

# BP神经网络

## 简介

BP神经网络是一种使用反向传播算法的ANN模型，其模型设置简单，便于使用。BP神经网络具有任意复杂的模式分类能力和多维函数映射能力。从本质上讲，BP算法就是以网络误差平方为目标函数、采用梯度下降法来计算目标函数的最小值。基本BP算法包括信号的前向传播和误差的反向传播两个过程。即计算误差输出时按从输入到输出的方向进行，而调整权值和阈值则从输出到输入的方向进行。正向传播时，输入信号通过隐含层作用于输出节点，经过非线性变换，产生输出信号，若实际输出与期望输出不相符，则转入误差的反向传播过程。误差反传是将输出误差通过隐含层向输入层逐层反传，并将误差分摊给各层所有单元，以从各层获得的误差信号作为调整各单元权值的依据。通过调整输入节点与隐层节点的联接强度和隐层节点与输出节点的联接强度以及阈值，使误差沿梯度方向下降，经过反复学习训练，确定与最小误差相对应的网络参数(权值和阈值)，训练即告停止。

## 初始权值优化

全链接神经网络无论在网络理论还是在性能方面已比较成熟。但是，学习速度慢、容易陷入局部极小值等问题使得网络参数的初始参数非常关键。这也成为了FCNN优化的一个主要方向。

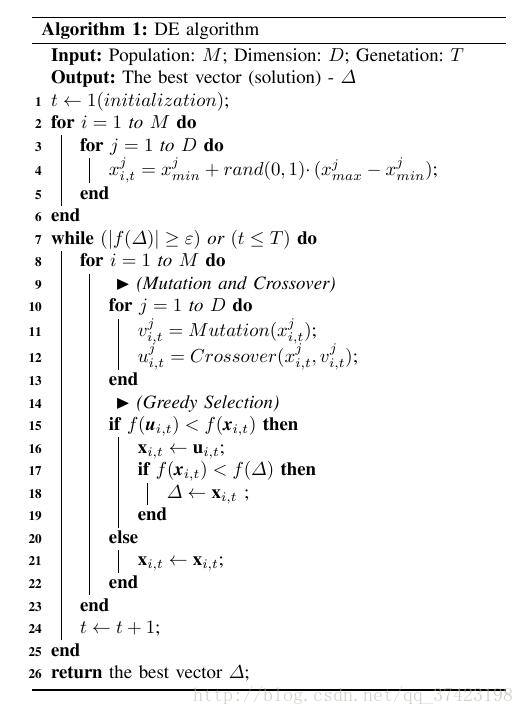
而在全连接神经网络中，初始权重值通常使用智能优化算法（如遗传算法等）和统计学习算法（如决策树、SVR等）。

# 差分进化算法

差分进化算法(Differential Evolution Algorithm，DE)是一种高效的全局优化算法，主要用于求解实数优化问题。差分进化思想来源即是早期提出的遗传算法（GeneticAlgorithm，GA），模拟遗传学中的杂交(crossover)、变异(mutation)、复制(reproduction)来设计遗传算子。

相对于遗传算法而言，相同点都是通过随机生成初始种群，以种群中每个个体的适应度值为选择标准，主要过程也都包括变异、交叉和选择三个步骤。不同之处在于遗传算法是根据适应度值来控制父代杂交，变异后产生的子代被选择的概率值，在最大化问题中适应值大的个体被选择的概率相应也会大一些。而差分进化算法变异向量是由父代差分向量生成，并与父代个体向量交叉生成新个体向量，直接与其父代个体进行选择。显然差分进化算法相对遗传算法的逼近效果更加显著。

