具有增加删除机制的正则化极端学习机的混沌时间序列预测*

赵永平† 王康康

(南京理工大学机械工程学院, 南京 210094)

(2013 年 8 月 13 日收到; 2013 年 9 月 17 日收到修改稿)

针对正则化极端学习机的隐层具有随机选择的特性, 提出了一种增加删除机制来自适应地确定正则化极端学习机的隐层节点数. 这种机制以对优化目标函数影响的大小作为评价隐层节点优劣的标准, 从而淘汰那些比较 "差" 的节点, 将那些比较 "优" 的节点保留下来, 起到一个优化正则化极端学习机隐层节点数的目的. 与已有的只具有增加隐层节点数的机制相比较, 本文提出的增加删除机制在减少正则化极端学习机隐层节点数、增强其泛化性能、提高其实时性等方面具有一定的优势. 典型混沌时间序列的实例证明了具有增加删除机制的正则化极端学习机的有效性和可行性.

关键词: 混沌时间序列, 人工神经网络, 极端学习机

PACS: 05.45.Tp, 05.45.−a, 02.50.Ey **DOI:** 10.7498/aps.62.240509

1 引 言

混沌现象是自然界中广泛存在的一种不规则运动, 诸如脑电波信号、地震波、太阳黑子、大气运动等都可以较好地运用混沌理论来解释 [1]. 混沌具有内随机性、整体稳定局部不稳定、短期可预测而长期不可预测性等特征 [2]. 随着混沌理论研究的不断深入及其在电力、金融、生物医学及短期气候预测等领域中的应用, 混沌时间序列的建模和预测已成为混沌信息处理领域中的一个研究热点. 目前, 已有多种方法被应用到混沌时间序列的建模和预测中. 在这众多的方法当中, 人工神经网络因其具有较强的自适应性、自学习能力与非线性映射能力而倍受重视, 像模糊神经网络、径向基函数神经网络、回声状态网络等都已被应用于混沌时间序列建模和预测研究中, 取得了较好的效果 [1−5].

近年来, Huang 等 [6−8] 提出了一种新颖的单隐层前馈神经网络极端学习机 (extreme learning machine, ELM), 并且将 ELM 应用于混沌时间序列建模和预测也取得了一些研究成果 [9−11]. 最近, 一种确定正则化 ELM (regularized ELM, RELM)[12] 隐层节点数目的方法被提了出来, 并且该方法借助于 Choleskey 分解加快其求解速度, 在本文将这种方法简记为 CRELM (Choleskey-assisted RELM)[13]. CRELM 选择隐层节点数目的方式是根据 RELM 优化目标的变化规律来确定的. 随着隐层节点数目的增加, RELM 的优化目标的数值逐渐减小, 当其变化比较缓慢或在可接受范围内时, 就可以认为 RELM 的结构已经饱和, 从而达到自动确定 RELM 隐层节点数目的目的.

在机器学习领域有一个非常著名的用来设计机器学习算法的一般化哲学原则, 即 Occam 剃刀原理 (Occam's razor)[14], 它忠告我们: 除非必要, "实体" 不应该随便增加. "实体" 在这里可以理解为 RELM 的隐层节点. 根据 Occam 剃刀原理, 如果有更好的优化 RELM 目标的方法, 其隐层节点数目就不应该随便增加. 既然 RELM 隐层节点都是随机选取的, 那么新增加的隐层节点并不一定比以前选取的隐层节点差. 在新增加隐层节点后, 可

* 国家自然科学基金 (批准号: 51006052) 和南京理工大学 "卓越计划" "紫金之星" 资助的课题.

† 通讯作者. E-mail: y.p.zhao@163.com

以对所有的隐层节点根据它们对 RELM 优化目标贡献值的大小进行评价. 当新增加的节点不是所有节点中最差的时, 可以将最差节点从当前网络中移除出去, 这样做的好处一是维持 RELM 隐层节点数目不变, 增加一个节点又剔除一个节点, 保持了其实时性 (实时性与隐层节点数成正比), 二是可以优化 RELM 的目标函数, 增强 RELM 的学习能力和泛化性能, 这种情况相当于用 "好" 的节点替换掉 "坏" 的节点, 致使 RELM 隐层中的节点越来越 "好", 达到优化 RELM 的目的; 当新增加的节点是隐层中最 "坏" 的节点时, 这时为了达到进一步优化 RELM 目标函数的目的就不得不增加新的节点了. 由此可见, 这种具有增加删除机制的 RELM (add-delete based RELM, AdRELM) 更加符合 Occam 剃刀原理. 与 CRELM 相比较, AdRELM 不仅可以自动确定 RELM 隐层节点数目, 而且具有优化隐层节点的功能, 这就会使 AdRELM 在网络泛化性能、结构规模、实时性等方面具有一定的优势. 另外, 为了提高 AdRELM 的计算效率, 可以采用 Sherman-Morrison 迭代机制降低其计算量. 为了验证所提 AdRELM 算法的有效性和可行性, 最后用一些典型的例子来验证其在混沌时间序列建模和预测方面的效果.

