第21卷 第1期 北京工业大学学报 Vol. 21 No. 1
1995年3月 JOURNAL OF BEIJING POLYTECHNIC UNIVERSITY Mar. 1994

神经网络BP算法的一种改进

吕宏伯 黄 铮 张 方
（北京工业大学应用数学系，100022）

摘 要 提出了一种改进算法，对原来的误差函数做了很小的改动．改进算法既保持了原BP算法的优点．又可以有效地避免假饱和，使学习过程可以较快地完成，取得了满意的结果．本文推导了改进后的算法，并结合实例对算法的效果进行了讨论．

关键词 神经网络，BP算法，人工智能
分类号 TP202.2

1 BP算法

人工神经网络是一门近年来重新受到人们重视的学科．它的理论、模型、算法和应用还有很多待研究的课题．误差反向传播训练算法 (Back Propagation，以下简称BP算法) 是人工神经网络的重要算法，使用BP算法进行学习的BP网络是重要的人工神经网络之一．BP网络的基本思想源于回归分析，但由于它不仅是非线性的，而且模型的复杂程度与参数的数量是一般回归分析所无法比拟的．正是由于这些特点，BP算法在实际应用中遇到了一些至今仍未很好的解决的问题．

BP算法是一种有指导的学习算法，靠调节各层的连接权值来学会所有的训练模式．文献[1]约定第 k 个训练模式中的输入数据，即第0层神经元的输入为

$$x_1^{(k)},\ x_2^{(k)},\ \cdots,\ x_n^{(k)}$$

相应的输出数据为

$$T_1^{(k)},\ T_1^{(k)},\ \cdots,\ T_m^{(k)}$$

其中 $k=1,\ 2,\ \cdots,\ K$，K 为训练模式的数量．

设第 l 层为神经元的数值为

$$y_j^l,\ (j=0,\ 1,\ \cdots,\ n_l)\ (l=0,\ 1,\ \cdots,\ L)$$

其中 n_l 为第 l 层神经元的个数．L 为神经网络层数．第0层神经元的数量 $n_0=n$，第0层神经元的输入，即输入数据

$$x_j=y_j^0(j=1,\ 2,\ \cdots,\ n)$$

需要说明的是这里 l 是上标，并非指数，而且

$$n_L=m$$

神经元的输出为

$$y_j^l=F(s_j^l)\ (j=1,\ 2,\ \cdots,\ n_l)$$

收稿日期：1994－07－01．

其中 F 为转移函数，常采用 S 型 (Sigmoid) 压缩函数 (见图 1). 有时，F 中需要一个控制门限值，只要令

$$y_0^l=-1\ (l=1,\ 2,\ \cdots,\ L-1)$$

对应的 w_{0j}^{l+1} 就成了 y_j^{l+1} 的控制门限值.

现在，对于确定的

w_{ij}^l $l=1,\ 2,\ \cdots,\ L;\ i=0,\ 1,\ \cdots,\ n_{l-1};\ j=0,\ 1,\ \cdots,\ n_l$. 由每组输入数据

$x_1^{(k)},\ x_2^{(k)},\ \cdots,\ x_n^{(k)}$ $k=1,\ 2,\ \cdots,\ K$就可以计算出

$$y_1^{L(k)},\ y_2^{L(k)},\ \cdots,\ y_m^{L(k)}$$

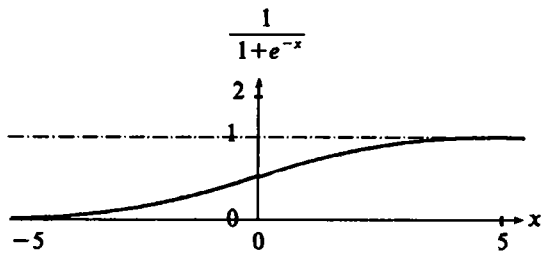

图 1 S型 (Sigmoid)压缩函数

问题是如何选择 w_{ij}^l 使得所有的 $y_j^{L(k)}$ 与 $T_j^{(k)}$ 的差异最小，按照最小二乘的惯例，通常取误函数为

$$E(W)=\frac{1}{2}\sum_j\sum_k[y_i^{L(k)}-T_j^{(k)}]^2$$

这里待定的参数集 W 是指全体 ω_{ij}^l.

