前向神经网络的一种快速分层线性优化算法

田传俊, 韦 岗
(华南理工大学无线电与自动控制研究所, 广东广州 510640)

摘 要: 本文利用数学分析的方法, 提出了一种前向神经网络快速分层线性优化算法, 其特点是: 用新方法构造了各层的目标函数; 无须计算 Hessian 矩阵, 加快了算法的收敛速度. 仿真实验表明, 与传统算法如误差反传法或 BP 法和含势态因子(Momentum factor)的 BP 法以及现有的分层优化算法相比, 新算法能加快收敛速度, 并降低学习误差.

关键词: 前向神经网络; 学习算法; 分层线性优化算法
中图分类号: TP183 文献标识码: A 文章编号: 0372-2112 (2001) 11-1495-04

A New Layer-Wise Linearized Algorithm for Feedforward Neural Network

TIAN Chuan-jun, WEI Gang
(*Inst. Electronic Engineering and Control, South China University of Technology, Guangzhou, Guangdong* 510640, *China*)

Abstract: This paper presents a fast layer-wise linearized algorithm for feedforward neural networks by mathematic methods with the following features: constructing target function for each layer by new methods; not calculating the Hessian matrix, thus greatly reducing the learning time. Simulation shows that the new algorithm can accelerate the convergence rate and reduce the error compared with the existing algorithms such as backpropagation (BP) algorithm, BP algorithm with momentum factor and existing layer-wise algorithms.

Key words: feedforward neural networks; learning algorithm; OLL-algorithm

1 引言

前向神经网络有着广泛的应用前景, 如可应用于函数逼近[2, 4]和时间序列预测[3, 4, 6]等方面. 常用的三层前向神经网络如图 1 所示. 关于前向神经网络, 目前有不少综述文章[7, 8]和专著[9]都详细地介绍过其相关知识, 可供参考.

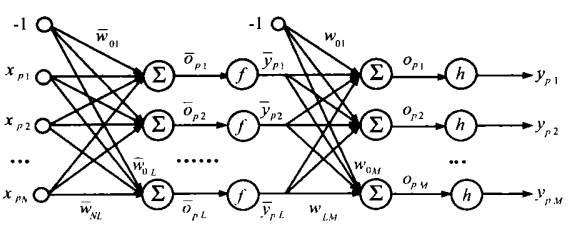

图 1 三层前向神经网络的结构

其中, 输入层、隐层和输出层的单元数分别为 N、L 和 M, 记为 N-L-M 网; 特性函数 f 是连续可微且可逆的, 输出层特性函数 h 为 f 或线性函数; $\boldsymbol{X}_p=(x_{p1}, \cdots, x_{pN})^T$ 表示第 p 个输入矢量; $\bar{w}_{nj}$ 是输入层第 n 个单元到隐层第 j 个单元的连接权值, $\bar{w}_{0j}$ 表示偏权值, 其中, $n=0, 1, \cdots, N$ 和 $j=1, \cdots, L$, 记 $\bar{\boldsymbol{W}}_j=(\bar{w}_{0j}, \bar{w}_{1j}, \cdots, \bar{w}_{Nj})^T$; $\bar{\boldsymbol{O}}_p=(\bar{o}_{p1}, \cdots, \bar{o}_{pL})^T$ 是隐层相应于 $\boldsymbol{X}_p$ 的线性输出矢量: $\bar{\boldsymbol{Y}}_p=(\bar{y}_{p1}, \cdots, \bar{y}_{pL})^T$ 是隐层的输出矢量: $\bar{y}_{pj}=f(\bar{o}_{pj})$; w_{jm} 是隐层第 j 个单元到输出层第 m 个单元间的连接权值, w_{0m} 是偏权值, 记 $\boldsymbol{W}_m=(w_{0m}, w_{1m}, \cdots, w_{Lm})^T$, $j=0, 1, \cdots, L$ 和 $m=1, \cdots, M$; $\boldsymbol{Y}_p=(y_{p1}, \cdots, y_{pM})^T$ 是输出层相应于 $\boldsymbol{X}_p$ 的网络输出; $\boldsymbol{O}_p=(o_{p1}, \cdots, o_{pM})^T$ 的输出层的线性输出: $o_{pm}=f^{-1}(y_{pm})$, 其中 f^{-1} 是 f 的反函数.