2 RELM

为了改善 ELM 的泛化性能, 通过引入正则化项, RELM 的数学模型可以描述为

$$\min_{\alpha}\left\{J=\frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\alpha}+\frac{C}{2}\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon}\right\},$$
$$\text{s.t.}\boldsymbol{y}=\boldsymbol{H}\boldsymbol{\alpha}+\boldsymbol{\varepsilon}, \tag{1}$$

其中 $y=[y_1,y_2,\cdots,y_N]^{\mathrm{T}}$, T 代表矩阵或向量的转置,

$$\boldsymbol{H}=\begin{bmatrix} g(\boldsymbol{w}_1\cdot\boldsymbol{x}_1+b_1) & \cdots & g(\boldsymbol{w}_L\cdot\boldsymbol{x}_1+b_L)\\ \vdots & \ddots & \vdots\\ g(\boldsymbol{w}_1\cdot\boldsymbol{x}_N+b_1) & \cdots & g(\boldsymbol{w}_L\cdot\boldsymbol{x}_N+b_L)\end{bmatrix}$$

为隐层输出矩阵, $\{(\boldsymbol{x}_i,y_i)\}_{i=1}^{N}$ 代表训练样本集, $g(\cdot)$ 为隐层激活函数, $\boldsymbol{w}_i$ $(i=1,2,\cdots,L)$ 为连接隐层和输入层的权值向量, $b_i(i=1,2,\cdots,L)$ 为隐层偏置值, L 为 RELM 隐层节点的数量, $\boldsymbol{\alpha}=[\alpha_1,\alpha_2,\cdots,\boldsymbol{\alpha}_L]^{\mathrm{T}}$ 为输出权值, $\boldsymbol{\varepsilon}=[\varepsilon_1,\varepsilon_2,\cdots,\varepsilon_N]^{\mathrm{T}}$ 为训练误差, $C\in\mathbb{R}^{+}$ 为正则化参数. 通过消除误差向量 $\boldsymbol{\varepsilon}$, 将 (1) 式中的约束条件代入到目标函数 J 中, 并令 $\dfrac{\partial J}{\partial\boldsymbol{\alpha}}=0$, 可得

$$\left(\boldsymbol{I}/C+\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\right)\boldsymbol{\alpha}=\boldsymbol{H}^{\mathrm{T}}\boldsymbol{y}, \tag{2}$$

其中 $\boldsymbol{I}$ 为适维单位矩阵. 通过解 (2) 式中的线性方程组, 可求出输出权值 $\boldsymbol{\alpha}$, 于是, 可以得到 RELM 的数学模型

$$\boldsymbol{y}=f(\boldsymbol{x})=\sum_{i=1}^{L}\alpha_i g(\boldsymbol{w}_i\cdot\boldsymbol{x}+b_i). \tag{3}$$

3 AdRELM

3.1 增加节点

假如 AdRELM 中隐层节点个数已经有 L 个, 由于模型的学习能力和泛化性能还不够, 需要进一步提高, 此时需要增加新的隐层节点. 在增加完新的隐层节点后, (2) 式变为

$$\left(\boldsymbol{I}/C+\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1}\right)\boldsymbol{\alpha}_{L+1}=\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{y}, \tag{4}$$

其中

$$\boldsymbol{H}_{L+1}=[\boldsymbol{H}_L,\boldsymbol{h}_{L+1}],$$
$$\boldsymbol{h}_{L+1}=\Big[g(\boldsymbol{w}_{L+1}\cdot\boldsymbol{x}_1+b_{L+1}),\cdots g(\boldsymbol{w}_{L+1}\cdot\boldsymbol{x}_N+b_{L+1})\Big]^{\mathrm{T}},$$

$\boldsymbol{w}_{L+1}$ 为随机选取的连接新增隐层节点和输入层的权值, b_{L+1} 为随机选取的隐层新增节点的偏置量. 如果在每增加一个节点后, 都直接进行 (4) 式的求解, 这会增加 AdRELM 的计算量, 影响其求解速度. 为了在一定程度上减轻 AdRELM 的计算量, 下面介绍一种迭代策略, 它不需要进行 (4) 式的直接求解, 而是在前面求解的基础上, 通过迭代更新策略来进行 (4) 式的求解.

令

$$\boldsymbol{R}_L=\left(\boldsymbol{I}/C+\boldsymbol{H}_L^{\mathrm{T}}\boldsymbol{H}_L\right)^{-1}, \tag{5}$$
$$\boldsymbol{t}_L=\boldsymbol{H}_L^{\mathrm{T}}\boldsymbol{y}, \tag{6}$$

则

$$\boldsymbol{\alpha}_L=\boldsymbol{R}_L\boldsymbol{t}_L. \tag{7}$$

根据 Sherman-Morrison 公式 [15], 可得下面的迭代公式

$$\boldsymbol{R}_{L+1}=\begin{bmatrix}\boldsymbol{R}_L & \boldsymbol{0}\\ \boldsymbol{0}^{\mathrm{T}} & 0\end{bmatrix}+\lambda\begin{bmatrix}\boldsymbol{\beta}\\ -1\end{bmatrix}\begin{bmatrix}\boldsymbol{\beta}^{\mathrm{T}} & -1\end{bmatrix}, \tag{8}$$

其中

$$\boldsymbol{\beta}=\boldsymbol{R}^{n}\boldsymbol{H}_L^{\mathrm{T}}\boldsymbol{h}_{L+1},$$

$$\lambda = \left(\boldsymbol{h}_{L+1}^{\mathrm{T}}\boldsymbol{h}_{L+1} + 1/C - \boldsymbol{h}_{L+1}^{\mathrm{T}}\boldsymbol{H}_L\beta\right)^{-1}.$$

同时

$$\boldsymbol{t}_{L+1} = \begin{bmatrix} \boldsymbol{t}_L \\ \boldsymbol{y}^{\mathrm{T}}\boldsymbol{h}_{L+1} \end{bmatrix}. \tag{9}$$