为了应用最速下降法寻找使 $E(W)$ 最小时的参数 w_{ij}^l，需要计算所有的 $\frac{\partial E}{\partial w_{ij}^l}$. 假定

$$\frac{\partial E}{\partial s_j^{l+1(k)}}=-\delta_j^{l+1(k)}\ (j=1,\ 2,\ \cdots,\ n_{l+1};\ k=1,\ 2,\ \cdots,\ K)\qquad(1.1)$$

已经计算出，则

$$-\delta_i^{l(k)}=\frac{\partial E}{\partial s_i^{l(k)}}=\sum_j\frac{\partial E}{\partial s_j^{l+1(k)}}\cdot\frac{\partial s_j^{l+1(k)}}{\partial y_i^{l(k)}}\cdot\frac{\partial y_i^{l(k)}}{\partial s_i^{l(k)}}\qquad(1.2)$$

$$=\sum_j(-\delta_j^{l+1(k)})\cdot w_{ij}^{l+1(k)}\cdot F'(s_i^{l(k)})$$

$$=-F'(s_i^{l(k)})\sum_j\omega_{ij}^{l+1(k)}\delta_j^{l+1(k)}$$

即

$$\delta_i^{l(k)}=F'(s_i^{l(k)})\sum_j w_{ij}^{l+1(k)}\delta_j^{l+1(k)}\qquad(1.3)$$

这里 $i=1,\ 2,\ \cdots,\ n_l;\ j=1,\ 2.\ \cdots,\ n_{l+1};\ k=1,\ 2,\ \cdots,\ K$.

显然

$$\delta_j^{L(k)}=-\frac{\partial E}{\partial s_j^{L(k)}}=-\frac{\partial E}{\partial y_j^{L(k)}}\cdot\frac{\partial y_j^{L(k)}}{\partial s_j^{L(k)}}$$

$$=-(y_j^{L(k)}-T_j^{L(k)})\cdot F'(s_j^{L(k)})$$

$$=(T_j^{L(k)}-y_j^{L(k)})\cdot F'(s_j^{L(k)})\qquad(1.4)$$

根据实际输出 $y_j^{L(k)}$ 与正确输出 $T_j^{L(k)}$ 的误差及转移函数可以计算出全部

$$\delta_j^{l(k)} \qquad l=1,\ 2,\ \cdots,\ L;\ \ j=1,\ 2,\ \cdots\ n_j;\ k=1,\ 2,\ \cdots,\ K.$$

进而，可以计算出全部

$$\frac{\partial E}{\partial w_{ij}^l}=\sum_k \frac{\partial E}{\partial s_j^{l(k)}}\cdot\frac{\partial s_j^{l(k)}}{\partial w_{ij}^l} \tag{1.5}$$

$$=\sum_k(-\delta_j^{l(k)}).\ y_i^{l-1(k)}$$

每一个待定参数的增量可计算出

$$\Delta w_{ij}^l=-\eta\frac{\partial E}{\partial w_{ij}^l}=\eta\sum_k\delta_j^{l(k)}.\ y_i^{l-1(k)} \tag{1.6}$$

$$i=0,\ 1,\ \cdots,\ n_{l-1};\ j=1,\ 2,\ \cdots,\ n_l;\ l=1,\ 2,\ \cdots,\ L$$

其中 η 为步长．

2　BP 算法的改进

BP 算法将多层前向网络的学习转换成了一个非线性规划问题．从理论上讲，$E(W)$ 是连接权 w_{ij}^l 的非线性可微函数，BP 算法是可行的．但由于 $E(W)$ 的复杂结构及参数数量极大，再加上训练数据的多样性，使得学习时间冗长，甚至出现无法继续计算的恼人局面．算法中采用的 S 型压缩函数 F 的导数具有如下的形式

