递归最小二乘循环神经网络

赵 杰[1] 张春元[1] 刘 超[1] 周 辉[1] 欧宜贵[2] 宋 淇[1]

摘 要 针对循环神经网络 (Recurrent neural networks, RNNs) 一阶优化算法学习效率不高和二阶优化算法时空开销过大, 提出一种新的迷你批递归最小二乘优化算法. 所提算法采用非激活线性输出误差替代传统的激活输出误差反向传播, 并结合加权线性最小二乘目标函数关于隐藏层线性输出的等效梯度, 逐层导出 RNNs 参数的迷你批递归最小二乘解. 相较随机梯度下降算法, 所提算法只在 RNNs 的隐藏层和输出层分别增加了一个协方差矩阵, 其时间复杂度和空间复杂度仅为随机梯度下降算法的 3 倍左右. 此外, 本文还就所提算法的遗忘因子自适应问题和过拟合问题分别给出一种解决办法. 仿真结果表明, 无论是对序列数据的分类问题还是预测问题, 所提算法的收敛速度要优于现有主流一阶优化算法, 而且在超参数的设置上具有较好的鲁棒性.

关键词 深度学习, 循环神经网络, 递归最小二乘, 迷你批学习, 优化算法

引用格式 赵杰, 张春元, 刘超, 周辉, 欧宜贵, 宋淇. 递归最小二乘循环神经网络. 自动化学报, 2022, **48**(8): 2050−2061

DOI 10.16383/j.aas.c190847

Recurrent Neural Networks With Recursive Least Squares

ZHAO Jie[1] ZHANG Chun-Yuan[1] LIU Chao[1] ZHOU Hui[1] OU Yi-Gui[2] SONG Qi[1]

Abstract In recurrent neural networks (RNNs), the first-order optimization algorithms usually converge slowly, and the second-order optimization algorithms commonly have high time and space complexities. In order to solve these problems, a new minibatch recursive least squares (RLS) optimization algorithm is proposed. Using the inactive linear output error to replace the conventional activation output error for backpropagation, together with the equivalent gradients of the weighted linear least squares objective function with respect to linear outputs of the hidden layer, the proposed algorithm derives the minibatch recursive least squares solutions of RNNs parameters layer by layer. Compared with the stochastic gradient descent algorithm, the proposed algorithm only adds one covariance matrix into each layer of RNNs, and its time and space complexities are almost three times as much. Furthermore, in order to address the adaptive problem of the forgetting factor and the overfitting problem of the proposed algorithm, two approaches are also presented, respectively, in this paper. The simulation results, on the classification and prediction problems of sequential data, show that the proposed algorithm has faster convergence speed than popular first-order optimization algorithms. In addition, the proposed algorithm also has good robustness in the selection of hyperparameters.

Key words Deep learning, recurrent neural network (RNN), recursive least squares (RLS), minibatch learning, optimization algorithm

Citation Zhao Jie, Zhang Chun-Yuan, Liu Chao, Zhou Hui, Ou Yi-Gui, Song Qi. Recurrent neural networks with recursive least squares. *Acta Automatica Sinica*, 2022, **48**(8): 2050−2061

循环神经网络 (Recurrent neural networks, RNNs) 作为一种有效的深度学习模型, 引入了数据在时序上的短期记忆依赖. 近年来, RNNs 在语言模型[1]、机器翻译[2]、语音识别[3] 等序列任务中均有不俗的表现. 但是相比前馈神经网络而言, 也正因为其短期记忆依赖, RNNs 的参数训练更为困难[4−5]. 如何高效训练 RNNs, 即 RNNs 的优化, 是 RNNs 能否得以有效利用的关键问题之一. 目前主流的 RNNs 优化算法主要有一阶梯度下降算法、自适应学习率算法和二阶梯度下降算法等几种类型.

最典型的一阶梯度下降算法是随机梯度下降 (Stochastic gradient descent, SGD)[6], 广泛应用于优化 RNNs. SGD 基于小批量数据的平均梯度对参数进行优化. 因为 SGD 的梯度下降大小和方向完全依赖当前批次数据, 容易陷入局部极小点, 故而学习效率较低, 更新不稳定. 为此, 研究者在 SGD

收稿日期 2019-12-12 录用日期 2020-04-07
Manuscript received December 12, 2019; accepted April 7, 2020
国家自然科学基金 (61762032, 61662019, 11961018) 资助
Supported by National Natural Science Foundation of China (61762032, 61662019, 11961018)
本文责任编委 曹向辉
Recommended by Associate Editor CAO Xiang-Hui
1. 海南大学计算机科学与技术学院 海口 570228 2. 海南大学理学院 海口 570228
1. School of Computer Science and Technology, Hainan University, Haikou 570228 2. School of Science, Hainan University, Haikou 570228

的基础上引入了速度的概念来加速学习过程, 这种算法称为基于动量的 SGD 算法[7], 简称为 Momentum. 在此基础上, Sutskever 等[8] 提出了一种 Nesterov 动量算法. 与 Momentum 的区别体现在梯度计算上. 一阶梯度下降算法的超参数通常是预先固定设置的, 一个不好的设置可能会导致模型训练速度低下, 甚至完全无法训练. 针对 SGD 的问题, 研究者提出了一系列学习率可自适应调整的一阶梯度下降算法, 简称自适应学习率算法. Duchi 等[9] 提出的 AdaGrad 算法采用累加平方梯度对学习率进行动态调整, 在凸优化问题中表现较好, 但在深度神经网络中会导致学习率减小过快. Tieleman 等[10] 提出的 RMSProp 算法与 Zeiler[11] 提出的 AdaDelta 算法在思路上类似, 都是使用指数衰减平均来减少太久远梯度的影响, 解决了 AdaGrad学习率减少过快的问题. Kingma 等[12] 提出的 Adam 算法则将 RMSProp 与动量思想相结合, 综合考虑梯度的一阶矩和二阶矩估计计算学习率, 在大部分实验中比 AdaDelta 等算法表现更为优异, 然而 Keskar 等[13] 发现 Adam 最终收敛效果比 SGD 差, Reddi 等[14] 也指出 Adam 在某些情况下不收敛.

基于二阶梯度下降的算法采用目标函数的二阶梯度信息对参数优化. 最广泛使用的是牛顿法, 其基于二阶泰勒级数展开来最小化目标函数, 收敛速度比一阶梯度算法快很多, 但是每次迭代都需要计算 Hessian 矩阵以及该矩阵的逆, 计算复杂度非常高. 近年来研究人员提出了一些近似算法以降低计算成本. Hessian-Free 算法[15] 通过直接计算 Hessian 矩阵和向量的乘积来降低其计算复杂度, 但是该算法每次更新参数需要进行上百次线性共轭梯度迭代. AdaQN[16] 在每个迭代周期中要求一个两层循环递归, 因此计算量依然较大. K-FAC 算法 (Kronecker-factored approximate curvature)[17] 通过在线构造 Fisher 信息矩阵的可逆近似来计算二阶梯度. 此外, 还有 BFGS 算法[18] 以及其衍生算法 (例如 L-BFGS 算法[19−20] 等), 它们都通过避免计算 Hessian 矩阵的逆来降低计算复杂度. 相对于一阶优化算法来说, 二阶优化算法计算量依然过大, 因此不适合处理规模过大的数据集, 并且所求得的高精度解对模型的泛化能力提升有限, 甚至有时会影响泛化, 因此二阶梯度优化算法目前还难以广泛用于训练 RNNs.

除了上面介绍的几种类型优化算法之外, 也有不少研究者尝试将递归最小二乘算法 (Recursive least squares, RLS) 应用于训练各种神经网络. RLS 是一种自适应滤波算法, 具有非常快的收敛速度. Azimi-Sadjadi 等[21] 提出了一种 RLS 算法, 对多层感知机进行训练. 谭永红[22] 将神经网络层分为线性输入层与非线性激活层, 对非线性激活层的反传误差进行近似, 并使用 RLS 算法对线性输入层的参数矩阵进行求解来加快模型收敛. Xu 等[23] 成功将 RLS 算法应用于多层 RNNs. 上述算法需要为每个神经元存储一个协方差矩阵, 时空开销很大. Peter 等[24] 提出了一种扩展卡尔曼滤波优化算法, 对 RNNs 进行训练. 该算法将 RNNs 表示为被噪声破坏的平稳过程, 然后对网络的状态矩阵进行求解. 该算法不足之处是需要计算雅可比矩阵来达到线性化的目的, 时空开销也很大. Jaeger[25] 通过将非线性系统近似为线性系统, 实现了回声状态网络参数的 RLS 求解, 但该算法仅限于求解回声状态网络的输出层参数, 并不适用于一般的 RNNs 训练优化.

针对以上问题, 本文提出了一种新的基于 RLS 优化的 RNN 算法 (简称 RLS-RNN). 本文主要贡献如下: 1) 在 RLS-RNN 的输出层参数更新推导中, 借鉴 SGD 中平均梯度的计算思想, 提出了一种适于迷你批样本训练的 RLS 更新方法, 显著减少了 RNNs 的实际训练时间, 使得所提算法可处理较大规模数据集. 2) 在 RLS-RNN 的隐藏层参数更新推导中, 提出了一种等效梯度思想, 以获得该层参数的最小二乘解, 同时使得 RNNs 仅要求输出层激活函数存在反函数即可采用 RLS 进行训练, 对隐藏层的激活函数则无此要求. 3) 相较以前的 RLS 优化算法, RLS-RNN 只需在隐藏层和输出层而非为这两层的每一个神经元分别设置一个协方差矩阵, 使得其时间和空间复杂度仅约 SGD 算法的 3 倍. 4) 对 RLS-RNN 的遗忘因子自适应和过拟合预防问题进行了简要讨论, 分别给出了一种解决办法.