由 (7), (8) 和 (9) 式可得

$$\begin{aligned}\boldsymbol{\alpha}_{L+1} &= \boldsymbol{R}_{L+1}\boldsymbol{t}_{L+1} \\ &= \begin{bmatrix} \boldsymbol{\alpha}_L \\ 0 \end{bmatrix} + \lambda \begin{bmatrix} \beta \\ -1 \end{bmatrix} \left(\beta^{\mathrm{T}}\boldsymbol{t}_L - \boldsymbol{y}^{\mathrm{T}}\boldsymbol{h}_{L+1}\right),\end{aligned} \tag{10}$$

运用迭代策略后, (10) 式的计算代价为 $O(L^2)$. 与线性方程组 (4) 式的直接求解代价 $O(L^3)$ 相比较, 计算量明显有所降低. 另外, 令

$$\boldsymbol{H}_{L+1} = [\boldsymbol{H}_L, \boldsymbol{h}_{L+1}], \tag{11}$$

在迭代求解出 $\boldsymbol{\alpha}_{L+1}$ 后, 就可以构造出新的 RELM 网络以期增加其泛化能力.

3.2 判断节点的优劣

在 CRELM 算法中, 已经给出了一种朴素的衡量隐层节点作用的标准. 当隐层节点对目标函数 J 的影响较大时, 就可以认为该节点比较重要, 反之, 则认为该节点不是那么重要, 从而给出一种衡量隐层节点 "优劣" 的标准. 其数学表达式为

$$\sigma_i = J_{L+1}^{(-i)} - J_{L+1}, \tag{12}$$

其中 $J_{L+1}^{(-i)}$ 为当第 i 个隐层节点去掉后, RELM 目标函数值的大小. 当通过 (10) 式计算出 $\boldsymbol{\alpha}_{L+1}$ 后, 就可以比较方便地求出 J_{L+1} 的大小. 然而直接根据 (1) 式计算 $J_{L+1}^{(-i)}$ 就比较麻烦, 需要计算 $L+1$ 次, 这当然是一个不明智的选择. 下面给出一种快速计算 σ_i 的方法.

当 RELM 有 $L+1$ 个隐层节点时, 将 (10) 式代入到 (1) 式中, 可得

$$\begin{aligned}J_{L+1} = \frac{1}{2C}\Big(&\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{y}^{\mathrm{T}}\boldsymbol{H}_{L+1} \\ &\times \left(\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C\right)^{-1}\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{y}\Big).\end{aligned} \tag{13}$$

然后, 考虑下面的优化问题

$$\begin{aligned}\min_{\tilde{\boldsymbol{\alpha}}}\Big\{\tilde{J} = &\frac{1}{2}\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\left(\boldsymbol{I} + \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}\right)\tilde{\boldsymbol{\alpha}} \\ &+ \frac{C}{2}\left\|\boldsymbol{y} - \boldsymbol{H}_{L+1}\tilde{\boldsymbol{\alpha}}\right\|^2\Big\}\end{aligned} \tag{14}$$

其中 $\theta_{ii} > 0$, 令 $\boldsymbol{u} = \sqrt{\theta_{ii}}\boldsymbol{e}_i$, $\boldsymbol{e}_i$ 为 $L+1$ 维单位矩阵的第 i 列. 令 $\dfrac{\partial \tilde{J}}{\partial \tilde{\boldsymbol{\alpha}}} = 0$, 得

$$\begin{aligned}\tilde{\boldsymbol{\alpha}}_{L+1} = \Big(&\boldsymbol{y}\boldsymbol{I}/C + \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}/C \\ &+ \boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1}\Big)^{-1}\boldsymbol{H}_{L+1}^{\mathrm{T}}.\end{aligned} \tag{15}$$

将 (15) 式代入到 (14) 式中, 可得

$$\begin{aligned}\tilde{J}_{L+1} = \frac{1}{2C}\Big(&y^{\mathrm{T}}y - y^{\mathrm{T}}\boldsymbol{H}_{L+1} \\ &\times \left(\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C + \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}/C\right)^{-1} \\ &\times \boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{y}\Big).\end{aligned} \tag{16}$$

从 (14) 式中可知, 当 $\theta_{ii} \to +\infty$ 时, $\tilde{\boldsymbol{\alpha}}_i \to 0$, 这就相当于从 $L+1$ 个隐层节点中去掉第 i 个节点后 RELM 所要优化的数学模型, 此时 (16) 式就退化为 $J_{L+1}^{(-i)}$. 令

$$\tilde{\sigma}_i = \tilde{J}_{L+1} - J_{L+1}, \tag{17}$$

则

$$\sigma_i = \lim_{\theta_{ii}\to+\infty}\tilde{\sigma}_i. \tag{18}$$

将 (13) 和 (16) 式代入到 (17) 式中, 可得

$$\begin{aligned}\tilde{\sigma}_i = \frac{1}{2C}&\boldsymbol{y}^{\mathrm{T}}\boldsymbol{H}_{L+1}\Big(\left(\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C\right)^{-1} \\ &- \left(\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C + \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}/C\right)^{-1}\Big) \\ &\times \boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{y}.\end{aligned} \tag{19}$$

根据 Sherman-Morrison 公式 [15], 可得

$$\begin{aligned}\tilde{\sigma}_i = \frac{1}{2C}&\Big[\boldsymbol{y}^{\mathrm{T}}\boldsymbol{H}_{L+1}\left(\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C\right)^{-1}\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}} \\ &\times \left(\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C\right)^{-1}\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{y}\Big] \\ &\times \Big[C + \boldsymbol{u}^{\mathrm{T}}\left(\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C\right)^{-1}\boldsymbol{u}\Big]^{-1}.\end{aligned} \tag{20}$$

由于

$$\boldsymbol{\alpha}_{L+1} = \left(\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C\right)^{-1}\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{y}, \tag{21}$$

