文章编号:1671 - 6833(2018)02 - 0028 - 05

一种基于共轭梯度法的广义单隐层神经网络

孙 峰,龚晓玲,张炳杰,柳毓松,王延江

(中国石油大学(华东),山东 青岛 266580)

摘 要:单隐层前馈神经网络是一种高效且结构简单的神经网络,它的一种典型的学习算法就是误差反向传播(error back propagation, BP)算法.这种算法基于最速下降法原理,主要缺点是学习速度过慢.超限学习机(extreme learning machine, ELM)极大地优化了单隐层神经网络的学习速度,却需要更多的隐层单元来达到与 BP 网络相当的效率,这不可避免地使网络结构冗余、测试时间变长.受到一种结合了 ELM 和最速下降法思想的 USA(upper-layer-solution-aware)算法的启发,提出一种基于共轭梯度法的单隐层神经网络快速算法,并把它应用于不同数据库中.试验结果表明,在相同网络结构情况下,本算法的效率要优于 ELM 和 USA 算法.

关键词:神经网络;反向传播;超限学习机;共轭梯度法;MNIST

中图分类号:TP389.1 **文献标志码**:A doi:10.13705/j.issn.1671 - 6833.2017.05.011

0 引言

人工神经网络是一种模仿人脑神经元的结构和信息传递过程所建立的数学模型,具有学习、记忆等功能.近三十年来神经网络得到迅速的发展并在模式识别等领域有着广泛的应用,尤其在未来智能化城市发展中会起到关键作用.其中,应用最广泛的就是采用了反向传播(back propagation)学习算法[1]的前馈神经网络,这种网络也称为 BP 神经网络,自 1986 年被提出之后,至今仍在神经网络中占据重要地位.该网络在训练时信息从前向后逐层传播,而网络中的权重则通过误差反向传播来调整.这种算法结构简单且能够以任意精度逼近任意非线性函数[2-3],但也存在一些缺陷:由于网络收敛速度慢,导致训练花费时间过长;由于采用梯度下降法易陷入局部极小值;易过拟合训练使其泛化能力变差等.

为了加快网络的训练速度,黄广斌等提出了一种针对单隐层前馈神经网络的快速构建与学习算法[4],因其快速特性而称之为超限学习机(extreme learning machine, ELM).ELM 算法的整个学习过程一次完成,使得网络训练大大简化,运算速度几十倍甚至几千倍于 BP 算法[5-6],ELM 在许多领域都取得了突出成果[7-9].虽然网络训练效果很好,但 ELM 相对于传统的神经网络,需要更多的隐层节点才能达到同样的训练精度[10].由于使用大量隐层节点,计算工作量会大大增加,特别是样本超高维时,测试速度明显变慢;过多的隐层节点也会使网络过拟合而降低泛化能力.

针对 ELM 存在的问题,文献[11]提出了用梯度下降法来选择合适权值的 upper-layer-solution-aware(USA)算法.使用最速下降法来更新从输入层到隐层的权值.它与传统的神经网络的区别在于是用广义逆一步得出,相当于传统 BP 算法和 ELM 算法的结合.与 BP 神经网络相比,它的速度大大提高了;与 ELM 算法相比,它需要的隐层节点数减少了.相比于 ELM,它在训练上需要浪费一些时间,但是实际应用中,往往测试时间比训练时间更加关键,减少隐层节点数,有利于缩短测试时间,更符合实际工程需要.

笔者借鉴 ELM 的一次学习思想,提出了一种基于共轭梯度法的快速学习算法,由于共轭梯度法计算复杂度不高于最速下降法,但是收敛速度优于最速下降法,使得笔者算法在保持

收稿日期:2017 - 07 - 16;**修订日期**:2017 - 09 - 01

基金项目:国家自然科学基金资助项目(61305075);山东省自然科学基金资助项目(ZR2013FQ004);教育部博士学科点科研基金资助项目(20130133120014);中央高校基本科研业务费专项基金资助项目(14CX05042A、15CX04065B、15CX05053A、15CX08011A)

作者简介:孙 峰(1978—),男,山东泰安人,中国石油大学讲师,硕士,主要从事机械工业设计方面的研究,E-mail:sunfeng@upc.edu.cn.

