基于MATLAB的软件可靠性BP神经网络模型

宋绍云，张玉忠
（玉溪师范学院 信息技术工程学院，云南 玉溪 653100）

摘 要：研究并分析了BP神经网络的结构和特点，针对不足之处提出改进方法。在改进的基础上建立神经网络软件可靠性新模型。通过MATLAB仿真工具进行了实例仿真，证实该新模型比传统模型预测精度高，泛化能力强。

关键词：神经网络；软件可靠性；MATLAB仿真
中图分类号：TP183 文献标识码：A

Research of software reliability BP neural network model based on MATLAB

SONG Shao Yun，ZHANG Yu Zhong
(College of Information Technology Engineering，Yuxi Normal University，Yuxi 653100，China)

Abstract：Structure and features of BP neural network are discussed,and improved method is proposed.Simulation results show that,new model of software reliability has higher predictable accuracy and generalization ability than traditional model.

Key words：neural network；software reliability；MATALB simulation

在众多的神经网络模型中，BP网络(Back Propagation Network)具有结构简单、工作状态稳定、易于硬件实现等优点，因此其应用最为广泛。然而，应用BP神经网络分析解决实际问题时，需要考虑用多少层网络，每层设多少个神经元节点、选择何种传输函数和训练算法等，均无可行的理论指导，只能通过大量的实验进行计算获得参考值。

本文主要研究BP神经网络结构中隐层数的选择及其神经元数的确定，改进训练算法，建立新BP神经网络软件可靠性模型，通过实例验证新模型的性能，并与传统模型进行比较。

1 基本概念

1.1 软件可靠性的定义

1983年美国IEEE计算机学会对“软件可靠性”作如下定义：

(1)在规定条件下，在规定的时间内，软件不引起系统失效的概率，该概率是系统输入和系统使用的函数，也是软件中存在的错误函数；系统输入将确定是否会遇到已存在的错误(如果错误存在的话)；

(2)在规定的时间周期内，在所述条件下程序执行所要求功能的能力。

1.2 软件可靠性参数

下面对几个主要的软件可靠性参数进行介绍[1]。

(1)软件错误：在软件生存期内不希望或不可接受的人为错误，其结果是导致软件缺陷的产生；

(2)软件缺陷：存在于软件(文档、数据或程序)之中的、不希望或不可接受的偏差，其结果导致软件在一定条件下运行出现故障；

(3)软件故障：软件运行过程中出现的一种不正常的内部状态。当软件运行时出现故障且没有采取措施处理时，便产生软件失效；

(4)软件失效：软件运行没有产生所预期的服务；

(5)软件失效时间的概率密度：简称失效概率密度，用$f(t)$表示，它是失效时间t的函数；

(6)软件失效时间的分布函数：从时刻0到给定时刻t间的失效概率，用$F(t)=\int_0^t f(x)\mathrm{d}x$表示；

(7)软件可靠性函数：在时刻t之前软件一直无失效

运行的概率用$R(t)=1-\int_0^t f(x)dx$表示；

(8)平均失效时间(MTTF)：从软件开始运行到出现一个故障的期望时间，也就是是概率密度函数的均值$MTTF=\int_0^{+\infty} tf(t)\,dt$；

(9)软件故障率：单位时间内软件发生故障的概率，用$\lambda(t)$表示，其与软件可靠性函数的关系为：$R(t)=\exp(-\int_0^t \lambda(x)dx)$。

2 软件可靠性建模

2.1 神经元数学模型

神经元一般是一个多路输入、单一输出的非线性结构，其作用是将可能的无限域输入变换到唯一指定的有限范围内。最基本的神经元数学模型如图1所示。

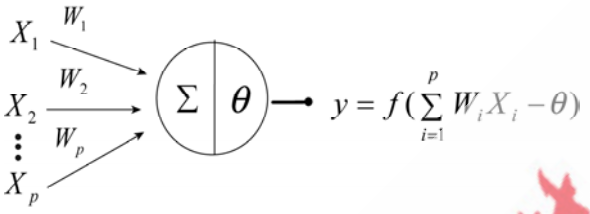

图1 神经元数学模型

其中，神经元的输入是X_1，X_2，…，X_p；对应的权系数分别是W_1，W_2，…，W_p。y是神经元的输出，f是神经元激发函数，常用的激发函数有：

(1)线性函数$f(x)=x$；

(2)阈值函数(阶跃函数)：$f(x)=\begin{cases}1, & x\geqslant 0\\ 0, & x<0\end{cases}$；

(3)Sigmoid函数(S函数)：$f(x)=\frac{1}{1+e^{-x}}$；

(4)双曲正切函数：$f(x)=\frac{1-e^{-x}}{1+e^{-x}}=\text{th}(x)$。

BP网络通常有一个或多个隐层，隐层中的神经元均采用Sigmoid型变换函数，输出层的神经元采用纯线变换函数。

2.2 BP神经网络的缺陷

BP神经网络有如下几点缺陷：

(1)网络学习收敛速度太慢，一般可采用变化的学习速率或者自适应的学习速率加以改进；

(2)易陷入局部极小问题，BP算法是建立在随机梯度下降法上的一个非线性优化，不可避免地存在局部极小问题，常得到局部最小点；

(3)网络结构难以确定，网络隐含层的层数及隐含节点数的选取非常困难，至今尚无理论依据；

(4)网络的学习和记忆具有不稳定性，一个训练成熟的网络，在原来的基础上再给它提供新的模式时，原有的连接权将被全部打乱，必须重新对网络进行训练才能继续使用，这无疑增加了网络学习的难度。