将 (21) 式代入到 (20) 式中, 可得

$$\tilde{\sigma}_i = \frac{1}{2C}\frac{\theta\boldsymbol{\alpha}_{L+1}^{\mathrm{T}}\boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{e}_i\boldsymbol{\alpha}_{L+1}}{C + \theta\boldsymbol{e}_i^{\mathrm{T}}\left(\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C\right)^{-1}\boldsymbol{e}_i}, \tag{22}$$

则

$$\begin{aligned}\sigma_i &= \lim_{\theta\to+\infty}\tilde{\sigma}_i \\ &= \lim_{\theta\to+\infty}\frac{1}{2C}\frac{\theta\boldsymbol{\alpha}_{L+1}^{\mathrm{T}}\boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{e}_i\boldsymbol{\alpha}_{L+1}}{C + \theta\boldsymbol{e}_i^{\mathrm{T}}\left(\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C\right)^{-1}\boldsymbol{e}_i} \\ &= \frac{\alpha_i^2}{2Cd_{ii}},\end{aligned} \tag{23}$$

其中 d_{ii} 为矩阵 $\left(\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C\right)^{-1}$ 的第 i 个对角元, 即 $\boldsymbol{R}_{L+1}$ 的第 i 个对角元. 由于 $\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C$ 为对称正定矩阵, 则 $\left(\boldsymbol{H}_{L+1}^{\mathrm{T}}\boldsymbol{H}_{L+1} + \boldsymbol{I}/C\right)^{-1}$ 也为对称正定矩阵, 可知 $d_{ii} > 0$. 因此, $\sigma_i > 0$, 即

$J_{L+1}^{(-i)} > J_{L+1}$, 也就是说 RELM 的目标函数 J 是随着隐层节点数目的增加而逐渐减小. 在根据 (8) 和 (10) 式计算得到 $\boldsymbol{\alpha}_{L+1}$ 后, 由 (23) 式就可以很容易计算得到 $\sigma_i\,(i=1,2,\cdots,L+1)$, 其计算量为 $O(L)$, 而不需要计算 $L+1$ 个 $J_{L+1}^{(-i)}$, 大大节省了计算量, 加快了 AdRELM 算法.

在根据 (23) 式获得 $\sigma_i\,(i=1,2,\cdots,L+1)$ 后, 就可以从 $L+1$ 个隐层节点中找出对目标函数 J 影响最小的节点 $i_{\min}$, 即拥有最小 $\sigma_{i_{\min}}$ 的节点. 此时, 分为两种情况.

第一种情况: $i_{\min}=L+1$, 说明新增节点是所有隐层节点中对目标函数贡献最小的节点, 这时候为了增强 RELM 的泛化能力, 就需要扩充 RELM 的结构, 增强其学习能力, 达到较好的学习效果.

第二种情况: $i_{\min}\neq L+1$, 说明新增节点不是所有隐层节点中对目标函数贡献最小的节点, 这时候可以用新增节点代替节点 $i_{\min}$. 下面介绍如何迭代删除当前对目标函数 J 贡献最小的节点, 也就是最 "坏" 的节点.

3.3 删除节点

令 $\boldsymbol{H}_{L+1}^{(\cdot,-i_{\min})}$ 为 $\boldsymbol{H}_{L+1}$ 的第 $i_{\min}$ 列删除后所得矩阵, $\boldsymbol{t}_{L+1}^{(-i_{\min})}$ 为 $\boldsymbol{t}_{L+1}$ 的第 $i_{\min}$ 个数据删除后所得向量, 为矩阵 $\boldsymbol{R}_{L+1}$ 的第 $i_{\min}$ 行和第 $i_{\min}$ 列分别删除后所得矩阵, $\boldsymbol{r}_{i_{\min}}$ 为矩阵的第 $i_{\min}$ 行, 在将 $\boldsymbol{r}_{i_{\min}}$ 的第 $i_{\min}$ 个数据 $r_{(i_{\min},i_{\min})}$ 删除后可得向量 $\boldsymbol{r}_{i_{\min}}^{(-i_{\min})}$. 根据 Sherman-Morrison 公式 [15], 则下式成立

$$\tilde{\boldsymbol{R}}_{L+1}^{(-i_{\min},-i_{\min})}=\boldsymbol{R}_{L+1}^{(-i_{\min},-i_{\min})}-\frac{\boldsymbol{r}_{i_{\min}}^{(-i_{\min})}\left(\boldsymbol{r}_{i_{\min}}^{(-i_{\min})}\right)^{\mathrm{T}}}{r_{(i_{\min},i_{\min})}},\tag{24}$$

则 $\tilde{R}_{L+1}^{(-i_{\min},-i_{\min})}$ 即为将第 $i_{\min}$ 个隐层节点删除后所得的 $\boldsymbol{R}_L$, 其更新计算量为 O(L^2). 令

$$\boldsymbol{R}_L=\tilde{R}_{L+1}^{(-i_{\min},-i_{\min})},\tag{25}$$

$$\boldsymbol{t}_L=\boldsymbol{t}_{L+1}^{(-i_{\min})},\tag{26}$$

$$\boldsymbol{H}_L=\boldsymbol{H}_{L+1}^{(\cdot,-i_{\min})},\tag{27}$$

则 $\boldsymbol{\alpha}_L$ 为

$$\boldsymbol{\alpha}_L=\boldsymbol{R}_L\boldsymbol{t}_L,\tag{28}$$

在通过增加第 $L+1$ 个隐层节点, 又删除掉第 $i_{\min}$ 个隐层节点后, RELM 的隐层节点个数没有发生变化, 其实时性也不会受到影响, 但是目标函数 J 却降低了, RELM 的泛化性能得到了改善, 预测精度得到提升.

