实验报告：基于 BP 算法的模糊神经网络控制器

王瑞哲　PB19071509

信息科学技术学院

摘　要：根据反向传播（Back-Propergation）网络设计方法，结合 Matlab 程序中 Neural network Toolbox 的程序，设计并实现给定算法下的模糊神经网络控制器。同时借助 BP 网络优化理论，着重研究不同的学习或改进方法、不同的隐含层神经元节点数设置和不同的学习速率对于网络训练效果的影响。

关键词：BP 网络；Matlab；模糊神经网络控制器；网络优化

反向传播网络(Back Propagation Network, 简称 BP 网络）是 1986 年由 Rumelhart 和 McClelland 为首的科学家提出的概念，是一种按照误差逆向传播算法训练的多层前馈神经网络，是应用最广泛的神经网络模型之一。

BP 网络的突出优点在于具有很强的非线性映射能力和灵活的网络结构，但也存在着学习速度慢、容易陷入局部极小值等缺点。对此已有基于标准梯度下降法和基于数值优化方法的网络训练算法等多种 BP 网络优化算法，目标在于加速网络的收敛速度和尽量避免陷入局部极小值的问题。

1 实验要求与基本设计

1.1 实验要求

根据所学过的 BP 网络设计及改进方案，设计实现模糊控制规则为 T = int((e+ec)/2)的模糊神经网络控制器，其中输入变量 e 和 ec 的变化范围分别是：e = int[-2, 2]，ec = int[-2, 2]。网络设计的目标误差为 0.001。设计要求为：

①输入、输出矢量及问题的阐述；

②给出网络结构

③学习方法（包括所采用的改进方法）

④初始化及必要的参数选取；

⑤最后的结果，循环次数，训练时间。在此过程中还需要着重讨论：

a) 不同隐含层 S1 时的收敛速度与误差精度的对比分析；

b) 当 S1 设置为较好的情况下，在训练过程中取始终不变的学习速率 lr 值时，对 lr 值为不同值时的训练时间，包括稳定性进行观察比较；

c) 当采用自适应值学习速率时，与单一固定的学习速率 lr 中最好的情况进行对比训练的观察；

d) 给出结论或体会。

⑥验证，采用插值法选取多于训练时的输入，对所设计的网络进行验证，给出验证的 A 与 T 值。

1.2 实验基本设计方案

根据实验要求，输入矢量 P 可以表述成两个行矢量 e 和 ec 矢量之和，即：

$$P=\begin{bmatrix} e \\ ec \end{bmatrix}$$

其中 e 矢量和 ec 矢量中各元素的变化范围均为 int[-2, 2]，即-2 到 2 之间的随机整数值。输入矢量 P（维数 R×Q）依题意知为 2 行，即 R=2，分别为 e 矢量和 ec 矢量；至于 Q 值，题目并未明确叙述，不妨取 Q=10。在 matlab 程序中，初始化两随机分布在[-2,2]间的具有 Q=10 个元素的行向量，这可以利用下列语句实现：

```
e = round(rand(1,Q)*4-2);
% 生成[-2,2]间的 10 个随机整数
ec = round(rand(1,Q)*4-2);
```

然后组合 e 和 ec 生成输入矢量 P：

```
P = [e;ec];
```

期望的输出矢量依题意即为 T = int((e+ec)/2)。在 matlab 程序中，利用 fix 函数对矩阵中各元素做去除小数的取整操作，即：

```
T = fix((e+ec)/2);
```

网络结构上，考虑单层隐含层神经网络，即 2-S1-1 型，其中隐含层神经元数 S1 待定，根据后续的实验结果确定最合适的 S1 值。隐含层采用 S

型正切激活函数，输出层采用线性激活函数。

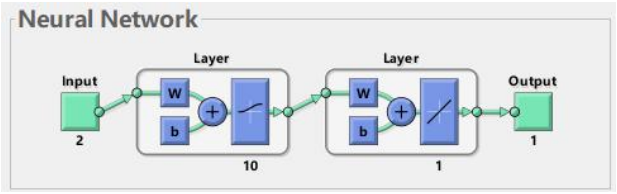


图 1　网络基本结构设计

备注：由于接下来的实验需要探究何样的 S1 是最佳的，故在初步调试程序无误后，固定输入 P 矢量进而固定输出矢量 T，以达到控制变量的目的。

2 实验项目

2.1 标准梯度下降法下的 S1 值选取

依据前初始化结果，训练网络，观察其训练循环次数、训练时间和误差精度（利用误差平方和 sse 来衡量）。其中初始化部分为：

e = [1,2,-1,2,1,-2,-1,0,2,2];

ec = [-1,2,2,0,1,-1,0,2,1,2];

