第 23 卷 第 6 期
1997 年 11 月

自 动 化 学 报
ACTA AUTOMATICA SINICA

Vol. 23, No. 6
Nov., 1997

一种前向神经网络快速学习算法及其在系统辨识中的应用[1)]

王正欧
（天津大学系统工程研究所 天津 300072）
林 晨
（福建亚洲银行 福州 350001）

摘 要 提出一种基于最小二乘的前向神经网络快速学习算法. 与现有同类算法相比，该算法无需任何矩阵求逆，计算量小，较适于需快速学习的系统辨识和其他应用. 文中推导了算法，并给出一种更为简便的局部化算法. 系统辨识的仿真实例表明了算法的优良性能.

关键词 前向神经网络，快速学习，系统辨识.

1 引言

自前向神经网络 BP 算法[1]提出以来，已在控制界和其他领域得到广泛应用[2,3]. 由于 BP 算法存在收敛慢和局部极小点等问题，迄今已提出大量改进算法，其中采用增广卡尔曼滤波的学习算法[4]引起了广泛地重视. 这类算法无需对学习速率和势态项系数进行猜测，且收敛速度快、精度高. 其基本思想是把网络权值作为一个相应动态系统的状态，运用增广卡尔曼滤波估计，即可得到好的效果. 然而这类算法由于维数过高且需矩阵求逆，限制了网络的规模. 多种局部化算法[5,6]的提出，虽降低了维数，但仍免不了矩阵求逆. 本文提出一种基于最小二乘的快速学习算法，无需任何矩阵求逆，其相应的局部化算法更极大地减少了计算量，提高了计算速度，较适于系统辨识等实际应用场合.

2 网络和相应的动态系统

考虑一个任意的具有一个输入层、一个输出层和若干隐层的前向网络，设此网络包含一个由全部连接权构成的权向量 $\theta \in R^M$，并认为一个单元的阈值是由单位强度输入连接到该单元的连接权. 假定网络中单元的激活函数（除输入层外）是 S 形函数 $\varphi(x)=1/(1+e^{-x})$，其中 x 为该单元的纯输入，它等于输入到该单元信号的加权和. 现在的问题是要确定 θ，使得对于一组给定的输入向量 $\boldsymbol{i}(k)\in R^n(k=1,\cdots,p)$，网络产生的实际输出向量 $\boldsymbol{o}(k)\in R^m(k=1,\cdots,p)$ 应尽可能接近期望的输出向量 $\boldsymbol{d}(k)\in R^m(k=1,\cdots,p)$. 这里考虑在线学习算法，即输入模式依次输入至输入层，权值则递推地改进. 将 θ 视为如下非线性

1） 天津市自然科学基金资助课题.
收稿日期 1995-03-13

动态系统的状态

$$\theta(k+1)=\theta(k)=\theta_0, \tag{1}$$

$$\boldsymbol{d}(k)=\boldsymbol{f}(\theta_0,\boldsymbol{i}(k))+\boldsymbol{e}(k). \tag{2}$$

这里 $\boldsymbol{f}(\theta_0,\boldsymbol{i}(k)=\boldsymbol{o}(k,\theta_0)$表示网络的输入、权值和输出关系的非线性函数，$\boldsymbol{e}(k)$为模型误差. $\boldsymbol{f}(\theta_0,\boldsymbol{i}(k))$可在当前估计值 $\hat{\theta}(k-1)$附近展开，故(1)，(2)式可改写为

$$\theta(k+1)=\theta(k)=\theta_0, \tag{3}$$

$$\boldsymbol{d}(k)=\boldsymbol{f}(\hat{\theta}(k-1),\boldsymbol{i}(k))+G^T(k)(\theta_0-\hat{\theta}(k-1))+\rho(k)+\boldsymbol{e}(k). \tag{4}$$

这里 $G(k)$是 $M\times m$ 维梯度矩阵

$$G(k)=\left.\frac{\partial \boldsymbol{f}(\theta,\boldsymbol{i}(k))}{\partial\theta}\right|_{\theta=\hat{\theta}(k-1)}; \tag{5}$$

$\rho(k)$是 $\boldsymbol{f}$ 的展开式的高阶余项. 若设

$$\xi(k)=\boldsymbol{f}(\hat{\theta}(k-1),\boldsymbol{i}(k))-G^T(k)\hat{\theta}(k-1)+\rho(k), \tag{6}$$