下面对算法 AdRELM 进行总结, 其实现过程如下:

步骤 1 选取 RELM 隐层的激活函数 $g(\cdot)$, 正则化参数 C, 学习精度 ξ 或者隐层节点的总数量 M, 令 $L=0$;

步骤 2 若 $L=0$, 随机产生连接隐层和输入层的权值 $\boldsymbol{w}_1$ 和偏置 b_1, 计算得到 $\boldsymbol{H}_1$, $\boldsymbol{R}_1=\left(\boldsymbol{H}_1^{\mathrm{T}}\boldsymbol{H}_1+1/C\right)^{-1}$, $t_1=\boldsymbol{H}_1^{\mathrm{T}}y$, 和 $\alpha_1=\boldsymbol{R}_1t_1$, 并令 $L=L+1$;

步骤 3 随机产生连接隐层和输入层的权值 $\boldsymbol{w}_{L+1}$ 和偏置 b_{L+1}, 根据 (8)—(11) 式, 分别得到 $\boldsymbol{R}_{L+1}$, $\boldsymbol{t}_{L+1}$, $\boldsymbol{\alpha}_{L+1}$ 和 $\boldsymbol{H}_{L+1}$, 然后根据 (23) 式计算出 $\sigma_i\,(i=1,\cdots,L+1)$, 并选择出最小的 $\sigma_{i_{\min}}$, 若 $i_{\min}=L+1$, 跳转到步骤 4; 否则跳转到步骤 5;

步骤 4 若 $\sigma_{i_{\min}}\leqslant\xi$ 或 $M=L+1$, 则算法终止; 否则令 $L=L+1$, 跳转到步骤 3;

步骤 5 根据 (25)—(28) 式, 分别更新 $\boldsymbol{R}_L$, $\boldsymbol{t}_L$, $\boldsymbol{H}_L$ 和 $\boldsymbol{\alpha}_L$, 然后将 $\boldsymbol{R}_L$ 和 $\boldsymbol{\alpha}_L$ 代入到 (23) 式中计算得到 $\boldsymbol{R}_{L+1}\sigma_i\,(i=1,\cdots,L)$, 并选择出最小的 $\sigma_{i_{\min}}$, 若 $\sigma_{i_{\min}}\leqslant\xi$, 则算法终止; 否则转到步骤 3.

4 混沌时间序列预测实例

为了验证本文所提算法的有效性, 下面用 3 个典型的混沌时间序列实例进行验证. 这 3 个典型的混沌时间序列描述如下

实例 1 Kawakami 混沌时间序列

$$\boldsymbol{x}_K(i+1)=\boldsymbol{x}_K(i)^2-0.1\boldsymbol{x}_K(i)-1.6,\tag{29}$$

其初始值为 0.6.

实例 2 Lorenz 混沌时间序列

$$\begin{cases}\dfrac{\mathrm{d}x_L}{\mathrm{d}t}=-A\,(x_L-y_L),\\[2mm]\dfrac{\mathrm{d}y_L}{\mathrm{d}t}=Cx-x_Lz_L-y_L,\\[2mm]\dfrac{\mathrm{d}z_L}{\mathrm{d}t}=x_Ly_L-Bz_L,\end{cases}\tag{30}$$

其中 $A=16$, $B=4$, $C=45.92$, 初始值为 $\begin{bmatrix}-1 & 0 & 1\end{bmatrix}^{\mathrm{T}}$.

实例 3 Chen's 混沌时间序列

$$\begin{cases}\dfrac{\mathrm{d}x_{\mathrm{C}}}{\mathrm{d}t}=A\,(y_{\mathrm{C}}-x_{\mathrm{C}}),\\[2mm]\dfrac{\mathrm{d}y_{\mathrm{C}}}{\mathrm{d}t}=(C-A)x_{\mathrm{C}}-x_{\mathrm{C}}z_{\mathrm{C}}+Cy_{\mathrm{C}},\\[2mm]\dfrac{\mathrm{d}z_{\mathrm{C}}}{\mathrm{d}t}=x_{\mathrm{C}}y_{\mathrm{C}}-Bz_{\mathrm{C}},\end{cases}\tag{31}$$

其中 $A=35, B=3, C=28$, 初始值为 $\begin{bmatrix}-1 & 0 & 1\end{bmatrix}^{\mathrm{T}}$.

对于每一种混沌时间序列, 分别产生 1000 个样本, 其中前 300 个样本用来训练, 后 700 个样本检验算法 AdRELM 和 CRELM 的性能. 重构相空间为嵌入维数 $m=4$, 延迟时间 $\tau=1$, 进行一步预测. 算法 CRELM 和 AdRELM 隐层选择的激活函数为 $g(x)=\dfrac{1}{1+\mathrm{e}^{-x}}$, 模型的正则化参数 C 用交叉验证的方法从 $\{10^0, 10^1, \cdots, 10^{10}\}$ 中选取. 为了衡量预测的精度, 定义一个性能指标, 即均方根误差 (rooted mean squared errors, RMSE), 其描述为

$$\mathrm{RMSE}=\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i-y_i)^2}, \tag{32}$$

其中 y_i 为混沌时间序列真实值, $\hat{y}_i$ 为 y_i 的预测值.

图 1 给出了 RELM 目标函数值 J 随隐层节点数 L 增多时的变化情况.