$$F'(s_j^{L(k)})=F(s_j^{L(k)})\ (1-F(s_j^{L(k)})) \tag{2.1}$$

只要 $s_j^{L(k)}$ 的绝对值稍大，就会落入 F 的饱和区，使 $F(s_j^{L(k)})$ 与 0 或 1 十分接近，由 (1.4) 与 (2.1) 可得

$$\delta_j^{L(k)}=(T_j^{L(k)}-y_j^{L(k)}).\ F'(s_j^{L(k)})$$

$$=(T_j^{L(k)}-y_j^{L(k)})\,F(s_j^{L(k)})\ (1-F(s_j^{L(k)})) \tag{2.2}$$

计算出的 $\delta_j^{L(k)}\approx 0$．这样通过反向传播计算出的 $\delta_j^{l(k)}\approx 0$．从而使 $\frac{\partial E}{\partial w_{ij}^l}\approx 0$，进一步 $\Delta w_{ij}^l\approx 0$．注意，由于 $s_j^{L(k)}$ 落入 F 的饱和区造成的 $\Delta w_{ij}^l\approx 0$ 与 $T_j^{L(k)}$，$y_j^{L(k)}$ 间的差异无关．如果这时 $y_j^{L(k)}$ 与 $T_j^{L(k)}$ 已经很接近了，$\Delta w_{ij}^l\approx 0$ 是合理的．但是，如果 $y_j^{L(k)}$ 与 $T_j^{L(k)}$ 的差别较大，仍需对某些 w_{ij}^l 进行修正时，$\Delta w_{ij}^l\approx 0$ 就会导致误差函数下降极其缓慢，迭代数万步都不能收敛的问题，此时继续计算实际上已经没有意义了．这就是在学习过程中很容易出现的“假饱和”现象．虽然可以在开始学习时将 w_{ij}^l 的初始值取得很小，使开始时不出现假饱和．但是由于误差函数中没有一种抑制假饱和的机制，所以，在学习过程中仍然可能出现假饱和．假饱和使学习不能完成，严重地消弱了 BP 算法的实用价值．在实际应用中，针对造成假饱和的原因，将 BP 算法做了一些改进，取得了较好的效果，改进的基本思想是，修正连接权时优先考虑使所有的 $y_j^{L(k)}$ 与 $T_j^{L(k)}$ 的误差趋向平均，以避免出现假饱和现象．具体作法如下．

使用当前的 w_{ij}^l 可以计算出一个输出向量 $\{y_j^{L(k)}\}$，它与正确的输出向量 $\{T_j^{L(k)}\}$ 之间存

在的差异，而

$$E(W)=\frac{1}{2}\sum_j\sum_k[y_j^{L(k)}-T_j^{(k)}]^{1+\alpha}$$

是对这种差异的综合度量．在维数极高时，这种度量是粗糙的．当然，可以考虑设多个函数来描述这种差异，但它无疑会大大增加BP算法的复杂性，从而降低了它的可行性，我们的改进是在 $E(W)$ 中加入了一个可调参数．将误差函数改变为

$$E_\alpha(W)=\frac{1}{1+\alpha}\sum_j\sum_k[y_j^{L(k)}-T_j^{(k)}]^{1+\alpha}$$

其中

$$\alpha=\frac{2q+1}{2p+1}$$

p, q 为两个自然数，且 $2p+1$ 与 $2q+1$ 互质，即

$$(2p+1,\ 2q+1)=1$$

显然，这样选取的 α 使 $E_\alpha(W)$ 与 $E(W)$ 有相同的最小值与最小值点．与前面的推导类似，可以顺序得出

$$\delta_j^{L(k)}=(T_j^{(k)}-y_j^{L(k)})^\alpha\cdot F'(s_j^{L(k)})$$

$$\delta_i^{l(k)}=F'(s_i^{l(k)})\sum_j w_{ij}^{l+1(k)}\,\delta_j^{l+1(k)}$$

$$i=1,\ 2,\ \cdots,\ n_l;\ \ j=1,\ 2,\ \cdots,\ n_{l+1};\ \ k=1,\ 2,\ \cdots,\ K. \qquad (2.3)$$