此外还设 $\boldsymbol{X}=(\boldsymbol{X}_1, \cdots, \boldsymbol{X}_P)$, $\boldsymbol{Y}=(\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_P)$, $\boldsymbol{W}=(\boldsymbol{W}_1, \cdots, \boldsymbol{W}_M)$, $\bar{\boldsymbol{W}}=(\bar{\boldsymbol{W}}_1, \cdots, \bar{\boldsymbol{W}}_L)$, $\boldsymbol{O}=(\boldsymbol{O}_1, \cdots, \boldsymbol{O}_P)$ 等; 并记 $\bar{\boldsymbol{X}}_p=(-1, \boldsymbol{X}_p^T)^T$, $\bar{\boldsymbol{X}}=(\bar{\boldsymbol{X}}_1, \cdots, \bar{\boldsymbol{X}}_P)$.

神经网络的一个重要性质是它具有学习能力. 有监督学习是前向神经网络的一种常见形式. 下面设有限的训练样本对集为 $\boldsymbol{S}=\{(\boldsymbol{X}_1, \boldsymbol{D}_1), \cdots, (\boldsymbol{X}_P, \boldsymbol{D}_P)\}$, 其中, $\boldsymbol{D}_p=(d_{p1}, \cdots, d_{pM})^T$ 表示相应于第 p 个输入矢量 $\boldsymbol{X}_p$ 的期望输出矢量.

自从 1986 年 Rumelhart 等人提出误差反传学习算法或 BP 法[1]以来, 前向神经网络学习算法的研究得到了很大发展. 目前常见的学习算法可分为两类: 一类是下降寻优算法, 包括著名的 BP 法和共轭梯度法等. 但下降算法存在局部极小、训练速度慢等不足. 另一类是分层优化算法或 OLL 法[2~4]. 它通常具有两个特征: 一是对各层的权值分别相互独立或一层层的进行优化; 二是每层的权值优化都可简化为解线性方程组. 研究表明 OLL 法能有效降低误差和加快收敛速度等. 一种有效

收稿日期: 2000-11-13; 修回日期: 2001-03-03
基金项目: 国家自然科学基金(No. 69772027); 教育部骨干教师培养计划及广东省"千百十"人才工程基金资助

的OLL法需要解决好以下两个问题[4]：一是各层目标函数的构造；二是线性方程组的求解.

关于目标函数的构造，文献[2]采用各层神经元的线性输出与期望线性输出之差的平方和来构造，但它在确定期望输出时常超出特性函数的取值范围，而易使训练失败；文献[3]在构造隐层的目标函数时，它是将特性函数作线性近似，再通过引入罚项来控制这种线性误差的大小，它要求权值的改变量也较小. 文献[4]指出了这一点并不是必要的，它在构造各层的目标函数时是将网络的输出看成各层权值的非线性算子，利用泰勒展式中的线性部分作近似代替，并将罚项定义为二次项绝对值的平均值. 这样就要计算非线性算子的Hessian矩阵，计算量很大，收敛速度慢.

为了避免上述问题，本文提出一种新的分层线性优化算法，其特点为：(1)输出层目标函数的构造与文献[2, 4]有明显不同，且更易计算；(2)隐层的目标函数构造较文献[3, 4]更加简单，无须计算 Hessian 矩阵，这样可减少计算量，加快算法收敛速度.

2 一种新的 OLL 算法

不失一般性，下面只就三层网络给出算法，并设 $h=f$. 当 h 为线性时，可类似推导.

对于三层神经网络，总误差函数通常选为如下形式

$$E=\frac{1}{2PM}\sum_{p=1}^{P}\sum_{m=1}^{M}(d_{pm}-y_{pm})^2=\frac{1}{2PM}\sum_{p=1}^{P}\sum_{m=1}^{M}(d_{pm}-f(\bar{\boldsymbol{Z}}_p^T\boldsymbol{W}_m))^2 \tag{1}$$

其中，$\bar{\boldsymbol{Z}}_p=(-1,\bar{\boldsymbol{Y}}_p^T)^T=(-1,\bar{y}_{p1},\cdots,\bar{y}_{pL})^T$ 和 $\boldsymbol{W}_m=(w_{0m},\boldsymbol{V}_m^T)^T$，$\boldsymbol{V}_m=(w_{1m},\cdots,w_{Lm})^T$.

新算法的过程是由以下两步交替进行：第一步是输出层权值的优化；第二步是隐层权值的优化.