1 背景

1.1 基于 SGD 优化的 RNN 算法

RNNs 处理时序数据的模型结构如图 1 所示. 一个基本的 RNN 通常由一个输入层、一个隐藏层 (也称为循环层) 和一个输出层组成. 在图 1 中, $X_{s,t} \in \mathbf{R}^{m\times a}$, $H_{s,t} \in \mathbf{R}^{m\times h}$ 和 $O_{s,t} \in \mathbf{R}^{m\times d}$ 分别为第 s 批训练样本数据在第 t 时刻的输入值、隐藏层和输出层的输出值, 其中, m 为迷你批大小, a 为一个训练样本数据的维度, h 为隐藏层神经元数, d 为输出层神经元数; $U_{s-1} \in \mathbf{R}^{a\times h}$, $W_{s-1} \in \mathbf{R}^{h\times h}$ 和 $V_{s-1} \in \mathbf{R}^{h\times d}$ 分别为第 s 批数据训练时输入层到隐藏层、隐藏层内部、隐藏层到输出层的参数矩阵; $\boldsymbol{b}^H_{s-1} \in \mathbf{R}^{1\times h}$ 和 $\boldsymbol{b}^O_{s-1} \in \mathbf{R}^{1\times d}$ 分别为隐藏层和输出层的偏置参数矩阵; τ 表示当前序列数据共有 τ 时间步. RNNs 的核心思想是在模型的不同时间步对参

数进行共享, 将每一时间步的隐藏层输出值加权输入到其下一时间步的计算中, 从而令权重参数学习到序列数据不同时间步之间的关联特征并进行泛化. 输出层则根据实际问题选择将哪些时间步输出, 比较常见的有序列数据的分类问题和预测问题. 对序列数据预测问题, 输出层每一时间步均有输出; 对序列数据分类问题, 输出层没有图 1 虚线框中的时间步输出, 即仅在最后一个时间步才有输出.

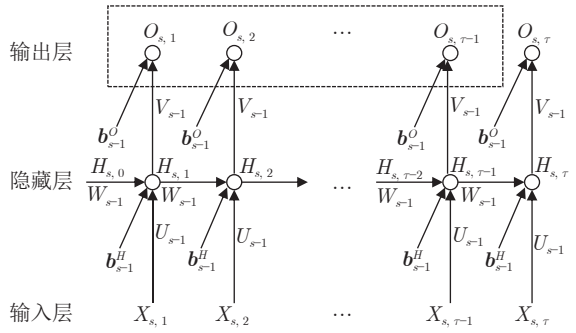


图 1 RNN 模型结构

Fig. 1 RNN model structure

RNNs 通过前向传播来获得实际输出, 其计算过程可描述为

$$H_{s,t}=\varphi(X_{s,t}U_{s-1}+H_{s,t-1}W_{s-1}+\mathbf{1}\times\boldsymbol{b}^H_{s-1}) \tag{1}$$

$$O_{s,t}=\sigma(H_{s,t}V_{s-1}+\mathbf{1}\times\boldsymbol{b}^O_{s-1}) \tag{2}$$

其中, $\mathbf{1}$ 为 m 行全 1 列向量; $\varphi(\cdot)$ 和 $\sigma(\cdot)$ 分别为隐藏层和输出层的激活函数, 常用的激活函数有 sigmoid 函数与 tanh 函数等. 为了便于后续推导和表达的简洁性, 以上两式可用增广矩阵进一步表示为

$$H_{s,t}=\varphi(R^H_{s,t}\Theta^H_{s-1}) \tag{3}$$

$$O_{s,t}=\sigma(R^O_{s,t}\Theta^O_{s-1}) \tag{4}$$

其中, $R^H_{s,t}\in\mathbf{R}^{m\times(a+h+1)}$, $R^O_{s,t}\in\mathbf{R}^{m\times(h+1)}$ 分别为隐藏层与输出层的输入增广矩阵; $\Theta^H_{s-1}\in\mathbf{R}^{(a+h+1)\times h}$, $\Theta^O_{s-1}\in\mathbf{R}^{(h+1)\times d}$ 分别为隐藏层与输入层的权重参数增广矩阵, 即

$$R^H_{s,t}=[X_{s,t}\quad H_{s,t-1}\quad\mathbf{1}] \tag{5}$$

$$R^O_{s,t}=[H_{s,t}\quad\mathbf{1}] \tag{6}$$

$$\Theta^H_{s-1}=\left[U^{\mathrm{T}}_{s-1}\quad W^{\mathrm{T}}_{s-1}\quad(\boldsymbol{b}^H_{s-1})^{\mathrm{T}}\right]^{\mathrm{T}} \tag{7}$$

$$\Theta^O_{s-1}=\left[V^{\mathrm{T}}_{s-1}\quad(\boldsymbol{b}^O_{s-1})^{\mathrm{T}}\right]^{\mathrm{T}} \tag{8}$$

RNNs 的参数更新方式和所采用的优化算法密切相关, 基于 SGD 算法的 RNNs 模型优化通常借助于最小化目标函数反向传播完成. 常用目标函数有交叉熵函数、均方误差函数、Logistic 函数等. 这里仅考虑均方误差目标函数

$$\hat{J}(\Theta_{s-1})=\frac{\sum\limits_{t=t_0}^{\tau}\left\|Y^*_{s,t}-O_{s,t}\right\|^2_{\mathrm{F}}}{2m(\tau-t_0+1)} \tag{9}$$

其中, $Y^*_{s,t}\in\mathbf{R}^{m\times d}$ 为 $X_{s,t}$ 对应的期望输出; Θ_{s-1} 为网络中的所有参数矩阵; t_0 表示输出层的起始输出时间步, 如果是分类问题, $t_0=\tau$, 如果是序列预测问题, 则 $t_0=1$, 下文延续该设定, 不再赘述.

令 $\hat{\nabla}^O_s=\frac{\partial\hat{J}(\Theta_{s-1})}{\partial\Theta^O_{s-1}}$, 由式 (9) 和链导法则, 则 $\hat{\nabla}^O_s$ 为

$$\hat{\nabla}^O_s=\sum_{t=t_0}^{\tau}(R^O_{s,t})^{\mathrm{T}}\hat{\Delta}^O_{s,t} \tag{10}$$

其中, $\hat{\Delta}^O_{s,t}=\frac{\partial\hat{J}(\Theta_{s-1})}{\partial Z^O_{s,t}}$, 即

$$\hat{\Delta}^O_{s,t}=\begin{cases}\dfrac{(O_{s,t}-Y^*_{s,t})\circ\sigma'(Z^O_{s,t})}{m(\tau-t_0+1)}, & t\geq t_0\\ 0, & t<t_0\end{cases} \tag{11}$$

式中, $\circ$ 为 Hadamard 积, $Z^O_{s,t}$ 为输出层非激活线性输出, 即

$$Z^O_{s,t}=R^O_{s,t}\Theta^O_{s-1} \tag{12}$$

则该层参数更新规则可定义为

$$\Theta^O_s=\Theta^O_{s-1}-\alpha\hat{\nabla}^O_s \tag{13}$$

其中, α 为学习率.

令 $\hat{\nabla}^H_s=\frac{\partial J(\Theta_{s-1})}{\partial\Theta^H_{s-1}}$, 根据 BPTT (Back propagation through time) 算法[26], 由式 (9) 和链导法则可得

$$\hat{\nabla}^H_s=\sum_{t=1}^{\tau}(R^H_{s,t})^{\mathrm{T}}\hat{\Delta}^H_{s,t} \tag{14}$$

其中, $\hat{\Delta}^H_{s,t}=\frac{\partial\hat{J}(\Theta_{s-1})}{\partial Z^H_{s,t}}$ 为目标函数对于隐藏层非激活线性输出的梯度, 即

$$\hat{\Delta}^H_{s,t}=\begin{cases}(\tilde{\Delta}^H_{s,t}(\tilde{\Theta}^H_{s-1})^{\mathrm{T}})\circ\varphi'(Z^H_{s,t}), & t<\tau\\ (\hat{\Delta}^O_{s,t}V^{\mathrm{T}}_{s-1})\circ\varphi'(Z^H_{s,t}), & t=\tau\end{cases} \tag{15}$$

其中, $\tilde{\Delta}^H_{s,t}=[\hat{\Delta}^O_{s,t},\hat{\Delta}^H_{s,t+1}]$, $\tilde{\Theta}^H_{s-1}=[V_{s-1},W_{s-1}]$, $Z^H_{s,t}$ 为隐藏层非激活线性输出, 即

$$Z^H_{s,t}=R^H_{s,t}\Theta^H_{s-1} \tag{16}$$

则该层参数更新规则可定义为

$$\Theta^H_s=\Theta^H_{s-1}-\alpha\hat{\nabla}^H_s \tag{17}$$

1.2 RLS 算法

RLS 是一种最小二乘优化算法的递推化算法,

不但收敛速度很快, 而且适用于在线学习. 设当前训练样本输入集 $X_t = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t\}$, 对应的期望输出集为 $\boldsymbol{Y}_t^* = \{y_1^*, \cdots, y_t^*\}$. 其目标函数通常定义为

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{s=1}^{t}\lambda^{t-s}\left(y_s^* - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_s\right)^2 \tag{18}$$

其中, $\boldsymbol{w}$ 为权重向量; $\lambda \in (0, 1]$ 为遗忘因子.