则(4)式可简写为

$$\boldsymbol{d}(k)=G^T(k)\theta_0+\xi(k)+\boldsymbol{e}(k). \tag{7}$$

如将网络中单元由 1 至 N 排列，并取权值向量为 $\theta=[\theta_1^T\quad\theta_2^T\cdots\theta_N^T]^T$，则(7)式可写为

$$\begin{bmatrix}d_1(k)\\d_2(k)\\\vdots\\d_m(k)\end{bmatrix}=\begin{bmatrix}\left(\frac{\partial o_1}{\partial\theta_1}\right)^T & \left(\frac{\partial o_1}{\partial\theta_2}\right)^T & \cdots & \left(\frac{\partial o_1}{\partial\theta_N}\right)^T\\ \left(\frac{\partial o_2}{\partial\theta_1}\right)^T & \left(\frac{\partial o_2}{\partial\theta_2}\right)^T & \cdots & \left(\frac{\partial o_2}{\partial\theta_N}\right)^T\\ \vdots & & & \\ \left(\frac{\partial o_m}{\partial\theta_1}\right)^T & \left(\frac{\partial o_m}{\partial\theta_2}\right)^T & \cdots & \left(\frac{\partial o_m}{\partial\theta_N}\right)^T\end{bmatrix}\begin{bmatrix}\theta_1\\\theta_2\\\vdots\\\theta_N\end{bmatrix}+\begin{bmatrix}c_1(k)\\c_2(k)\\\vdots\\c_m(k)\end{bmatrix}. \tag{8}$$

这里 θ_i 为连接到第 i 个单元的权向量，$\boldsymbol{c}(k)=\xi(k)+\boldsymbol{e}(k)$. $G(k)$中的偏导数可通过网络反传 $\boldsymbol{o}(k)$求得.

3　学习算法

3.1　全局递推最小二乘算法(GRLS)

基于模型(8)给定一个输入/输出序列对(i(j)，d(j))(j=1，…，k)，选择如下的目标函数

$$\begin{aligned}\varepsilon(\hat{\theta}(k))&=\sum_{j=1}^{k}\|\boldsymbol{d}(j)-\boldsymbol{f}(\hat{\theta}(j),\boldsymbol{i}(j))\|_2\lambda^{k-j}\\&=\sum_{j=1}^{k}\|\boldsymbol{e}(j)\|_2\lambda^{k-j}\\&=\sum_{j=1}^{k}\sum_{i=1}^{m}e_i^2(j)\lambda^{k-j}.\end{aligned} \tag{9}$$

这里‖·‖表示向量的欧几里得范数，$e_i(j)$是(8)式中第 i 个建模误差，λ 为遗忘因子 $0<\lambda\leqslant 1$，估计值 $\hat{\theta}(k)$是由使 $\varepsilon(k)$达到极小而得的. 此解可从下列正则方程推导

$$\nabla_{\theta_0}\varepsilon(k)=2\sum_{j=1}^{k}\sum_{i=1}^{m}\boldsymbol{g}_i(j)[d_i(j)-\boldsymbol{g}_i^T(j)\theta_0-\xi_i(j)]\lambda^{k-j}=0, \tag{10}$$

其中 $1\times M$ 维向量 $\boldsymbol{g}_i^T(j)$是矩阵 $G^T(j)$的第 i 行. 由(10)式可得

$$\hat{\boldsymbol{\theta}}(k) = S^{-1}(k)\boldsymbol{t}(k). \tag{11}$$

其中

$$S(k) = \sum_{j=1}^{k}\sum_{i=1}^{m}\boldsymbol{g}_i(j)\boldsymbol{g}_i^T(j)\lambda^{k-j}, \tag{12}$$

$$\boldsymbol{t}(k) = \sum_{j=1}^{k}\sum_{i=1}^{m}\boldsymbol{g}_i(j)[d_i(j) - \xi_i(j)]\lambda^{k-j}. \tag{13}$$

由(11)式可推得 GRLS 算法如下：

$$\hat{\boldsymbol{\theta}}(k) = \hat{\boldsymbol{\theta}}(k-1) + P(k)\sum_{i=1}^{m}\boldsymbol{g}_i(k)[d_i(k) - f_i(\hat{\theta}(k-1), i(k))], \tag{14}$$

$$P(k) = \lambda^{-1}P(k-1) - \lambda^{-2}P(k-1)\Big[\sum_{i=1}^{m}\boldsymbol{g}_i(k)\boldsymbol{g}_i^T(k)\Big]$$

$$\times P(k-1)\Big[1 + \lambda^{-1}\sum_{i=1}^{m}\boldsymbol{g}_i^T(k)P(k-1)\boldsymbol{g}_i(k)\Big]^{-1}. \tag{15}$$

算法初值可取 $p(0)=r^{-1}I$，r 为任一小正数，I 为单位阵，$\hat{\boldsymbol{\theta}}(0)$可为非零的随机向量.(14)，(15)式的推导见附录.