快速这一优势前提下,网络训练精度得到进一步提高.

1　ELM 和 USA 算法描述

在介绍笔者算法前,先简单介绍一下 ELM 的基本算法以及 ELM 算法和 BP 算法相结合的 USA 算法. 这两种算法都应用于单隐层前馈神经网络且有良好的数值表现,ELM 算法随机赋予输入层到隐层的权值,无需迭代,因此训练时间极快. USA 算法用最速下降法对输入层到隐层的权值进行优化,虽训练时间稍慢,但达到相同网络训练精度所需隐层节点个数要大大少于 ELM.

1.1　超限学习机

对于 N 个不同的训练样本$(\boldsymbol{x}_j,\boldsymbol{t}_j)$,其中输入向量为 $\boldsymbol{x}_j = (x_{j1},x_{j1},\cdots,x_{jn})^{\mathrm{T}} \in \mathbf{R}^n$,理想输出向量为 $\boldsymbol{t}_j = (t_{j1},t_{j1},\cdots,t_{jm})^{\mathrm{T}} \in \mathbf{R}^m$,若选取单隐层神经网络的隐层节点个数为 $\tilde{N}$ 个,隐层激活函数为 $g(x)$,输出层激活函数为简单线性函数$f(x) = x$,那么该网络的实际输出为

$$\sum_{i=1}^{\tilde{N}} \boldsymbol{u}_i g_i(\boldsymbol{w}_i \cdot \boldsymbol{x}_j + b_i), j = 1,2,\cdots,N, \quad (1)$$

式中:$\boldsymbol{w}_i = [w_{i1},w_{i2},\cdots,w_{in}]^{\mathrm{T}}$ 为连接输入层到隐层第 i 个单元之间的权值;$\boldsymbol{u}_i = [u_{i1},u_{i2},\cdots,u_{in}]^{\mathrm{T}}$ 为连接隐层第 i 个单元到输出层之间的权值;b_i 为第 i 个隐节点的阈值.

如果这个网络可以零误差地学习这 N 个样本,即

$$\sum_{i=1}^{\tilde{N}} \boldsymbol{u}_i g_i(\boldsymbol{w}_i \cdot \boldsymbol{x}_j + b_i) = \boldsymbol{t}_j, j = 1,2,\cdots,N. \quad (2)$$

上式表示成矩阵的形式为:

$$\boldsymbol{H}\boldsymbol{U} = \boldsymbol{T}, \quad (3)$$

式中:$\boldsymbol{H}$ 为隐层输出矩阵[12-13].

$$\boldsymbol{H}(\boldsymbol{w}_1,\cdots,\boldsymbol{w}_{\tilde{N}},b_1,\cdots,b_{\tilde{N}},\boldsymbol{x}_1,\cdots,\boldsymbol{x}_N) = \begin{bmatrix} g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_1 + b_1) & \cdots & g(\boldsymbol{w}_{\tilde{N}} \cdot \boldsymbol{x}_1 + b_{\tilde{N}}) \\ \vdots & \vdots & \vdots \\ g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_N + b_1) & \cdots & g(\boldsymbol{w}_{\tilde{N}} \cdot \boldsymbol{x}_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}. \quad (4)$$

当输入层到隐层间的权值 $\boldsymbol{w}_i$ 和阈值 b_i 给定后,可以求出隐层输出矩阵 $\boldsymbol{H}$,公式(3)就等价于一个求解 $\boldsymbol{U}$ 的线性系统[14]. 当隐层节点个数 $\tilde{N}$ 等于输入样本数 N 时,可以证明 $\boldsymbol{H}$ 以概率 1 可逆,所以该式有唯一解,

$$\boldsymbol{U} = \boldsymbol{H}^{-1}\boldsymbol{T}, \quad (5)$$

即网络可以零误差的拟合训练样本[12-13]. 在实际情况中,样本数远多于隐层节点个数,即 $N >> \tilde{N}$, 这时可以利用 Moore-Penrose 广义逆来求解,

$$\boldsymbol{U} = \boldsymbol{H}^{\dagger}\boldsymbol{T}, \quad (6)$$

式中:$\boldsymbol{H}^{\dagger} = (\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H})^{-1}\boldsymbol{H}^{\mathrm{T}}$.