P = [e;ec];

T = fix((e+ec)/2);

其中，为了控制变量，固定了一组输入矢量以比对性能（该组输入矢量为先前随机生成的一组值）。以 S1 取 10 为例，网络训练程序部分为：

```
% 网络结构：考虑单层隐含层神经网络，即2-S1-1型
% 隐含层采用S型正切激活函数，输出层采用线性激活函数
% 隐含层节点数为S1，根据后续实验过程确定一个合适的数值
S1 = 10;

net = newff(minmax(P),[S1,S2],{'logsig','purelin'},'traingd');
% 隐含层logsig，输出层线性，标准梯度下降
net.performFcn='sse';                    %执行函数为误差平方和函数
net.trainParam.epochs = 10000;           %最大训练步长
net.trainParam.goal = 0.001;             %执行函数目标值
net.trainParam.lr = 0.03;          %调整学习速率
[net,tr] = train(net,P,T);          %训练网络
W = net.iw{1,1};       %训练后的网络权值
B = net.b{1};          %训练后的网络偏差

Y = sim(net,P);            %查看训练时间
t=tr.time(end)             %计算均方差
SSE = perform(net,T,Y)
```

图 2　标准梯度法下降程序部分

运行程序，根据代码内容，终端上可以显示出训练时间和均方误差为：

t = 4.1480

SSE = 9.9999e-04

根据 matlab NNT 工具箱界面，可得到本次训练的各项指标为：

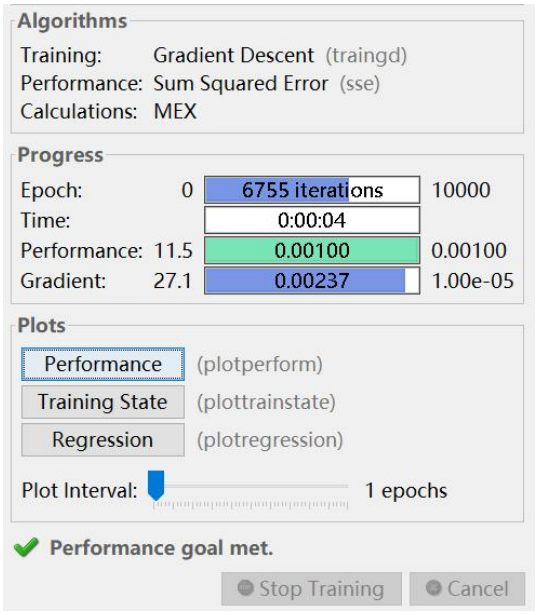


图 3　S1=10 时训练成果参数界面

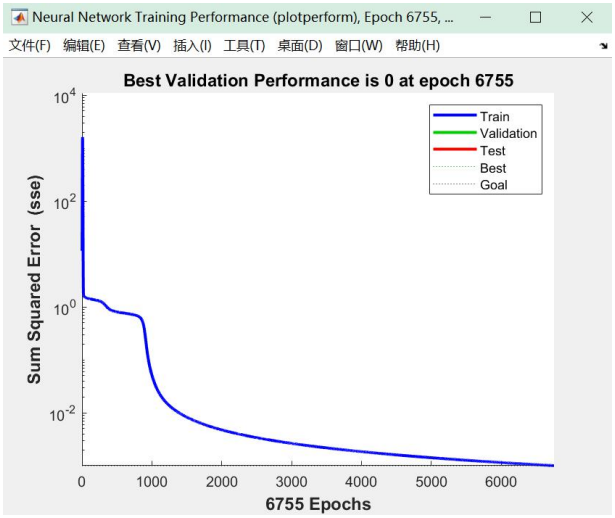


图 4　S1=10 时均方误差曲线界面

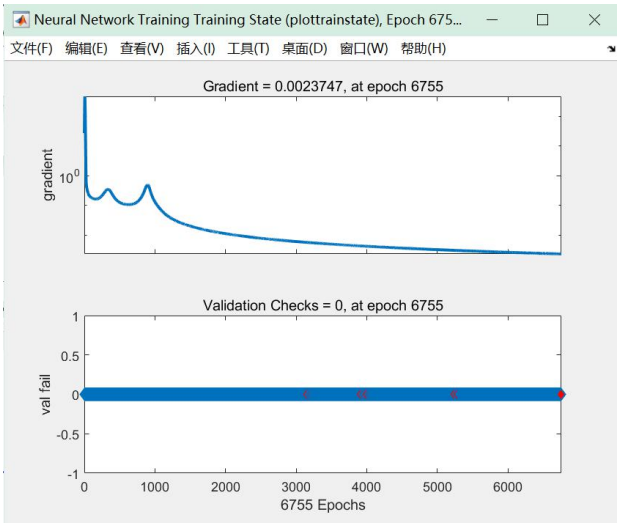


图 5　S1=10 时梯度变化曲线界面

matlab 工具箱对神经网络的训练结果提供了可视化的分析。图 3 展示了训练参数，如 Algorithms 界面展现了训练方法 Gradient Desent（梯度下降法）、展示方法 mean Squared Error（均方误差），Progress 界面展示了训练进程，包括训练次数（6755 次）、Performance 数值（绿色表示达到了设定值 0.001）、Gradient 数值（0.00237），Plot 界面则提供了图形界面来展现训练效果。图 4 和图 5 则分别展现了均方误差和梯度数值随着训练循环次数的增长的变化情况。

