp分别是P的最小 值和最大值

        Tn为T归一化变化后对应的数据

        mint,maxt分别是T的最小值和最大值

* tramnmx在训练网络时使用的数据是经过归一化的样本数据，那么以后使用网络时所用的数据应该和样本数据接收相同的预处理

         pn=tramnmx(p,minp,maxp)

         p,pn分别是变换前，后的输入数据

         minp,maxp分别是premnmx函数找到P的最大值和最小值

* postmnmx网络输出结果需要进行反归一化还原成原始的数据

          tn=postmnmx(t,mint,maxt)

          t,tn分别是变换前，后的输出数据

          mint,maxtp分别是premnmx函数找到T的最大值和最小值

2）仿真函数

      [t,x,y]=sim(model,timespan,options,ut)

      model:需要进行仿真的系统模型框图名称

      timespan：设置仿真终止时间tFinal与起止时间tStart

      options:其他仿真参数设置

     ut：系统模型顶层的外部可选输入

     t：系统仿真的时间向量

     x：系统仿真的状态变量矩阵

y:系统仿真的输出矩阵

**4、实际例子**

 close all

clear all

clc

n=5;%隐藏层的神经元个数

m=10;%输出层的神经元个数

for i=1:10

    P(:,i)=i\*ones(4,1);

end

% P = [1 2 3 4 5 6 7 8 9 10];%输入数据

% t=eye(5);

% T = [t t];%期望结果

T=eye(10);

TF1='tansig';%第一个隐藏层的传递函数

TF2='tansig';%第二个隐藏层的传递函数

TF3='purelin';%输出层的传递函数

[pn,minp,maxp,tn,mint,maxt]=premnmx(P,T);  %将数据归一化

net = newff(minmax(pn),[n,m],{TF1,TF2},'trainlm');%产生一个前向网络

% net = newff(minmax(P),[n,m],{TF1,TF2},'trainlm');%产生一个前向网络

T1=sim(net,pn);%未经训练BP神经网络的仿真输出

% T1=sim(net,P);%未经训练BP神经网络的仿真输出

% pause%程序暂停，按任意键可继续执行程序

net=init(net);%初始化网络

net.trainparam.epochs=1000;%最大迭代次数

net.trainparam.goal=0.001;%神经网络训练的目标误差

net.trainparam.show=2;%显示中间结果的周期

net.trainParam.lr=0.01;%设置学习系数

net=train(net,pn,tn);%训练网络

p2n=tramnmx(P,minp,maxp);%测试数据的归一化

T2=sim(net,p2n);

% net=train(net,P,T);%训练网络

% T2=sim(net,P);