令 $\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = 0$, 可得

$$\boldsymbol{w}^* = \sum_{s=1}^{t}\lambda^{t-s}(\boldsymbol{x}_s\boldsymbol{x}_s^{\mathrm{T}})^{-1}\boldsymbol{x}_s y_s^* \tag{19}$$

整理后可表示为

$$\boldsymbol{w}_t = \boldsymbol{w}^* = A_t^{-1}\boldsymbol{b}_t \tag{20}$$

其中,

$$A_t = \sum_{s=1}^{t}\lambda^{t-s}\boldsymbol{x}_s\boldsymbol{x}_s^{\mathrm{T}} \tag{21}$$

$$\boldsymbol{b}_t = \sum_{s=1}^{t}\lambda^{t-s}\boldsymbol{x}_s y_s^* \tag{22}$$

为了避免昂贵的矩阵求逆运算且适用于在线学习, 令

$$P_t = A_t^{-1} \tag{23}$$

将式 (21) 和式 (22) 改写为如下递推更新形式

$$A_t = \lambda A_{t-1} + \boldsymbol{x}_t\boldsymbol{x}_t^{\mathrm{T}} \tag{24}$$

$$\boldsymbol{b}_t = \lambda\boldsymbol{b}_{t-1} + \boldsymbol{x}_t y_t^* \tag{25}$$

由 Sherman-Morrison-Woodbury 公式[27] 易得

$$P_t = \frac{1}{\lambda}P_{t-1} - \frac{1}{\lambda}\boldsymbol{g}_t\boldsymbol{u}_t^{\mathrm{T}} \tag{26}$$

其中,

$$\boldsymbol{u}_t = P_{t-1}\boldsymbol{x}_t \tag{27}$$

$$\boldsymbol{g}_t = \frac{\boldsymbol{u}_t}{\lambda + \boldsymbol{u}_t^{\mathrm{T}}\boldsymbol{x}_t} \tag{28}$$

其中, $\boldsymbol{g}_t$ 为增益向量. 进一步将式 (23)、(25) 和 (26) 代入式 (20), 可得当前权重向量的更新公式为

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \boldsymbol{g}_t e_t \tag{29}$$

其中,

$$e_t = \boldsymbol{w}_{t-1}^{\mathrm{T}}\boldsymbol{x}_t - y_t^* \tag{30}$$

2 基于 RLS 优化的 RNNs 算法

RLS 算法虽然具有很快的学习速度, 然而只适用于线性系统. 我们注意到在 RNNs 中, 如果不考虑激活函数, 其隐藏层和输出层的输出计算依旧是线性的, 本节将基于这一特性来构建新的迷你批 RLS 优化算法. 假定输出层激活函数 $\sigma(\cdot)$ 存在反函数 $\sigma^{-1}(\cdot)$, 并仿照 RLS 算法将输出层目标函数定义为

$$J(\Theta) = \sum_{n=1}^{s}\sum_{t=t_0}^{\tau}\frac{\lambda^{s-n}\left\|Z_{n,t}^{O*} - Z_{n,t}^{O}\right\|_{\mathrm{F}}^2}{2m(\tau - t_0 + 1)} \tag{31}$$

其中, s 代表共有 s 批训练样本; $Z_{n,t}^{O*}$ 为输出层的非激活线性期望值, 即

$$Z_{n,t}^{O*} = \sigma^{-1}(Y_{n,t}^*) \tag{32}$$

因此, RNNs 参数优化问题可以定义为

$$\Theta_s = \arg\min_{\Theta} J(\Theta) \tag{33}$$

由于 RNNs 前向传播并不涉及权重参数更新, 因此本文所提算法应用于 RNNs 训练时, 其前向传播计算与第 1.1 节介绍的 SGD-RNN 算法基本相同, $H_{s,t}$ 同样采用式 (3) 计算, 唯一区别是此处并不需要计算 $O_{s,t}$, 而是采用式 (12) 计算 $Z_{s,t}^{O}$. 本节将只考虑 RLS-RNN 的输出层和隐藏层参数更新推导.

2.1 RLS-RNN 输出层参数更新推导

令 $\nabla_{\Theta^O} = \frac{\partial J(\Theta)}{\partial \Theta^O}$, 由式 (31) 和链导法则可得

$$\nabla_{\Theta^O} = \sum_{n=1}^{s}\sum_{t=t_0}^{\tau}(R_{n,t}^{O})^{\mathrm{T}}\Delta_{n,t}^{O} \tag{34}$$

其中, $\Delta_{n,t}^{O} = \frac{\partial J(\Theta)}{\partial Z_{n,t}^{O}}$, 即

$$\Delta_{n,t}^{O} = \begin{cases}\dfrac{\lambda^{s-n}\left(Z_{n,t}^{O} - Z_{n,t}^{O*}\right)}{m\left(\tau - t_0 + 1\right)}, & t \geq t_0\\ 0, & t < t_0\end{cases} \tag{35}$$

为了求取最优参数 Θ^{O*}, 进一步令 $\nabla_{\Theta^O} = 0$, 即

$$\sum_{n=1}^{s}\sum_{t=t_0}^{\tau}(R_{n,t}^{O})^{\mathrm{T}}\Delta_{n,t}^{O} = 0 \tag{36}$$

将式 (35) 代入式 (36), 得

$$\sum_{n=1}^{s}\sum_{t=t_0}^{\tau}\frac{\lambda^{s-n}(R_{n,t}^{O})^{\mathrm{T}}(R_{n,t}^{O}\Theta^{O} - Z_{n,t}^{O*})}{m(\tau - t_0 + 1)} = 0 \tag{37}$$

整理可得 Θ_s^O 的最小二乘解

$$\Theta_s^O = \Theta^{O*} = (A_s^O)^{-1}B_s^O \tag{38}$$

其中,

$$A_s^O = \sum_{n=1}^{s}\sum_{t=t_0}^{\tau}\frac{\lambda^{s-n}(R_{n,t}^{O})^{\mathrm{T}}R_{n,t}^{O}}{m(\tau - t_0 + 1)} \tag{39}$$

$$B_s^O = \sum_{n=1}^{s}\sum_{t=t_0}^{\tau}\frac{\lambda^{s-n}(R_{n,t}^{O})^{\mathrm{T}}Z_{n,t}^{O*}}{m(\tau - t_0 + 1)} \tag{40}$$

类似于 RLS 算法推导, 以上两式可进一步写成

如下递推形式

$$A_s^O = \lambda A_{s-1}^O + \sum_{t=t_0}^{\tau}\sum_{k=1}^{m}\frac{\boldsymbol{R}_{s,t,k}^O(\boldsymbol{R}_{s,t,k}^O)^{\mathrm{T}}}{m(\tau-t_0+1)} \tag{41}$$

$$B_s^O = \lambda B_{s-1}^O + \sum_{t=t_0}^{\tau}\sum_{k=1}^{m}\frac{\boldsymbol{R}_{s,t,k}^O(\boldsymbol{Z}_{s,t,k}^{O*})^{\mathrm{T}}}{m(\tau-t_0+1)} \tag{42}$$

其中, $\boldsymbol{R}_{s,t,k}^O \in \mathbf{R}^{h+1}$为 $(R_{s,t}^O)^{\mathrm{T}}$的第 k列向量, $\boldsymbol{Z}_{s,t,k}^{O*} \in \mathbf{R}^d$ 为$(Z_{s,t}^{O*})^{\mathrm{T}}$ 的第 k 列向量. 但是, 由于此处 RNNs 基于迷你批训练, 式 (41) 并不能像式 (24) 那样直接利用 Sherman-Morrison-Woodbury 公式求解 A_s^O 的逆.