从图 1 可以很明显地看出, RELM 优化函数目标值 J 随着隐层节点数的增多而逐渐减小. 刚开始 J 减少得较快, 而随着隐层节点数的增多减少量逐渐减小, 最后趋于平缓. 这主要是因为刚开的时候, RELM 处于欠学习状态, 每增加一个节点都对其影响较大, 而随着隐层节点数的增多, RELM 的学习能力逐渐趋于饱和状态, 再想通过节点数的增加对其泛化能力产生较大的影响就比较困难了. 等隐层节点数达到一定的数量, 如果再增加隐层节点数, 这时候的隐层节点就处于冗余状态, 只会增加 RELM 的复杂度, 而对改善其性能几乎起不到作用. 同时, 图 1 也显示了在相同隐层节点数的情况下, AdRELM 对目标函数 J 的影响比 CRELM 要大, 这主要是因为 AdRELM 算法的增加删除机制优化了隐层节点的质量, 使其和 CRELM 相比较, 在相同节点数的情况, RELM 的泛化能力更强, 这就证明了本文提出的增加删除机制是有效的. CRELM 算法要想取得和 AdRELM 算法相同的 J 值, 往往需要更多的隐层节点数, 这就会增加 RELM 网络的复杂度, 影响其实时性.

图 2 给出了性能指标 RMSE 随隐层节点数 L 增多时的变化趋势.

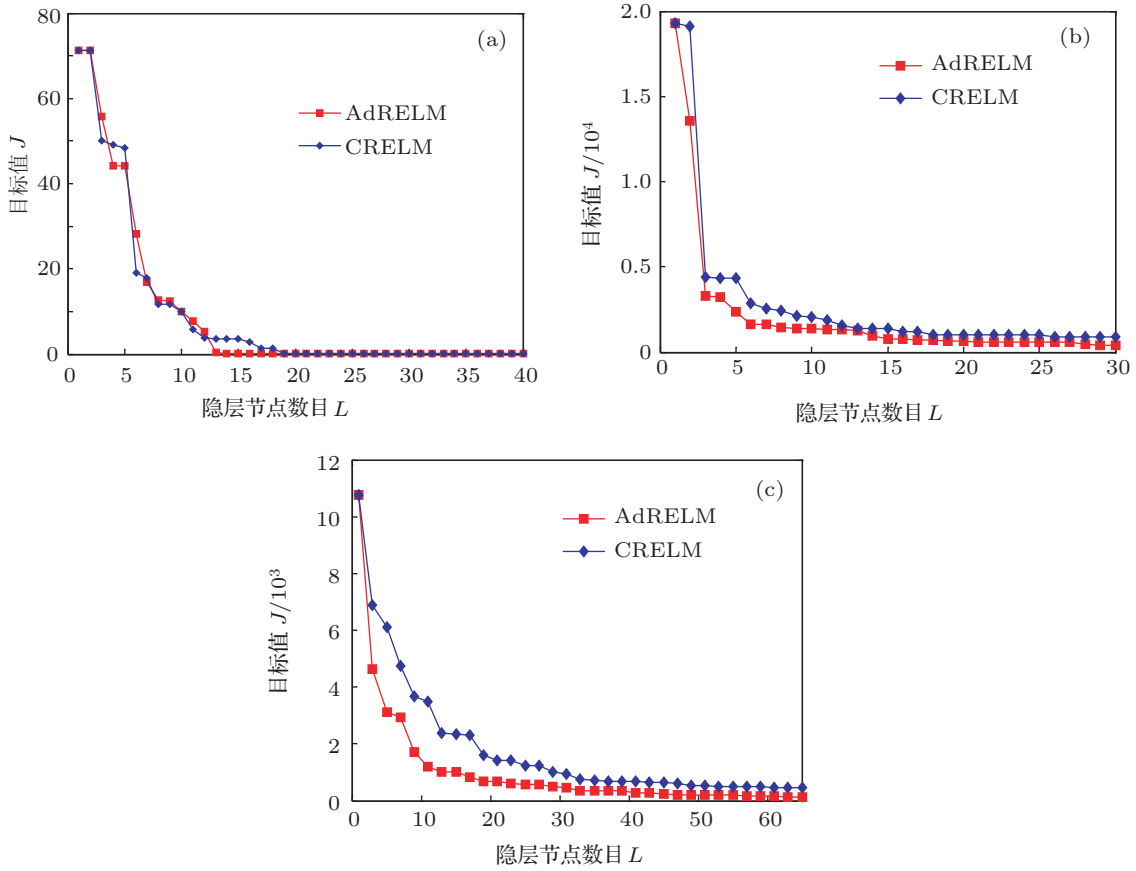

图 1 RELM 优化函数目标值随隐层节点数目增多时的变化情况 (a) Kawakami; (b) Lorenz; (c) Chen's

图 2 的变化趋势和图 1 的变化趋势相同, RMSE 随着隐层节点数目的增多而逐渐减小, 刚开始减少得较快, 而随着隐层节点数的增多而减少得较慢, 最后趋于稳定. 在网络达到饱和之前, 在相同隐层节点数目的情况下, AdRELM 具有较小的 RMSE, 说明其泛化能力比较好, 表明它拥有的隐层节点的质量比较高, 证明了 AdRELM 的增加删除机制能够优化隐层节点的质量, 使 RELM 在相同泛化能力的情况拥有较少的隐层节点数, 具有较简单的网络结构和较好的实时性. 目标函数值 J 和 RMSE 的变化趋势一致, 说明依据目标函数值来选择隐层节点的数目是有效可行的, 对目标函数值影响较大的节点认为是较好的节点, 反之则认为是较差的节点, 佐证了本文给出的衡量隐层节点质量好坏的标准是有效性的.

将本文进行实验的部分结果列于表 1 当中, 以便进行对比.

