文章编号: 1001-0920(2001)05-0627-03

神经网络的新型二阶学习算法及其应用

刘铁男, 段玉波, 于 镝, 刘志德, 张长江
(大庆石油学院 自动控制系, 黑龙江 安达 151400)

摘 要: 针对 BP 算法和 Karayiannis 的二阶学习算法存在的不足, 提出多层前向网络的新型二阶学习算法。该算法具有二阶收敛速度, 其计算量与通常的递推最小二乘法相当。算法性能分析和仿真应用表明新算法是有效的。

关键词: 多层前向网络; 二阶学习算法; Newton 算法
中图分类号: TP 18　　文献标识码: A

New Second-order Learning Algorithm for Neural Networks and Its Application

LIU Tie-nan, DUAN Yu-bo, YU Di, LIU Zhi-de, ZHANG Chang-jiang
(Department of Automatic Control, Daqing Petroleum Institute, Anda 151400, China)

Abstract: A new second-order recursive learning algorithm is presented. The algorithm is equivalent to Newton iterative method and has second-order convergent speed. It achieves the recurrence calculation of Newton search divections and the inverse of Hessian matrices. The analysis and application results show the efficiency of the new algorithm.

Key words: multilayer feedforward networks; second-order learning algorithm; Newton algorithm

1 引　　言

对学习算法的研究是神经网络(NN) 研究的重要问题之一。目前对多层前向 NN, 主要应用 BP 算法估计其权值, 而 BP 算法是一阶算法, 收敛速度较慢。为此, Karayiannis 等[1,2] 提出一种二阶学习算法, 即递推近似Newton 法。文献[1] 对两层网(或多层网络的输出层) 给出如下二阶算法

$$\boldsymbol{P}_{ik} = \boldsymbol{P}_{ik-1} - \delta_{ik}\boldsymbol{P}_{ik-1}\boldsymbol{X}_k\boldsymbol{X}_k^{\mathrm{T}}\boldsymbol{P}_{ik-1} \tag{1}$$

$$\begin{cases} \delta_{ik} = c_{ik}[1 + c_{ik}\boldsymbol{X}_k^{\mathrm{T}}\boldsymbol{P}_{ik-1}\boldsymbol{X}_k]^{-1} \\ c_{ik} = [(\tanh(\bar{y}_{ik}))]^2 = [1 - \hat{y}_{ik}^2]^2 \end{cases} \tag{2}$$

$$\boldsymbol{W}_{ik} = \boldsymbol{W}_{ik-1} + \alpha\epsilon_{ik}\boldsymbol{P}_{ik}\boldsymbol{X}_k \tag{3}$$

文献[1] 算法对 Hessian 阵的递推公式做了若干简化, 丢失了目标函数二阶导数的部分有用信息, 这对算法性能有一定的影响。另外由于算法的推导不彻底, 致使算法的计算量仍比较大。针对文献[1] 存在的问题, 本文首先提出并证明了一种不需任何简化的递推 Newton 算法, 它与 Newton 法等价, 具有二阶收敛速度; 然后分析了新算法的性能, 并给出了新算法的仿真实例。

2 递推 Newton 学习算法

为阐述方便, 本文与文献[1] 一样, 考虑单输出

收稿日期: 2000-06-09; 修回日期: 2001-05-28
基金项目: 黑龙江省自然科学基金项目(F9812)
作者简介: 刘铁男(1945—), 男, 黑龙江青岗人, 教授, 从事非线性系统辨识、预测、滤波和控制等研究; 段玉波(1951—), 男, 黑龙江木兰人, 教授, 从事电气自动化的研究。

两层NN(或多层网络的输出层),但结果可推广到NN的其它层。假设训练样本为$(\boldsymbol{X}_k, y_k)$, $k = 1, 2, \cdots, m$, $\boldsymbol{X}_k$和y_k分别为输入向量和期望输出。$\boldsymbol{X}_k = (x_{0k}, x_{1k}, \cdots, x_{nk})^{\tau}$, $x_{0k} = 1$。NN的输入输出关系为

$$\hat{y}_k = \sigma(\bar{y}_k), \quad \bar{y}_k = \boldsymbol{W}_k^{\tau}\boldsymbol{X}_k \tag{4}$$

其中$\sigma(\cdot)$是神经元作用函数。目标函数为

$$\begin{cases} E_m = \dfrac{1}{2}\sum_{k=1}^{m} e_k^2 = E_{m-1} + \dfrac{1}{2}e_m^2 \\ e_k = y_k - \hat{y}_k \end{cases} \tag{5}$$