考虑到同一批次中各样本 Θ_{s-1}^O, A_{s-1}^O 和 B_{s-1}^O 是相同的, 借鉴 SGD 计算迷你批平均梯度思想, 接下来采用平均近似方法来处理这一问题. 因为式 (41) 和式 (42) 可以重写为如下形式

$$A_s^O = \sum_{t=t_0}^{\tau}\sum_{k=1}^{m}\frac{\mathcal{A}_{s,t,k}^O}{m(\tau-t_0+1)} \tag{43}$$

$$B_s^O = \sum_{t=t_0}^{\tau}\sum_{k=1}^{m}\frac{\mathcal{B}_{s,t,k}^O}{m(\tau-t_0+1)} \tag{44}$$

其中,

$$\mathcal{A}_{s,t,k}^O = \lambda A_{s-1}^O + \boldsymbol{R}_{s,t,k}^O(\boldsymbol{R}_{s,t,k}^O)^{\mathrm{T}} \tag{45}$$

$$\mathcal{B}_{s,t,k}^O = \lambda B_{s-1}^O + \boldsymbol{R}_{s,t,k}^O(\boldsymbol{Z}_{s,t,k}^{O*})^{\mathrm{T}} \tag{46}$$

因而可使用如下公式来近似求得 $(A_s^O)^{-1}$ 和 Θ_s^O 为

$$(A_s^O)^{-1} \approx \sum_{t=t_0}^{\tau}\sum_{k=1}^{m}\frac{(\mathcal{A}_{s,t,k}^O)^{-1}}{m(\tau-t_0+1)} \tag{47}$$

$$\Theta_s^O \approx \sum_{t=t_0}^{\tau}\sum_{k=1}^{m}\frac{(\mathcal{A}_{s,t,k}^O)^{-1}\mathcal{B}_{s,t,k}^O}{m(\tau-t_0+1)} \tag{48}$$

令 $P_s^O = (A_s^O)^{-1}$, 根据式 (47) 和式 (38) 以及 Sherman-Morrison-Woodbury 公式, 整理后得如下更新式为

$$P_s^O \approx \frac{1}{\lambda}P_{s-1}^O - \sum_{t=t_0}^{\tau}\sum_{k=1}^{m}\frac{\boldsymbol{G}_{s,t,k}^O(\boldsymbol{\Lambda}_{s,t,k}^O)^{\mathrm{T}}}{\lambda m(\tau-t_0+1)} \tag{49}$$

$$\Theta_s^O \approx \Theta_{s-1}^O - \sum_{t=t_0}^{\tau}\sum_{k=1}^{m}\boldsymbol{G}_{s,t,k}^O(\boldsymbol{\Delta}_{s,t,k}^O)^{\mathrm{T}} \tag{50}$$

其中, $\boldsymbol{\Delta}_{s,t,k}^O \in \mathbf{R}^d$ 为 $(\Delta_{s,t}^O)^{\mathrm{T}}$ 的第 k 列向量, 且

$$\boldsymbol{\Lambda}_{s,t,k}^O = P_{s-1}^O\boldsymbol{R}_{s,t,k}^O \tag{51}$$

$$\boldsymbol{G}_{s,t,k}^O = \frac{\boldsymbol{\Lambda}_{s,t,k}^O}{\lambda + (\boldsymbol{\Lambda}_{s,t,k}^O)^{\mathrm{T}}\boldsymbol{R}_{s,t,k}^O} \tag{52}$$

2.2 RLS-RNN 隐藏层参数更新推导

令 $\nabla_{\Theta^H} = \frac{\partial J(\Theta)}{\partial \Theta^H}$, 由式 (31) 和链导法则可得

$$\nabla_{\Theta^H} = \sum_{n=1}^{s}\sum_{t=1}^{\tau}(R_{n,t}^H)^{\mathrm{T}}\Delta_{n,t}^H \tag{53}$$

其中, $\Delta_{n,t}^H = \frac{\partial J(\Theta)}{\partial Z_{n,t}^H}$, 使用 BPTT 算法计算其具体形式为

$$\Delta_{n,t}^H = \begin{cases}(\acute{\Delta}_{n,t}^H(\tilde{\Theta}_{n-1}^H)^{\mathrm{T}}) \circ \varphi'(Z_{n,t}^H), & t<\tau\\ (\Delta_{n,t}^O V_{n-1}^{\mathrm{T}}) \circ \varphi'(Z_{n,t}^H), & t=\tau\end{cases} \tag{54}$$

其中, $\acute{\Delta}_{n,t}^H = \left[\Delta_{n,t}^O,\, \Delta_{n,t+1}^H\right]$. 进一步令 $\nabla_{\Theta^H}=0$, 可得

$$\sum_{n=1}^{s}\sum_{t=1}^{\tau}(R_{n,t}^H)^{\mathrm{T}}\Delta_{n,t}^H = 0 \tag{55}$$

然而, 式 (54) 非常复杂, 且 $\varphi'(Z_{s,t}^H)$ 一般为非线性, 我们并不能将式 (54) 代入式 (55) 求得隐藏层参数 Θ^H 的最小二乘解.

接下来我们提出一种新的方法来导出 $\Delta_{n,t}^H$ 的等价形式, 藉此来获得 Θ^H 的最小二乘解. 临时定义一个新的隐藏层目标函数 $J^H(\Theta^H)$

$$J^H(\Theta^H) = \sum_{n=1}^{s}\sum_{t=1}^{\tau}\frac{\lambda^{s-n}}{2m\tau}\left\|Z_{n,t}^{H*} - Z_{n,t}^H\right\|_{\mathrm{F}}^2 \tag{56}$$

其中, $Z_{n,t}^{H*}$ 为该层非激活线性输出期望值. 显然, 如果 $J(\Theta)\to 0$, 那么 $J^H(\Theta^H)\to 0$. 即

$$\Theta_s^H = \arg\min_{\Theta^H} J^H(\Theta^H) \tag{57}$$

令 $\frac{\partial J^H(\Theta^H)}{\partial \Theta^H} = 0$, 得

$$\sum_{n=1}^{s}\sum_{t=1}^{\tau}\frac{\lambda^{s-n}}{m\tau}(R_{n,t}^H)^{\mathrm{T}}(Z_{n,t}^H - Z_{n,t}^{H*}) = 0 \tag{58}$$

对比式 (55) 和式 (58), 可以得到 $\Delta_{n,t}^H$ 的另一种等价定义形式

$$\Delta_{n,t}^H = \frac{\eta\lambda^{s-n}}{m\tau}\left(Z_{n,t}^H - Z_{n,t}^{H*}\right) \tag{59}$$

其中, η 为比例因子. 理论上讲, 不同迷你批数据对应的 η 应该有一定的差别. 但考虑到各批迷你批数据均是从整个训练集中随机选取, 因此可忽略这一差别. 根据式 (16) 可知 $Z_{n,t}^H = R_{n,t}^H\Theta^H$, 且将式 (59) 代入式 (55), 得

$$\sum_{n=1}^{s}\sum_{t=1}^{\tau}\frac{\eta\lambda^{s-n}}{m\tau}(R_{n,t}^H)^{\mathrm{T}}\left(R_{n,t}^H\Theta^H - Z_{n,t}^{H*}\right) = 0 \tag{60}$$

进一步整理, 可得 Θ_s^H 的最小二乘解

$$\Theta_s^H = \Theta_s^{H*} = (A_s^H)^{-1}B_s^H \tag{61}$$

其中,

$$A_s^H = \sum_{n=1}^{s}\sum_{t=1}^{\tau}\frac{\eta\lambda^{s-n}}{m\tau}(R_{n,\,t}^H)^{\mathrm{T}}R_{n,\,t}^H \tag{62}$$

$$B_s^H = \sum_{n=1}^{s}\sum_{t=1}^{\tau}\frac{\eta\lambda^{s-n}}{m\tau}(R_{n,\,t}^H)^{\mathrm{T}}Z_{n,\,t}^{H*} \tag{63}$$

式 (61) 的递归最小二乘解推导过程类似于输出层参数更新推导. 令 $P_s^H = (A_s^H)^{-1}$, 同样采用上文的近似平均求解方法, 易得

$$P_s^H \approx \frac{1}{\lambda}P_{s-1}^H - \frac{1}{\lambda m\tau}\sum_{t=1}^{\tau}\sum_{k=1}^{m}\boldsymbol{G}_{s,\,t,\,k}^H(\boldsymbol{\Lambda}_{s,\,t,\,k}^H)^{\mathrm{T}} \tag{64}$$

$$\Theta_s^H \approx \Theta_{s-1}^H - \frac{1}{\eta}\sum_{t=1}^{\tau}\sum_{k=1}^{m}\boldsymbol{G}_{s,\,t,\,k}^H(\boldsymbol{\Delta}_{s,\,t,\,k}^H)^{\mathrm{T}} \tag{65}$$

其中, $\boldsymbol{\Delta}_{s,\,t,\,k}^H \in \mathbf{R}^h$ 为 $(\Delta_{s,\,t}^H)^{\mathrm{T}}$ 的第 k 列向量, 且

$$\boldsymbol{\Lambda}_{s,\,t,\,k}^H = P_{s-1}^H\boldsymbol{R}_{s,\,t,\,k}^H \tag{66}$$

$$\boldsymbol{G}_{s,\,t,\,k}^H = \frac{\boldsymbol{\Lambda}_{s,\,t,\,k}^H}{\frac{\lambda}{\eta} + (\boldsymbol{\Lambda}_{s,\,t,\,k}^H)^{\mathrm{T}}\boldsymbol{R}_{s,\,t,\,k}^H} \tag{67}$$

需要说明的是, 因为我们并不知道隐藏层期望输出 $Z_{s,\,t}^{H*}$, 所以实际上不能通过式 (59) 来求取 $\Delta_{s,\,t}^H$. 幸运的是, 式 (54) 与 (59) 等价, 因此在算法具体实现中, 采用式 (54) 来替换式 (59).

综上, RLS-RNN 算法如算法 1 所示.