通过以上原理,可以看出 ELM 最大的优点在于它无需迭代,一步就能求出隐层权值 $\boldsymbol{U}$.

1.2　最速下降法

USA 算法是文献[11]中提出的一种算法. 下面简单介绍其原理.

给定训练样本集 $\aleph = \{(\boldsymbol{x}_j,\boldsymbol{t}_j) \mid \boldsymbol{x}_j \in \mathbf{R}^n, \boldsymbol{t}_j \in \mathbf{R}^m\}$ $(j = 1,2,\cdots,N)$,选取单隐层神经网络的隐层节点个数为 $\tilde{N}$,隐层激活函数 $g(x)$ 为 Sigmoid 函数,输出层激活函数为简单线性函数 $f(x) = x$,给定网络的误差函数 $E = \|\boldsymbol{Y} - \boldsymbol{T}\|^2 = \mathrm{Tr}[(\boldsymbol{Y} - \boldsymbol{T})(\boldsymbol{Y} - \boldsymbol{T})^{\mathrm{T}}]$. 那么该网络的隐层输出矩阵仍为 $\boldsymbol{H}$,隐层到输出层的权值 $\boldsymbol{U} = \boldsymbol{H}^{\dagger}\boldsymbol{T} = (\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H})^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{T}$. 由此可以看出,从隐层到输出层的权值 $\boldsymbol{U}$ 是从隐层到输入层权值 $\boldsymbol{W}$ 的函数,即可以由 $\boldsymbol{W}$ 确定. 所以在网络训练时,我们在每次迭代中用最速下降的方法更新 $\boldsymbol{W}$,即先求误差函数的梯度,

$$\frac{\partial E}{\partial \boldsymbol{W}} = \frac{\partial \mathrm{Tr}[(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{H} - \boldsymbol{T})(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{H} - \boldsymbol{T})^{\mathrm{T}})]}{\partial \boldsymbol{W}}. \quad (7)$$

然后沿负梯度方向优化 $\boldsymbol{W}$,即

$$\boldsymbol{W} = \boldsymbol{W} - \eta \frac{\partial E}{\partial \boldsymbol{W}}, \quad (8)$$

式中:η 为学习率.

这样在每次迭代后只优化更新连接输入层和隐藏层的权值 $\boldsymbol{W}$,使误差逐步减小,达到可接受误差或最大迭代次数时迭代结束,得到了优化的网络结构.

2　笔者工作

要想降低网络的隐层节点个数,一种有效方法就是对其权值进行优化. 从最优化原理来说,优化权值的方法有最速下降法、共轭梯度法、牛顿法等. BP 算法就是采用最速下降法达到优化权值的目的.

2.1　共轭梯度法

共轭梯度法是一类效果较好的共轭方向法,最早由 Hestenes 和 Stiefel 提出并用于求解线性方程组[14],后来被 Fletcher 和 Reeves 引入求解无约束最优化问题[15]. 共轭梯度法的原理是:在寻优过程中,利用当前点 $\boldsymbol{x}^k$ 处的梯度向量 $\boldsymbol{g}(\boldsymbol{x}^k)$ 和前一迭代点 $\boldsymbol{x}^{k-1}$ 处的搜索方向 $\boldsymbol{d}^{k-1}$ 对最速下降方向

进行如下修正:

$$\boldsymbol{d}^k = -\boldsymbol{g}(\boldsymbol{x}^k) + \beta_k \boldsymbol{d}^{k-1}, \tag{9}$$

并保证新的搜索方向 $\boldsymbol{d}^k$ 与之前的搜索方向 $\boldsymbol{d}^{k-1}$, $\boldsymbol{d}^{k-2}, \cdots, \boldsymbol{d}^0$ 之间满足共轭关系. 其中,修正系数 β_k 的不同又进一步形成了不同的共轭梯度法.