$$\Delta w_{ij}^l=-\eta\frac{\partial E}{\partial w_{ij}^l}=\eta\sum_k\delta_j^{l(k)}\cdot y_i^{l-1(k)}$$

$$l=1,\ 2,\ \cdots,\ L;\ i=1,\ 2,\ \cdots,\ n_{l-1};\ j=1,\ 2,\ \cdots,\ n_l$$

设

$$\Delta_j^{(k)}=|y_j^{L(k)}-T_j^{(k)}|$$

原来的误差函数 $E(W)$ 梯度的各分量正比于相应的 $\Delta_j^{(k)}$，改进的误差函数 $E_\alpha(W)$ 梯度的各分量正比于相应的 $(\Delta_j^{(k)})^\alpha$，所以，可以通过调节 α，对大小差异不同的 $\Delta_j^{(k)}$ 予以侧重，使它们的差异不要过大．当 $\Delta_j^{(k)}$ 差异较大时，可取 $\alpha>1$．此时，尽管对各个满足 $\Delta_j^{(k)}\neq0$ 项都有 $(\Delta_j^{(k)})^\alpha<\Delta_j^{(k)}$，但是，很显然较小的 $\Delta_j^{(k)}$ 缩小率较大，所以，与原来的 $E(W)$ 相比．较小的 $\Delta_j^{(k)}$ 对应的梯度的缩小率较大．这样就使 $\Delta_j^{(k)}$ 较大的项在 $E\alpha(W)$ 中占的比重加大．使较大的 $\Delta_j^{(k)}$ 较快地减小，以使 $\Delta_j^{(k)}$ 达到比较均衡的状态．实践证明，在这种情况下，改进算法尽管在学习的开始阶段收敛速度比过去略低，但有效地抑制了假饱和现象．当 $\Delta_j^{(k)}$ 已比较均衡时，可以调小 α，当调成 $\alpha=1$ 时，$E\alpha(W)$ 即成为原来的 $E(W)$，但此时 w_{ij}^l 已进入了比较有利的初始位置．如果需要，甚至可以调成 $0<\alpha<1$．这样可以加快收敛速度．

需要注意的是，尽管 α 调整前后 $E\alpha(W)$ 是单调变化的，但毕竟是不同的误差函数了，因此 α 不宜调整的过于频繁．

可以把改进的学习过程形象地比喻为将一个产品表面磨光．当 α 较大时，用的是粗砂

纸，首先去掉产品表面的大颗粒；当 α 调小时，用的是细砂纸，再去掉小颗粒，以使产品表面光滑．

比较 (1.4)～ (1.6) 式与 (2.1) 可以发现，与原 BP 算法相比，本算法的改动极小，所以保持了原算法的优点．从下面的实例可以看到这些改动对抑制假饱和又是极为有效的．

3 实例

为检验改进算法的效果，选择了多组具有不同特点的训练数据及不同拓扑结构的 BP 网络，用改进的 BP 算法分别训练这些网络，均取得了较好的结果．下面仅以其中的一组为例进行分析．

这种训练数据共有 221 对模式，每个模式包含 14 个输入，对应一个二值的输出．相应的 BP 网络输出层有一个神经元，输入层与唯一的隐层均有 14 个神经元．为便于对比，每次训练过程中，都固定了 $E\alpha(W)$ 中的 α．由于输出层神经元的个数仅为一，所以在学习过程中容易出现假饱和的现象．事实上，训练这个网络时，选用了多组 w_{ij}^{l} 的初始值，总是在某一阶段后，出现假饱和现象使训练无法有效的继续下去．例如，在某一组初值下，训练到 800 步时，虽然随着 w_{ij}^{l} 的调整，网络对绝大多数的训练模式可以产生正确的输出．但是也有 3 对训练模式已处于假饱和状态，这时误差曲线逐渐呈水平状态，见图 2．为避免 $E\alpha(W)$ 中 α 的影响，便于对结果进行比较，本文各图中的误差均为绝对误差之和，即