2.1 输出层权值的优化方法

设 $\boldsymbol{A}_p=(a_{p1},\cdots,a_{pM})^T$，$a_{pi}=f^{-1}(d_{pi})$，并记 $\boldsymbol{A}=(A_1,\cdots,A_p)$. 在调整输出层权值时，将隐层权值当成固定的常数. 由微分学的中值定理，可得

$$y_{pi}-d_{pi}=f(\boldsymbol{o}_{pi})-f(\boldsymbol{a}_{pi})=f'(\xi_{pi})(\boldsymbol{o}_{pi}-\boldsymbol{a}_{pi}) \tag{2}$$

其中，$\xi_{pi}\in(a_{pi},o_{pi})$ 或 (o_{pi},a_{pi}). 因此，由式(2)和特性函数的连续可导性，可得

$$E=\frac{1}{2PM}\sum_{p=1}^{P}\sum_{i=1}^{M}[(f'(\boldsymbol{a}_{pi})(\boldsymbol{a}_{pi}-\boldsymbol{o}_{pi}))]^2+o\left(\sum_{p=1}^{P}\sum_{i=1}^{M}(\boldsymbol{a}_{pi}-\boldsymbol{o}_{pi})^2\right)=\bar{\boldsymbol{E}}+\boldsymbol{E} \tag{3}$$

其中，$o(\cdot)$ 表示高阶无穷小. 参照文献[3, 4]中的方法，可通过引入罚项来控制 $\boldsymbol{a}_{pi}$ 和 $\boldsymbol{o}_{pi}$ 的差异大小的方法来构造目标函数. 考虑到式(3)，将罚项定义为

$$\tilde{\boldsymbol{E}}=\frac{1}{2PM}\sum_{p=1}^{P}\sum_{i=1}^{M}(\boldsymbol{a}_{pi}-\boldsymbol{o}_{pi})^2 \tag{4}$$

因此输出层的目标函数定义为

$$E_J=\bar{\boldsymbol{E}}+\alpha\tilde{\boldsymbol{E}}=\frac{1}{2PM}\sum_{p=1}^{P}\sum_{i=1}^{M}\boldsymbol{c}_{pi}(\boldsymbol{a}_{pi}-\boldsymbol{o}_{pi})^2 \tag{5}$$

其中 $\boldsymbol{c}_{pi}=[f'(\boldsymbol{a}_{pi})]^2+\alpha$，$\alpha$，$\alpha$ 是非负常数. 显然 α 的大小将影响到 $\tilde{\boldsymbol{E}}$ 和 $\bar{\boldsymbol{E}}$ 在目标函数 E_J 中作用的大小. 本文将采用文献[3, 4]的方法来动态调整 α 的大小，以便增强训练效果.

下面将利用 E_J 作为目标函数来确定输出层的最佳权值，为此令

$$\frac{\partial E_J}{\partial\bar{W}_i}=0\Rightarrow\left(\sum_{p=1}^{P}\boldsymbol{c}_{pi}\bar{\boldsymbol{Z}}_p\bar{\boldsymbol{Z}}_p^T\right)\boldsymbol{W}_i=\sum_{p=1}^{P}\boldsymbol{c}_{pi}\boldsymbol{a}_{pi}\bar{\boldsymbol{Z}}_p,\ i=1,\cdots,M \tag{6}$$

易知线性方程组(6)的未知数向量为 $\boldsymbol{W}_i$，而系数矩阵 $\boldsymbol{Q}_i$ 和常数列向量 $\boldsymbol{U}_i$ 分别为

$$\boldsymbol{Q}_i=\left(\sum_{p=1}^{P}\boldsymbol{c}_{pi}\bar{y}_{pk}\bar{y}_{pk}\right)_{(L+1)\times(L+1)}\text{和}$$

$$\boldsymbol{U}_i=\left(\sum_{p=1}^{P}\boldsymbol{c}_{pi}\boldsymbol{a}_{pi}\bar{y}_{p0},\cdots,\sum_{p=1}^{P}\boldsymbol{c}_{pi}\boldsymbol{a}_{pi}\bar{y}_{pL}\right)^T_{(N+1)\times1}, \tag{7}$$

其中，$i=1,2,\cdots,M$ 和 $\bar{y}_{p0}=-1$. 于是，由式(6)可得

$$\boldsymbol{Q}_i\boldsymbol{W}_i=\boldsymbol{U}_i\Rightarrow\boldsymbol{W}_i^{opt}=\boldsymbol{Q}_i^{+}\boldsymbol{U}_i,\ i=1,2,\cdots,M \tag{8}$$