由于每次训练时，网络的初始权值都是随机取定的，因此实际情况下对比不同 S1 时的训练效果时并不能做到严格意义上的控制变量。由于 S1 不同时，权值矩阵和偏差矩阵的维度都不同，因此也不可能取定统一的训练初始权值。因此采取的折中策略为，每取定一个 S1 数值时，重复训练 5 次，取其各训练效果参数的平均值作为对比依据。如取定 S1=10 时：

表 1 S1=10 时各次网络训练效果

次数	均方误差	训练次数	训练时间/秒
1	9.9999e-04	6755	4.1480
2	9.9989e-04	4361	2.4670
3	9.9995e-04	5465	3.1580
4	9.9942e-04	6010	3.4560
5	9.9442e-04	4446	2.5820

取定更多的 S1 数值，重复上面的实验过程，记录得到下面的对比表格：

表 2 不同 S1 值下的网络训练效果对比

S1	平均 均方误差	平均 训练次数	平均 训练时间/秒
5	9.99496e-04	6305.4	3.8164
10	9.98734e-04	5407.4	3.1622
11	9.99825e-04	4242.4	2.4452
12	9.99106e-04	2521.0	1.5438
13	9.98465e-04	3305.4	2.0946
14	9.99493e-04	3395.0	2.2342
15	9.98933e-04	3495.4	2.1738
20	9.99466e-04	5155.6	3.5494

由表格可见，S1 取各值时 sse 数值相差不大，即意味着神经网络均在限定的 10000 次循环内满足了目标误差 0.001 的要求；而当 S1=12 时，训练次数和训练时间均取得最小值，且是在输入矢量不变，5 次训练取平均的情况下得到的结果，故我们有理由说明，S1=12 为此神经网络最合适的取值；当 S1 取得过大（如 20）和过小（如 5）时，需要训练的次数和训练时间都会有较大的增长。

2.2 取定 S1 时学习速率 lr 的选取

在固定 S1 的取值后，进而研究学习速率 lr 对训练效果的影响。除训练网络参数赋值处进行微调外，其余程序部分均不做调整。每取定一个 lr 数值时，重复训练 5 次，取其各训练效果参数的平均值作为对比依据：

表 3 不同 lr 值下的网络训练效果对比

lr	平均 均方误差	平均 训练次数	平均 训练时间/秒
0.001	0.2294	10000	5.8812
0.02	9.9977e-04	3440.8	2.2152
0.025	9.9795e-04	3247.5	2.3250
0.03	9.9799e-04	2898.4	1.6948
0.035	9.9873e-04	4717.2	2.7686
0.04	9.9997e-04	4653.0	2.6810
0.05	0.0029	10000	6.0068
0.1		发散	
0.2		发散	

由表 3 可以看出，在本次实验的神经网络中，取定速率 lr 为 0.03 时有较好的表现结果，此时平均训练次数和平均训练时间都最小；当 lr 较小或较大时，训练情况明显不佳，超过一定范围后在所设定的 10000 次最大训练次数内不能达到所要求的误差精度 0.0001；当 lr 值取得过大时，网络不收敛，无法得到预期训练结果。