共轭梯度法与最速下降法同样用到一阶导数信息,但它克服了梯度下降法收敛慢的缺点,又避免了牛顿法需要存储和计算 Hesse 矩阵并求逆的缺点. 其优点是所需存储量小,具有有限步收敛性,稳定性高,而且不需要任何外来参数.

2.2 笔者算法

基于共轭梯度法优化权值的思想,提出一种训练单隐层前馈神经网络的新算法:给定 N 个不同训练样本 $(x_1, t_1), (x_2, t_2), \cdots, (x_N, t_N)$,其中 $\boldsymbol{x}_i = (x_{1i}, x_{2i}, \cdots, x_{ni})^{\mathrm{T}}$, $\boldsymbol{t}_i = (t_{1i}, t_{2i}, \cdots, t_{ni})^{\mathrm{T}}$,则网络的输入节点个数为 n,输出节点个数为 m. 网络的输入矩阵为 $\boldsymbol{X}_{n\times N}$,理想输出为 $\boldsymbol{T}_{m\times N}$. 若隐层节点个数为 $\tilde{N}$, 则输入层到隐层间的权值矩阵为:

$$\boldsymbol{W} = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1\tilde{N}} \\ w_{21} & w_{22} & \cdots & w_{2\tilde{N}} \\ \vdots & \vdots & \vdots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{n\tilde{N}} \end{pmatrix}_{n\times\tilde{N}}. \tag{10}$$

隐层到输出层间的权值矩阵为:

$$\boldsymbol{U} = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1m} \\ u_{21} & u_{22} & \cdots & u_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ u_{\tilde{N}1} & u_{\tilde{N}2} & \cdots & u_{\tilde{N}m} \end{pmatrix}_{\tilde{N}\times m}. \tag{11}$$

那么该网络的隐层输出矩阵为:

$$\boldsymbol{H} = g(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}), \tag{12}$$

式中:$g(\cdot)$ 为隐层激活函数,一般选用 Sigmoid 函数. 根据输出值是否包含负值,分为 Log-Sigmoid 函数和 Tan-Sigmoid 函数,本算法选用函数值在 0 ~ 1 的简单 Log-Sigmoid 函数,即

$$\boldsymbol{g}(x) = \frac{1}{1+\mathrm{e}^{-x}}. \tag{13}$$

网络的输出层激活函数为简单线性函数,则网络的实际输出为:

$$\boldsymbol{Y} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{H}. \tag{14}$$

对于确定网络的权值 $\boldsymbol{U}$,可以将网络结构看成一个线性系统,

$$\boldsymbol{U}^{\mathrm{T}}\boldsymbol{H} = \boldsymbol{T}. \tag{15}$$

当 $N = \tilde{N}$ 时,该方程的解为:$\boldsymbol{U} = (\boldsymbol{T}\boldsymbol{H}^{-1})^{\mathrm{T}}$. 但是实际情况中,一般 $N >> \tilde{N}$, 这时利用广义逆求解:

$$\boldsymbol{U} = (\boldsymbol{H}^{\mathrm{T}})^{\dagger}\boldsymbol{T}^{\mathrm{T}} = (\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})^{-1}\boldsymbol{H}\boldsymbol{T}^{\mathrm{T}}. \tag{16}$$

由于 $\boldsymbol{H} = g(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X})$,所以隐层到输出层的权值 $\boldsymbol{U}$ 可以看作是输入层到隐层的权值 $\boldsymbol{W}$ 的函数. 本算法采用共轭梯度法来对输入层到隐层之间的权值 $\boldsymbol{W}$ 进行优化. 网络的误差函数可以定义为:

$$E = \left\|\boldsymbol{Y} - \boldsymbol{T}\right\|^2 = \mathrm{Tr}[(\boldsymbol{Y} - \boldsymbol{T})(\boldsymbol{Y} - \boldsymbol{T})^{\mathrm{T}}]. \tag{17}$$