其中 $\boldsymbol{C}^{+}$ 表示式(6)的任意矩阵 $\boldsymbol{C}$ 的 Moore-Penrose 伪逆矩阵，$\boldsymbol{W}_i^{opt}$ 表示欧氏范数为最小的最佳权值解. 不难证明，若 $\alpha>0$，则式(8)可统一写成(见附录)

$$W^{opt}=(\bar{\boldsymbol{Z}}^{+})^T\boldsymbol{A}^T \tag{9}$$

其中，$\boldsymbol{A}=(\boldsymbol{a}_{pi})_{P\times M}^T$，$\boldsymbol{W}^{opt}=(\boldsymbol{w}_{ij}^{opt})_{(L+1)\times M}$. 和 $\bar{\boldsymbol{Z}}=(\bar{\boldsymbol{Z}}_1,\cdots,\bar{\boldsymbol{Z}}_p)$ 值得注意的是式(9)与调节因子 α 无关.

2.2 隐层权值的优化方法

若将隐层权值调整更新前的权值记为 $\bar{W}_{old}$，更新后为 $\bar{W}_{new}$，引起的隐层权值改变量记为 $\Delta\bar{W}=(\Delta\bar{w}_{nj})$，则 $\bar{w}_{new,nj}=\bar{w}_{old,nj}+\Delta\bar{w}_{nj}$，$n=0,1,\cdots,N$ 和 $j=1,\cdots,L$. 设调整后的输出层权值 $W=W^{opt}$ 保持不变，隐层权值为 $\bar{W}_{old}=(\bar{w}_{old,nj})$ 和 $\bar{W}_{new}=(\bar{w}_{new,nj})$ 所对应的实际输出分别记为 $Y_{old}=(y_{old,pm})$ 和 $Y_{new}=(y_{new,pm})$，则隐层权值调整前后的网络误差分别为

$$E_{old}=\frac{1}{2PM}\sum_{p=1}^{P}\sum_{m=1}^{M}(e_{old,pm})^2\text{ 和}$$

$$E_{new}=\frac{1}{2PM}\sum_{p=1}^{P}\sum_{m=1}^{M}(e_{old,pm}-\Delta y_{pm})^2 \tag{10}$$

其中 $e_{old,pm}=d_{pm}-y_{old,pm}$ 和 $\Delta y_{pm}=y_{new,pm}-y_{old,pm}$，$p=1,\cdots,P$ 和 $m=1,\cdots M$.

将网络输出 y_{pm} 看成隐层权值 $\bar{W}$ 的非线性算子，并利用其泰勒展式作一次线性近似，可得

$$y_{new,pm}\approx y_{old,pm}+\nabla^T y_{pm}(\bar{\boldsymbol{W}}_{old})\cdot\Delta\bar{\boldsymbol{W}} \tag{11}$$

其中，$\bar{\boldsymbol{W}}=(\bar{w}_{01},\cdots,\bar{w}_{0L},\cdots,\bar{w}_{NL})^T$，$\nabla y_{pm}(\bar{\boldsymbol{W}}_{old})$ 表示梯度和 $\Delta\bar{\boldsymbol{W}}=(\Delta\bar{\boldsymbol{w}}_{jk})_{(N+1)L\times1}$. 因此

$$E_{new}\approx E_{lin}=\frac{1}{2PM}\sum_{p=1}^{P}\sum_{m=1}^{M}(e_{old,pm}-\nabla^T y_{pm}(\bar{\boldsymbol{W}}_{old})\Delta\bar{\boldsymbol{W}})^2 \tag{12}$$

显然，E_{lin} 和 E_{new} 的近似程度由 y_{pm} 的泰勒展式的二阶和高阶项决定的，但一般主要由二次项决定. 易知存在正常数 σ，使该二次项的绝对值满足

$$|\varepsilon_{pm}|=|0.5\Delta\bar{\boldsymbol{W}}^T\nabla^2 y_{pm}(\bar{\boldsymbol{W}}_{old})\Delta\bar{\boldsymbol{W}}|\le\sigma\cdot\Delta\bar{\boldsymbol{W}}^T\Delta\bar{\boldsymbol{W}} \tag{13}$$

其中 $\nabla^2 y_{pm}(\bar{\boldsymbol{W}}_{old})$ 表示 y_{pm} 关于隐层权值 $\bar{\boldsymbol{W}}$ 的 Hessian 矩阵. 因此只需要限制过大的 $\sigma\cdot\Delta\bar{\boldsymbol{W}}^T\Delta\bar{\boldsymbol{W}}$ 就可以控制过大的 $|\varepsilon_{pm}|$.