2.3 自适应与固定学习速率的比较

前文所做实验都采用的是 BP 网络的标准梯度下降法。实际上，为了改进性能，现在已经有很多 BP 算法的改进算法，其目的都在于优化网络权值的修正过程，以达到更快的计算速度和更好的计算

精度。其中，自适应学习速率法是基于标准梯度下降法的一种改良算法，其原理为根据网络输出的SSE 误差，自适应地调整网络的学习速率，这与 2.2 小节中所提到的寻求最佳 lr 值的思路有些许类似。在 matlab 程序中，新建网络语句改写为：

net = newff (minmax(P), [S1,S2], {'logsig', ...
'purelin'}, 'traingda');

其中参数'traingda'即表示新建自适应学习速率网络。net.trainParam.lr 参数仍设为 0.3，表示网络的初始学习速率。其他参数不变，对比自适应学习速率与标准梯度下降法训练的差别：

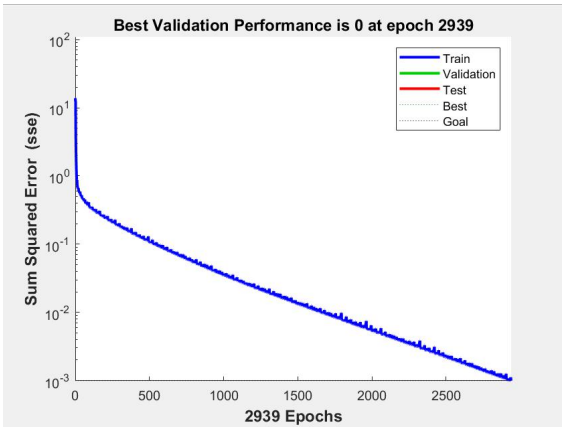
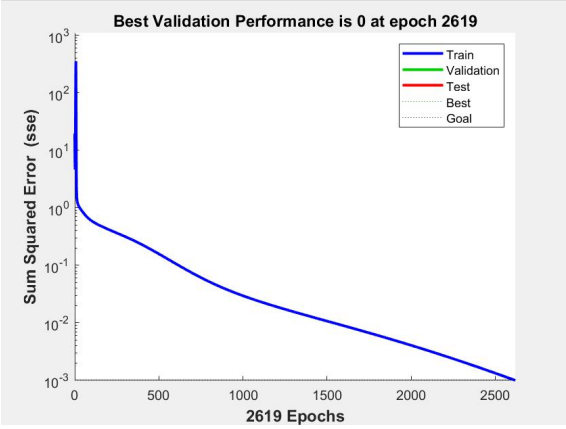


图 6 自适应学习速率法训练效果



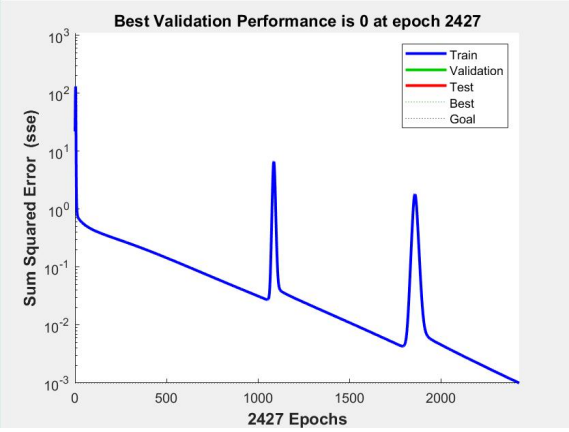


图 7 标准梯度下降法训练效果

从图 6 和图 7 的对比可以看出，采用自适应学习速率网络的训练结果和前所使用的各参数均取最优的标准梯度下降法训练网络表现大致相同，所消耗的训练时间和训练次数均相差不大。

但值得注意的是，标准梯度下降法的训练过程中，SSE 的整体变化曲线或呈平缓下降状，或会出现突然的阶跃峰值，这可能与训练速率固定而初始权值随机有关；而在自适应学习速率网络中，整体曲线基本呈平缓下滑态势，但曲线中始终有毛刺与波动，这是因为自适应学习速率算法中随着网络的训练会对学习速率时刻进行调整，则每次输出的 SSE 也自然地会产生小幅度的波动。