由最优化理论[3]知,对目标函数(5)的Newton迭代法如下

$$\begin{cases} \boldsymbol{W}_m = \boldsymbol{W}_{m-1} + \Delta\boldsymbol{W}_m = \\ \boldsymbol{W}_{m-1} - \boldsymbol{H}^{-1}(m)\nabla E(m) \\ \Delta\boldsymbol{W}_m = -\boldsymbol{H}^{-1}(m)\nabla E(m) = \\ -\boldsymbol{H}^{-1}(m)\left.\dfrac{\partial E_m}{\partial \boldsymbol{W}_m}\right|_{\boldsymbol{W}=\boldsymbol{W}_{m-1}} \end{cases} \tag{6}$$

由式(4)和(5)可知

$$\frac{\partial e_k}{\partial \boldsymbol{W}} = -\frac{\partial \hat{y}_k}{\partial \boldsymbol{W}} = -\sigma'(\bar{y}_k)\frac{\partial \bar{y}_k}{\partial \boldsymbol{W}} = -\sigma'(\bar{y}_k)\boldsymbol{X}_k \tag{7}$$

由式(5)和(7)得

$$\nabla E(m) = \nabla E(m-1) - e_m\sigma'(\bar{y}_m)\boldsymbol{X}_m = \nabla E(m-1) - \epsilon_m\boldsymbol{X}_m \tag{8}$$

其中$\epsilon_m = e_m\sigma'(\bar{y}_m)$是输出层局部误差。对式(5)求$E_m$的Hessian阵,并注意式(8),得

$$\boldsymbol{H}(m) = \frac{\partial}{\partial \boldsymbol{W}}\left[\frac{\partial E_m}{\partial \boldsymbol{W}}\right]^{\tau} = \sum_{k=1}^{m} z_k\boldsymbol{X}_k\boldsymbol{X}_k^{\tau} = \boldsymbol{H}(m-1) + z_m\boldsymbol{X}_m\boldsymbol{X}_m^{\tau} \tag{9}$$

$$z_k = (\sigma'(\bar{y}_k))^2 - e_k\sigma''(\bar{y}_k) \tag{10}$$

为了推导递推Newton算法,现给出如下引理:

引理1 设P为$n\times n$可逆阵,$\boldsymbol{X}$为$n\times 1$向量,z为非零标量,则有矩阵求逆公式

$$(\boldsymbol{P}^{-1} + z\boldsymbol{X}\boldsymbol{X}^{\tau})^{-1} = \boldsymbol{P} - \boldsymbol{P}\boldsymbol{X}(z^{-1} + \boldsymbol{X}^{\tau}\boldsymbol{P}\boldsymbol{X})^{-1}\boldsymbol{X}^{\tau}\boldsymbol{P}$$

证明 直接计算可知$(\boldsymbol{P}^{-1} + z\boldsymbol{X}\boldsymbol{X}^{\tau})(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{X}(z^{-1} + \boldsymbol{X}^{\tau}\boldsymbol{P}\boldsymbol{X})^{-1}\boldsymbol{X}^{\tau}\boldsymbol{P}) = \boldsymbol{I}$,其中$\boldsymbol{I}$为单位阵。(证毕)

定理1 估计上述NN权值的如下递推算法与Newton迭代法(6)等价,并具有二阶收敛速度。

$$\begin{cases} \boldsymbol{W}_m = \boldsymbol{W}_{m-1} + \Delta\boldsymbol{W}_m \\ \Delta\boldsymbol{W}_m = \Delta\boldsymbol{W}_{m-1} + \boldsymbol{M}_m[\beta_m\epsilon_m - \boldsymbol{X}_m^{\tau}\Delta\boldsymbol{W}_{m-1}] \end{cases} \tag{11}$$

$$\beta_m = 1/z_m = 1/[(\sigma'(\bar{y}_m))^2 - \sigma''(\bar{y}_m)e_m] \tag{12}$$

$$\boldsymbol{M}_m = \boldsymbol{P}_{m-1}\boldsymbol{X}_m/[\beta_m + \boldsymbol{X}_m^{\tau}\boldsymbol{P}_{m-1}\boldsymbol{X}_m] \tag{13}$$

$$\boldsymbol{P}_m = \boldsymbol{P}_{m-1} - \boldsymbol{M}_m\boldsymbol{X}_m^{\tau}\boldsymbol{P}_{m-1} \tag{14}$$

证明 令$\boldsymbol{P}_m = \boldsymbol{H}^{-1}(m)$,由式(9)和引理1得

$$\boldsymbol{P}_m = \boldsymbol{H}^{-1}(m) = [\boldsymbol{P}_{m-1}^{-1} + z_m\boldsymbol{X}_m\boldsymbol{X}_m^{\tau}]^{-1} = \boldsymbol{P}_{m-1} - \boldsymbol{M}_m\boldsymbol{X}_m^{\tau}\boldsymbol{P}_{m-1} \tag{15}$$

$$\boldsymbol{M}_m = \boldsymbol{P}_{m-1}\boldsymbol{X}_m/[\beta_m + \boldsymbol{X}_m^{\tau}\boldsymbol{P}_{m-1}\boldsymbol{X}_m] \tag{16}$$