若将正常数 σ 合入调节因子中, 则可将罚项定义为

$$E_{pen}=(\Delta \boldsymbol{W}^{T}\cdot\Delta \boldsymbol{W})/2PM \tag{14}$$

显然以上定义的罚项没有利用Hessian矩阵. 这样在调整隐层权值时, 构造的目标函数为

$$E_{hid}=E_{lin}+\mu E_{pen} \tag{15}$$

为优化隐层的误差函数 E_{hid}, 可令 $U_{pm}=\nabla y_{pm}(\boldsymbol{W}_{old})=(u_{pm,01},\cdots,u_{pm,NL})^{T}$ 和

$$\frac{\partial E_{hid}}{\partial \Delta \boldsymbol{W}}=\frac{\partial E_{lin}}{\partial \Delta \boldsymbol{W}}+\frac{\partial E_{pen}}{\partial \Delta \boldsymbol{W}}=0=\frac{1}{PM}[(Q+\mu I)\Delta \boldsymbol{W}-H] \tag{16}$$

其中$\partial E_{pen}/\partial \Delta \boldsymbol{W}=\Delta \boldsymbol{W}/PM$ 和

$$\frac{\partial E_{lin}}{\partial \Delta \boldsymbol{W}}=\frac{1}{PM}\sum_{p=1}^{P}\sum_{m=1}^{M}(\boldsymbol{U}_{pm}^{T}\Delta \boldsymbol{W}-e_{old,pm})\cdot \boldsymbol{U}_{pm}=\frac{Q\Delta \boldsymbol{W}-H}{PM} \tag{17}$$

$$\boldsymbol{Q}=\boldsymbol{U}\boldsymbol{U}^{T}=\sum_{p=1}^{P}\sum_{m=1}^{M}\boldsymbol{U}_{pm}\boldsymbol{U}_{pm}^{T}\text{和}\boldsymbol{H}=\boldsymbol{U}\cdot e_{old}=\sum_{p=1}^{P}\sum_{m=1}^{M}\boldsymbol{U}_{pm}\cdot e_{old,pm} \tag{18}$$

因式(16)是一个有$(N+1)L$ 个未知数和$(N+1)L$ 个方程的线性方程组, 其最佳解为

$$(\boldsymbol{Q}+\mu I)\Delta \boldsymbol{W}=\boldsymbol{H}\Rightarrow\Delta \boldsymbol{W}^{opt}=(\boldsymbol{Q}+\mu I)^{+}\cdot\boldsymbol{H} \tag{19}$$

在求解方程(8)或(19)时, 文献[4]是利用 Kaczmarz 迭代方法求解, 并指出在迭代次数趋于无穷时可以得到方程的最小范数解. 而利用 Moore-Penrose 伪逆矩阵的迭代计算公式, 只需要有限次迭代就可求出方程的最小范数解. 因此, 本文的方法有可能比文献[4]的方法更快.

由此可得到一种三层网络新的 OLL 法, 其过程如下:

(1) 初始化

® 初始化隐层权值 $\boldsymbol{W}=\boldsymbol{W}(0)$, 并置初始迭代次数 $T=0$;

® 选择式(15)中的调节因子 μ 和误差门限值或上限值 ε, 选择最大的迭代次数 T_{max};

(2) 输出层权值的优化

® 增加迭代次数 $T=T+1$, 并利用 $\boldsymbol{W}$, 由式(9)计算输出层的最佳权值 W^{opt};

® 更新输出层权值 $W=W^{opt}$; 利用 $\boldsymbol{W}$ 和 W, 由式(1)计算此时的误差 $E=E(\boldsymbol{W},W)$;

(3) 隐层权值的优化

® 在当前调节因子 μ 下, 利用式(19)计算最佳隐层权值的改变量 $\Delta \boldsymbol{W}^{opt}$;

® 利用权值 $\boldsymbol{W}^{test}=\boldsymbol{W}+\Delta \boldsymbol{W}^{opt}$ 和 W, 由式(1)计算网络的测试误差 E^{test};