2.4 其他改进算法

如 2.3 小节所述，除了自适应学习速率算法，还有许多被用于改善 BP 网络的算法，如附加动量法、BFGS 拟牛顿法、Levenberg-Marquardt 法等，他们的核心都在于优化网络权值的改进过程，以谋求更好的训练效果。

设计实验：对于同一组输入矢量和初始权值，分别建立标准梯度下降法、自适应学习速率法、附加动量法、弹性 BP 算法、BFGS 拟牛顿法、Levenberg-Marquardt 法网络，对比分析他们的训练结果：

表 4 不同算法下的网络训练效果对比

算法	平均均方误差	平均训练次数	平均训练时间/秒
标准梯度下降法	9.9949e-04	2521.0	1.5438
自适应学习速率	9.9873e-04	2731.5	1.7790
附加动量法	9.9936e-04	6373.3	3.8025
弹性 BP 算法	9.6212e-04	380.5	0.4480
BFGS 拟牛顿法	9.2503e-04	22.5	0.1271
Levenberg-Marquardt 法	5.2732e-07	4.2	0.0831

可以看出，标准梯度下降法和自适应学习速率法的表现差异不大，附加动量法在本次实验的神经网络中表现不佳；而 BFGS 拟牛顿法、弹性 BP 算法和 Levenberg - Marquardt 法具有非常强大的计算能力，相比于标准梯度下降法，他们在平均训练次数和平均训练时间上都有了极大的改进，尤其是 Levenberg - Marquardt 法，每次训练均只需个位数

的训练循环次数，甚至还可以将平均误差 SSE 缩减至 10^{-6} 量级以下。

3 实验分析与插值验证

采用插值方法对所训练的神经网络进行人工验证（选取前面已调整参数的标准梯度下降法网络）。对原输入矢量采取每两元素均值插值法，增加每行元素从 10 个至 19 个，期望输出矢量 T 也从 1×10 矢量扩充至 1×19 矢量：

```
e1 = [1,1.5,2,0.5,-1,0.5,2,1.5,1,-0.5,-2,-1.5,-1, ...
-0.5,0,1,2,1,2];
ec1 = [-1,0.5,2,2,2,1,0,0.5,1,0,-1,-0.5,0,1,2, ...
1.5,1,1.5,2];
P1 = [e1;ec1];
T1 = fix((e1+ec1)/2);
```

因为网络已经训练完毕，所以直接将 P1 矢量输入进网络，观察其输出的均方误差：

```
A = sim(net,P1);
SSE1 = perform(net,T1,A)  %计算均方差
```

运行结果为：

```
SSE = 9.9702e-04
SSE1 = 3.0569
```

可以看出，相比于先前训练结果，采用插值改变后的输入矢量送入网络后，输出与预期误差相差较大。这可能是由于模糊神经网络的算法所导致的：原输入矢量各元素均是[-2,2]范围内的整数值，输出矢量算法 T = fix((e+ec)/2)也做了清除小数部分取整的操作，在某种程度上这就是所谓“模糊”算法的体现。而在插值算法中，会出现 0.5、1.5 这类非整数插值输入值的产生；但与此对应的输出矢量算法并未改变，即对于更复杂多变的输入，输出值的范围并未改变，这样就更增加了“模糊”算法出错的概率。

4 实验分析及心得体会

本次实验借助 matlab 神经网络工具箱，分析、设计、编写一个简单的模糊神经网络控制器。基于反向传播算法（BP 算法）理论，本次实验重点研究了不同隐含层神经元节点数 S1，和学习速率 lr 这两个参数对于网络训练结果的影响。同时借助 BP 网络优化理论，研究了不同的学习或改进方法对于网络训练效果的影响。对于不同实验效果好坏的衡量，本次实验采用固定预期误差值，观察网络达到这个预期值所消耗的训练时间和总共的训练次数的方法来进行。

通过本次实验，对 BP 网络原理有了更深层次的认识，进一步加深了对于 BP 网络设计和训练方法的理解，体会了神经网络识别精度训练的重要性，根据多组数据明显感受到了重要参数的改变对神经网络识别效果的影响，并掌握评估训练识别效果的一般方法；同时体会到了 matlab 软件及其 Neural Network 工具箱在进行神经网络建立及训练时的方便快捷。

参考文献：

[1] 丛爽. 面向 MATLAB 工具箱的神经网络理论与应用（第三版）[M]. 合肥：中国科学技术大学出版社, 2009.4: 293-305.