Least-Squares Temporal Difference Learning with Eligibility Traces based on Regularized Extreme Learning Machine

Dazi Li*, Luntong Li, Tianheng Song, Qibing Jin

College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, P R China
E-mail: lidz@mail.buct.edu.cn

Abstract: The task of learning the value function under a fixed policy in continuous Markov decision processes (MDPs) is considered. Although ELM has fast learning speed and can avoid tuning issues of traditional artificial neural network (ANN), the randomness of the ELM parameters would result in fluctuating performance. In this paper, a least-squares temporal difference algorithm with eligibility traces based on regularized extreme learning machine (RELM-LSTD(λ)) is proposed to overcome these problems caused by ELM in Reinforcement Learning problem. The proposed algorithm combined the LSTD(λ) algorithm with RELM. The RELM is used to approximate value functions. Furthermore, the eligibility trace term is introduced to increase data efficiency. In experiments, the performances of the proposed algorithm are demonstrated and compared with those of LSTD and ELM-LSTD. Experiment results show that the proposed algorithm can achieve a more stable and better performance in approximating the value function under a fixed policy.

Key Words: Reinforcement learning; Markov decision processes, Function approximation, Least-squares temporal difference learning, Regularized Extreme learning machine.

1 INTRODUCTION

Reinforcement learning (RL) method is a machine learning framework for solving sequential decision-making problems [1]. Since such problems arise in the crossing field of artificial intelligence, control theory and many other related research areas [2, 3, 4, 5], RL has been widely studied as an universal optimization method. The environments in RL are usually modelled as Markov decision processes (MDPs). As the standard approach of MDPs, dynamic programming (DP) is inefficient when the MDP has a large number of states and actions or the precise model is unknown. However, RL is considered as a way of turning DP into practical algorithms to solve large-scale problems [6, 7].

Most of RL algorithms learn the value function under a fixed policy in a MDP [6]. Temporal difference learning (TD) is one of the most well-known algorithms for estimating value function through the observed samples [6]. However, TD with function approximation diverges under off-policy sampling. In order to make TD learning suitable for off-policy situation, gradient-based TD learning and least-square-based TD learning have been developed. Gradient-based TD learning contains a series of first-order algorithms, such as off-policy TD(λ) [8], off-policy TD with corrections (TDC) and off-policy gradient TD ver. 2 (GTD2) [9], etc. Meanwhile, least-squares-based TD learning contains several batch methods such as off-policy least-squares TD (LSTD) [10], off-policy least-squares policy evaluation (LSPE) [10], and off-policy convergence least-squares TD with gradient correction (LSTDC) [11], etc. Algorithms in both of the above two categories have been extended to eligibility traces [12], and combined with linear function approximation (LFA) as the value function estimators. Most of the LFAs in RL use local feature functions, but the limitation is that the number of required features grows exponentially with the dimension of input space [13]. Thus, local feature functions are not suitable for high dimensional problems [13-15].

Recently, extreme learning machine (ELM), a novel kind of artificial neural network (ANN) using global feature function, has attracted great attention because of extremely fast learning speed and good generalization performance [16]. ELM has been used to solve some value prediction problems successfully, such as Q-value function approximation by using an ELM-based Q-learning method [17], linked multicomponent robotic systems dynamics approximation for RL [18], and a least-squares temporal difference combined with the ELM (ELM-LSTD) [15], etc. All of these successful applications benefit from the global approximate ability of ELM to improve the performance of approximation. In addition, the nonlinear parameterized structure of ELM has more powerful generalization ability than LFA.

However, the random initialization of ELM parameters can result in fluctuating performance [19-21]. Especially for sparse data, the effect of random parameters imposed on the generalization performance can be quite significant [22]. Various algorithms have been studied to overcome this difficulty lately [19-23]. Among these methods, a regularized variant named Regularized ELM (RELM) is an efficient method to achieve stable performance [23]. Therefore, a good performance can be expected by combining LSTD with RELM.

This work is supported by National Nature Science Foundation under Grant 61573052.

This paper focus on using RELM combined with LSTD (RELM-LSTD) to improve the approximate performance in RL. Also, RELM-LSTD is extended to RELM-LSTD(λ) to speed up the learning convergence. The proposed algorithm not only reduces the tuning work comparing with LSTD, but also achieves more stable and higher accuracy performance than ELM-LSTD. In our experiments, the learning performances of RELM-LSTD(λ) are compared with those of LSTD and ELM-LSTD, and the experimental results demonstrate the effectiveness of the proposed algorithm.

