1996 年 11 月 重庆大学学报 (自然科学版) Vol. 19, №. 6
第 19 卷第 6 期 Journal of Chongqing University (Natural Science Edition) Nov. 1996

(7) 39-43

多层前馈神经网络的动态规划算法*

Dynamical Programming Algorithm for Multilayer feedforward Neural Networks

TP18
O221.3

叶仲泉[①] 张邦礼[②] 曹长修[②]
Ye Zhongquan Zhang Bangli Cao Changxiu
(① 重庆大学系统工程及应用数学系，重庆，630044；② 重庆大学信息工程学院；第一作者 35 岁，男，讲师，博士)

摘　要　利用动态规划来训练多层前馈网络，即逐层修改网络的权值。其算法采用有关文献提出的矩阵的广义逆的正交反向传播算法(OBP)，经有限次迭代即可得到每一层的最优权值。

关键词　动态最佳化；广义逆 / OBP 算法；多层前馈网络(DEDS)
中国图书资料分类法分类号　O221.3

神经网络
动态规划算法

ABSTRACT　This paper trains the multilayer neural networks using dynamical programming. The weithts are adjusted layer by layer. The optimal weights of each layer are calculated using OBP algorithm proposed in the related reference. The globally optimal weights of each layer can be obtained by finite iterations.

KEYWORDS　dynamic optimization; generalized inverse / OBP algorithm; multilayer feedforward networks(DEDS)

1　动态规划与最优性原理

基于对多级决策过程的研究，R. Belleman 在 50 年代首先提出动态规划法。目前该方法已在许多领域得到了广泛的应用，在生产、资源分配、设备更新、多级工艺设备的优化设计以及信息处理、模式识别等方面都得到了成功的应用。

动态规划是一种分步(级)优化方法，在多级决策寻优过程中，它是一种有效的方法。事实上，泛函极小化问题可以看成是一控制过程的结果。

动态规划法的基本思想是将一个多步最优决策问题化成多个一步最优控制(决策问题)。

Bellman 所提出的动态规划法概括起来是建立在最优性原理的基础之上。

最优性原理：在多步控制(决策)过程中的最优控制(决策)序列具有下列性质，即不论初始阶段、初始状态及初始控制如何，余下的控制序列对新的初始状态(即初始阶段的末值状态)而言，仍然构成最优控制序列。

* 收文日期　1996-04-05

从最优轨线的角度看,最优性原理也可表述为:最优轨线的一部分必为最优轨线。

2 矩阵广义逆的 OBP 算法

考察矩阵方程 $$W^*B = C \tag{1}$$

其中 $B \in R^{m\times n}$, $C \in R^{r\times n}$ 为已知的矩阵,$W^* \in R^{r\times m}$ 为未知的矩阵。文献[1]提出了一种解矩阵方程(1)的正交反向传播算法(OBP)

OBP 算法[1]: $$W(k) = W(k-1) - \eta(k)e(k)\xi^T(k) \tag{2}$$

其中 $$e(k) = W(k-1)z(k) - Cx(k)$$

$x(k)$ 是结构性神经网络的输入,$z(k) = Bx(k)$

$$\xi(k) = Q(k-1)z(k) \tag{3}$$

$$\eta(k) = \begin{cases} 1/|\xi(k)|^2 & \text{如果 } |\xi(k)| \neq 0 \\ 0 & \text{如果 } |\xi(k)| = 0 \end{cases} \tag{4}$$

$$Q(k) = Q(k-1) - \eta(k)\xi(k)\xi^T(k), Q(0) = I \tag{5}$$

由 $Q(k)$ 的定义知 $Q(k)$ 为 m 阶对称矩阵。

在上面的算法中,$z(k) = Bx(k)$, $x(k) \in R^n$.

定理 1[1] 由(5)定义的矩阵 $Q(k)$ 满足:

a) $Q^2(k) = Q(k)$

b) 若 $Q(k)z = 0$,则 z 为 $z(1), z(2), \cdots, z(k)$ 的一个线性组合。

定理 2[1] 如果输入 $x(k)$ 满足 $x(1), x(2), \cdots, x(n)$ 线性无关,且(1)有解,则由(2)~(5)定义的 OBP 算法保证。 $$W(n) \in W^* \tag{6}$$

其中 W^* 为方程(1)的解集。即该算法经 n 步迭代即收敛于方程(1)的解。

设 $A \in R^{m\times n}$, $R(A) = r$,记 A 的广义逆为 A^+ 假定 $m \geqslant n$.