如果网络误差 $E^{test}<E$, 则

• 更新隐层权值 $\boldsymbol{W}=\boldsymbol{W}+\Delta \boldsymbol{W}^{opt}$, 并置误差 $E=E^{test}$;

• 降低罚项的影响: 置 $\mu=\mu\cdot\beta$, $0<\beta<1$, 转(4);

® 否则, 增加罚项的影响; 置 $\mu=\mu\cdot\gamma$, $\gamma>1$; 重新开始(3);

(4) 学习过程的终止

® 如果 $T<T_{max}$ 且 $E>\varepsilon$, 则转(2); 否则

® 保存网络权值为 $\boldsymbol{W}^{*}=\boldsymbol{W}$ 和 $W^{*}=W$, 学习过程结束.

在以上的学习过程中, 各种参数可选为固定的值, 如 $\mu=0.1$, $\beta=0.9$ 和 $\gamma=1.2$ 等.

3　仿真实验

下面将新算法与传统的几种算法进行比较, 以此来说明新算法优越之处. 用于训练的两个问题分别为: (1)编码问题; (2)函数逼近和推广能力问题.

3.1　编码问题的仿真

考虑 4 阶编码问题, 其 16 个输入和相应的输出矢量由表 1 给出. 选择 4- 6- 4 神经网络对上述问题进行训练, 其学习的效果见表 2. 由表 2 可知, 三种算法都可成功用于上述 4 阶编码问题的训练, 但新算法的训练时间比 BP 法减少了 49~68%, 比含势态因子的 BP 法或 DBD 法[5]减少了 5~ 25%.

表 1　4阶编码的输入与相应的期望输出矢量

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}
输入矢量	1	1	1	1	-1	1	1	-1	1	-1	-1	1	-1	-1	-1	-1
	1	1	1	-1	1	1	-1	1	-1	1	-1	-1	1	-1	-1	-1
	1	1	-1	1	1	-1	1	1	-1	-1	1	-1	-1	1	-1	-1
	1	-1	1	1	1	-1	-1	-1	1	1	1	-1	-1	-1	1	-1
输出矢量	1	1	-1	1	1	-1	1	-1	1	-1	-1	1	-1	-1	1	-1
	1	-1	1	1	-1	1	-1	-1	1	1	-1	1	-1	1	-1	-1
	-1	1	1	1	-1	-1	1	1	-1	-1	1	1	-1	1	-1	-1
	1	-1	1	1	1	1	1	1	-1	-1	-1	-1	1	1	-1	-1
	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12}	d_{13}	d_{14}	d_{15}	d_{16}

表 2　4 阶编码的训练结果

	门限值 ε	学习率 η	势态率 μ	迭代次数 k	时间 t(秒)	总误差 E
BP 法	$\varepsilon_1=0.00005$	$\eta=0.3$	无	$k_1=123$	$t_1=8.5$	$E_1=0.0109$
	$\varepsilon_2=0.00005$	$\eta=0.2$		$k_2=183$	$t_2=12.3$	$E_2=0.0111$
DBD 法	$\varepsilon_1=0.00005$	$\eta=0.3$	$\mu_1=0.4$	$k_1=105$	$t_1=4.5$	$E_1=0.0091$
	$\varepsilon_2=0.00005$	$\eta=0.2$	$\mu_2=0.5$	$k_2=121$	$t_2=5.2$	$E_2=0.0098$
新算法	$\beta=0.82$ $\gamma=1.2$ $T_{max}=40$			$k=38$	$t=4.3$	$E=0.0104$
	$\beta=0.8$ $\gamma=1.25$ $T_{max}=36$			$k=34$	$t=3.9$	$E=0.0108$