注意，上述算法要求存储和改进一个 $M\times M$ 维矩阵，但不要求任何矩阵求逆.

3.2 局部化递推最小二乘算法(LRLS)

在 GRLS 算法中 P 阵的维数相当高，仍难实用. 一种有效的简化方法是将系统分解成若干较小的子系统，例如分解到层一级、神经元一级乃至到单个连接权一级等. 这里拟在神经元一级推导 LRLS 算法.

考虑网络中第 j 个单元，假设它具有 M_j+1 维连接权向量 $\boldsymbol{\theta}_j$ 和 M_j+1 维的输入向量，$\boldsymbol{\theta}_j$ 中包含了该单元的阈值. 该单元对总输出 $\boldsymbol{O}(k)$的影响可用梯度矩阵 $G^{j^T}(k)$描述，它是 $G^T(k)$的第 j 个子块

$$G^{j^T}(k) = \begin{bmatrix} \left(\dfrac{\partial O_1}{\partial \boldsymbol{\theta}_j}\right)^T \\ \vdots \\ \left(\dfrac{\partial O_m}{\partial \boldsymbol{\theta}_j}\right)^T \end{bmatrix}. \tag{16}$$

此矩阵的第 i 行是$\left(\dfrac{\partial O_i}{\partial \boldsymbol{\theta}_j}\right)^T$，用 $\boldsymbol{g}_i^{j^T}(\boldsymbol{k})$表示. 如相对于第 $\boldsymbol{j}$ 个神经元的权值线性化 $\boldsymbol{f}$，则类似于(4)式有

$$\boldsymbol{d}(k) = \boldsymbol{f}(\hat{\boldsymbol{\theta}}(k-1), \boldsymbol{i}(k)) + G^{j^T}(k)[\boldsymbol{\theta}_j - \hat{\boldsymbol{\theta}}_j(k-1)] + \boldsymbol{\rho}_j(k) + \boldsymbol{e}(k)$$

$$= G^{j^T}(k)\boldsymbol{\theta}_j + \boldsymbol{\xi}_j(k) + \boldsymbol{e}(k). \tag{17}$$

其中

$$\boldsymbol{\xi}_j(k) = \boldsymbol{f}(\hat{\boldsymbol{\theta}}(k-1), \boldsymbol{i}(k)) - G^{j^T}(k)\hat{\boldsymbol{\theta}}_j(k-1) + \rho_j(k). \tag{18}$$

这里台劳展开仅对第 j 个单元的权值进行，其余权值采用 $k-1$ 时刻以前的估计值. 此非线性函数的线性化可对每个单元平行地进行.

对于每个单元改进权值的局部化算法类似于(14)，(15)式可得

$$\hat{\boldsymbol{\theta}}_j(k) = \hat{\boldsymbol{\theta}}_j(k-1) + P_j(k)\sum_{i=1}^{m}\boldsymbol{g}_i^j(k)[d_i(k) - f_i(\hat{\boldsymbol{\theta}}(k-1), \boldsymbol{i}(k))], \tag{19}$$

$$P_j(k) = \lambda^{-1}P_j(k-1) - \lambda^{-2}P_j(k-1)\Big[\sum_{i=1}^{m}\boldsymbol{g}_i^j(k)\boldsymbol{g}_i^{j^T}(k)\Big]$$

$$\times P_j(k-1)\Big[1+\lambda^{-1}\sum_{i=1}^{m}\boldsymbol{g}_i^{jT}(k)P_j(k-1)\boldsymbol{g}_i^j(k)\Big]^{-1},\quad j=1,\cdots,n. \tag{20}$$

注意到 LRLS 仅要求存储和改进一个$(M_j+1)\times(M_j+1)$维矩阵(随单元不同 M_j 可不同)，此矩阵维数大大小于 GRLS 中相应矩阵的维数，故 LRLS 的存储和计算量远小于 GRLS.

