基于构造性设计和 GA的 ANN建模方法

张 胜[1], 王 蔚[2], 刘红星[3], 余水宝[1]
(1. 浙江师范大学数理与信息工程学院, 浙江 金华 321004)
(2. 南京师范大学教育科学学院, 江苏 南京 210097)
(3. 南京大学电子科学与工程系, 江苏 南京 210093)

[摘要] 强调了激活函数在 ANN设计中的重要性, 提出一种基于构造性设计及 GA的网络结构及神经元激活函数类型自动优化的 ANN模型 (constructived and GA based activation function, 简称为 CGBAF), 并给出其一般形式和算法. 本模型用于多层前向神经网络时, 其网络结构及激活函数类型可自动优化, 进而可大大提高 ANN的泛化能力. 通过例子验证了本方法的有效性, 并进行了分析.

[关键词] 人工神经网络, 构造性设计, GA, 激活函数类型优化
[中图分类号] O233 [文献标识码] A [文章编号] 1001-4616(2008)04-0061-05

An ANN Model Based on Constructive Algorithm and GA

Zhang Sheng[1], Wang Wei[2], Liu Hongxing[3], Yu Shuibao[1]
(1. College of Mathematics, Physics and Infomation Engineering, Zhejiang Normal University, Jinhua 321004, China)
(2. School of Educational Science, Nanjing Normal University, Nanjing 210097, China)
(3. Department of Electronic Science and Engineering, Nanjing University, Nanjing 210093, China)

Abstract The importance of the activation functions in ANN is emphasized. A new ANN modeling method is proposed based on constructive algorithm and GA. This method can be used to realize the automatic optimization of the net structure and the types of the activation functions. As a result, the ANN's generalization capability is greatly improved. This improvement is verified experimentally.

Key words ANN, constructive algorithm, GA, activation functions tapes

人工神经网络 (ANN)是从神经元模型、神经元连接模型和学习模型等三个方面对人脑进行模拟的一种人工智能方法. 目前, ANN技术已有了相当的发展, 各种 ANN的模型和算法不断涌现, 并得到了广泛的应用.

ANN在训练完成后输入其训练样本之外的新数据时获得正确输出的能力称为泛化能力. 泛化能力在 ANN的应用过程中表现出来, 但由 ANN的设计过程所决定.

文[1]指出, 目前 ANN泛化能力改善方法都有合理的一面, 都有其产生的背景, 因此, 它们在一定的条件下都会对提高所设计 ANN的泛化能力起到积极的作用, 但是他们对较为复杂的问题还不能奏效, 网络泛化问题仍然十分突出; 文献[2]也指出了 ANN外推泛化能力弱的问题.

要得到好的 ANN泛化能力, 必须对网络进行完整的彻底的学习优化才可能达到目的. 而完整的彻底的 ANN学习应该包括网络权值大小的学习、拓扑结构的学习、神经元模型的学习等. 但是, 目前很少有讨论如何对神经元模型进行自动学习的文献, 在传统 ANN设计中, 神经元模型一经人工选定就固定不变, 而且同层的神经元模型总是选的相同.

论文[3]对神经网络激活函数的重要性进行了研究. 提出了一种优化神经元激活函数类型的方法: 根据先验知识人为选择若干基本激活函数类型, 以基本激活函数类型的线性组合作为各神经元的初始激活函数, 然后用 BP算法实现包括激活函数类型在内的网络学习. 该方法相当于在学习过程中从由若干基本激活函数类型组成的函数库中优选出各神经元的激活函数类型, 它具有训练后的 ANN各神经元激活函数

收稿日期: 2008-10-09
基金项目: 国家自然科学基金 (59905011)、浙江省自然科学基金 (Y207738)资助项目.
通讯联系人: 张 胜, 博士, 副教授, 研究方向: 电子建模. E-mail: zs@zjnu.cn

类型不一定相同这样的效果. 但从本质上说该方法还是不能优化出所给基本激活函数类型以外的新的激活函数类型; 另激活函数异常复杂的初始化会导致学习误差函数的异常复杂和局部最小问题的突出, 提出的 BP算法在学习中肯定会遇到收敛的麻烦.

本文提出基于构造性设计的以自动优化神经元激活函数类型为核心的网络泛化能力的改善方法.