表 3　函数 $\sin x$ 和 $\sin x\cos 2x$ 逼近的训练结果

	$\sin(x)$	$\sin(x)\cos(2x)$	
BP 法	$\varepsilon=0.0000001$ $\eta=0.065$	$\varepsilon=0.0000001$ $\eta=0.08$	参数
	$T=10000$	$T=10000$	迭代次数
	$t=175$ 秒	$t=386$ 秒	时间
	$E=0.004544$	$E=0.04692$	总误差
DBD 法	$\varepsilon=10^{-7}$ $\eta=0.065$ $\mu=0.4$	$\varepsilon=10^{-7}$ $\eta=0.008$ $\mu=0.3$	参数
	$T=10000$	$T=10000$	迭代次数
	$t=158$ 秒	$t=385$ 秒	时间
	$E=0.002033$	$E=0.04505$	总误差
OLL 法	$\beta=0.9$ $\gamma=1.4$ $T_{max}=40$	$\beta=0.9$ $\gamma=1.2$ $T_{max}=90$	参数
	$T=1$	$T=2$	迭代次数
	$t=0.172$ 秒	$t=0.579$ 秒	时间
	$E=0.0000003$	$E=0.00079$	总误差
新算法	$\beta=0.9$ $\gamma=1.4$ $T_{max}=40$	$\beta=0.9$ $\gamma=1.2$ $T_{max}=90$	参数
	$T=1$	$T=4$	迭代次数
	$t=0.036$ 秒	$t=0.198$ 秒	时间
	$E=0.0000003$	$E=0.00073$	总误差

3.2　函数逼近和推广能力的仿真

下面对函数 $y=\sin(x)$ 和 $y=\sin(x)\cdot\cos(2x)$ 进行逼近, x

$\in(0, 2\pi)$, 分别利用 1- 7- 1 网和 1- 10- 1 网来逼近这两个函数, 并且在定义域$[0, 2\pi]$上分别均匀地取 12 和 20 个样本点来训练网络, 其训练结果可参见表 3, 其中的 OLL 法是指文献[4]中的分层优化法.

表 4 各种差异限下的合格率

差异上限		0.005	0.01	0.05	0.1	0.15	0.2
	函数名	合格率					
BP 法	$\sin(x)$	4.2%	7.5%	49.2%	79.2%	89.2%	99.2%
	$\sin(x)\cos(2x)$	1.5%	2.5%	14%	33%	40.5%	51%
DBD 法	$\sin(x)$	5%	10.8%	64.2%	90%	100%	100%
	$\sin(x)\cos(2x)$	3%	4.5%	21.5%	34.5%	42.5%	52%
OLL 法	$\sin(x)$	99.5%	99.96%	100%	100%	100%	100%
	$\sin(x)\cos(2x)$	8.1%	17.2%	73.6%	95.5%	99.1%	99.8%
新算法	$\sin(x)$	99.6%	99.96%	100%	100%	100%	100%
	$\sin(x)\cos(2x)$	9.2%	18.8%	85%	97.5%	99.2%	99.9%

为检验用新算法训练的网络的推广能力, 在函数 $y=\sin(x)$ 和 $y=\sin(x)\cdot\cos(2x)$ 的定义域$[0, 2\pi]$上分别均匀地取 120 和 200 样本点, 让它们通过所得的网络, 则在各种差异上限下, 这些样本点的合格率可见表 4, 差异上限是指每点值与网络输出值之差绝对值的上界.

由表 3- 4 可知, 在逼近正弦函数 $y=\sin(x)$ 时, 无论在学习时间、迭代次数、误差大小和推广能力等方面, 新算法的学习效果都要明显优于 BP 法和 DBD 法; 而在逼近 $y=\sin(x)\cdot\cos(2x)$ 时, 从合格率方面看, BP 法和 DBD 法基本上训练失败, 新算法的训练效果却是成功的. 由表 3- 4 还可知, 虽然新算法可能比 OLL 法的迭代次数要多一些, 但整个训练时间却减少了 79% 或 66% .

4 结论

本文提出了一种三层前向神经网络快速 OLL 算法, 并用仿真实验说明了新算法在一些方面要优于现有的算法, 是一种切实可行的学习方法. 类似地可以得到三层以上前向网络的相应算法.

附录:

令对角矩阵 $\boldsymbol{C}_i=\mathrm{diag}(c_{1i}, 2_{2i}, \cdots, c_{pi})$ 和 $\overline{\boldsymbol{A}}_i=(a_{1i}, a_{2i}, \cdots, a_{pi})^T$, 则利用 $\overline{\boldsymbol{Z}}=(\overline{Z}_1, \cdots, \overline{Z}_P)$ 和式(7)不难计算出 $\boldsymbol{Q}_i=\overline{\boldsymbol{Z}}\boldsymbol{C}_i\overline{\boldsymbol{Z}}^T$ 和 $\boldsymbol{U}_i=\overline{\boldsymbol{Z}}\boldsymbol{C}_i\overline{\boldsymbol{A}}_i$. 设 $\widetilde{\boldsymbol{C}}_i=\mathrm{diag}(\sqrt{c_{1i}}, \sqrt{c_{2i}}, \cdots, \sqrt{c_{Pi}})$, $\widetilde{\boldsymbol{Z}}_i=\overline{\boldsymbol{Z}}\widetilde{\boldsymbol{C}}_i$ 和 $\widetilde{\boldsymbol{A}}_i=\widetilde{\boldsymbol{C}}_i\overline{\boldsymbol{A}}_i=(\sqrt{c_{1i}}a_{1i}, \sqrt{c_{2i}}a_{2i}, \cdots, \sqrt{c_{Pi}}a_{Pi})^T$, 则

$$\boldsymbol{Q}_i=\overline{\boldsymbol{Z}}\boldsymbol{C}_i\overline{\boldsymbol{Z}}^T=\widetilde{\boldsymbol{Z}}_i\widetilde{\boldsymbol{Z}}_i^T \text{ 和 } \boldsymbol{U}_i=\overline{\boldsymbol{Z}}\boldsymbol{C}_i\overline{\boldsymbol{A}}^T=\widetilde{\boldsymbol{Z}}_i\widetilde{\boldsymbol{A}}_i \qquad (a)$$

若选择调节因子 $\alpha>0$, 则对一切 p 和 i, 都有 $c_{pi}>0$. 此时方程组(8)等价于

$$W_i^{opt}=\boldsymbol{Q}_i^{+}\boldsymbol{U}_i=(\widetilde{\boldsymbol{Z}}_i\widetilde{\boldsymbol{Z}}_i^T)^{+}\widetilde{\boldsymbol{Z}}_i\widetilde{\boldsymbol{A}}_i=(\widetilde{\boldsymbol{Z}}_i^{+})^T\widetilde{\boldsymbol{A}}_i=(\overline{\boldsymbol{Z}}_i^{+})^T\overline{\boldsymbol{A}}_i,\ i=1, 2, \cdots, M \qquad (b)$$

上式中利用了等式: $\widetilde{\boldsymbol{Z}}_i^{+}=(\overline{\boldsymbol{Z}}\widetilde{\boldsymbol{C}}_i)^{+}=\widetilde{\boldsymbol{C}}_i^{-1}\overline{\boldsymbol{Z}}^{+}$. 不难发现, 式(b)可统一写成式(9).

参考文献:

[1] Rumelhart E E, Hinton G E, Williams R J. Learning Internal Representations by Error Propagation [M]. Cambridge, MA: MIT Press, 1986.

[2] Wang G J, Chen C C. A fast multilayer neural network training algorithm based on the layer-by-layer optimizing procedures [J]. IEEE Trans. Neural Networks, 1996, 7(3): 768- 775.

[3] Ergezinger S, Thomsen E. An accelerated learning algorithm for multilayer perceptrons: Optimization layer by layer [J]. IEEE Trans. Neural Networks, 1995, 6(1): 31- 42.

[4] Rubanov N. The layer-wise method and the backpropagation hybrid approach to learning a feedforward neural network [J]. IEEE Trans. Neural Networks, 2000, 11(2): 295- 305.

[5] Jacobs R A. Increased rates of convergence through learning rate adaptation [J]. Neural Networks. 1988, 1: 295- 307.

[6] 潘维民, 沈理. 基于神经网络的时间序列动态预测器的调整学习算法 [J]. 电子学报, 1999, 27(11): 1- 3.

[7] 阎平凡. 对多层神经网络研究的进一步看法 [J]. 电子学报, 1999, 27(5): 82- 85.

[8] 陈国良, 韩文廷. 人工神经网络理论研究进展 [J]. 电子学报, 1996, 24(1): 70- 75.

[9] 韦岗, 贺前华. 神经网络模型、学习及应用 [M]. 电子工业出版社, 1994.

作者简介:

田传俊 男. 1964 年 5 月出生于湖北荆州. 副教授, 现为华南理工大学电子信息学院博士研究生. 研究兴趣包括神经网络、信号处理、微分和差分方程稳定性等.

韦 岗 男. 1963 年出生于广西宾阳. 教授, 博士生导师. 国家自然科学基金电子与信息学科评委; 国际学术刊物"Real Time Systems"(美国) Associate Editor; 研究兴趣包括通信、信号处理、神经网络等.