4　LRLS 在系统辨识中的应用

系统辨识通常和自适应控制相联系，要求快速及时地辨识以利于控制，但通常的学习算法受收敛速度限制，效果难以理想. LRLS 算法可较好地弥补这一缺陷.

为研究该算法在系统辨识中的性能，这里将它同原始 BP 算法作了仿真比较. 为实用计，只研究 LRLS 的性能. 由于前向多层网络可对任意线性和非线性函数映射. 不失一般性，这里仅研究线性系统辨识例子. 为简单计，在仿真中略去了系统模型误差，以突出说明算法的收敛性和精度. 由经验选取 $\lambda=0.995$，以利于算法收敛. 以下用 4 个例子说明新算法的优越性.

例 1. 考虑简单单变量系统

$$y(k)=0.8y(k-1)+0.2u(k),$$

这里输入 $u(k)$是在(0,1)区间中均匀分布的随机变量，网络结构为两个输入 $u(k)$和 $y(k-1)$、一个 15 单元隐层及一个输出 $y(k)$(2-15-1). 初始权值为 0 与 0.01 间均匀分布的随机值，P_j初始阵中 $r=0.001$. 原始 BP 算法中学习速率和势态项系数分别取为 0.9 和 0.3. 训练样本数为 30. 训练停止准则为 $e_i<0.05$，其中 e 表示 30 个样本中输出误差最大值，下标 i 表示迭代次数. 图 1 表示采用 LRLS 和原始 BP 算法在不同初始条件下 10 次训练的平均误差随迭代次数的变化曲线.

例 2. 考虑动态系统

$$y(k)=1.368y(k-1)-0.368y(k-2)+0.368u(k-1)+0.364u(k-2),$$

这里 $u(k)$是 0 和 1 间均匀分布随机变量，网络结构为 4-10-1，初始条件和停止准则同例 1，训练样本为 30. 图 2 表示 LRLS 和原始 BP 算法收敛速度比较，其中 BP 算法参数取法同例 1.