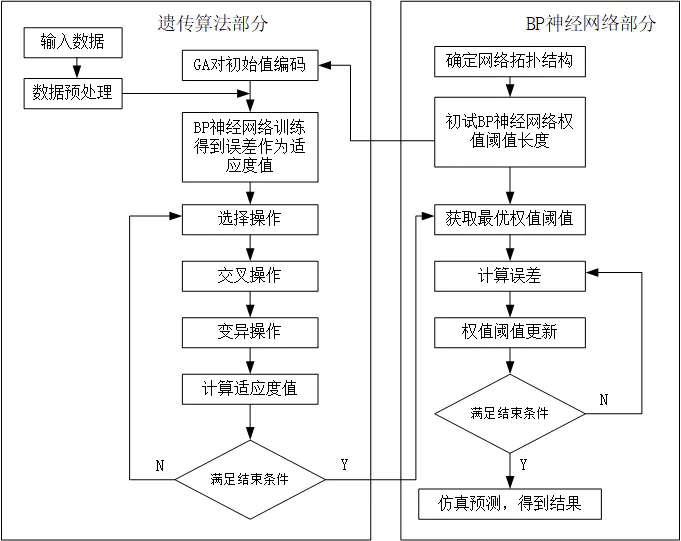
伪代码如下：



# DE-BP

## 参数设置

参考GA-BP的结构，只需更换其中遗传算法部分为差分进化算法即可。



主要参数如下：

DE参数设置

1. %% 设置DE参数
2. sizepop = 20;                      %种群规模
3. dim = net.numWeightElements;       %优化参数
4. lb = [-2];ub = [2];                %上下界
5. mutation = 0.5;                     %变异概率选择，0和1之间
6. crossover = 0.2;                    %交叉概率选择，0和1之间
7. maxgen = 50;                        %进化代数，即迭代次数

BPNN参数设置

1. S(1) = 20;           % 第 1 隐层的神经元数
2. S(2) = 12;           % 第 2 隐层的神经元数
3. S(3) = 8;            % 第 3 隐层的神经元数
4. net = feedforwardnet(S);    % 建立网络 trainFcn = default = 'trainlm'
5. % 根据输入、输出数据结构，配置输入和输出层的神经元数
6. net = configure(net,p,t);
7. % 设置输入层之后的第 i 层的传递函数
8. net.layers{1}.transferFcn = 'tansig';   % 第 1 隐层的传递函数为 tansig %help tansig
9. net.layers{2}.transferFcn = 'tansig';   % 第 2 隐层的传递函数为 tansig
10. net.layers{3}.transferFcn = 'tansig';   % 第 3 隐层的传递函数为 tansig
11. net.layers{4}.transferFcn = 'purelin';  % 第 4 层（输出层）的传递函数为 purelin
12. % 设置参数
13. net.trainFcn='trainlm';     %训练函数：中型网络，内存需求最大，收敛速度最快
14. net.trainParam.goal = 1e-8;    %训练目标最小误差1e-5
15. net.trainParam.show = 50;    %每间隔50步显示一次训练结果
16. net.trainParam.lr = 0.05;    %学习速率0.05
17. net.trainParam.mc = 0.7;    %附加动量因子0.9
18. net.trainparam.mu = 1e-10;      %随着 mu 的增大，学习过程按梯度下降法进行

## 关键代码

对差分进化的染色体进行解码，转换为模型参数

1. %网络权值赋值
2. %input weight
3. net.iw{1,1}=reshape(w1,20,5);%将w1由1行inputnum\*hiddennum列转为hiddennum行inputnum列的二维矩阵
4. %layer weight
5. net.lw{2,1}=reshape(w2,[12 20]);%更改矩阵的保存格式
6. net.lw{3,2}=reshape(w3,8,12);
7. net.lw{4,3}=reshape(w4,1,8);
8. net.b{1}=reshape(b1,20,1);%1行hiddennum列，为隐含层的神经元阈值
9. net.b{2}=reshape(b2,12,1);
10. net.b{3}=reshape(b3,8,1);

变异操作

1. %变异操作
2. ri = randperm(sizepop,3); % 随机选择三个个体以备变异使用
3. while isempty(find(ri==j))==0
4. ri = randperm(sizepop,3);
5. end
6. mpop = position(ri(1),:)+mutation\*(position(ri(2),:)-position(ri(3),:));%差分向量
7. if isempty(find(mpop>ub'))== 0
8. mpop(find(mpop>ub')) = (ub(find(mpop>ub')))';
9. end
10. if isempty(find(mpop<lb'))== 0
11. mpop(find(mpop<lb')) = (lb(find(mpop<lb')))';
12. end

交叉操作

1. %交叉操作
2. tmp = zeros(1,dim);
3. for i = 1:dim
4. if rand < crossover
5. tmp(i) = mpop(i);
6. else
7. tmp(i) = position(j,i);
8. end
9. end

选择操作

1. %选择操作
2. cpredict = DE\_fit(net,train\_input,train\_output,tmp);
3. cpredict = mapminmax('reverse',cpredict,outputps);
4. cfit = sqrt(sum((cpredict'-y\_train).^2)/length(y\_train));