当 $r = n$ 时,有著名的计算公式 $$A^+ = (A^TA)^{-1}A^T \tag{7}$$

且 $(A^TA)^{-1}$ 的计算也有著名的递推公式:

令 $$A = \begin{bmatrix} \alpha_1^T \\ \alpha_2^T \\ \vdots \\ \alpha_m^T \end{bmatrix}$$

则 $$A^T = [\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_m]$$

令 $$P_k = \left(\sum_{i=1}^{k} \alpha_i\alpha_i^T\right)^{-1}$$

则 $$(A^TA)^{-1} = P_m = \left(\sum_{i=1}^{m} \alpha_i\alpha_i^T\right)^{-1}$$

$$= (P_{m-1}^{-1} + \alpha_m\alpha_m^T)^{-1} = P_{m-1} - \frac{P_{m-1}\alpha_m\alpha_m^TP_{m-1}}{1 + \alpha_m^TP_{m-1}\alpha_m} \tag{8}$$

当 $R(A) = r < n$ 时,上述算法就不能使用了。[2]提出一种新算法,先利用上面介绍的解矩阵方程(1)的 OBP 算法将矩阵 A 进行满秩分解,然后再用矩阵求逆递推公式进行计算。

定理 3[3] 设 $A \in R^{m\times n}$, $R(A) = r > 0$,则存在 $F \in R_r^{m\times r}$ 和 $G \in R_r^{r\times n}$ 使 $A = FG$.

此定理即矩阵的满秩分解定理，其中 F 可由 A 的列极大线性无关组构成，G 可由 A 的行极大线性无关组构成。

定理 4([3])　如果 $A \in R_r^{m\times n}$ 的满秩分解为

$$A = FG \tag{9}$$

则

$$A^+ = G^T(F^TAG^T)^{-1}F^T = G^T(GG^T)^{-1}(F^TF)^{-1}F^T \tag{10}$$

因此，求 A^+ 的关键是求 A 的满秩分解。为此，先利用(3) ～ (5) 式计算 A 的行极大线性无关组。

设

$$A = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_m^T \end{bmatrix}$$

在(3) ～ (5) 式中令　$z(k) = a_k,\quad 1 \leqslant k \leqslant m$

由定理 1 知，存在 L 个向量　$a_{k_1}, a_{k_2}, \cdots, a_{k_L}\quad (1 \leqslant L \leqslant m)$

满足：1) $Q(k_i)a_{k_i} \neq 0, 1 \leqslant i \leqslant L$；2) $\forall k \notin \{k_1, k_2, \cdots, k_L\}$，$\exists\, k_i \in \{k_1, k_2, \cdots, k_L\}$ 使 $Q(k_i)a_k = 0$，则 $a_{k_1}, \cdots, a_{k_L}$ 即为 $a_1, a_2, \cdots, a_m$ 的最大无关组，且 $L = R(A) = r$.

故 G 可取为

$$G = \begin{bmatrix} a_{k_1}^T \\ a_{k_2}^T \\ \vdots \\ a_{k_L}^T \end{bmatrix} \tag{11}$$

求得 G 后，再求 F，即求解矩阵方程　$FG = A$　(12)

在(12) 中令 $F = W^* \in R^{m\times r}$，$G = B \in R^{r\times n}$，$A = C$，于是(12) 就化为　$W^*B = C$

这和方程(1) 完全一样，只是矩阵的阶数的字母不完全一样。由定理 3 知方程(12) 的解是存在的。因此可用(2) ～ (5) 定义的 OBP 算法来求方程(12) 的解。由定理 2 知，只要 $x(1)$，$x(2)$，$\cdots$，$x(n)$ 线性无关，则 $W(n) = F$ 即是方程(12) 的解。

这样，矩阵 A 的满秩分解 $A = FG$ 的问题就解决了。在(10) 中，逆矩阵$(GG^T)^{-1}$和$(F^TF)^{-1}$的计算可用与(8) 一样的递推公式来计算。

最后，将矩阵广义逆的正交反向传播算法小结如下：

1) 初始化权值 $W(0)$；

2) 用(3) ～ (5)($z(k) = a_k$) 求矩阵 A 的行最大线性无关组构成的矩阵 G；

3) 用(2) ～ (5) 式解方程(12)，其中 $x(1), x(2), \cdots, x(n)$ 线性无关，得 $W(n) = F$ 即为(12) 式的解；

4) 用与(8) 式一样的求逆递推公式求矩阵$(GG^T)^{-1}$与$(F^TF)^{-1}$；

5) 用公式(10) 求 A^+.