从表 1 可以看出, 对于每一个实例来说, 当 AdRELM 和 CRELM 最后趋于稳定时, AdRELM 具有较小的目标值 J, 和其相应较小的 RMSE, 这一点可以从每个实例的第 2 行和第 4 行对比看出来. 这说明了 AdRELM 能够优化隐层节点的质量. 如果要使 AdRELM 达到和 CRELM 近似相同的泛化能力, AdRELM 往往需要较少的隐层节点数, 这可以从每个实例的第 2 行和第 3 行对比看出来, 这也就意味着 AdRELM 具有较简单的网络结构, 其实时性也比较好 (测试时间可以反映出来). 可是, 反过来, CRELM 在和 AdRELM 拥有相同隐层节点数时, 其泛化性能就比较差, 这可以从每一个实例的第 1 行和第 3 行对比表现出来. 这说明了 AdRELM 的增加删除机制的有效性和可行性, 它可以使 RELM 具有质量更高的隐层节点, 更优化的网络结构. 当然, AdRELM 也有缺点, 这就是它需要的训练时间比较长, 这一点可以从训练时间的对比上看出来. CRELM 每增加一个隐层节点, 其计算量为 $O(L^2)$. AdRELM 增加一个隐层节点的过程的计算量也为 $O(L^2)$, 但当新增加的节点不是所有节点中最 "坏" 的节点时, 就要进行 (24) 式的删除计算, 其计算量为 $O(L^2)$, 并且这个增加删除过程在 AdRELM 优化隐层节点的过程当中经常出现. 因而, 和 CRELM 相比较, AdRELM 就会需要较多的计算量, 花费一

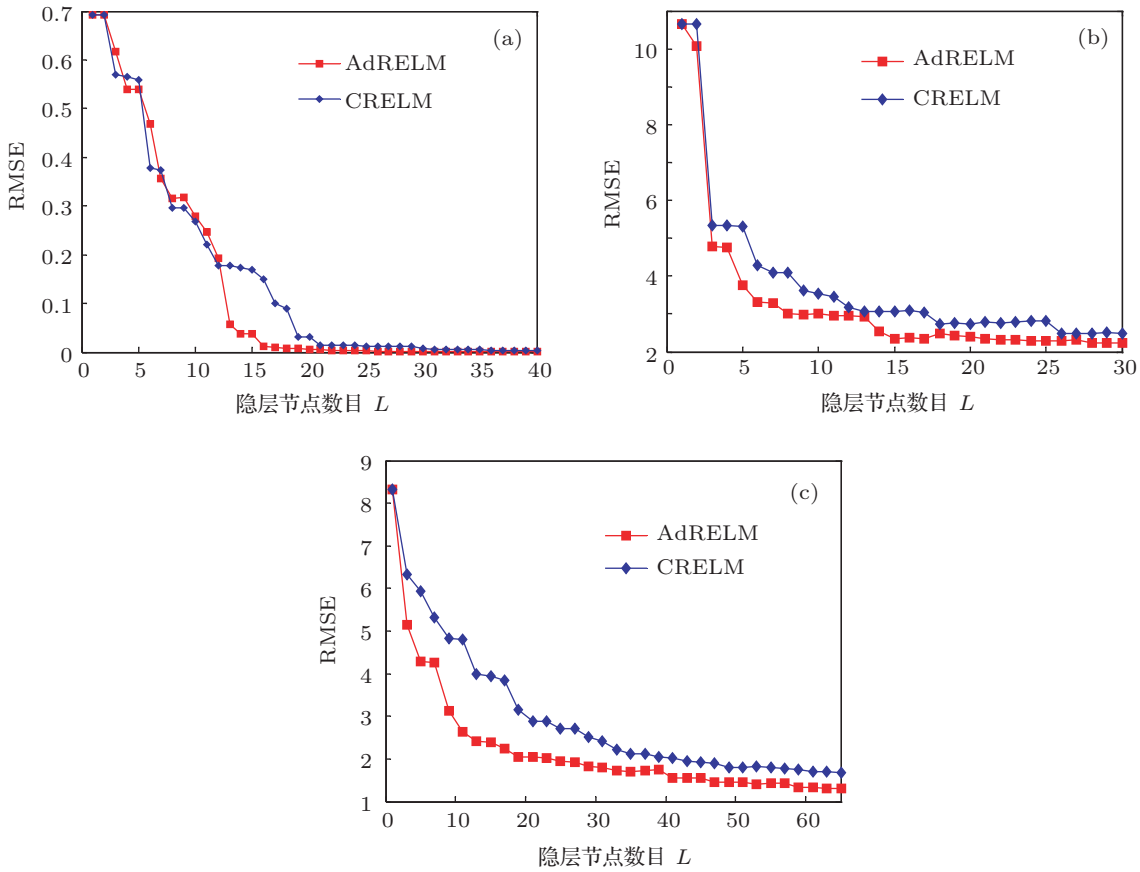

图 2 RMSE 随隐层节点数目增多时的变化趋势 (a) Kawakami; (b) Lorenz; (c) Chen's

表 1 不同混沌时间序列 AdRELM 和 CRELM 的比较结果

混沌时间序列	算法	优化目标 J	RMSE	隐层节点数 L	训练时间/s	测试时间/s
Kawakami	CRELM	5.089×10^{-2}	0.0136	24	0.0051	0.0011
	CRELM	6.640×10^{-3}	0.0037	36	0.0084	0.0013
	AdRELM	7.730×10^{-3}	0.0037	24	0.0209	0.0010
	AdRELM	9.443×10^{-4}	0.0012	36	0.0339	0.0012
Lorenz	CRELM	1377.967	3.060	15	0.0036	0.0008
	CRELM	892.344	2.498	26	0.0049	0.0011
	AdRELM	770.825	2.342	15	0.0146	0.0007
	AdRELM	576.849	2.287	26	0.0206	0.0011
Chen's	CRELM	705.925	2.110	35	0.0087	0.0013
	CRELM	454.599	1.701	61	0.0129	0.0020
	AdRELM	357.422	1.709	35	0.0291	0.0012
	AdRELM	152.875	1.338	61	0.0591	0.0019

定的训练时间. 由于 AdRELM 和 CRELM 都是离线算法, 训练时间一般都是有保证的, 所以训练时间较长这个缺点也是可以克服的.