优化 ANN的拓扑结构是改善 ANN特性的重要手段之一, 现有的调整 ANN拓扑结构以匹配任务复杂度的努力大概包括三个方面. 其一是量化法, 即想办法精确度量一个网络的与任务无关的拟合能力及一个任务的与网络无关的复杂度, 并将二者进行比较以调整匹配[4]; 其二是逐步增大法, 即先将网络初始化成非常简单, 然后依据某种准则自动地逐步增加神经元个数以匹配[5]; 其三是自动修剪法, 即将网络尺寸初始成足够大, 再依某种准则逐步地修剪冗余权值或神经元[6].

现有的方法中, 构造性设计方法[7-9]是比较理想的方法, 初始化易操作, 计算量小[8]. 所谓构造性设计方法, 即先将网络的结构初始得非常简单, 然后根据需要在训练中逐步增加权值、神经元甚至隐层个数, 从而获得有利于改善 ANN泛化特性的最紧凑的结构. 因而也不易出现 ANN过拟合现象、对改善 ANN泛化特性有利, 但构造性设计方法还不能从根本上解决上述 ANN泛化能力差的问题.

把新的优化神经元激活函数类型的方法和 ANN构造性设计结合起来, 提出了在 ANN构造性设计框架下基于 GA实现神经元激活函数类型自动优化的思路.

1　基于构造性设计及 GA自动优化网络结构及激活函数类型的实现

1.1　ANN构造性设计的原理与实现

文献[10]提出, 输出层为线性层、隐层神经元激活函数类型为 S型函数的单隐层 FNN能够实现任意复杂的映射. 当隐层激活函数类型在包括 S型函数在内的更大函数空间内优化产生时, 该结论自然仍成立. 另外, 输出为 n维的 FNN可分解为 n个输出为一维的 FNN. 因此, 只讨论典型的一维输出且输出层线性的单隐层 FNN的情况.

考虑如下的多维输入、一维输出的函数拟合问题: 已知 m个 n维输入矢量构成输入样本集 $p = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m\}$, 与之对应的 m个输出标量构成输出目标集 $T = \{y_1, y_2, \cdots, y_m\}$. 以这些数据为训练集, 训练一个输出层线性的单隐层 FNN实现函数拟合.

若 FNN的结构和参数如图 1所示, 则整个网络的计算模型可描述为式(1):

$$y = \sum_{i=1}^{N} w_{2i} f_i(w_{1i} \cdot \boldsymbol{x} - b_{1i}) - b_2, \tag{1}$$

其中, $\boldsymbol{x}$为 n维输入矢量, y为一维输出, N为隐层神经元的个数, $f_i(\cdot)$, w_{1i}, b_{1i}分别为第 i个隐层神经元的激活函数、输入权值矢量和偏置, w_{2i}为输出层与第 i个隐层神经元间的连接权值, b_2为输出层的偏置.

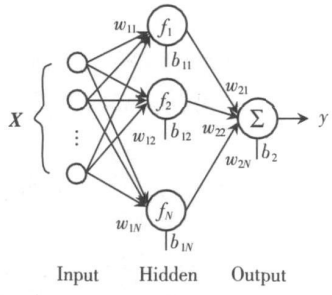

图 1　前馈神经网络的结构和参数

Fig.1　Structure and parameters of feed-forword neural network

令 $b_2 = \sum_{i=1}^{N} b_{2i}$, 则(1)式可进一步表示为:

$$y = \sum_{i=1}^{N} (w_{2i} f_i(\vec{w}_{1i} \cdot \boldsymbol{x} - b_{1i}) - b_{2i}) \tag{2}$$

其中, b_{2i}为输出层的偏置的分解. 再设

$$y_i = w_{2i} f_i(\vec{w}_{1i} \cdot \boldsymbol{x} - b_{1i}) - b_{2i}, \quad i = 1, \cdots, N \tag{3}$$

则有

$$y = \sum_{i=1}^{N} y_i \tag{4}$$

这样, FNN就可分解成如(4)式所示的 N个神经元的组合, 每个神经元的待定参数包括 w_{1i}, b_{1i}, w_{2i}, b_{2i}和激活函数 f_i的类型及其本身的参数.

显然, 至此 FNN的设计问题分解成了各单个神经元的优化问题. 在构造性设计的框架下, 我们可以采

用如下的步骤逐个地设计每个神经元 (包括激活函数类型在内):

(1) 生成第一个神经元, 待定参数见式 (3), 将初始的 P 和 T 设为当前训练集.