首先求误差函数的梯度,由式(14)和式(17)得:

$$\frac{\partial E}{\partial \boldsymbol{W}} = \frac{\partial \mathrm{Tr}[(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{H} - \boldsymbol{T})(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{H} - \boldsymbol{T})^{\mathrm{T}}]}{\partial \boldsymbol{W}}. \tag{18}$$

再将式(16)带入式(18),得到梯度的计算公式为:

$$\begin{aligned} \frac{\partial E}{\partial \boldsymbol{W}} &= \frac{\partial \mathrm{Tr}[(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{H} - \boldsymbol{T})(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{H} - \boldsymbol{T})^{\mathrm{T}}]}{\partial \boldsymbol{W}} \\ &= \frac{\partial \mathrm{Tr}[([(\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})^{-1}\boldsymbol{H}\boldsymbol{T}^{\mathrm{T}}]^{\mathrm{T}}\boldsymbol{H} - \boldsymbol{T})([(\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})^{-1}\boldsymbol{H}\boldsymbol{T}^{\mathrm{T}}]^{\mathrm{T}}\boldsymbol{H} - \boldsymbol{T})^{\mathrm{T}}]}{\partial \boldsymbol{W}} \\ &= \frac{\partial \mathrm{Tr}[\boldsymbol{T}\boldsymbol{T}^{\mathrm{T}} - \boldsymbol{T}\boldsymbol{H}^{\mathrm{T}}(\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})^{-1}\boldsymbol{H}\boldsymbol{T}^{\mathrm{T}}]}{\partial \boldsymbol{W}} \\ &= \frac{\partial \mathrm{Tr}[(\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})^{-1}\boldsymbol{H}\boldsymbol{T}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{T}^{\mathrm{T}}]}{\partial \boldsymbol{W}} \\ &= \frac{\partial \mathrm{Tr}[(\sigma(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X})[\sigma(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X})]^{\boldsymbol{T}})^{-1}\sigma(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X})\boldsymbol{T}^{\mathrm{T}}\boldsymbol{T}[\sigma(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X})]^{\mathrm{T}}]}{\partial \boldsymbol{W}} \\ &= 2\boldsymbol{X}[\boldsymbol{H}^{\mathrm{T}}\mathrm{o}(\boldsymbol{I} - \boldsymbol{H})^{\mathrm{T}}\mathrm{o}[\boldsymbol{H}^{\dagger}(\boldsymbol{H}\boldsymbol{T}^{\mathrm{T}})(\boldsymbol{T}\boldsymbol{H}^{\dagger} - \boldsymbol{T}^{\mathrm{T}}(\boldsymbol{T}\boldsymbol{H}^{\dagger}))]], \end{aligned} \tag{19}$$

式中:

$$\boldsymbol{H}^{\dagger} = \boldsymbol{H}^{\mathrm{T}}(\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})^{-1}. \tag{20}$$

本算法采用 F-R 共轭梯度法,搜索方向为:

$$\boldsymbol{d}^k = \begin{cases} -\boldsymbol{g}^k, & k = 0 \\ -\boldsymbol{g}^k + \beta_k^{FR}\boldsymbol{d}^{k-1}, & k \geqslant 1 \end{cases}, \tag{21}$$

式中:$\boldsymbol{g}^k$ 即为第 k 次迭代的梯度,修正项系数为:

$$\beta_k^{FR} = \frac{\boldsymbol{g}(\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{g}(\boldsymbol{x}^k)}{\boldsymbol{g}(\boldsymbol{x}^{k-1})^{\mathrm{T}}\boldsymbol{g}(\boldsymbol{x}^{k-1})}. \tag{22}$$

从输入层到隐层的权值 $\boldsymbol{W}$ 的更新公式为:

$$\boldsymbol{W}^{k+1} = \boldsymbol{W}^k + \eta_k \boldsymbol{d}^k. \tag{23}$$

式中:学习率 η_k 通过线搜索的方式获得.