## 算法实现过程中的问题及对应解决

1.DE\_BP性能较低：

推测是DEoptimization函数处理不当，多读几个代码

2.初始化参数有好几次，使用自定义和minmax函数修正

3.优化目标不明确，需要查看sample中给出的代码，他优化的是什么，为什么只用了一点数据

4.计算一下VC-dimension是不是有必要，参数数量太多，明显过拟合

5.收敛速度的优化是否因为多次fit，matlab内存残留了训练过程，而非是算法实际有效？ 比较收敛的epoch位置，计算更多参数，如r，mse

4.方法和gd相比多余？加大了计算量只是为获得更优的初始权值。

5.三次实验BPNN代码不一样。统一bp参数

## 对于DE\_BP的理解

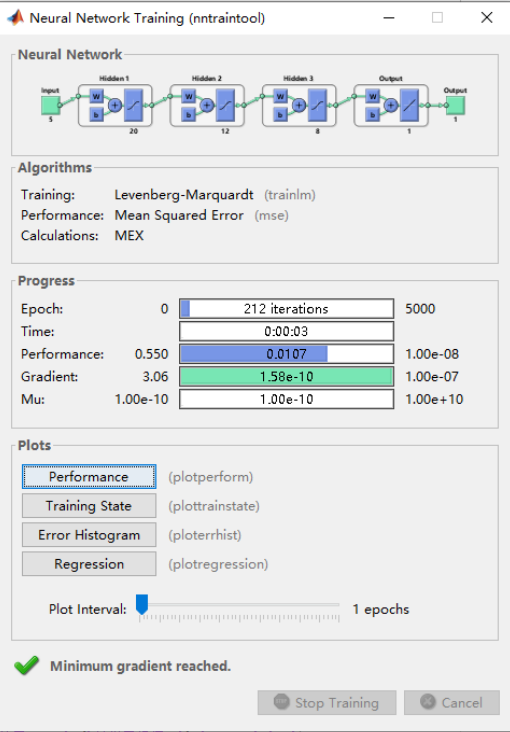
DE算法在本任务中近似于Boosting，差分的过程就是小模型(向量差)以拟合优度为加权权重值融合在一起的过程。因此得以以较小的计算代价将待训练参数优化到使反向传播算法快速收敛。