(2) 用当前训练集训练新生成的神经元 (本文将基于 GA进行), 直至训练误差下降趋势变缓时停止. 若训练停止时训练误差达到要求, 整个设计过程结束, 否则转 (3).

(3) 对刚结束训练的神经元, 分别计算各输入样本 x_1, x_2, …, x_m 的实际输出 Y_1, Y_2, …, Y_m, 并将 T 修正为 $\{y_j = y_j - Y_j,\ j = 1, \cdots, m\}$.

(4) 生成一个新的神经元, 待定参数见式 (3), 将 P 和修正后的 T 设为当前训练集, 转 (2).

1.2　基于 GA 优选神经元激活函数类型的实现

ANN 构造性设计中基于 GA 优选神经元激活函数类型意思是: 在上述的 ANN 构造性设计原理中, 基于 GA逐个优化式 (3) 所示的单个神经元的所有的参数, 包括从一个激活函数类型库中优选一个激活函数类型.

由于单个神经元的参数不是太多, 因此遗传算法这种不易陷入局部最小的优化方法可以在此找到用武之地. 遗传算法的引入还有一个特别的好处: 通过建立一个激活函数类型库, 可以实现神经元激活函数类型的自动优选. 本文的设计如下:

(1) 建立激活函数类型库

Hornik在 [11] 中提出, 任何连续、有界、非常值的函数都可以作为激活函数. 我们可以把任何满足上述条件的函数类型加入到激活函数类型库中, 如可以把常用的 S型函数类型、正弦型函数类型、幂函数类型等放入.

(2) GA 的神经元个体

由 (3) 式可见, 一个神经元的参数包括: w_{1i}, b_{1i}, w_{2i}, b_{2i} 和激活函数 f_i 的类型及其本身的参数. 一个神经元的所有参数构成 GA 的一个个体; 一个参数即为个体的一个变量或者说一个基因位.

(3) 目标函数与适合度函数

一般的, 训练目标是理想输出与实际输出的均方误差 E_{sn} 最小. 由于正则化方法能够有效地提高泛化性能, 因此可以对目标函数进行修正. 本文将样本点处输出对输入的导数的均方值最小作为正则化项对目标函数做了修正, 修正后的目标函数为 $\min F(f, w_1, b_1, w_2, b_2, \lambda) = \min(E_{sn} + \lambda \cdot P_d)$, 其中 $P_d = \sum_{j=1}^{n} Y_j'^2 / n$, λ为拉格朗日乘数. 适合度函数可定义为 $FI = C - F$, 其中 C 为一常数. 本文后面的实验直接用目标函数进行个体评价.

(4) GA 进化策略

本文采用两种进化算子: 一般算子和精化算子. 精化算子只是对父代中的优质个体进行激活函数类型不变的小范围参数调整优化, 一般用于优化的最后阶段; 一般算子则会产生与父代中的个体完全不同的新个体 (包括激活函数类型), 旨在进行大范围搜索, 一般用在最后优化阶段之前.

2　实验验证

为了便于研究, 我们选取最常见的 3个函数构成函数类型库.

(Ⅰ) S型函数 $y = 2/(1 + \exp(-2x)) - 1$

(Ⅱ) 幂函数 $y = x^u$

(Ⅲ) 正弦函数 $y = \sin(x)$

下面分别用本文方法和其他一些算法来训练 FNN, 进行函数拟合, 并比较其性能. 这些方法包括: Broyden等提出的 BFGS算法[12], DavidMacKay提出的 BR (BayesianRegularization) 算法[13], Moller提出的 SCG (scaled conjugate gradient) 算法[14].

测试时, 分别以激活函数类型库中所包含的函数及其线性组合作为待拟合函数. 这些函数包括: ① $Y = \sin(x)$, ② $Y = 0.5\sin(x) + x$, ③ $Y = 0.1\sin(x) + \tansig(x)$, ④ $y = 5\tansig(x) + x^2$. 所用的测试数据如表 1所示.

表 1　测试用数据

Table 1　Test data

待拟合函数	训练集	内插测试集	外推测试集 1	外推测试集 2
①②③④	$x = 0:0.1:8\pi$	$x = 0:0.05:8\pi$	$x = 8\pi:0.1:12\pi$	$x = 12\pi:0.1:16\pi$

(1) 测试 Ⅰ

首先测试网络大小相同的情况下各个方法的泛化性能. 先用本文方法进行构造性设计, 当目标函数减小的趋势变得很缓慢时停止训练, 记录下所用隐层神经元的个数, 然后以此为基准, 用其他方法, 选取同样数目的隐层神经元进行函数拟合. 几种方法的测试结果如表 2所示.