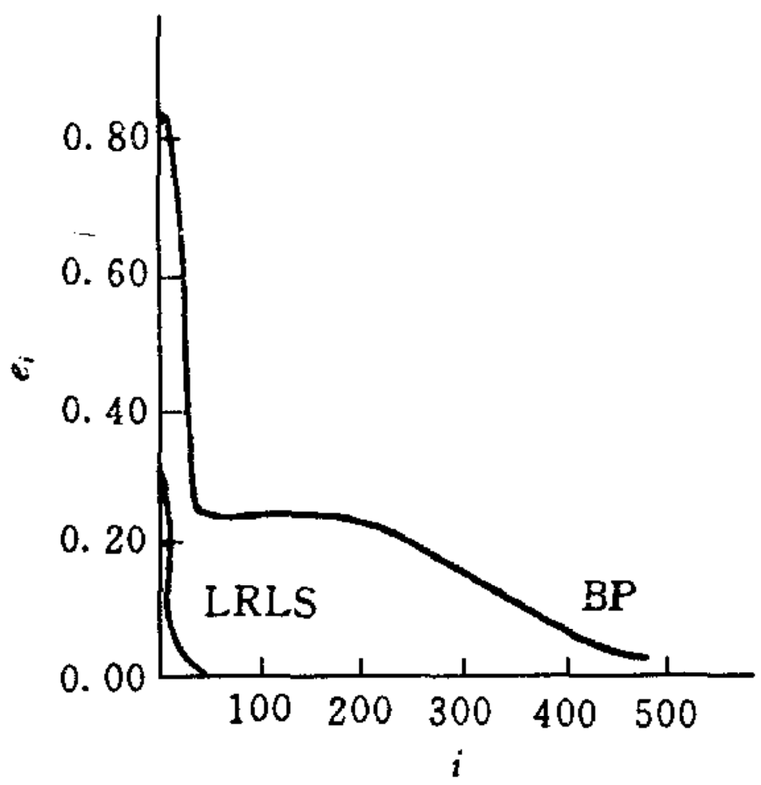

图 1　对例 1 采用 LRLS 和 BP 的平均输出误差曲线

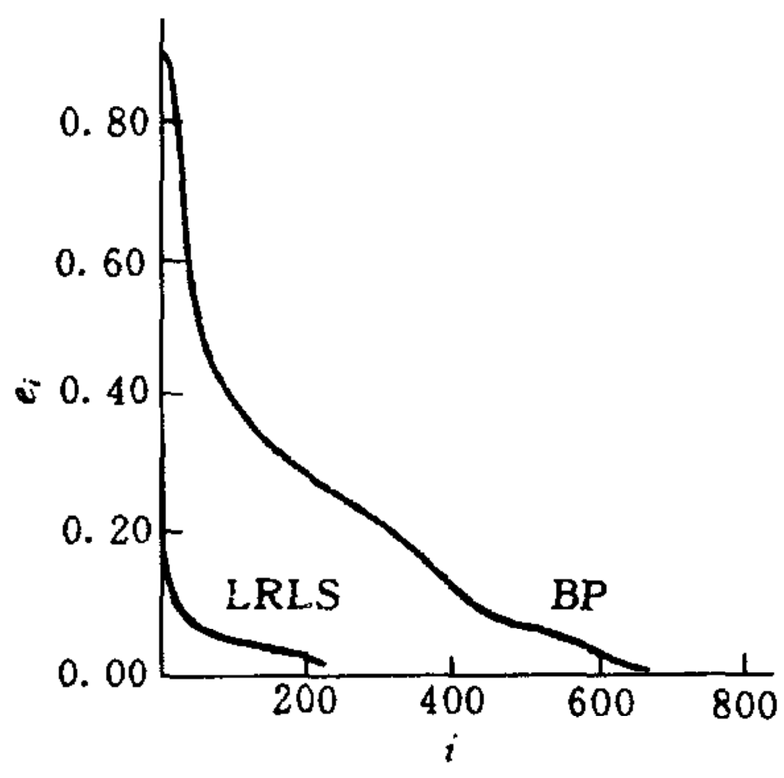

图 2　对例 2 采用 LRLS 和 BP 的平均输出误差曲线

例 3. 考虑单输入二输出系统

$$y_1(k) = u(k-1),$$
$$y_2(k) = u(k-1) + u(k-2).$$

网络取为双隐层结构 2-10-10-2，$u(k)$取为 0—0.5 间均匀随机变量. P_j 初始阵中取 $r=1000$，网络初始权重取为 0—0.5 间均匀随机数，阈值全部取为 0.5. 判断收敛的误差函数取为均方误差 $RMS=\left[\sum_{i=1}^{n} e_i^2\right]^{1/2}/n$，其中样本数 $n=30$，e_i 为第 i 个样本的学习误差. 图 3 表示 LRLS 与原始 BP 算法收敛速度的比较，其中 BP 算法取学习速率为 0.83，势态项系数为 0.23.

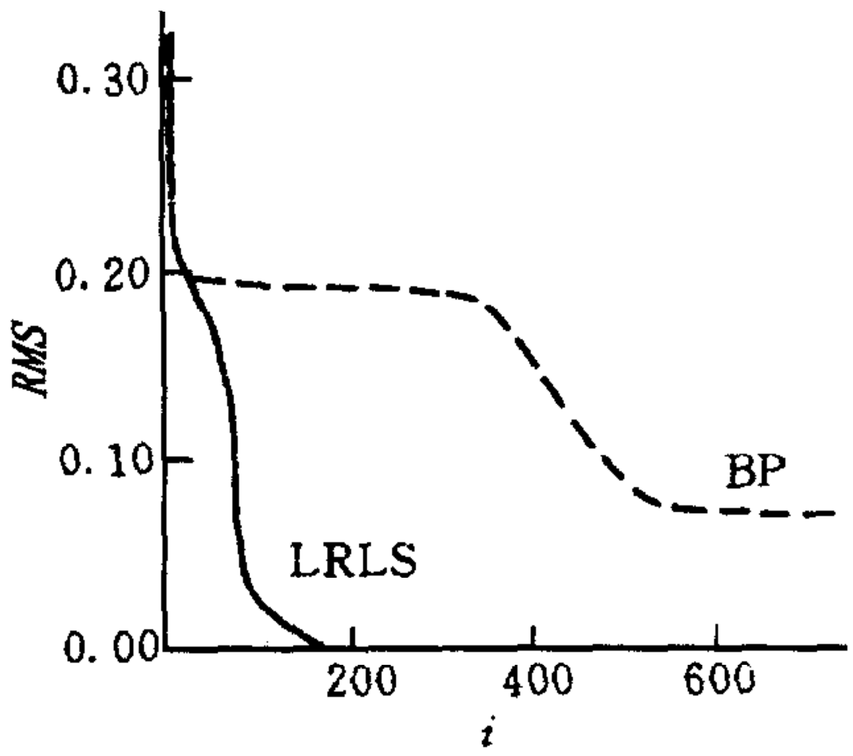

图 3 对例 3 采用 LRLS 和 BP 的均方输出误差曲线

注. LRLS 迭代至 129 次精度达 0.004979，而 BP 到 18700 次精度才达 0.01347.