算法 1. 基于 RLS 优化的 RNN 算法

Require: 迷你批样本 $\{(X_1, Y_1^*), (X_2, Y_2^*), \cdots, (X_N, Y_N^*)\}$, 时间步 τ, 遗忘因子 λ, 比例因子 η, 协方差矩阵初始参数 α;

Initialize: 初始化权重矩阵 Θ_0^H 和 Θ_0^O, 初始化协方差矩阵 $P_0^H = \alpha I^H$, $P_0^O = \alpha I^O$;

for $s = 1, 2, \cdots, N$ **do**
  设置 $H_{s,\,0} = 0$;
  for $t = 1, 2, \cdots, \tau$ **do**
    用式 (3) 计算 $H_{s,\,t}$;
    用式 (12) 计算 $Z_{s,\,t}$;
  end for
  for $t = \tau, \tau-1, \cdots, 1$ **do**
    用式 (35) 计算 $\Delta_{s,\,t}^O$;
    用式 (54) 计算 $\Delta_{s,\,t}^H$;
    for $k = 1, \cdots, m$ **do**
      用式 (51), (52) 计算 $\boldsymbol{\Lambda}_{s,\,t,\,k}^O$, $\boldsymbol{G}_{s,\,t,\,k}^O$;
      用式 (66), (67) 计算 $\boldsymbol{\Lambda}_{s,\,t,\,k}^H$, $\boldsymbol{G}_{s,\,t,\,k}^H$;
    end for
  end for
  用式 (49), (50) 更新 P_s^O, Θ_s^O;
  用式 (64), (65) 更新 P_s^H, Θ_s^H;
end for.

3 分析与改进

3.1 复杂度分析

在 RNNs 当前所用优化算法中, SGD 是时间和空间复杂度最低的算法. 本节将以 SGD-RNN 为参照, 来对比分析本文提出的 RLS-RNN算法的时间和空间复杂度. 两个算法采用一个迷你批样本数据集学习的时间和空间复杂度对比结果如表 1 所示. 从第 1 节介绍可知, τ 表示序列数据时间步长度, m 表示批大小, a 表示单个样本向量的维度, h 表示隐藏层神经元数量, d 表示输出层神经元数量. 在实际应用中, a 和 d 一般要小于 h, 因而 RLS-RNN 的时间复杂度和空间复杂度大约为 SGD-RNN 的 3 倍. 在实际运行中, 我们发现 RLS-RNN 所用时间和内存空间大约是 SGD-RNN 的 3 倍, 与本节理论分析结果正好相吻合.

所提算法只需在 RNNs 的隐藏层和输出层各设置一个矩阵, 而以前的 RLS 优化算法则需为 RNNs 隐藏层和输出层的每一个神经元设置一个与所提算法相同规模的协方差矩阵, 因而所提算法在时间和空间复杂度上有着大幅降低. 此外, 所提算法采用了深度学习广为使用的迷你批训练方式, 使得其可用于处理较大规模的数据集.

3.2 λ 自适应调整

众多研究表明, 遗忘因子 λ 的取值对 RLS 算法性能影响较大[28], 特别是在 RLS 处理时变任务时影响更大. 由于本文所提算法建立在传统 RLS 基础之上, 因而 RLS-RNN 的收敛质量也易受 λ 的取值影响. 在 RLS 研究领域, 当前已有不少关于 λ 自适应调整方面的成果[28−29], 因此可以直接利用这些成果对 RLS-RNN 作进一步改进.

在文献 [29] 基础上, 本小节直接给出一种 λ 自适应调整方法. 对第 s 迷你批样本, RLS-RNN 各层中的遗忘因子统一定义为

$$\lambda_s = \begin{cases} \lambda_{\max}, & \text{若 } \sigma_s^e \le \kappa\sigma_s^v \\ \min\left(\dfrac{q_s\sigma_s^v}{\xi + |\sigma_s^e - \sigma_s^v|}, \lambda_{\max}\right), & \text{否则} \end{cases} \tag{68}$$

其中, $\lambda_{\max}$ 接近于 1, $\kappa > 1$ 用于控制 λ_s 更新, 一般建议取 2, 通常 κ 取值越小, λ_s 更新越频繁; ξ 是一个极小的常数, 防止在计算 λ_s 时分母为 0; q_s, σ_s^e

表 1 SGD-RNN 与 RLS-RNN 复杂度分析
Table 1 Complexity analysis of SGD-RNN and RLS-RNN

		SGD-RNN	RLS-RNN
时间复杂度	O_s	$\mathrm{O}(\tau mdh)$	—
	Z_s	—	$\mathrm{O}(\tau mdh)$
	H_s	$\mathrm{O}(\tau mh(h+a))$	$\mathrm{O}(\tau mh(h+a))$
	Δ_s^O	$\mathrm{O}(4\tau md)$	$\mathrm{O}(3\tau md)$
	Δ_s^H	$\mathrm{O}(\tau mh(h+d))$	$\mathrm{O}(\tau mh(h+d))$
	P_s^O	—	$\mathrm{O}(2\tau mh^2)$
	P_s^H	—	$\mathrm{O}(2\tau m(h+a)^2)$
	Θ_s^O	$\mathrm{O}(\tau mdh)$	$\mathrm{O}(\tau mdh)$
	Θ_s^H	$\mathrm{O}(\tau mh(h+a))$	$\mathrm{O}(\tau mh(h+a))$
	合计	$\mathrm{O}(\tau m(3dh+3h^2+2ha))$	$\mathrm{O}(\tau m(7h^2+2a^2+3dh+6ha))$
空间复杂度	Θ_s^O	$\mathrm{O}(hd)$	$\mathrm{O}(hd)$
	Θ_s^H	$\mathrm{O}(h(h+a))$	$\mathrm{O}(h(h+a))$
	P_s^H	—	$\mathrm{O}((h+a)^2)$
	P_s^O	—	$\mathrm{O}(h^2)$
	合计	$\mathrm{O}(h^2+hd+ha)$	$\mathrm{O}(hd+3ha+a^2+3h^2)$

和 σ_s^v 定义为

$$q_s=\mu_0 q_{s-1}+\frac{(1-\mu_0)\sum\limits_{t=t_0}^{\tau}\sum\limits_{k=1}^{m}(\boldsymbol{\Lambda}_{s,t,k}^O)^{\mathrm{T}}\boldsymbol{R}_{s,t,k}^O}{m(\tau-t_0+1)}$$

$$\sigma_s^e=\mu_1\sigma_{s-1}^e+\frac{(1-\mu_1)\sum\limits_{t=t_0}^{\tau}\sum\limits_{k=1}^{m}||\boldsymbol{\Delta}_{s,t,k}^O||_2^2}{md(\tau-t_0+1)}$$

$$\sigma_s^v=\mu_2\sigma_{s-1}^v+\frac{(1-\mu_2)\sum\limits_{t=t_0}^{\tau}\sum\limits_{k=1}^{m}||\boldsymbol{\Delta}_{s,t,k}^O||_2^2}{md(\tau-t_0+1)}$$

其中, μ_0 建议取 7/8; $\mu_1=1-1/(\varsigma_1 m)$, 通常 $\varsigma_1\geq 2$; $\mu_2=1-1/(\varsigma_2 m)$, 且 $\varsigma_2>\varsigma_1$.

当然, 采用以上方式更新 λ_s 将会引入新的超参数, 给 RLS-RNN 的调试带来一定困难. 从使用 RLS-RNN 的实际经验来看, 也可采用固定的 λ 进行训练, 建议将 λ 取值设置在 0.99 至 1 之间.

3.3 过拟合预防

传统 RLS 算法虽然具有很快的收敛速度, 但也经常面临过拟合风险, RLS-RNN 同样面临这一风险. 类似于第 3.2 节, 同样可以利用 RLS 领域关于这一问题的一些研究成果来改进 RLS-RNN.

Ekşioğlu[30] 提出了一种 L_1 正则化 RLS 方法, 即在参数更新时附加一个正则化项. 对其稍加改进, 则在式 (50) 和式 (65) 的基础上可分别重新定义为

$$\Theta_s^O\approx\Theta_{s-1}^O-\sum_{t=1}^{\tau}G_{s,t}^O\Delta_{s,t}^O-\gamma P_{s-1}^O\mathrm{sgn}(\Theta_{s-1}^O)\quad(69)$$

$$\Theta_s^H\approx\Theta_{s-1}^H-\frac{1}{\eta}\sum_{t=t_0}^{\tau}G_{s,t}^H\Delta_{s,t}^H-\gamma P_{s-1}^H\mathrm{sgn}(\Theta_{s-1}^H)\quad(70)$$

其中, γ 为正则化因子, $G_{s,t}^O=\left[\boldsymbol{G}_{s,t,1}^O,\cdots,\boldsymbol{G}_{s,t,m}^O\right]$, $G_{s,t}^H=\left[\boldsymbol{G}_{s,t,1}^H,\cdots,\boldsymbol{G}_{s,t,m}^H\right]$.

实际上, 除了这种方法外, 读者也可采用其他正则化方法对 RLS-RNN 作进一步改进.

4 仿真实验

为了验证所提算法的有效性, 本节选用两个序列数据分类问题和两个序列数据预测问题进行仿真实验. 其中, 两个分类问题为 MNIST 手写数字识别分类[31] 和 IMDB 影评正负情感分类, 两个预测问题为 Google 股票价格预测[32] 与北京市 PM2.5 污染预测[33]. 在实验中, 将着重验证所提算法的收敛性能、超参数 α 和 η 选取的鲁棒性. 在收敛性能验证中, 选用主流一阶梯度优化算法 SGD、Momentum 和 Adam 进行对比, 所有问题的实验均迭代运行 150 Epochs; 在超参数鲁棒性验证中, 考虑到所提算法收敛速度非常快, 所有问题的实验均只迭代运行 50 Epochs. 为了减少实验结果的随机性, 所有实验均重复运行 5 次然后取平均值展示结果. 此外, 为了观察所提算法的实际效果, 所有优化算法在 RNNs 参数更新过程均不进行 Dropout 处理. 需要特别说明的是: 对前两个分类问题, 由于时变性不强, 所提算法遗忘因子采用固定值方式而不采用第 3.2

节所提方式; 对后两个预测问题, 所提算法遗忘因子将采用第 3.2 节所提方式; 所提算法对 4 个问题均将采用第 3.3 节所提方法防止过拟合.