5 结 论

影响 RELM 性能的一个重要因素就是隐层节点的数目, 选择得太多或太少都不合适. 隐层节点数选择得太少, 影响 RELM 的泛化性能; 而选择得太多的话, 不仅不能改善 RELM 的性能, 由于有冗余隐层节点的存在, 相反还会增加网络的复杂度, 影响其算法的实时性等. 因此, 确定一种选择 RELM 隐层节点数目的算法就显得比较重要了. 本文在他人工作的基础上, 提出了一种增加删除机制来自适应地确定 RELM 隐层节点的数目. 这种机制充分考虑到了 RELM 随机选择隐层节点的特点, 优化了隐层节点, 使其和一味单纯地增加隐层节点来改善 RELM 性能的算法相比较, 在隐层节点数、泛化性能以及实时性等方面都占有优势. 同时, 典型混沌时间序列实例验证了本文所提算法的有效性.

[1] Li D C, Han M 2011 *Acta Phys. Sin.* **60** 108903 (in Chinese) [李德才, 韩敏 2011 物理学报 **60** 108903]

[2] Li H, Yang Z, Zhang Y M, Wen B C 2011 *Acta Phys. Sin.* **60** 070512 (in Chinese) [李鹤, 杨周, 张义民, 闻邦椿 2011 物理学报 **60** 070512]

[3] Ma Q L, Zheng Q L, Peng H, Qin J W 2009 *Acta Phys. Sin.* **58** 1410 (in Chinese) [马千里, 郑启伦, 彭宏, 覃姜维 2009 物理学报 **58** 1410]

[4] Zhang S, Liu H X, Gao D T, Du S D 2003 *Chin. Phys.* B **12** 594

[5] Zhang J S, Xiao X C 2000 *Chin. Phys. Lett.* **17** 88

[6] Huang G B, Zhu Q Y, Siew C K 2004 *Proceedings of IEEE International Conference on Neural Networks* Budapest, Hungary, July 25–29, 2004 p985

[7] Huang G B, Chen L, Siew C K 2006 *IEEE Trans. Neural Netw.* **17** 879

[8] Huang G B, Zhou H M, Ding X J, Zhang R 2012 *IEEE Trans. Syst. Man Cybern.* B *Cybern.* **42** 513

[9] Wang X Y, Han M 2012 *Acta Phys. Sin.* **61** 080507 (in Chinese) [王新迎, 韩敏 2012 物理学报 **61** 080507]

[10] Gao G Y, Jiang G P 2012 *Acta Phys. Sin.* **61** 040506 (in Chinese) [高光勇, 蒋国平 2012 物理学报 **61** 040506]

[11] Zhang X, Wang H L 2011 *Acta Phys. Sin.* **60** 080504 (in Chinese) [张弦, 王宏力 2011 物理学报 **60** 080504]

[12] Deng W Y, Zheng Q H, Chen L 2009 *Proceedings of 2009 IEEE Symposium on Computational Intelligence and Data Mining* Nashville, TN, United states, March 30 – April 2, 2009 p389

[13] Zhang X, Wang H L 2011 *Acta Phys. Sin.* **60** 110201 (in Chinese) [张弦, 王宏力 2011 物理学报 **60** 110201]

[14] Duda R O, Hart P E, Stork D G 2001 *Pattern Classification* (New York: John Wiley & Sons, Inc.)

[15] Zhang X D 2004 *Matrix Analysis and Applications* (Beijing: Tsinghua University Press) (in Chinese) [张贤达 2004 矩阵分析与应用 (北京: 清华大学出版社)]

Chaotic time series prediction using add-delete mechanism based regularized extreme learning machine*

Zhao Yong-Ping† Wang Kang-Kang

(*School of Mechanical Engineering, Nanjing University of Science and Technology, Nanjing* 210094*, China*)

(Received 13 August 2013; revised manuscript received 17 September 2013)

Abstract

Considering a regularized extreme learning machine (RELM) with randomly generated hidden nodes, an add-delete mechanism is proposed to determine the number of hidden nodes adaptively, where the extent of contribution to the objective function of RELM is treated as the criterion for judging each hidden node, that is, the large the better, and vice versa. As a result, the better hidden nodes are kept. On the contrary, the so-called worse hidden nodes are deleted. Naturally, the hidden nodes of RELM are selected optimally. In contrast to the other method only with the add mechanism, the proposed one has some advantages in the number of hidden nodes, generalization performance, and the real time. The experimental results on classical chaotic time series demonstrate the effectiveness and feasibility of the proposed add-delete mechanism for RELM.

Keywords: chaotic time series, artificial neural networks, extreme learning machine

PACS: 05.45.Tp, 05.45.−a, 02.50.Ey **DOI:** 10.7498/aps.62.240509

* Project supported by the National Natural Science Foundation of China (Grant No. 51006052) and the Outstanding Scholar Supporting Program of Nanjing Univeristy of Science and Technology, China.

† Corresponding author. E-mail: y.p.zhao@163.com