一种新的全局优化 BP 网络*

盛　立, 刘希玉, 高　明
(山东师范大学 信息管理学院, 山东 济南 250014)

摘　要: 将 L-M 算法与填充函数法相结合, 提出一种训练前向网络的混合型全局优化 GOBP(Global Optimization BP) 算法。L-M 算法的收敛速度快, 利用它先得到一个局部极小点, 然后利用填充函数算法跳出局部最小, 得到一个更低的局部极小点, 重复计算即可得到全局最优点。经实验验证, 该算法收敛速度很快, 避免了局部收敛, 而且性能稳定。

关键词: L-M 算法; 填充函数; 全局优化; BP 网络
中图法分类号: TP183　　**文献标识码:** A　　**文章编号:** 1001-3695(2006)02-0211-02

A Global Optimization BP Neural Networks

SHENG Li, LIU Xi-yu, GAO Ming
(School of Information Management, Shandong Normal University, Jinan Shandong 250014, *China)*

Abstract: Proposes GOBP(Global Optimization BP) neural networks in which combines the filled function method and Levenberg-Marquardt algorithm for training feed forward neural networks. With the L-M algorithm whose astringency is good, it can find one of local minimal points quickly. Afterwards, the filled function method will be used to find the point that is lower than the minimal point previously found. By repeating these processes, a global minimal point can be obtained at last. Practical examples indicate that the method has a higher accuracy in astringency and works well in avoiding sticking in local minima.
Key words: L-M Algorithm; Filled Function; Global Optimization; BP Neural Networks

人工神经网络是近年来发展起来的一门新兴学科, 它具有很强的自适应、自学习功能。其中 BP 网络是目前应用最为广泛的一种网络模型, 它采用多层前向网络的反向传播算法, 在函数逼近、模式识别、故障诊断、智能控制、信号处理等领域都有大量的应用。但标准的 BP 算法存在着易形成局部极小、训练陷入瘫痪和收敛速度很慢的问题, 从而影响了它的实际使用。为此, 人们在标准 BP 算法的基础上进行了许多有益的改进[9], 改进方法主要有两类: ①基于标准梯度下降的改进方法如附加动量的 BP 算法、自适应学习率调整法、弹性 BP 算法等; ②基于标准数值优化的改进算法如共扼梯度法、拟牛顿法和 Levenberg-Marquardt(L-M) 法等。

由于 BP 算法[1,2] 是通过用梯度下降方法来最小化二次误差目标函数推导出来的, BP 算法以及各种改进的 BP 算法均为局部优化算法, 训练结果与初始权值的选择有关。虽然可以选取多组初始权值, 多次训练, 选用最好的一组结果, 但是这样做不仅需要花费大量时间, 而且如何选取初始权值, 满足什么条件终止训练, 目前尚无理论指导。近年来, 各种随机型全局优化算法被相继提出[7], 如遗传算法、模拟退火法、随机搜索方法和下山单纯形搜索方法。

文献[3 ~6] 提出了确定型全局优化的填充函数法。本文利用一种新的易于计算的单参数填充函数[8], 把填充函数法与 L-M 算法相结合, 提出一种训练前向神经网络的混合型全局优化新算法。

收稿日期: 2005-03-04; 修返日期: 2005-04-18
基金项目: 山东省自然科学基金重大项目(Z2004G02) ; 山东省中青年科学家奖励基金资助项目(03BS003)

1　Levenberg-Marquardt 算法

Levenberg-Marquardt 算法类似于优化设计中的牛顿法, 其误差函数为平均平方误差。

$$x_{k+1} = x_k - A_k^{-1} g_k \tag{1}$$

式中 A_k 为 Hessian 矩阵, g_k 为梯度, 即

$$A_k = {}^2F(x_k) \quad g_k = F(x_k) \tag{2}$$

设 $F(x) = \sum_{i=1}^{N} v_i^2(x) = v^T(x)v(x)$　($v(x)$ 为输出映射函数) (3)

第 j 个梯度元素为 $F(x)_j = \frac{5F(x)}{5x_j} = 2\sum_{i=1}^{N} v_i(x)\frac{5v_i(x)}{5x_j}$ (4)

写成矩阵形式为 $F(x) = 2J^T(x)v(x)$ (5)

式中 $J(x)$ 称为雅可比矩阵, 其表达式为

$$J(x) = \begin{bmatrix} \frac{5v_1(x)}{5x_1} & \frac{5v_1(x)}{5x_2} & \cdots & \frac{5v_1(x)}{5x_n} \\ \frac{5v_2(x)}{5x_1} & \frac{5v_2(x)}{5x_2} & \cdots & \frac{5v_2(x)}{5x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{5v_N(x)}{5x_1} & \frac{5v_N(x)}{5x_2} & \cdots & \frac{5v_N(x)}{5x_n} \end{bmatrix} \tag{6}$$