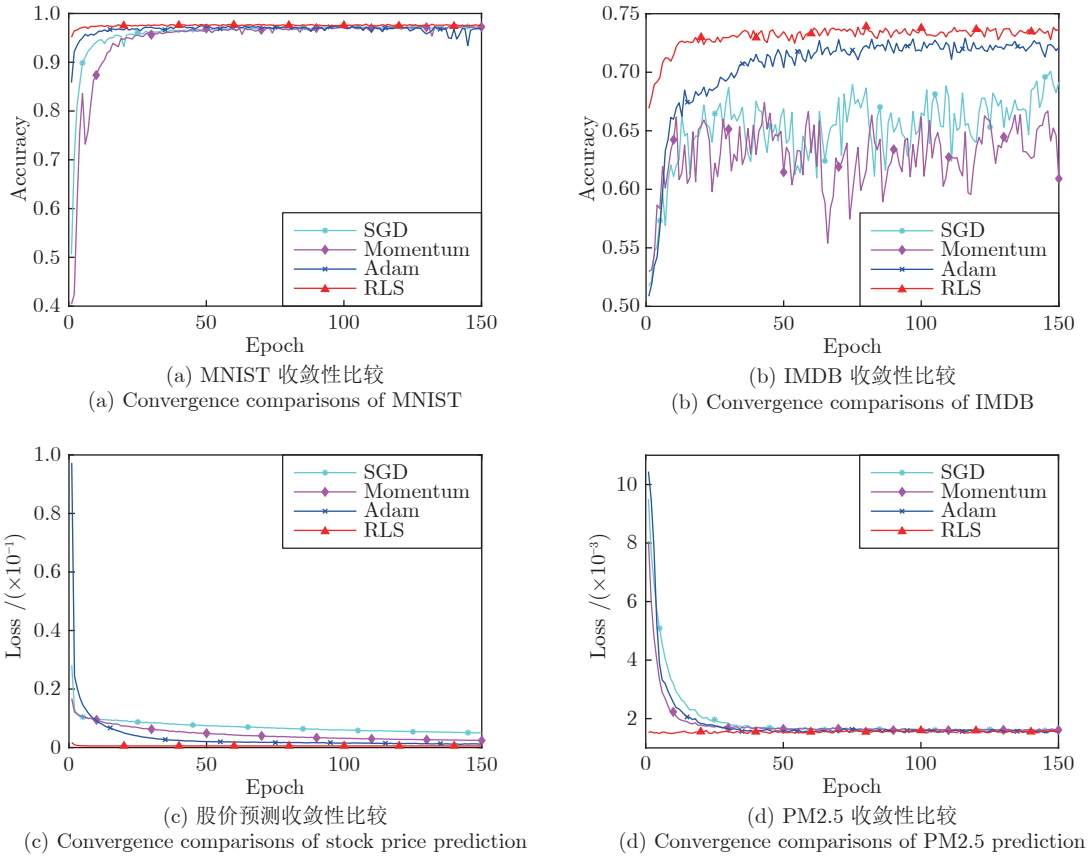
4.1　MNIST 手写数字识别分类

MNIST 分类问题的训练集与测试集分别由 55 000 和 10 000 幅 28×28 像素、共 10 类灰度手写数字图片组成, 学习目标是能对给定手写数字图片进行识别. 为了适应 RNNs 学习, 将训练集和测试集中的每张图片转换成一个 28 时间步的序列, 每时间步包括 28 个像素输入, 图片类别采用 One-hot 编码.

该问题所用 RNN 模型结构设置如下: 1) 输入层输入时间步为 28, 输入向量维度为 28. 2) 隐藏层时间步为 28, 神经元数为 100, 激活函数为 $\tanh(\cdot)$. 3) 输出层时间步为 1, 神经元数为 10, 激活函数为 $\tanh(\cdot)$. 由于 $\tanh^{-1}(1)$ 和 $\tanh^{-1}(-1)$ 分别为正、负无穷大, 在具体实现中, 对 $\tanh^{-1}(x)$, 我们约定: 若 $x\geq 0.997$, 则 $\tanh^{-1}(x)=\tanh^{-1}(0.997)$; 若 $x\leq -0.997$, 则 $\tanh^{-1}(x)=\tanh^{-1}(-0.997)$. RNN 模型权重参数采用 He 初始化[34].

在收敛性能对比验证中, 各优化算法超参数设置如下: RLS 遗忘因子 λ 为 0.9999, 比例因子 η 为 1, 协方差矩阵初始化参数 α 为 0.4, 正则化因子 γ 为 0.0001; SGD 学习率为 0.05; Momentum 学习率为 0.05, 动量参数 0.5; Adam 学习率 0.001, β_1 设为 0.9, β_2 为 0.999, ϵ 设为 10^{-8}. 在超参数 α 和 η 选取的鲁棒性验证中, 采用控制变量法进行测试: 1) 固定 $\lambda=0.9999$, $\gamma=0.0001$ 和 $\eta=1$, 依次选取 $\alpha=0.01, 0.1, 0.2, \cdots, 1$ 验证; 2) 固定 $\lambda=0.9999$, $\gamma=0.0001$ 和 $\alpha=0.4$, 依次选取 $\eta=0.1, 1, 2, \cdots, 10$ 验证.

在上述设定下, 每一 Epoch 均将训练集随机划分成 550 个迷你批, 批大小为 100. 每训练完一个 Epoch, 便从测试集中随机生成 50 个迷你批进行测试, 统计其平均分类准确率. 实验结果如图 2(a)、表 2 和表 3 所示. 由图 2(a) 可知, RLS 在第 1 个 Epoch 便可将分类准确率提高到 95% 以上, 其收敛速度远高于其他三种优化算法, 且 RLS 的准确率曲线比较平滑, 说明参数收敛比较稳定. 表 2 和表 3 记录了该实验取不同的 α 和 η 时第 50 Epoch 的平均分类准确率. 从表 2 中不难看出, 不同初始化因子 α 在第 50 Epoch 的准确率都在 97.10% 到 97.70% 之间波动, 整体来说比较稳定, 说明 α 对算法性能



(a) MNIST 收敛性比较
(a) Convergence comparisons of MNIST

(b) IMDB 收敛性比较
(b) Convergence comparisons of IMDB

(c) 股价预测收敛性比较
(c) Convergence comparisons of stock price prediction

(d) PM2.5 收敛性比较
(d) Convergence comparisons of PM2.5 prediction

图 2　收敛性比较实验结果

Fig. 2　Experimental results on the convergence comparisons

表 2 初始化因子 α 鲁棒性分析

Table 2 Robustness analysis of the initializing factor α

α	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
MNIST分类准确率 (%)	97.10	97.36	97.38	97.35	97.57	97.70	97.19	97.27	97.42	97.25	97.60
IMDB分类准确率 (%)	72.21	73.50	73.24	73.32	74.02	73.01	73.68	73.25	73.20	73.42	73.12
股价预测MSE ($\times 10^{-4}$)	5.32	5.19	5.04	5.43	5.42	5.30	4.87	4.85	5.32	5.54	5.27
PM2.5预测MSE ($\times 10^{-3}$)	1.58	1.55	1.53	1.55	1.61	1.55	1.55	1.54	1.57	1.58	1.57

表 3 比例因子 η 鲁棒性分析

Table 3 Robustness analysis of the scaling factor η

η	0.1	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0
MNIST分类准确率 (%)	97.80	97.59	97.48	97.61	97.04	97.62	97.44	97.33	97.38	97.37	97.45
IMDB分类准确率 (%)	73.58	73.46	73.62	73.76	73.44	73.82	73.71	72.97	72.86	73.12	73.69
股价预测MSE ($\times 10^{-4}$)	5.70	5.32	5.04	5.06	5.61	4.73	5.04	5.14	4.85	4.97	5.19
PM2.5预测MSE ($\times 10^{-3}$)	1.53	1.55	1.56	1.59	1.56	1.53	1.58	1.55	1.54	1.50	1.52

影响较小. 从表 3 中可知, 不同 η 取值的准确率均在 97.04% 到 97.80% 之间, 波动较小, η 取值对算法性能的影响也不大. 综上, RLS 算法的 α 和 η 取值均具有较好的鲁棒性.

4.2 IMDB 影评情感分类

IMDB 分类问题的训练集和测试集分别由 25 000 和 10 000 条电影评论组成, 正负情感评论各占 50%, 学习目标是能对给定评论的感情倾向进行识别. 为了适应 RNNs 学习, 首先从 Keras 内置数据集加载训练集和测试集的各条评论, 选取每条评论前 32 个有效词构成一个时间步序列, 然后对该评论中的每个有效词以 GloVe.6B 预训练模型[35] 进行词嵌入, 使得每个时间步包括 50 个输入维度, 评论的正负情感类别采用 One-hot 编码.

该问题所用 RNN 模型结构设置如下: 1) 输入层输入时间步为 32, 输入向量维度为 50. 2) 隐藏层时间步为 32, 神经元数为 100, 激活函数为 $\tanh(\cdot)$. 3) 输出层时间步为 1, 神经元数为 2, 激活函数为 $\tanh(\cdot)$. $\tanh^{-1}(x)$ 问题和 RNN 模型权重参数的初始化按第 4.1 节方式同样处理.