3　数值试验

将笔者提出的算法与 USA 算法[11]、ELM 算法[4]在 3 种数据库上进行实数值试验,验证算法效果. 为了公平评判算法的性能,网络的初始权值均相同,并重复选取不同的随机初始权值进行 10 次试验,从训练及测试误差(函数拟合)或精度(分类)、训练时间几个方面来评价试验结果(结果为平均值). 因为测试时间仅与隐藏层节点数有关,与算法无关,所以在这里不做比较.

3.1　数据库介绍

Sin C 函数表达式为:

$$y(x) = \begin{cases} \dfrac{\sin x}{x}, & x \neq 0 \\ 1, & x = 0 \end{cases}. \tag{24}$$

Sin C 数据的产生方法为在(-10,10)内随机产生 5 000 组训练样本和 5 000 组测试样本. 通过网络在训练集及测试集上的误差可以衡量网络的学习能力.

MNIST 手写数字数据库也是一种衡量分类算法性能的数据库,它源于美国国家标准与技术局收集的 NIST 数据库,数字图像已被标准化为一张 28×28 的图片. 该数据库以矩阵的形式存放,其中每个样本即每个手写数字都是一个 1×784 的向量,向量中的每个元素都是 0~255 的数字,代表的是该像素点的灰度值. MNIST 数据库一共有 60 000 个训练样本和 10 000 个测试样本.

3.2　试验结果

在 Sin C 数据库上的试验选取网络的隐层节点数为 30,在 Diabetes 数据库上的试验选取网络的隐层节点数为 150,在这两个数据库上 USA 和笔者算法的迭代次数为 10 次. 训练及测试结果见表 1 和表 2. 可以看出,ELM 算法训练时间最短,符合 ELM 算法原理. 笔者算法在训练时间与 USA 算法相当的情况下,训练及测试误差是 3 种算法中最小的,精度是 3 种算法中最高的.

由于 MNIST 数据库较大,分别选取隐层节点个数为:64、128、256、512、1 024 进行试验,计算相应的分类精度. USA和笔者算法的迭代次数为 15 次. 训练及测试结果见表 3. 可以看出在时间上本算法虽然没有优势,但是精度相比于其他两种算法有所提高.

表 1　不同算法在 Sin C 数据库上的误差比较

Tab. 1　Error comparisons on Sin C for different algorithms

算法	训练误差	测试误差	训练时间/s
ELM	0.129 0	0.056 0	0.087 2
USA	0.128 2	0.055 1	0.123 4
笔者算法	0.121 5	0.036 5	0.123 7

表 2　不同算法在 Diabetes 数据库上的精度比较

Tab. 2　Accuracy comparisons on Diabetes for different algorithms

算法	训练精度/%	测试精度/%	训练时间/s
ELM	79.17	78.64	0.015 4
USA	83.85	79.17	0.154 3
笔者算法	85.42	80.21	0.153 9

表 3　不同算法在 MNIST 数据库上的误差比较

Tab. 3　Error comparisons on MNIST for different algorithms

隐层节点	算法	训练精度/%	测试精度/%	训练时间/s
64	ELM	67.13	67.88	1.251 4
	USA	85.80	84.27	64.975 0
	笔者算法	86.70	85.89	64.884 0
128	ELM	78.32	78.99	1.916 2
	USA	89.71	88.06	189.800 0
	笔者算法	90.40	89.62	191.160 0
256	ELM	85.08	85.55	3.462 2
	USA	92.97	90.82	326.590 0
	笔者算法	93.35	92.68	330.790 0
512	ELM	89.50	89.65	6.694 3
	USA	95.18	93.79	608.980 0
	笔者算法	96.03	95.58	613.320 0
1 024	ELM	92.32	94.68	13.847 0
	USA	98.93	97.86	1 128.750 0
	笔者算法	98.99	98.95	1 137.270 0

从表 3 中一方面可以看出,若要达到相同的测试精度,如 90%,ELM 大约需要 512 个隐节点,USA 算法大约需要 128 个隐节点,而笔者算法大约需要 90 个隐层节点. 说明笔者算法在保证相同精度的前提下,可有效地减少了隐层节点数,简化了网络结构,增强了网络的泛化能力. 另一方面,对于相同的隐层节点个数,ELM、USA 和笔者算法的测试精度依次增高,且 USA 算法和笔者算法的精度均明显高于 ELM 算法. 说明笔者算法在保证

训练时间不是太长的情况下,有效地优化了初始权值,使网络在相同的隐层节点个数下,得到更高的训练精度.