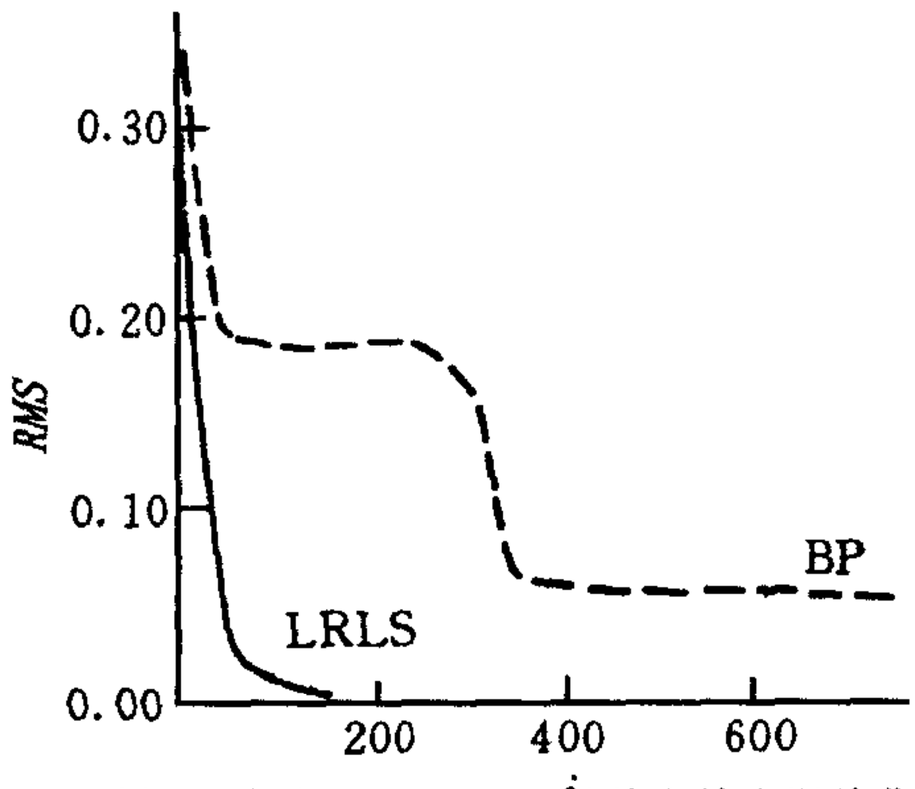

图 4 对例 4 采用 LRLS 和 BP 的均方输出误差曲线

注. LRLS 迭代至 112 次精度达 0.007984，而 BP 到 9651 次精度才达 0.007981.

例 4. 考虑二输入二输出动态系统

$$y_1(k) = 0.8y_1(k-1) + 3u_1(k-1) + u_2(k-1),$$
$$y_2(k) = -0.8y_2(k-1) - 4u_1(k-1) + u_2(k-1).$$

网络取为双隐层结构 4-10-10-2，其中 $u_1(k)$，$u_2(k)$均取为 0—1 间均匀随机变量. 模型初始值取为 $y_1(0)=y_2(0)=0$，网络和算法的初始值同例 3，判断收敛用的准则仍为均方误差. 图 4 显示了 LRLS 与原始 BP 算法收敛速度的比较，其中 BP 算法的参数同例 3.

从上述四个例子可见 LRLS 具有极好的性能，其收敛速度远比原始 BP 算法快.

5 结论

本文推导了一种前向网络的快速学习算法 GRLS，给出了一种更为简便的局部化算法 LRLS. 该算法避免了矩阵求逆，减少了计算量，其中 LRLS 算法更使存储量和计算量大为降低. 仿真结果表明了这种算法具有收敛快速、精度高的特点，是一种实用的前向网络快速学习的有效算法，适用于系统辨识等实际应用场合.

参 考 文 献

[1] Rumelhart D E, Hinon D E, Williams R J. Learning internal representations by error propagation, parallel distributed processing: Explorations in the microstructure of cognition, Cambridge MA: MIT Press, 1986, 318—362.

[2] Narendra K S, Parthasarathy K. Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks*, 1990, **1**(1): 4—27.