$$\sum_{j}\sum_{k}|y_j^{L(k)}-T_j^{(k)}| \tag{3.1}$$

使用改进的 BP 算法，分别取 $\alpha=11/9$, $\alpha=9/7$, $\alpha=11/7$ 在相同的初始值下对网络进行训练，结果见图 3．α 取得较小时，不足以抑制假饱和的出现，如 $\alpha=11/9$．但如 α 超过一定值时，就可以抑制假饱和的发生，使训练得以在较短的时间内完成，如 $\alpha=9/7$, $\alpha=11/7$, 当然，对不同的训练模式组和不同的网络，可以抑制假饱和的 α 一般是不同的．进一步的观察可以发现，与较小的 α 相比，当 α 较大时，在学习的开始阶段误差曲线下降较慢．但

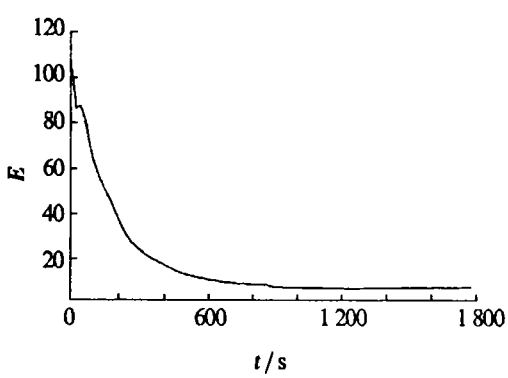

图 2　使用 BP 法时的误差

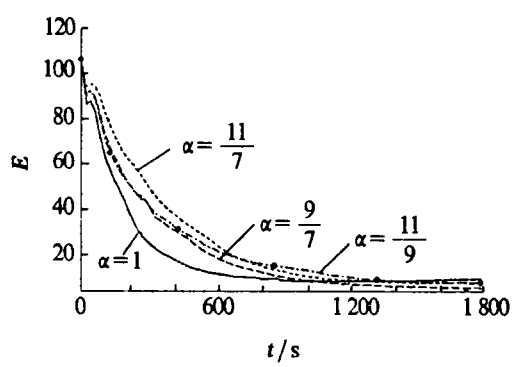

图 3　使用不同 α 时的误差曲线

到了一定的时候以后，由于改进算法使误差趋于均匀，可以取比较大的学习率，且没有出现假饱和．误差曲线的下降速度相对提高了，总体收敛时间得以缩短．可以比较$\alpha=1$, $\alpha=11/9$, $\alpha=9/7$的情况．但当α过大时，误差曲线在相当长的一段时间内下降缓慢．由本例可见，较大的α可以有效的抑制假饱和，取适当α可以得到更快的收敛速度．但不是α越大越好．本例中α选为9/7时的综合效果较好．关于α的选择，将在另文中讨论．

4 结论

通过理论分析及实际验算，可以得到如下结论：

1 采用改进的BP算法，可以较为有效地避免神经元的输出落入S型函数的饱和区，抑制假饱和的发生．

2 选择适当的α可以提高收敛速度．

3 α不宜过大，以恰好抑制假饱和为最好．

4 本方法的主要作用在于抑制假饱和的出现，使学习得以完成，并且可以稍稍提高学习速度，如何从根本上提高学习速度的问题仍有待探讨．

参 考 文 献

1 周继成，周青山，韩飘扬.人工神经网络——第六代计算机的实现．北京：科学普及出版社，1993

An Improvement in the BP Algorithm

Lu Hongbo Huang Zheng Zhang Fang

(Department of Applied Mathematics, Beijing Polytechnic University, 100022)

Abstract This paper presents an improvement in the Back Propagation (BP) algorithm. The new algorithm retains the best part of the old algorithm yet avoids fault saturation although it changes a small part of the error function. With these changes the study process can be completed quickly and with good results. This paper derives the impoved algorithm and discusses it using practical examples.

Keywords neuron net, Back Propagation(BP) algorithm, artifical intelligence