3　多层前馈网络的动态规划算法

多层前馈网络可以看作为一多级决策问题，每一层最优权值的选择类似于多级决策问

题中每一级的最优决策选择，且每一层的权值选择与多级决策过程一样只影响后面各层的输出。文献[4]给出了一种基于多层动态规划网络的学习算法：它从网络的输出层开始逐层计算最优权值，也给出了完全根据隐层权值和输出表示的误差函数，应用最优性原理最小化这一误差函数，即可得到隐层的最优权值，并且它是一种递推算法。

但文献[4]采用的优化方法仍是梯度下降法，所以存在收敛速度慢和难于选择学习率的问题。

笔者利用第二节提出的矩阵广义逆的正交反向传播算法来求每一层的最优权值，主要是因为从输入层的输出向量到第一隐层的输入向量，第一隐层的输出向量到第二隐层的输入向量，…，最后一个隐层的输出向量到输出层的输入向量之间都呈线性关系。这样，求矩阵的求广义逆就成为解决问题的关键了，第二节提出的求广义逆的方法刚好可以用上。

本文采用文献[5]的符号。

设多层网络有 L 层，一个输入层和一个输出层，$L-2$ 个隐层，设第 K 层有 t_K 个神经元，第 K 层的输出和第 $K+1$ 层的输入有如下的关系式：

$$y_{in}^{(K+1)} = W^{(K)} \times y_{out}^{(K)}, \qquad K = 1,2,\cdots,L-1 \tag{13}$$

$$y_{out}^{(K)} = f(y_{in}^{(K)}), \qquad K = 2,3,\cdots,L-1 \tag{14}$$

$$y_{out}^{(1)} = \bar{y}_{in}^{(1)}, \qquad y_{out}^{(L)} = \bar{y}_{in}^{(L)} \tag{15}$$

其中

$$f(x) = \frac{1}{1+e^{-x}} \tag{16}$$

给定样本集

$$(\bar{y}_{in}^{(1)}(1), \bar{y}_{out}^{(L)}(1)), \qquad (\bar{y}_{in}^{(1)}(2), \bar{y}_{out}^{(L)}(2)), \cdots, (\bar{y}_{in}^{(1)}(S), \bar{y}_{out}^{(L)}(S))$$

给定 L，学习的目的是要寻找：

$$\{W^{(K)}\}_1^{L-1} \tag{17}$$

使得

$$\sum_{i=1}^{s} \|\bar{y}_{out}^{(L)}(i) - y_{out}^{(L)}(i)\| \leqslant \varepsilon \tag{18}$$

其中 ε 为给定的充分小的正数，$\{y_{out}^{(L)}(i)\}$ 由下式确定

$$\begin{aligned}
&y_{in}^{(1)}(i) = \bar{y}_{in}^{(1)}(i) \qquad i = 1,2,\cdots,s \\
&y_{out}^{(K)}(i) = f(y_{in}^{(K)}(i)), \qquad i = 1,2,\cdots,s; K = 2,3,\cdots,L-1 \\
&y_{out}^{(1)}(i) = \bar{y}_{in}^{(1)}(i), \qquad y_{out}^{(L)}(i) = y_{in}^{(L)}(i) \qquad i = 1,2,\cdots,s \\
&y_{in}^{(K+1)}(i) = W^{(K)} \times y_{out}^{(K)}(i) \qquad i = 1,2,\cdots,s; K = 1,2,\cdots,L-1
\end{aligned} \tag{19}$$

注：　上述算法中阈值归入结点。

上面的计算过程可以用矩阵表示

$$Y_{in}^{(K)} = (y_{in}^{(K)}(1) \quad \cdots\cdots \quad y_{in}^{(K)}(S))_{t_K \times s} \tag{20}$$

$$Y_{out}^{(K)} = (y_{out}^{(K)}(1) \quad \cdots\cdots \quad y_{out}^{(K)}(S))_{t_K \times s} \tag{21}$$

从而(19)可以表示为

$$\begin{aligned}
&Y_{in}^{(1)} = \bar{Y}_{in}(1) \\
&Y_{out}^{(k)} = f(Y_{in}^{k}) \qquad k = 2,3,\cdots,L-1 \\
&y_{out}^{(1)} = \bar{Y}_{in}^{(1)}, \qquad y_{out}^{(L)} = Y_{in}^{(L)} \\
&Y_{in}^{(K+1)} = W^{(K)} \times Y_{out}^{(L)}, \qquad K = 1,2,\cdots,L-1
\end{aligned} \tag{22}$$