Hessian 矩阵的第 k, j 元素为

$${}^2F(x)_{k,j} = \frac{5^2F(x)}{5x_k5x_j} = 2\sum_{i=1}^{N}\frac{5v_i(x)}{5x_k}\cdot\frac{5v_i(x)}{5x_j} + v_i(x)\frac{5^2v_i(x)}{5x_k5x_j} \tag{7}$$

其矩阵形式为 ${}^2F(x) = 2J^T(x)J(x) + 2S(x)$ (8)

式中 $S(x) = \sum_{i=1}^{N} v_i(x)\ {}^2v_i(x)$ (9)

假设 $S(x)$ 很小,则有 $\nabla^2 F(x) \approx 2J^T(x)J(x)$ (10)

将式(10)和式(5)代入式(1)中,得高斯 - 牛顿法:

$$x_{k+1} = x_k - [2J^T(x_k)J(x_k)]^{-1} 2J^T(x_k)v(x_k) = x_k - [J^T(x_k)J(x_k)]^{-1} J^T(x_k)v(x_k) \quad (11)$$

与标准牛顿法相比,高斯 - 牛顿法不计算二阶导数,但高斯 - 牛顿法中,$H = J^T \times J$ 可能是不可逆的。因此设

$$G = H + \mu I \quad (12)$$

式(12)中,I 为单位矩阵。

假定 H 的特征值和特征向量分别为 $\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$ 和 $\{z_1, z_2, \cdots, z_n\}$,则有

$$Gz_i = [H + \mu I]z_i = Hz_i + \mu z_i = \lambda_i z_i + \mu z_i = (\lambda_i + \mu)z_i \quad (13)$$

因此 G 的特征向量和 H 的相同,而特征值为 $\lambda_i + \mu$。若增加 μ 使 $\lambda_i + \mu > 0$,可使 G 变为正定矩阵。而正定矩阵是可逆的,这就有 Levenberg-Marquardt 算法,即

$$x_{k+1} = x_k - [J^T(x_k)J(x_k) + \mu_k I]^{-1} J^T(x_k)v(x_k) \quad (14)$$

随着 μ_k 的增加,它接近于最速下降法,即当 μ_k 很大时

$$x_{k+1} = x_k - \frac{1}{\mu_k} J^T(x_k)v(x_k) = x_k - \frac{1}{2\mu_k} \nabla F(x) \quad (15)$$

而当 μ_k 减少到 0 时,它变成高斯 - 牛顿法。

Levenberg-Marquardt 算法在开始时,可将 μ_k 取得很小,如 $\mu_k = 0.01$。如果迭代一次后 $F(x)$ 的函数值仍很大,可将 μ_k 乘以一个大于 1 的因子 $v > 1$(如 $v = 10$),以使函数值减少,类似于最速下降法;否则将 μ_k 除以一个大于 1 的因子 v,以加速收敛速度,此时算法接近高斯 - 牛顿法。

由此可见,从收敛速度和收敛性来看 L-M 算法是介于牛顿法和最速下降法的一种折中算法。

2 填充函数全局寻优法

本节所考虑的数学模型为 $\min(F(X))$,其中 $X \in \Omega \subset R^n$。$F(X)$ 为多值函数,$F(X)$ 在 Ω 上有最小值,假设 $F(X)$ 二次连续可微。

定义 1 设 X_1 是 $F(X)$ 的一个局部极小点,X_1 的盆(或称为吸引域) B_1 是一个具有下列性质的连通域:①如果 $X_0 \in B_1$,$X_0 \neq X_1$,$F(X_0) > F(X_1)$,则从 X_0 出发的下降轨线必然终止于 X_1;②如果 $X_0 \mid B_1$,则从 X_0 出发的下降轨线比不终止于 X_1。

定义 2 称局部极小点 X_2 高于局部极小点 X_1,当且仅当 $F(X_2) > F(X_1)$,此时也称 X_2 的盆高于 X_1 的盆。

填充函数法的基本思想是,根据目标函数 $F(X)$(图 1)的一个已知极小点 X_1 构造一个填充函数 $P(X)$,它有极大点 X_1,而且在任何比 X_1 盆高的盆中没有极小点,但在某个比 X_1 盆低的盆中有一个极小点 $\bar{X}$;以 $\bar{X}$ 作为初始点取极小 $F(X)$,并找到 $F(X)$ 的一个新的极小点 X_2,使得 $F(X_2) \leq F(X_1)$。用 X_2 代替 X_1 重复上述过程,直到找到 $F(X)$ 的全局极小点。