表 2　网络大小相同时几种方法的性能比较

Table 2　Performance comparison under different network size

拟合函数	训练算法	训练误差	内插误差	外推误差 1	外推误差 2	隐层神经数
①	BFGS	4.38e-1	4.38e-1	5.07e-1	5.07e-1	1个
	BR	4.43e-1	4.43e-1	5.07e-1	5.07e-1	1个
	SCG	4.38e-1	4.38e-1	5.07e-1	5.07e-1	1个
	本文方法	3.21e-14	3.19e-14	1.04e-13	3.44e-13	1个
②	BFGS	1.10e-1	1.10e-1	4.45	6.05e+1	2个
	BR	1.10e-1	1.10e-1	5.17	6.86e+1	2个
	SCG	1.11e-1	1.11e-1	3.64	5.01e+1	2个
	本文方法	2.53e-03	2.72e-03	2.09e-03	2.85e-03	2个
③	BFGS	4.03e-3	4.04e-3	4.99e-3	4.99e-3	2个
	BR	4.37e-3	4.37e-3	5.07e-3	5.07e-3	2个
	SCG	4.45e-3	4.46e-3	5.02e-3	5.02e-3	2个
	本文方法	3.23e-04	3.23e-04	1.39e-04	3.11e-04	2个
④	BFGS	2.90e-07	2.93e-07	5.39e-01	1.61e+02	12个
	BR	6.98e-05	6.82e-08	1.29e+03	2.94e+05	12个
	SCG	7.90	7.79	4.18e+04	8.83e+05	12个
	本文方法	5.99e-04	1.16e-03	1.74e+01	3.36e+02	12个

显然, 网络大小相同的情况下, 在拟合 ①②③时, 本文方法的误差比其他方法都小很多, 尤其是两个外推误差, 这说明此时本文方法的泛化性能很好. 另一方面, 在拟合 ④时, 相对于其他方法, 本文方法的训练误差和内插误差比较大, 外推误差则稍好一点.

(2) 测试 Ⅱ

接着测试在达到相同训练误差的情况下各个方法的泛化性能及网络大小. 先用本文方法进行训练, 记录下所达到的训练误差的量级, 然后以此为基准, 用其他方法, 调整网络大小使训练误差达到或者小于该量级. 几种方法的测试结果如表 3所示.

可以看出, 在误差性能方面, 测试 Ⅱ 与测试 Ⅰ 的结果相同, 即拟合 ① ② ③时本文方法泛化性能很好, 但拟合 ④时性能一般. 同时, 在隐层神经元个数方面, 拟合 ① ② ③时, 本文方法仅用了 1 ~ 2个隐层神经元, 其他方法都用了 5个以上的隐层神经元; 但拟合 ④时, 本文方法用到的隐层神经元略多于其他方法.

(3) 实验结果分析

从测试结果中可以看出, 拟合 ①②③时, 本文方法各方面性能远远优于其他方法, 拟合 ④时, 本文方法性能与其他方法差不多. 对于拟合 ①而言, 显然是激活函数类型库中包含待拟合函数 $y = \sin(x)$ 的缘故, 但这也说明了激活函数类型的优选对 ANN 性能有着很大的影响, 说明了对激活函数进行选择的必要性. 对于拟合 ②③④而言, 待拟合函数不再是像 $y = \sin(x)$ 一样的简单函数, 而是激活函数类型库中所包含的函数的线性组合. 仔细分析这 3个待拟合函数可以看出, ②中的主要分量是 $y = x$, 分量 $y = 0.5\sin(x)$ 相对来说很小. 同样的, ③中的主要分量是 $y = \tan\mathrm{sig}(x)$, 分量 $y = 0.1\sin(x)$ 很小. 而 ④中的两个分量都是差不多大小. 在拟合 ②③时, 由于待拟合函数中主分量很明显, 因此本文方法能够在设计第一个神经元时就正确的优选出与主分量相一致的激活函数, 在此基础上设计的 FNN 泛化性能自然很好. 而在拟合 ④时, 由于待拟合函数中包含 2个没有明显区别的分量, 在逐个设计神经元时, 本文方法无法正确的优选出激活函数, 造成泛化性能不理想. 这同样显示出了优选神经元激活函数的重要性.