# 实验

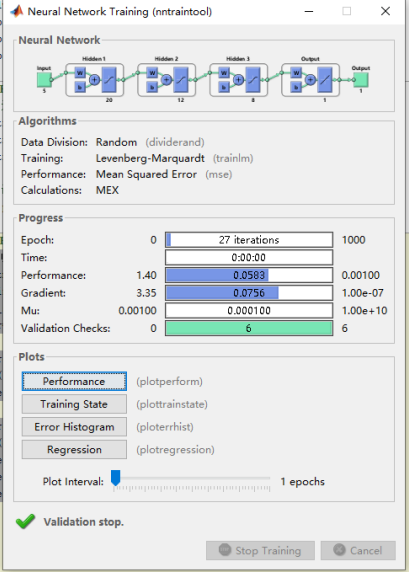
## 实验分析

### 不同优化方法下BPNN迭代次数的对比

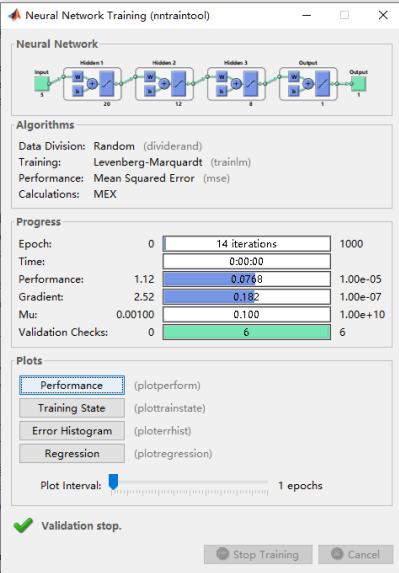
不实用任何初始权值优化时迭代epoch为212。



使用GA优化时迭代epoch为27。



使用BP优化时迭代epoch为14。



因此，DE\_BP显著提升了网络初始化性能。

### BPNN训练效果

1.预测误差



2.预测误差偏差率



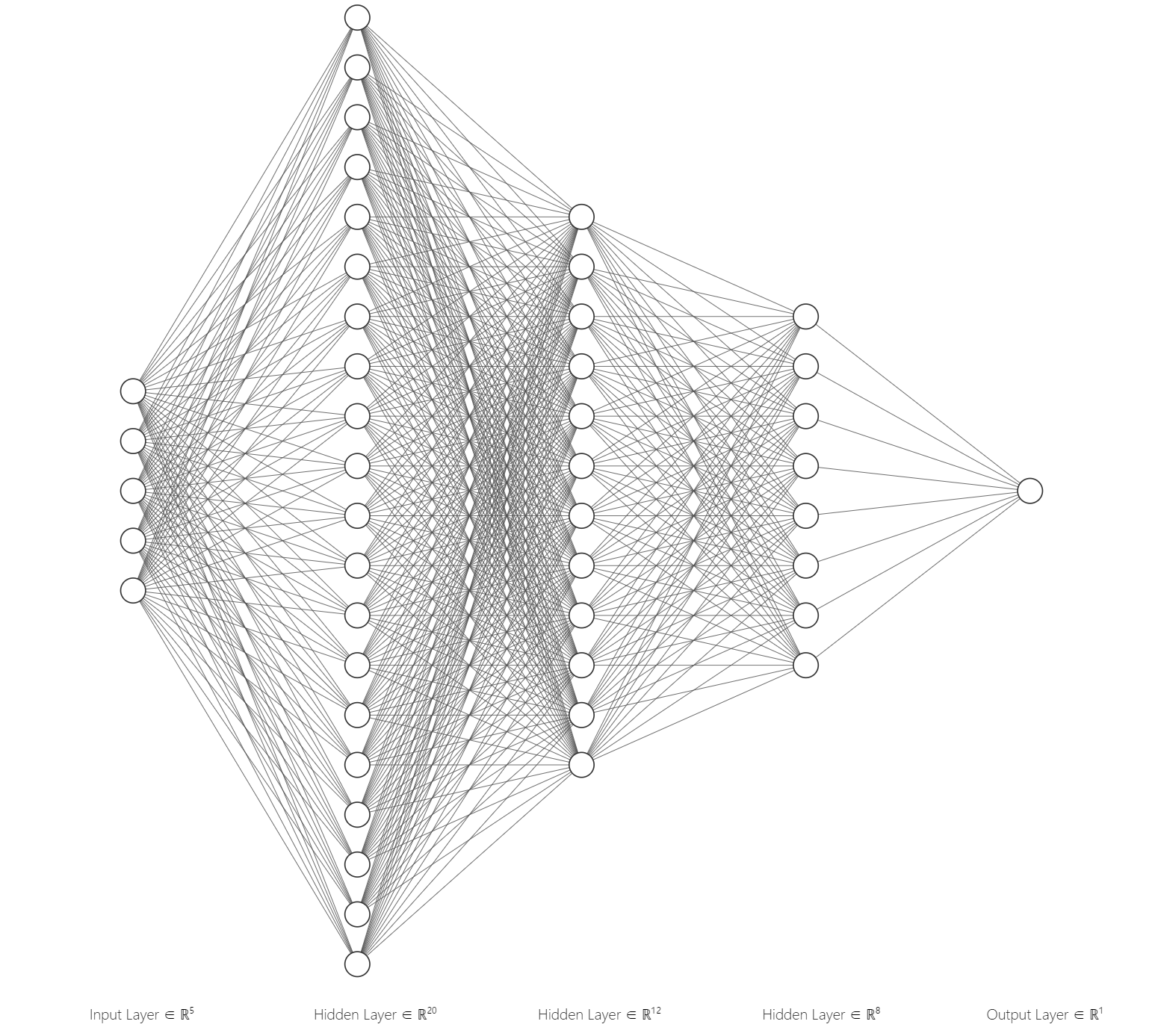
3.对所有样本的RMSE



观察可知网络拟合性能较好，BPNN训练有效。

## 思考

按照已给出的网络设计，本次实验设置的BP神经网络如下图：



其参数量D为5\*20+20+20\*12+12+12\*8+8+8\*1+1=485

神经元个数V为：5\*20+20\*12+8\*1=348

VC-dimension近似为O(VD)=O(16870)

要充分训练该神经网络，所需样本数量为10倍的VC-dimension。因此，此模型存在样本量不足的问题，模型对数据过拟合，缺乏泛化能力。使用参数更少、VC-dimension更低的传统统计学习方法进行拟合可能会有效地改善此种情况。

# 引用

Sontag E D. VC dimension of neural networks[J]. NATO ASI Series F Computer and Systems Sciences, 1998, 168: 69-96.

<https://github.com/Felix660/HeuristicApproach>

<https://github.com/fcampelo/DEMO>

王晓敏,刘宏伟,李石妍.一种基于差分进化的BP神经网络学习算法[J].电子设计工程,2011,19(15):16-18.