在收敛性能对比验证中, 各优化算法超参数设置如下: RLS 遗忘因子 λ 为 0.9999, 比例因子 η 为 1, 协方差矩阵初始化参数 α 为 0.4, 正则化因子 γ 为 0.001; SGD 学习率为 0.05; Momentum 学习率为 0.05, 动量参数 0.5; Adam 学习率 0.0001, β_1 设为 0.9, β_2 设为 0.999, ϵ 设为 10^{-8}. 在超参数 α 和 η 选取的鲁棒性验证中, 同样采用控制变量法进行测试: 1) 固定 $\lambda = 0.9999$, $\gamma = 0.001$ 和 $\eta = 1$, 依次选取 $\alpha = 0.01, 0.1, 0.2, \cdots, 1$ 验证; 2) 固定 $\lambda = 0.9999$, $\gamma = 0.001$ 和 $\alpha = 0.4$, 依次选取 $\eta = 0.1, 1, 2, \cdots, 10$ 验证.

在上述设定下, 每一 Epoch 均将训练集随机划分成 250 个迷你批, 批大小为 100. 每训练完一个 Epoch, 便从测试集中随机生成 50 个迷你批进行测试, 统计其平均分类准确率. 实验结果如图 2(b)、表 2 和表 3 所示. 由图 2(b) 可知, SGD 与 Momentum 的收敛不太稳定, 波动比较大, 而 Adam 的准确率曲线则比较平滑, 这三者在训练初期的准确率都比较低. 相比之下, RLS 在训练初期的准确率已经比较接近后期预测准确率, 前期收敛速度极快, 整体准确率也明显优于其余三种优化算法. 表 2 和表 3 记录了 IMDB 实验取不同的 α 和 η 时第 50 Epoch 的平均分类准确率. 由表 2 易知不同 α 的情况下准确率浮动范围比较小, 因此不同 α 对算法的影响比较小. 由表 3 可知, 采用不同 η 时其准确率在 72.86% 到 73.82% 之间浮动, 可见 η 的取值对算法性能影响较小. 综上, RLS 算法的 α 和 η 取值在本实验中同样都具有较好的鲁棒性.

4.3 Google 股票价格预测

Google 股票价格预测问题的数据源自 Google 公司从 2010 年 1 月 4 日到 2016 年 12 月 30 日的股价记录, 每日股价记录包括当日开盘价、当日最低价、当日最高价、交易笔数及当日调整后收盘价五种数值, 学习目标是能根据当日股价预测调整后次日收盘价. 为了适应 RNNs 学习, 首先对这些数值进行归一化处理, 然后以连续 50 个交易日为单位进行采样, 每次采样生成一条 5 维输入序列数据, 同时将该次采样后推一个交易日选取各日调整后收盘价生成对应的一维期望输出序列数据, 取前 1 400 条序列数据的训练集, 后续 200 条序列数据为测试

集, 并将训练集和测试集的样本分别随机置乱.

该问题所用 RNN 模型结构设置如下: 1) 输入层输入时间步为 50, 输入向量维度为 5. 2) 隐藏层时间步为 50, 神经元数为 50, 激活函数为 tanh(·). 3) 输出层时间步为 50, 神经元数为 1, 激活函数为 identity(·). RNN 模型权重参数采用高斯分布随机初始化.

在收敛性能对比验证中, 各优化算法超参数设置如下: RLS 遗忘因子采用第 3.2 节自适应方式, 其参数 κ=1.5, $\lambda_{\max}$=0.9999, $q_0=10$, μ_0=7/8, ς_1=6, $\varsigma_2=18$, $\xi=10^{-15}$, 鲁棒性实验中自适应参数与此相同, RLS 的比例因子 η 为 1, 协方差矩阵初始化参数 α 为 0.3, 正则化因子 γ 为 0.000001; SGD 学习率为 0.05; Momentum 学习率为 0.05, 动量参数 0.8; Adam 学习率 0.001, β_1 设为 0.9, β_2 设为 0.999, ϵ 设为 10^{-8}. 在超参数 α 和 η 选取的鲁棒性验证中, 采用控制变量法进行测试: 1) 固定 γ=0.000001 和 η=1, 依次选取 α=0.01, 0.1, 0.2, $\cdots$, 1 验证; 2) 固定 $\gamma=0.000001$ 和 $\alpha=0.3$, 依次选取 $\eta=0.1, 1, 2, \cdots, 10$ 验证.

在上述设定下, 每一 Epoch 均将训练集随机划分成 28 个迷你批, 批大小为 50. 每训练完一个 Epoch, 便从测试集中随机生成 50 个迷你批进行测试, 统计其均方误差 (mean square error, MSE). 实验结果如图 2(c)、表 2 和表 3 所示. 由图 2(c) 可知, Adam 在前期误差较大, 但是很快降低到 SGD 与 Momentum 的误差之下, Momentum 则比 SGD 稍强一些, RLS 则在迭代初期就几乎收敛. 表 2 和表 3 记录了该实验取不同的 α 和 η 时在第 50 Epoch 的平均 MSE. 由表 2 可知, 在不同 α 取值下, 实验结果都比较接近, 波动较小, 这说明 α 的取值对于 RLS 算法性能的影响较小. 由表 3 可知, 当 η 取不同值时, MSE 值变化不大, 因此 η 对算法影响较小. 综上, RLS 算法的 α 和 η 取值在本实验中也具有较好的鲁棒性.

4.4 北京市 PM2.5 污染预测

北京市 PM2.5 污染预测问题的数据源自 2010 年 1 月 1 日至 2014 年 12 月 31 日每小时的监测记录. 每条记录由 PM2.5 浓度、露点、摄氏温度、气压、组合风向、累计风速、降雪量、降水量共 8 个属性值组成, 学习目标是能根据当前小时的监测记录预测两个小时后的 PM2.5 浓度值. 为了适应 RNNs 学习, 首先对风向数据按序编码, 然后对属性数值归一化, 接着以连续 24 小时为单位进行采样, 每次采样生成一条 8 维输入序列数据, 同时将该次采样后推两个小时选取各小时的 PM2.5 浓度值生成一条对应的一维期望输出序列数据, 并取前 35 000 条序列数据为训练集, 后续 8 700 条序列数据为测试集. 需要说明的是, 由于这一问题比较简单且训练集较大, 所提算法和 Adam 运行一个 Epoch基本即可收敛, 为了更好地对各算法性能进行区分, 只取训练集前 7 000 条和测试集前 3 700 条数据并分别随机置乱进行训练和测试.

该问题所用 RNN 模型结构设置如下: 1) 输入层输入时间步为 24, 输入向量维度为 8. 2) 隐藏层时间步为 24, 神经元数为 50, 激活函数为 tanh(·). 3) 输出层时间步为 24, 神经元数为 1, 激活函数为 identity(·). RNN 模型权重参数的初始化按第 4.3 节方式同样处理.

在收敛性能对比验证中, 各优化算法超参数设置如下: RLS 遗忘因子采用第 3.2 节自适应方式, 其参数 $\kappa=2$, $\lambda_{\max}=0.9999$, $q_0=10$, $\mu_0=7/8$, $\varsigma_1=6$, $\varsigma_2=18$, $\xi=10^{-15}$, 鲁棒性实验中自适应参数与此相同, RLS 的比例因子 η 为 1, 协方差矩阵初始化参数 α 为 0.4, 正则化因子 γ 为 0.000001; SGD 学习率为 0.05; Momentum 学习率为 0.05, 动量参数 0.8; Adam 学习率 0.001, β_1 设为 0.9, β_2 设为 0.999, ϵ 设为 10^{-8}. 在超参数 α 和 η 选取的鲁棒性验证中, 同样采用控制变量法进行测试: 1) 固定 γ=0.000001 和 η=1, 依次选取 α=0.01, 0.1, 0.2, $\cdots$, 1 验证; 2) 固定 $\gamma=0.000001$ 和 $\alpha=0.4$, 依次选取 $\eta=0.1, 1, 2, \cdots, 10$ 验证.

在上述设定下, 每一 Epoch 均将训练集随机划分成 140 个迷你批, 批大小为 50. 每训练完一个 Epoch, 便从测试集中随机生成 50 个迷你批进行测试, 统计其均方误差损失. 实验结果如图 2(d)、表 2 和表 3 所示. 由图 2(d) 可知, SGD、Momentum、Adam 的损失初期均较大, 收敛速度较缓慢; 而 RLS 的曲线几乎在第 1 个 Epoch 就收敛完成, 因此其收敛速度要优于 3 个对比优化算法. 表 2 和表 3 记录了该实验取不同的 α 和 η 时在第 50 Epoch 的平均 MSE. 表 2 中在取不同 α 时, 其 MSE 上下波动幅度较小, 且没有明显的变化趋势, 因此我们认为 α 对算法性能影响较小. 由表 3 可知, 对 η 取不同值时, 其平均 MSE 在 1.50×10^{-3} 到 1.59×10^{-3} 间波动, 整体来说浮动范围较小. 综上, RLS 算法的 α 和 η 取值都具有较好的鲁棒性.