[3] Chen S, Billings S A, Grant P M. Non-linear system identification using neural networks. *Int. J. Control*, 1990, **51**(6): 1191—1214.

[4] Singhal S, Wu L. Training feed-forward networks with the extended Kalman algorithm. In: Proc. of the IEEE Int. Conf. on Acoustics, Speech and Singal Processing, Glasgow, 1989, 1187—1190.

[5] Kollias S, Amastassion D. An adaptive least squares algorithm for the efficient training of artificial neural networks. *IEEE Trans. Circuits and Systems*, 1990, **36**: 1092—1101.

[6] Shah S, Palmieri F, Datum M. Optimal filtering algorithms for fast learning in feed-forward neural networks. *Neural Networks*, 1992, **5**: 779—787.

附　录

算法(14),(15)的推导

从(12)式有

$$S(k) = \lambda \sum_{j=1}^{k-1} \sum_{i=1}^{m} \lambda^{k-1-j} \boldsymbol{g}_i(j) \boldsymbol{g}_i^T(j) + \sum_{i=1}^{m} \boldsymbol{g}_i(k) \boldsymbol{g}_i^T(k)$$

$$= \lambda S(k-1) + \sum_{i=1}^{m} \boldsymbol{g}_i(k) \boldsymbol{g}_i^T(k). \tag{A1}$$

同理可得

$$\boldsymbol{t}(k) = \lambda \boldsymbol{t}(k-1) + \sum_{i=1}^{m} \boldsymbol{g}_i(k) [d_i(k) - \xi_i(k)]. \tag{A2}$$

假定下面的等式成立

$$\sum_{i=1}^{m} \boldsymbol{g}_i(k) \boldsymbol{g}_i^T(k) = \boldsymbol{h}(k) \boldsymbol{h}^T(k), \tag{A3}$$

这里 $\boldsymbol{h}(k)$ 有与 $\boldsymbol{g}_i(k)$ 相同的维数. 需要指出,严格地说,(A3)式不能精确成立,此处仅为推导方便起见;后面,$\boldsymbol{h}(k)\boldsymbol{h}^T(k)$ 仍将用 $\sum \boldsymbol{g}_i(k)\boldsymbol{g}_i^T(k)$ 替代,故不会出现误差. 应用矩阵求逆公式

$$A = B^{-1} + CC^T, \quad A^{-1} = B + BC(I + C^T BC)^{-1} C^T B,$$

并设 $B = S^{-1}(k-1)\lambda^{-1}$ 及 $C = \boldsymbol{h}(k)$. 由(A1)和(A3)式可得

$$S^{-1}(k) = \lambda^{-1} S^{-1}(k-1) - \lambda^{-2} S^{-1}(k-1) \boldsymbol{h}(k) \boldsymbol{h}^T(K)$$
$$\times S^{-1}(k-1)[1 + \lambda^{-1} \boldsymbol{h}^T(k) S^{-1}(k-1) \boldsymbol{h}(k)]^{-1}. \tag{A4}$$

定义 $P(k) = S^{-1}(k)$,则(A4)式可写为

$$P(k) = \lambda^{-1} P(k-1) - \lambda^{-2} P(k-1) \boldsymbol{h}(k) \boldsymbol{h}^T(k) P(K-1)$$
$$\times [1 + \lambda^{-1} \boldsymbol{h}^T(k) P(k-1) \boldsymbol{h}(k)]^{-1}. \tag{A5}$$

由(A3)式知

$$\boldsymbol{h}^T(k) \boldsymbol{h}(k) = \sum_{i=1}^{m} \boldsymbol{g}_i^T(k) \boldsymbol{g}_i(k) \tag{A6}$$

必成立,即 $\boldsymbol{h}(k)\boldsymbol{h}^T(k)$ 的对角元素之和等于所有构成 $\boldsymbol{h}(k)\boldsymbol{h}^T(k)$ 的矩阵的对角元素之和,因此有

$$\boldsymbol{h}^T(k) P(k-1) \boldsymbol{h}(k) = \sum_{i=1}^{m} \boldsymbol{g}_i^T(k) P(k-1) \boldsymbol{g}_i(k). \tag{A7}$$

由(A3),(A5)及(A7)式可得

$$P(k) = \lambda^{-1}P(k-1) - \lambda^{-2}P(k-1)\Big[\sum_{i=1}^{m}\boldsymbol{g}_i(k)\boldsymbol{g}_i^T(k)\Big]P(k-1)$$

$$\times\Big[1+\lambda^{-1}\sum_{i=1}^{m}\boldsymbol{g}_i^T(k)P(k-1)\boldsymbol{g}_i(k)\Big]^{-1} \tag{A8}$$