式(15)两边右乘以$\boldsymbol{X}_m$,得

$$\boldsymbol{P}_m\boldsymbol{X}_m = \boldsymbol{P}_{m-1}\boldsymbol{X}_m - \boldsymbol{M}_m\boldsymbol{X}_m^{\tau}\boldsymbol{P}_{m-1}\boldsymbol{X}_m \tag{17}$$

式(16)两边同时乘以该式右边的分母,经整理得

$$\beta_m\boldsymbol{M}_m = \boldsymbol{P}_{m-1}\boldsymbol{X}_m - \boldsymbol{M}_m\boldsymbol{X}_m^{\tau}\boldsymbol{P}_{m-1}\boldsymbol{X}m \tag{18}$$

由式(17)和(18)知

$$\boldsymbol{P}_m\boldsymbol{X}_m = \beta_m\boldsymbol{M}_m \tag{19}$$

将式(8)和(15)代入式(6),并应用式(19)得

$$\Delta\boldsymbol{W}_m = (\boldsymbol{I} - \boldsymbol{M}_m\boldsymbol{X}_m^{\tau})\Delta\boldsymbol{W}_{m-1} + \beta_m\boldsymbol{M}_m\epsilon_m \tag{20}$$

$$\Delta\boldsymbol{W}_m = \Delta\boldsymbol{W}_{m-1} + \boldsymbol{M}_m[\beta_m\epsilon_m - \boldsymbol{X}_m^{\tau}\Delta\boldsymbol{W}_{m-1}] \tag{21}$$

Newton迭代法有二阶收敛速度[3],上述推导已证明了这两种算法是等价的,故递推Newton法也具有二阶收敛速度。(证毕)

在上述算法中,β_m和z_m与网络神经元作用函数形式有关,基于式(10)和$\epsilon_m = e_m\sigma'(\bar{y}_m)$,对常用作用函数有

$$\beta_m = \frac{1}{z_m} = \begin{cases} 1/[\hat{y}_m^2(1-\hat{y}_m)^2 - (1-2\hat{y}_m)\epsilon_m] \\ \hat{y}_m = 1/[1+\exp(-\bar{y}_m)] \\ 1/[(1-\hat{y}_m^2)^2 + 2\hat{y}_m\epsilon_m] \\ \hat{y}_m = \tanh(\bar{y}_m) \\ 4/[(1-\hat{y}_m^2)^2 + 4\hat{y}_m\epsilon_m] \\ \hat{y}_m = \dfrac{1-\exp(-\bar{y}_m)}{1+\exp(-\bar{y}_m)} \\ 0.5/[2\hat{y}_m^2(1-\hat{y}_m)^2 - (1-2\hat{y}_m)\epsilon_m] \\ \hat{y}_m = \dfrac{1}{2}[1+\tanh(\bar{y}_m)] \end{cases} \tag{22}$$

式中tanh(·)表示双曲正切函数。

3 算法性能分析

新算法在式(14)和(11)中分别实现了E_m的Hessian阵的逆和Newton搜索方向的递推运算。由式(20)知,其第1项为搜索方向的"惯性"项,第2项为当前搜索方向。与文献[1]算法相比,本文算法未做任何简化。文献[1]的第1次简化,相当于令式(8)中的$\nabla E(m-1) = 0$,因而式(3)中的"惯性"项消失了。与带惯性项的BP算法同理[2],新算法的"惯性"项能有效地防止算法产生振荡,从而可加快收敛过程,并且此惯性项能在线自适应计算,不必人为试凑。文献[1]的另一简化等效于令式(9)中的$e_k\frac{\partial}{\partial \boldsymbol{W}}\left[\frac{\partial e_k}{\partial \boldsymbol{W}}\right]^{\tau} = 0$,所以其算法仅用到$E_m$的一阶导数.

信息, 不是真正的二阶算法。而新算法充分利用了 E_m 的一阶和二阶导数的全部信息, 因此是真正的二阶算法。

从计算量上看, 新算法比文献[1] 算法的计算量小。这主要反映在式(11) 和(3) 上(其它公式计算量相当)。令 $v = n + 1$, 则 $\boldsymbol{P}_m$ 是 $v \times v$ 方阵, $\boldsymbol{X}_m$ 是 $v \times 1$ 向量, 式(3) 中 $\epsilon_m \boldsymbol{P}_m \boldsymbol{X}_m$ 要做 $(v + 1)v$ 次乘法, 而式(11) 仅做 $2v$ 次乘法。显然, 新算法的计算量略大于通常的递推最小二乘法。