5 结论与展望

在 RNNs 优化训练中, 现有一阶优化算法学习速度较慢, 而二阶优化算法和以前的 RLS 类型优化算法时空复杂度又过高. 为此, 本文提出了一种新的 RLS 优化算法. 该算法吸收了深度学习中广为应

用的迷你批训练学习模式, 在推导过程中我们将研究重点放置在隐藏层和输出层的非激活线性输出上, 通过等价梯度替换, 最终得到各层权重参数的递归最小二乘解. 所提算法只需在 RNNs 的隐藏层和输出层各添加一个协方差矩阵, 解决了长期以来 RLS 优化算法应用时需要为隐藏层和输出层的每一神经元设置一个协方差矩阵的问题, 极大地降低了时空复杂度, 使得 RLS 可以适用于较大规模的 RNNs 训练. 在此基础上, 采用遗忘因子自适应调整和正则化技术对所提算法作了改进, 进一步提高了所提算法的性能. 4 组仿真实验表明, 所提算法在收敛性能、稳定性以及超参数选取的鲁棒性等方面均要明显优于主流一阶优化算法, 能够有效加快 RNNs 模型的训练速度, 降低超参数的选择难度. 此外, 在实验过程中我们还发现所提算法可缓解梯度消失导致 RNNs 无法训练的问题. 如何将本算法扩展到 RNNs 以外的其他深度学习网络以及如何进一步降低所提算法的时空复杂度将是我们下一步工作的重点.

References

1 Mikolov T, Karafiát M, Burget L, Cernockỳ J, Khudanpur S. Recurrent neural network based language model. In: Proceedings of the 11th Annual Conference of the International Speech Communication Association. Chiba, Japan: ISCA, 2010. 1045−1048

2 Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks. In: Proceeding of the 27th International Conference on Neural Information Processing Systems. Montréal, Canada: MIT Press, 2014. 3104−3112

3 Graves A, Mohamed A R, Hinton G. Speech recognition with deep recurrent neural networks. In: Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. Vancouver, Canada: IEEE, 2013. 6645−6649

4 Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. In: Proceedings of the 30th International Conference on Machine Learning. Atlanta, USA: JMLR.org, 2013. 1310−1318

5 Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 1994, **5**(2): 157−166

6 Bottou L. On-line learning and stochastic approximations. *On-Line Learning in Neural Networks*. New York: Cambridge University Press, 1999. 9−42

7 Polyak B T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 1964, **4**(5): 1−17

8 Sutskever I, Martens J, Dahl G, Hinton G. On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on International Conference on Machine Learning. Atlanta, USA: JMLR.org, 2013. 1139−1147

9 Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 2011, **12**: 2121−2159

10 Tieleman T, Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude [Online], available: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, February 19, 2020

11 Zeiler M D. ADADELTA: An adaptive learning rate method [Online], available: https://arxiv.org/abs/1212.5701\newblock, February 19, 2020

12 Kingma D P, Ba J. Adam: A method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations. San Diego, USA: ICLR, 2015.

13 Keskar N S, Socher R. Improving generalization performance by switching from adam to SGD [Online], available: https://arxiv.org/abs/1712.07628, February 19, 2020

14 Reddi S J, Kale S, Kumar S. On the convergence of Adam and beyond. In: Proceedings of the 6th International Conference on Learning Representations. Vancouver, Canada: OpenReview.net, 2018.

15 Martens J. Deep learning via hessian-free optimization. In: Proceedings of the 27th International Conference on Machine Learning. Haifa, Israel: Omnipress, 2010. 735−742

16 Keskar N S, Berahas A S. adaQN: An adaptive quasi-newton algorithm for training RNNs. In: Proceedings of the 2016 Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Riva del Garda, Italy: Springer, 2016. 1−16.

17 Martens J, Grosse R. Optimizing neural networks with kronecker-factored approximate curvature. In: Proceedings of the 32nd International Conference on Machine Learning. Lille, France: JMLR.org, 2015. 2408−2417

18 Shanno D F. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 1970, **24**(111): 647−656

19 Liu D C, Nocedal J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 1989, **45**(1−3): 503−528

20 Le Q V, Ngiam J, Coates A, Lahiri A, Prochnow B, Ng A Y. On optimization methods for deep learning. In: Proceedings of the 28th International Conference on Machine Learning. Bellevue, USA: Omnipress, 2011. 265−272

21 Azimi-Sadjadi M R, Liou R J. Fast learning process of multilayer neural networks using recursive least squares method. *IEEE Transactions on Signal Processing*, 1992, **40**(2): 446−450

22 Tan Yong-Hong. RLS training algorithm for multilayer feedforward neural networks and its application to identification. *Control Theory and Applications*, 1994, **11**(5): 594−599
(谭永红. 多层前向神经网络的RLS训练算法及其在辨识中的应用. 控制理论与应用, 1994, **11**(5): 594−599)

23 Xu Q, Krishnamurthy K, McMillin B, Lu W. A recursive least squares training algorithm for multilayer recurrent neural networks. In: Proceedings of the 1994 American Control Conference-ACC'94. Baltimore, USA: IEEE, 1994. 1712−1716

24 Peter T, Pospíchal J. Neural network training with extended Kalman filter Using graphics processing unit. In: Proceedings of the 18th International Conference on Artificial Neural Networks. Berlin, Germany: Springer, 2008. 198−207

25 Jaeger H. Adaptive nonlinear system identification with echo state networks. In: Proceedings of the 15th International Conference on Neural Information Processing Systems. Vancouver, Canada: MIT Press, 2002. 609−616

26 Werbos P J. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 1990, **78**(10): 1550−1560

27 Sherman J, Morrison W J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 1950, **21**(1): 124−127

28 Paleologu C, Benesty J, Ciochina S. A robust variable forgetting factor recursive least-squares algorithm for system identification. *IEEE Signal Processing Letters*, 2008, **15**: 597−600

29 Albu F. Improved variable forgetting factor recursive least square algorithm. In: Proceedings of the 12th International Conference on Control Automation Robotics and Vision (ICARCV).

Guangzhou, China: IEEE, 2012. 1789−1793

30 Ekşioğlu E M. RLS adaptive filtering with sparsity regularization. In: Proceedings of the 10th International Conference on Information Science, Signal Processing and Their Applications (ISSPA 2010). Kuala Lumpur, Malaysia: IEEE, 2010. 550−553

31 LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998, **86**(11): 2278−2324

32 Gawlik D. New York stock exchange [Online], available: https://www.kaggle.com/dgawlik/nyse, February 19, 2020

33 Liang X, Zou T, Guo B, Li S, Zhang H Z, Zhang S Y, et al. Assessing Beijing's PM2.5 pollution: Severity, weather impact, APEC and winter heating. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2015, 471(2182): Article No. 20150257

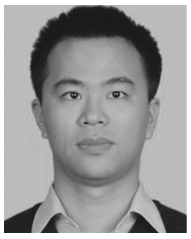
34 He K M, Zhang X Y, Ren S Q, Sun J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). Santiago, Chile: IEEE, 2015. 1026−1034

35 Pennington J, Socher R, Manning C. GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: ACL, 2014. 1532−1543

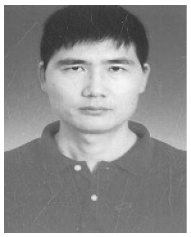
赵 杰 海南大学计算机科学与技术学院硕士研究生. 主要研究方向为深度学习和强化学习.
E-mail: zhaojie@lonelyme.cn
(**ZHAO Jie** Master student at the School of Computer Science and Technology, Hainan University. His research interest covers deep learning and reinforcement learning.)

张春元 海南大学计算机科学与技术学院副教授. 2016 年获得电子科技大学计算机软件与理论博士学位. 主要研究方向为深度学习与强化学习. 本文通信作者.
E-mail: zcy7566@126.com
(**ZHANG Chun-Yuan** Associate professor at the School of Computer Science and Technology, Hainan University. He received his Ph.D. degree in computer software and theory from University of Electronic Science and Technology of China in 2016. His research interest covers deep learning and reinforcement learning. Corresponding author of this paper.)

刘 超 海南大学计算机科学与技术学院硕士研究生. 主要研究方向为深度学习与强化学习.
E-mail: lcdyx0618@126.com
(**LIU Chao** Master student at the School of Computer Science and Technology, Hainan University. His research interest covers deep learning and reinforcement learning.)



周 辉 海南大学计算机科学与技术学院副教授. 2008 年获得中国科学院软件研究所博士学位. 主要研究方向为自然语言处理, 人工智能写作与数据可视化.
E-mail: zhouhui@hainanu.edu.cn
(**ZHOU Hui** Associate professor at the School of Computer Science and Technology, Hainan University. He received his Ph.D. degree from the Software Institute, Chinese Academy of Sciences in 2008. His research interest covers natural language processing, artificial intelligence writing, and data visualization.)



欧宜贵 海南大学理学院教授. 2003 年获得中国科学技术大学博士学位. 主要研究方向为最优化算法.
E-mail: ouyigui@126.com
(**OU Yi-Gui** Professor at the School of Science, Hainan University. He received his Ph.D. degree from University of Science and Technology of China in 2003. His main research interest is numerical optimization algorithm.)

宋 淇 海南大学计算机科学与技术学院硕士研究生. 主要研究方向为深度学习与强化学习.
E-mail: songqihnu@163.com
(**SONG Qi** Master student at the School of Computer Science and Technology, Hainan University. Her research interest covers deep learning and reinforcement learning.)