表 3　相同训练误差时几种方法的性能比较

Table 3　Performance comparison under the same training errors

拟合函数	训练算法	训练误差	内插误差	外推误差 1	外推误差 2	隐层神经元数
①	BFGS	难以达到 - 14量级				
	BR					
	SCG					
	本文方法	3.21e-14	3.19e-14	1.04e-13	3.44e-13	1个
②	BFGS	5.07e-03	5.08e-03	5.33	1.53e+02	8个
	BR	1.87e-03	1.87e-03	4.13e+01	3.34e+02	5个
	SCG	2.51e-05	2.47e-05	3.48e+01	3.12e+02	16个
	本文方法	2.53e-03	2.72e-03	2.09e-03	2.85e-03	2个
③	BFGS	1.37e-05	1.36e-05	6.31e+01	9.52e+02	6个
	BR	1.17e-05	1.16e-05	1.67e+01	1.30e+02	6个
	SCG	6.94e-05	6.78e-05	7.28e-03	7.07e-03	8个
	本文方法	3.23e-04	3.23e-04	1.39e-04	3.11e-04	2个
④	BFGS	6.30e-04	5.77e-04	1.27e+01	2.29e+03	5个
	BR	4.49e-11	4.41e-11	2.33e-02	6.21e+01	5个
	SCG	难以达到 - 4量级				
	本文方法	5.99e-04	1.16e-03	1.74e+01	3.36e+02	12个

3　总结

本文提出了一种基于 GA和构造性设计的神经元激活函数类型及网络结构可自动优化的 ANN设计方法. 经验证其有较好的泛化能力及简捷的拓扑结构, 同时也能较好的解决 ANN的过拟合问题.

诚然, 实验中由于激活函数类型库中函数类型较少, 同时也简化了神经元之间的协同作用, 对有些复杂问题难以实现真正的激活函数类型优化有待进一步的研究, 可考虑引进 GP等优化方法实现自动优化出新的激活函数类型, 这样效果会更好.

[参考文献]

[1] Zhang Sheng. Surveying the methods of improving ANN generalization capability[C]// Proceedings of the Second International Conference on Machine Learning and Cybernetics. Xi'an, 2003: 1259-1263.

[2] 张建勋, 贺京同, 王维, 等. 人工神经网络对时间增长序列预测能力分析[J]. 预测, 1999, 5: 60-63.

[3] 吴佑寿, 赵明生. 激活函数可调的神经元模型及其有监督学习的应用[J]. 中国科学(E辑), 2001, 31(3): 263-272.

[4] Wang Z O, Zhu T. An efficient learning algorithm for improving generalization performance of radial basis function neural networks[J]. Neural Networks, 2000, 13(4/5): 545-553.

[5] 鲁子弈, 杨绿溪, 无球, 等. 提高前馈神经网络泛化能力的新算法[J]. 电路与系统学报, 1997, 2(4): 7-12.

[6] 彭汉川, 甘强, 韦钰. 提高前馈神经网络推广能力的若干实际方法[J]. 电子学报, 1998, 26(4): 116-119.

[7] Giles C L. Constructive learning of recurrent neural networks: problems with recurrent cascade correlation and a simple solution[J]. IEEE Transactions on Neural Networks, 1995, 6(4): 829-836.

[8] Kwok T Y, Yeung D Y. Constructive algorithms for structure learning in feedforward neural networks for regression problems[J]. IEEE Transactions on Neural Networks, 1997, 8(3): 630-645.

[9] Treadgold N K, Gedeon T D. Exploring constructive cascade networks[J]. IEEE Transactions on Neural Networks, 1999, 10(6): 1335-1350.

[10] Cybenco G. Approximations by superpositions of a sigmoid function[J]. Math Cont Signal Syst, 1989, 2: 303-314.

[11] Hornik K. Approximation capabilities of multilayer feedforward networks[J]. Neural Networks, 1991, 4: 251-257.

[12] MacKay. Bayesian interpolation[J]. Neural Computation, 1992, 4(3): 415-447.

[13] Foresee F D, Hagan M T. Gauss-Newton approximation to Bayesian regularization[C]// Proceedings of the 1997 International Joint Conference on Neural Networks. New York, 1997: 1930-1935.

[14] Moller M F. A scaled conjugate gradient algorithm for fast supervised learning[J]. Neural Networks, 1993, 6: 525-533.

[责任编辑: 顾晓天]