其中 $W^{(K)}$ 和 $t^{(K+1)} \times t^{K}$ 矩阵。

对网络的训练采用集中样本训练，训练算法如下：

1) 初始化权值矩阵：　$W^{(K)} = W_0^{(K)}, K = 1,2,\cdots,L-1$

2) 修改从最后一个隐层到输出层的权值，解矩阵方程：

$$\bar{Y}_{out}^{(L)} = W^{(L-1)} Y_{out}^{(L-1)} \tag{23}$$

其中 $\bar{Y}_{out}^{(L)}$ 与 $Y_{out}^{(L-1)}$ 为已知矩阵，$W^{(L-1)}$ 为未知矩阵。上述方程不一定有解，但可求最小二乘解：

$$\min_{W^{(L-1)}} \| \bar{Y}_{out}^{(L)} - W^{(L-1)} Y_{out}^{(L-1)} \|_F^2 \tag{24}$$

其中 $\|\cdot\|_F$ 表 Frobenious 范数。(24) 的最小二乘解为

$$W_{new}^{(L-1)} = \bar{Y}_{out}^{(L)} [Y_{out}^{(L-1)}]^+ \tag{25}$$

其中 A^+ 表示 A 的广义逆。$[Y_{out}^{(L-1)}]^+$ 可用第二节提出的正交反向传播算法来得

3) 解矩阵方程：

$$\bar{Y}_{out}^{(L)} = W_{new}^{(L-1)} Y_{out}^{(L-1)}(\text{new}) \tag{26}$$

其中　$\bar{Y}_{out}^{(L)}$ 与 $W_{new}^{(L-1)}$ 为已知矩阵，$Y_{out}^{(L-1)}(\text{new})$ 为未知矩阵。同理可得(26) 的最小二乘解为

$$Y_{out}^{(L-1)}(\text{new}) = (W_{new}^{(L-1)})^+ \bar{Y}_{out}^{(L)} \tag{27}$$

$(W_{new}^{(L-1)})^+$ 的计算仍采用第二节提出的 OBP 算法。

但(27) 中的 $Y_{out}^{(L-1)}(\text{new})$ 可能不属于 f 的值域 $R(f)$，这时可解如下的约束优化问题：

$$\min_{Y_{out}^{(L-1)}(\text{new}) \in R(f)} \| \bar{Y}_{out}^{(L)} - W^{(L-1)} Y_{out}^{(L-1)}(\text{new}) \|_F^2$$

4) 修改第$(L-2)$ 层到第$(L-1)$ 层的权值矩阵 $W^{(L-2)}$

从　$Y_{out}^{(L-1)}(\text{new}) = f(Y_{in}^{(L-1)}(\text{new}))$

求得　$Y_{in}^{(L-1)}(\text{new}) = f^{-1}(Y_{out}^{(L-1)}(\text{new}))$

解矩阵方程：　$Y_{in}^{(L-1)}(\text{new}) = W_{new}^{(L-2)} Y_{out}^{(L-2)}$

其中 $W_{new}^{(L-2)}$ 为未知矩阵。得最小二乘解

$$W_{new}^{(L-2)} = Y_{in}^{(L-1)}(\text{new}) (Y_{out}^{(L-2)})^+$$

5) 解矩阵方程：

$$Y_{in}^{(L-1)}(\text{new}) = W_{new}^{(L-2)} Y_{out}^{(L-2)}(\text{new})$$

其中　$Y_{out}^{(L-2)}(\text{new})$ 为未知矩阵。与第 3) 过程一样，求得

$$Y_{out}^{(L-2)}(\text{new})$$

$$\vdots$$

余此类推，可分别求得

$$W^{(L-1)}, W^{(L-2)}, \cdots, W^{(2)}, W^{(1)}$$

参 考 文 献

1 Polycarpou M M, Ioannou P A. Learning and convergence analysis of neural-type structured newtworks. IEEE Trans. Neural Networks, 1992, 3(1): 39～50

2 叶仲泉. 多层前馈神经网络的若干新算法研究及应用：[学位论文]，重庆：重庆大学博士学位论文，1996

3 孙继广. 矩阵扰动分析. 北京：科学出版社，1987. 1～50

4 焦李成. 神经网络计算. 西安：西安电子科技大学出版社，1993. 1～400

5 叶世伟，胡宏，史忠植. 多层网络并行学习算法及隐单元设置，C²N²－93，1993. 1～200