4 结论

针对前馈神经网络中传统算法所需时间过长,而 ELM 算法所需要的隐层节点数过多的缺陷,笔者算法在保证算法训练速度的前提下,有效提高了网络的精度和泛化能力.从各类数据库的试验可以证明,笔者算法是一个更有效的单隐层前馈神经网络学习算法.

参考文献:

[1] WERBOS P J. Beyond regression: new tools for prediction and analysis in the behavioral sciences[D]. Harvard University, Cambridge,1974.

[2] HORNIK K. Approximation capabilities of multilayer feedforward networks[J]. Neural networks, 1991, 4(2): 251 - 257

[3] LESHNO M, LIN V Y, PINKUS A, et al. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function[J]. Neural networks, 1993, 6(6): 861 - 867.

[4] HUANG G B, ZHU Q Y, SIEW C K. Extreme learning machine: theory and applications[J]. Neurocomputing, 2006, 70(123): 489 - 501.

[5] BARTLETT P L. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network[J]. IEEE trans. Inf. theory, 1998, 44(2): 525 - 536.

[6] WIDROW B, GREENBLATT A, KIM Y, et al. The No-Prop algorithm: a new learning algorithm for multilayer neural networks[J]. Neural networks, 2013(37): 182 - 188.

[7] 郝向东, 毛晓波, 梁静. ELM 与 Mean Shift 相结合的抗遮挡目标跟踪算法[J]. 郑州大学学报(工学版), 2016, 37(1):1 - 5.

[8] 王杰, 苌群康, 彭金柱. 极限学习机优化及其拟合性分析[J]. 郑州大学学报(工学版), 2016, 37(2):20 - 24.

[9] 邓万宇, 李力, 牛慧娟. 基于 Spark 的并行极速神经网络[J]. 郑州大学学报(工学版), 2016, 37(5):47 - 56.

[10] ZHU Q Y, QIN A K, SUGANTHAN P N, et al. Evolutionary extreme learning machine[J]. Pattern recognition, 2005, 38(10): 1759 - 1763.

[11] YU D, DENG L. Efficient and effective algorithms for training single-hidden-layer neural networks[J]. Pattern recognition letters, 2012, 33(5):554 - 558.

[12] HUANG G B, BABRI H A. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions[J]. IEEE transactions on neural networks, 1998, 9(1): 224 - 229.

[13] HUANG G B. Learning capability and storage capacity of two hidden-layer feedforward networks[J]. IEEE transactions on neural networks, 2003, 14(2): 274 - 281.

[14] ARMIJO L. Minimization of functions having Lipschitz continuous first partial derivatives[J]. Pacific journal of mathematics, 1966(16):1 - 3.

[15] GOLDSTEIN A. On steepest descent[J]. SIAM journal on control, 1965(3):147 - 151.

An Efficient Generalized Single Hidden Layer Neural Networks Based on Conjugate Gradient Method

SUN Feng, GONG Xiaoling, ZHANG Bingjie, LIU Yusong, WANG Yanjiang

(China University of Petroleum, 266580, China)

Abstract: The single hidden layer feedforward neural network was efficient with simple structure. Back Propagation Error (BP) algorithm was one of its typical learning algorithm which had one main shortcoming of the slow learning speed because of the use of the steepest descent method. Extreme Learning Machine (ELM) which could greatly accelerate the learning speed of networks was put forward. However, it demanded much more hidden neurons than BP algorithm to get the match accuracy, which led to redundant structure of networks and more testing time. Motived by the USA (upper-layer-solution-aware) which was a combination of the steepest descent method and ELM, in this paper, we proposed an algorithm based on the conjugate gradient method and train the network on different data sets. The Simulation results showed our algorithm had a better performance than USA and ELM with the same structure of the network.

Key words: neural network; back propagation; extreme learning machine; conjugate gradient; Mnist