4　仿真实例

例 **1**　以如下 Logistic 模型[4] 作为仿真对象

$$y(k) = 1/[a^{-1} + (y_0^{-1} - a^{-1}) \times \exp(-b(k - k_0))] + e(k) \quad (23)$$

例 **2**　以如下 Gompertz 模型[4] 作为仿真对象

$$y(k) = a\exp\left\{\ln\frac{y_0}{a}\exp\frac{-b(k-k_0)}{\ln(a)}\right\} + e(k) \quad (24)$$

在式(23) 和(24) 中, 初始条件为: 当 $k = k_0$ 时, $y(k) = y_0$, a 和 b 是参数, 并且 a 是 $y(k)$ 的极限值。仿真时, 例 1 中取 $a = 50, b = 0.15, k_0 = 1, y_0 = 2$; 例 2 中取 $a = 60, b = 0.36, k_0 = 1, y_0 = 3$。式(23) 和(24) 中附加了幅值为 a 的 1% 的白噪声 $e(k)$, 目的是能分辨出 $y(k)$ 及其计算值。

基于式(23) 或(24) 可生成训练样本对 $\{X(k), y(k)\}$, $X(k)$ 是 n 维输入向量。当 $k \leq n$ 时有 $X^{\tau}(1) = [y_0, 0, \cdots, 0], \cdots, X^{\tau}(n) = [y(n-1), \cdots, y(1), y_0]$, 当 $k > n$ 时, 有 $X^{\tau}(k) = [y(k-1), \cdots, y(k-n)]$。采用单输出三层前向网来逼近上述仿真系统。网络结构为: 例 1 中输入、隐层和输出层神经元个数为 5-10-1; 例 2 中为 6-15-1。网络训练采用混合式学习算法, 对输出层分别用本文或文献[1] 的二阶学习算法估计权值(分别称为方案 1 和方案 2); 对隐层权值均采用带惯性项的BP 算法进行训练。隐层和输出层神经元作用函数均采用 Sigmoid 函数。两种方案的对比如表 1 所示。若只用带惯性项的BP 算法训练, 要达到与表 1 类似的精度, 需要较多的迭代次数(例 1 用 2 012 步, 例 2 用 2 562 步)。可见引入二阶算法后, 与单纯用 BP 算法相比具有更快的收敛速度。由表 1 可见, 在逼近精度相近的情况下, 本文方案计算量更小, 收敛速度更快。本文方案仿真结果如图 1 和图 2 所示, 文献[1] 仿真结果与上述结果相近, 故省略。

表 **1**　方案 **1** 与方案 **2** 对比

	例 1		例 2	
方　案	1	2	1	2
控制精度	0.015	0.015	0.01	0.01
迭代步数	225	340	238	378
运行时间(s)	36	63	38	70
平均相对误差	0.012 9	0.013 1	0.007 8	0.008 9

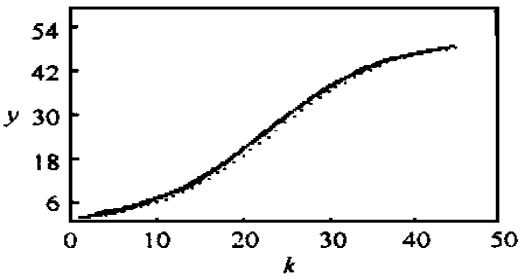

图 **1**　例 **1** 的仿真结果

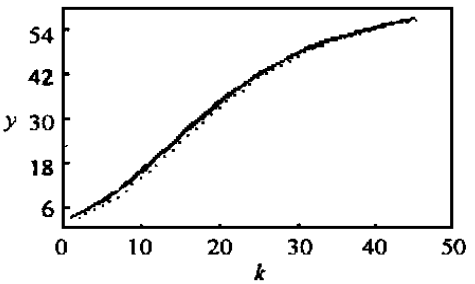

图 **2**　例 **2** 的仿真结果

5　结　　论

本文针对多层前向网络, 提出一种新型的二阶学习算法, 它与 Newton 迭代法等价, 具有二阶收敛速度, 其计算量与 RLS 法相当。算法性能分析和仿真应用表明新算法优于 Karayiannis 的二阶学习算法。

参考文献:

[1] Karayiannis N B, Venetsanopoulos A N. Artificial neural networks, learning algorithm, performance evaluation and applications[M]. New York: Kluwer Academic Publishers, 1993.

[2] 王永骥, 涂健. 神经元网络控制[M]. 北京: 机械工业出版社, 1998.

[3] 席少霖, 赵风治. 最优化计算方法[M]. 上海: 上海科学技术出版社, 1983.

[4] 陈玉详, 张汉亚. 预测技术与应用[M]. 北京: 机械工业出版社, 1985.