2.3 改进的BP神经网络

对BP网络的改进主要包括2方面：一方面是算法的改进(包括学习参数的选取及多种提高收敛速度的算法的运用等);另一方面是网络结构的设计(包括输入与输出层节点数的确定，隐层数、隐层节点数和初始权值的选取等)。

2.3.1 网络结构的设计

(1)初始权值的选取

使用逐步搜索法对网络权值进行初始化，可以避免随机取值的局部行为。由于神经网络训练算法中采用Sigmoid函数($f(x)=1/(1+e^{-x})$)，实际输出值在[0，1]之间，因而初始权值不宜太大，一般都选在[–1，1]之间。先将初始权值区域(记为H)N等分，在这N个区域内分别随机产生初始权值进行学习，选取对应误差函数E最小的那个区域再N等分，再在这N个小区域内重复上述步骤，当误差函数E不再减小时，则认为找到了最优点并停止迭代。

(2)隐层层数设计

理论分析证明，具有单隐层的前馈网可以映射所有连续函数，只有当学习不连续函数(如锯齿波等)时，才需要2个隐层，所以多层前馈网最多只需要2个隐层。在设计多层前馈网时，一般先考虑设一个隐层，当一个隐层的隐节点数很多仍不能改善网络性能时，才考虑再增加一个隐层。经验表明，采用2个隐层时，如在第一个隐层设置较多的隐节点而第二个隐层设置较少的隐节点，则有利于改善多层前馈网的性能。此外，对于有些实际问题，采用双隐层所需要的隐节点总数可能少于单隐层所需的隐节点数。所以，对于增加隐节点仍不能明显降低训练误差的情况，应该尝试一下增加隐层数。

2.3.2 隐层神经元个数

隐节点的作用是从样本中提取并存储其内在规律。每个隐节点有若干个权值，而每个权值都是增强网络映射能力的一个参数。隐节点数量太少，网络从样本中获取的信息能力就差，不足以概括和体现训练集中的样本规律；隐节点数量过多，又可能把样本中非规律性的内容如噪声等也学会记牢，从而出现所谓“过度吻合”问题，反而降低了泛化能力。此外隐节点数太多还会增加训练时间。

确定最佳隐节点数的一个常用方法称为试凑法，可先设置较少的隐节点训练网络，然后逐渐增加隐节点数，用同一样本集进行训练，从中确定网络误差最小时对应的隐节点数。在用试凑法时，可用一些确定隐节点数的经验公式。这些公式计算出来的隐节点数只是

一种粗略的估计值，可作为试凑法的初始值。下面介绍几个公式：

$$m=\sqrt{n+l}+\alpha \quad , \quad m=\log 2^n \quad , \quad m=\sqrt{nl}$$

式中，m为隐层节点数，n为输入层节点数，l为输出层节点数，α为1～10之间的常数。

2.3.3 算法设计

(1)增加动量因子

在BP算法中加入动量因子，将对各连接权增量的计算方式加以调整，目的是平滑权的变化。将动量因子m加到权值和闭值的修正公式中，可得到反向传播的动量改进权值修正公式：

$$\Delta w_{ij}(k+1)=(1-m_c)\eta\delta_i P_i+m_c\Delta w_{ij}(k)$$

$$\Delta b_i(k+1)=(1-m_c)\eta\delta_i+m_c\Delta b_i(k)$$

其中，k为训练次数；P_j为第j个输入向量；η为学习率；$\Delta b_i(k+1)$为第k+1次阀值；δ_i为反向传播误差导数；m_c为动量因子，在此取0.95；$m_c\Delta w_{ij}(k)$称为动量项。增加动量因子，可以在维持算法稳定的前提下具有更高的学习速度，而且能提高收敛速度和改善算法的动态性能。另外，当收敛轨迹进入某个一致的方向后，它可以加速收敛。增加动量项的方法最终可以归结为对学习步长的动态调整。因此，动量项的加入，使得算法可以考虑前后两次调整量，实现加速收敛并避免振荡的有效学习过程。而且可以通过迭代增量作对比来确定选择缩小或者放大调整因子，使算法的学习过程得到最佳。