定义
$$\boldsymbol{a}(k)=\lambda^{-1}P(k-1)\boldsymbol{h}(k)[1+\lambda^{-1}\boldsymbol{h}^T(k)P(k-1)\boldsymbol{h}(k)]^{-1}. \tag{A9}$$

由(A5)式得

$$P(k) = \lambda^{-1}P(k-1) - \lambda^{-1}\boldsymbol{a}(k)\boldsymbol{h}^T(k)P(k-1), \tag{A10}$$

则(A9)式可写为

$$\boldsymbol{a}(k) + \lambda^{-1}\boldsymbol{a}(k)\boldsymbol{h}^T(k)P(k-1)\boldsymbol{h}(k) = \lambda^{-1}P(k-1)\boldsymbol{h}(k) \tag{A11}$$

或
$$\boldsymbol{a}(k) = [\lambda^{-1}P(k-1) - \lambda^{-1}\boldsymbol{a}(k)\boldsymbol{h}^T(k)P(k-1)]\boldsymbol{h}(k) = P(k)\boldsymbol{h}(k). \tag{A12}$$

另外,(11)式中的参数估计 $\hat{\theta}(k)$ 可写为

$$\hat{\theta}(k) = P(k)\boldsymbol{t}(k) = P(k)\lambda\boldsymbol{t}(k-1) + P(k)\sum_{i=1}^{m}\boldsymbol{g}_i(k)[d_i(k) - \xi_i(k)]. \tag{A13}$$

把(A10)式代入(A13)式得

$$\hat{\theta}(k) = P(k-1)\boldsymbol{t}(k-1) - \boldsymbol{a}(k)\boldsymbol{h}^T(k)P(k-1)\boldsymbol{t}(k-1)$$

$$+ P(k)\sum_{i=1}^{m}\boldsymbol{g}_i(k)[d_i(k) - \xi_i(k)]. \tag{A14}$$

由(11),(A12)和(A14)式可得

$$\hat{\theta}(k) = \hat{\theta}(k-1) - P(k)\boldsymbol{h}(k)\boldsymbol{h}^T(k)\hat{\theta}(k-1) + P(k)\sum_{i=1}^{m}\boldsymbol{g}_i(k)[d_i(k) - \xi_i(k)]$$

$$= \hat{\theta}(k-1) + P(k)\sum_{i=1}^{m}\boldsymbol{g}_i(k)[d_i(k) - \xi_i(k) - \boldsymbol{g}_i^T(k)\hat{\theta}(k-1)]. \tag{A15}$$

略去高阶项 $\rho_i(k)$,$i=1,\cdots,m$,即得

$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)\sum_{i=1}^{m}\boldsymbol{g}_i(k)[d_i(k) - f_i(\hat{\theta}(k-1),\boldsymbol{i}(k))]. \tag{A16}$$

从上面的推导中已得到了(14)和(15)式.

A FAST LEARNING ALGORITHM OF FEEDFORWARD NEURAL NETWORKS AND ITS APPLICATION TO SYSTEM INDENTIFICATION

WANG ZHENG'OU

(*Institute of Systems Engineering, Tianjin University, Tianjin* 300072)

LIN CHEN

(*Fujian Asia Bank Limited, Fuzhou* 350001)

Abstract In this paper, we propose a fast learning algorithm of feedforward networks based on the least squares. Compared with existing similar algorithms, the present algorithm does not require any matrix inversion, therefore, it has a less computational cost and can be better suited for system indentification and other areas where fast learning is required. We derive the

algorithm and also give an even simpler and more convenient localized algorithm. Simulation results for system identification show the effectiveness of the algorithm.

Key words　Feedforward neural networks, fast learning, system identification.

王正欧　1961 年毕业于天津大学自动化仪表专业. 现为该校教授和美国纽约科学院成员及 IEEE 成员,长期从事系统辨识、系统建模理论及应用的教学科研工作. 近年来在国内外发表论文 60 余篇. 近期研究方向为神经网络理论及应用、人工智能等.

林　晨　1969 年生,1991 年毕业于天津大学,1994 年于天津大学系统工程研究所获工学硕士学位. 现任职于福建亚洲银行. 研究兴趣为神经网络、系统辨识和控制、经济金融等领域.