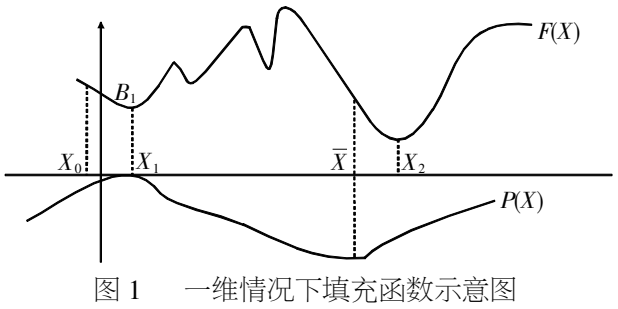

图 1 一维情况下填充函数示意图

文献[8]给出了一个只有单参数(α)的填充函数:

$$L(X, \alpha) = -\frac{1}{\nabla F(X) - F(X_1) + \alpha \| X - X_1 \|} \quad (16)$$

式中,X_1 为一个已知的局部极小点。

3 利用基于 L-M 算法的填充函数法训练神经网络

对于 N 组训练数据对、输出节点为 M 的神经网络,可以定义第 k 个样本的训练误差和总误差为

$$E(k, w) = \frac{1}{M}\sum_{i=1}^{M}(y_i(k) - y_i(k, w))^2 \quad (17)$$

$$E(w) = \frac{1}{N}\sum_{k=1}^{N} E(k, w) \quad (18)$$

式(17)中,$y_i(k)$ 和 $y_i(k, w)$ 分别为输出节点 i 对第 k 个样本的网络输出和期望输出,$w \in R^n$ 为权值向量,训练神经网络就是利用已知训练数据对,调整连接权系数以使 $E(w)$ 最小。实际训练中可以 $E(w)$ 小于预先指定的某个很小的正数作为训练终止条件。

利用基于 L-M 算法的填充函数法训练神经网络的算法 GOBP(Global Optimization BP)的步骤如下:

(1)在[0,1]区间内取很小的随机数作为初始权值,给定参数 α 的初始值(如 $\alpha = 100$)以及训练精度 $h = 0.001$。用 L-M 算法求出 $E(w)$ 的一个局部极小点 w_1。

(2)如果 $E(w_1) < h$,则 w_1 是最优权值,结束训练;否则转第(3)步。

(3)构造 w_1 处的填充函数:

$$L(w, \alpha) = -\{[F(w) - F(w_1)]^{1/\beta} + \alpha \| w - w_1 \|\} \quad (19)$$

(4)在 w_1 附近取一点 w_0,如 $w_0 = w_1 + \delta e_j$,其中,δ 为很小正数(如 $\delta = 0.02$),e_j 为第 j 个分量为 1,其余分量为 0 的单位向量。从 w_0 出发,极小化 $L(w, \alpha)$ 的极小点 w 终止准则为 w' 满足下列条件之一:

①$L(w', \alpha) \geq 0$;

②$(w' - w_1)^T \nabla L(w', \alpha) \geq 0$;

③$\| \nabla L(w', \alpha) \| \leq \varepsilon$。

其中 ε 是实现给定的精度,若 w 落入目标函数的定义域之外,则取 w_0 为最优权值,训练结束。

(5)以 w 为初始权值,用 L-M 算法训练网络得到 $E(w)$ 的另一个极小点 w_2。

(6)如果 $E(w_2) \leq E(w_1)$,令 $w_1 = w_2$,返回第(2)步。

(7)如果 $E(w_2) > E(w_1)$,放大 α,返回第(3)步。

4 应用实例

贷款评估问题,就是要求设计一个神经网络,能够根据借贷申请人的月收入、生活费用、房租、水、电、交通费用的支出以及其他费用的支出来实时地分析这一借贷申请是否合格,如合格则批准申请给予贷款,否则给予拒绝。

我们取上述四个参数作为诊断参数。BP 网络结构采取 4-16-1,输入输出训练对为 96 个,平均平方误差为 0.001,隐层转移函数为 S 型,输出层为线性函数。图 2、图 3 分别为用基本 BP 算法训练和用本文所述算法训练的误差平方和随训练步数的变化图。

由图 2 和图 3 可以看出基本 BP 算法需要 5 222 代才能达到精度要求,而 GOBP 算法只需要 9 代就能满足 (下转第 255 页)