(2)动态改变可变学习速率

在标准BP算法中，学习速率是不变的，而实际上学习速率对收敛速度的影响很大，通过对它的在线调整可以大大提高收敛速度。学习速率调整的一般规则是：当总误差减小(新误差比老误差小)，则学习速率增加(将实际值乘以某一放大因子)；当总误差增大，则学习速率减小。具体规则如下：

①在整个训练集上，当平方误差在权值更新后增加了，且超过某个设定的百分数θ(1%～5%)，则权值更新被取消，学习速度η被乘以一个因子ρ(0<ρ<1)，并且动量因子m_c被设置为0；

②当平方误差在权值更新后减小了，则权值更新被接受，而且学习速度η被乘以一个因子γ>1。如果m_c被设置为0，则恢复到以前的值；

③当平方误差的增长小于θ，则权值更新被接受，学习速度η保持不变，如果m_c被设置为0，则恢复到以前的值。

3 模型的实现及验证

3.1 数据来源

采用表1所示测试数据，将全部19个数据样本分成2个部分，其中16个作为训练集来建立模型，剩下的3个作为测试集来评价模型的效果，由于分组方法对实验效果有一定的影响，故将样本数据进行3组实验，以减轻分组产生的影响，如表2所示。

3.2 数据归一化处理

用$M_i=\dfrac{X_i-X_{\min}}{X_{\max}-X_{\min}}$对表1中的数据进行处理，把所有的数据缩放到[0，1]范围之间，以更好地提高神经网络的训练速度，得到表3的数据。

3.3 实验过程

采用3层BP网络：1个输入层，1个隐层和1个输出层。输入层由于只有1个16维的向量，故有1个神经元；隐层用上述方法试凑后应有19个神经元；输出层由于每次输出1个值，所以具有1个神经元。按2.3.3节所设计的BP算法，在MATLAB下带有动量项的权值修正法用learnbpm.m函数实现，动态改变可变学习速率采用trainbpa.m设置递增因子和递减因子来实现。用测试集进行仿真。第1组测试结果与标准BP网络测试结果如表4所示。

有关第二组及第三组的测试结果同第一组测试结

表1 某软件的测试数据

测试时间T_K	1	2	3	4	5	6	7	8	9	10
执行时间W_K	2.45	2.45	1.96	0.98	1.68	3.37	4.21	3.37	0.96	1.92
发现错误总数Y_K	15	14	66	103	105	110	146	175	179	206
测试时间T_K	11	12	13	14	15	16	17	18	19	
执行时间W_K	2.88	1.44	3.26	3.84	3.84	2.30	1.76	1.99	2.99	
发现错误总数Y_K	233	255	276	298	304	311	320	325	328	

表2 实验样本数据分组

实验分组	训练集	测试集
1	1，2，4，5，7，8，9，10，11，12，13，14，16，17，18，19	3，6，15
2	2，3，4，5，6，7，8，9，10，11，12，14，15，16，17，18	1，13，19
3	1，2，3，4，5，6，7，8，9，10，12，13，14，15，17，19	8，11，16

表3 数据归一化处理

测试时间T_K	1	2	3	4	5	6	7	8	9	10
执行时间W_K	0.458	0.458	0.308	0.006	0.222	0.742	1	0.742	0	0.295
发现错误总数Y_K	15	14	66	103	105	110	146	175	179	206
测试时间T_K	11	12	13	14	15	16	17	18	19	
执行时间W_K	0.591	0.148	0.708	0.886	0.886	0.413	0.246	0.317	0.625	
发现错误总数Y_K	233	255	276	298	304	311	320	325	328	

表4 改进型网络和标准BP网络测试结果比较

序号	改进后的预测结果	标准BP网络的预测结果	实际结果	改进后的误差	标准BP网络误差
3	65.028 6	63.275 6	66	−0.971 4	−2.724 4
6	110.230 9	112.287 4	110	0.230 9	2.287 4
15	305.727 4	306.713 0	304	1.727 4	2.713

果相仿,限于篇幅从略。

从表4可以看出，改进后的BP网络比标准BP网络的误差要小得多。总体上看改进型神经网络模型比标准BP神经网络模型效果更好。实验中改进型神经网络的训练要比标准BP神经网络的训练速度要快，而且泛化能力更强，收敛速度更快。

参考文献

[1] 朱磊. 基于BP神经网络的软件可靠性模型选择研究[D]. 重庆：重庆大学，2006. 10.

[2] LAPRIE J. C. Dependability evaluation of software systems in operation. IEEE Trans. on Software Engineering. 1984，10(6)：701–714.

[3] 尹乾. 基于神经网络的软件可靠性模型[D]. 北京：北京师范大学，2006. 6.

[4] HAHKIN S. 神经网络原理 (第2版)[M]. 叶世伟，史忠植，译. 北京: 机械工业出版社，2004.

[5] 刘耦耕，贺素良. 神经网络结构参数的计算机自动确定[J]. 计算机工程与应用，2004(13)：72–74.

(收稿日期：2009-02-02)