The rest of this paper is organized as follows. An introduction to ELM is summarized in Section 2. In Section 3, the background on RL especially the LSTD(λ) and the proposed LSTD base on RELM is given. In Section 4, the experimental results are shown to evaluate the performance of the proposed method. Section 5 draws conclusions and future works.

2 REGULARIZED EXTREME LEARNING MACHINE

ELM is a simple and efficient learning method proposed by Hung et al [16]. Good generalization property, simple structure and convenient calculation are the advantages of this algorithm. However, the randomness of the ELM parameters would result in unstable performance. A simple and effective way to overcome this problem is to introduce regularization. The essence of RELM is that the input weights and hidden biases are initialized randomly and the output weights are calculated analytically by using Moore–Penrose generalized inverse with ridge regression [23]. In this way, RELM provides faster learning speed than the methods based on gradient-descent and global search.

For the sake of simplicity, only the case of single output regression problem is considered. The output y with M hidden nodes is given by

$$y = \sum_{i=1}^{M} \beta_i f_i(x \mid w_i, b_i) = H\beta \tag{1}$$

where x is inputs vector of samples, and (ω_i, b_i) are the weights and biases vectors of the hidden layer respectively, f_i is the active function, i is the number of hidden node, H is the output matrix of hidden layer, and β is the weight matrix of output layer. Given samples $(x_j, t_j), j = 1,2,...N$, Eq.(1) can be rewritten as

$$T = H\beta \tag{2}$$

where

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} f(x_1 \mid \omega_1, b_1) & \cdots & f(x_1 \mid \omega_M, b_M) \\ \vdots & \ddots & \vdots \\ f(x_N \mid \omega_1, b_1) & \cdots & f(x_N \mid \omega_M, b_M) \end{bmatrix}_{N \times M} \tag{3}$$

$$\beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_N \end{bmatrix}_{M \times 1} \text{ and } T = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}_{N \times 1} \tag{4}$$

The only parameter vector β can be calculated by least-squares method:

$$\beta = H^* T \tag{5}$$

where H^* is the Moore–Penrose generalized inverse of H:

$$H^* = (H^T H + gI)^{-1} H^T \tag{6}$$

where I is an identity matrix, g is a small positive value.

3 REINFORCEMENT LEARNING

The goal of policy evaluation is to estimate the value function under a fixed policy according to samples observed from MDPs. A MDP is defined as a tuple $\{S, A, R, P\}$, where S is the state space, A is the action space, R is the reword function, P is the transition probability function. For each transition $s_t \to s_{t+1}$, an immediate reward r_t is given to the agent, according to R. The value function V^π under policy π is defined as the expected value of the future discounted total rewards [6]:

$$V^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s] \tag{7}$$

where $\mathbb{E}[\cdot]$ refers to the expectation and $\gamma \in [0,1)$ is the discount factor. V^π can be estimated through function approximation method [7, 15]:

$$V_\theta(s) = F(\theta, s) \tag{8}$$

where θ is a vector of parameters and $F(\cdot)$ can be nonlinear or linear function. The linear $F(\theta, s) = \theta^{\mathrm{T}} \phi(s)$ is more popular, because the theoretical analysis and application is more convenient, where $\phi(s) : \mathbb{R} \to \mathbb{R}^d$ is the feature basis and d is the dimension of feature basis. However, it has been proved that the nonlinear parameterized $F(\cdot)$ have more powerful generalization ability than that of the linear function.

3.1 LSTD learning with eligibility traces (LSTD(λ))

LSTD(λ) is derived from temporal difference learning with eligibility traces (TD(λ)) [24]. Given a finite sample set $D_n = \{(s_0, r_1, s_1), (s_1, r_2, s_2), ..., (s_{n-1}, r_n, s_n)\}$ generated under policy π, LSTD(λ) finds a parameter vector of linear function approximation $\theta \in \mathbb{R}^d$:

$$\begin{aligned} \mathbb{E}[(r_{t+1} + \gamma \phi_{t+1}^{\mathrm{T}} \theta_t - \phi_t^{\mathrm{T}} \theta_t) z_{t+1}] &= 0 \\ \mathbb{E}[(\phi_t^{\mathrm{T}} \theta_t - \gamma \phi_{t+1}^{\mathrm{T}} \theta_t) z_{t+1}] &= \mathbb{E}[r_{t+1} z_{t+1}] \\ \frac{1}{n} \sum_{t=0}^{n-1} (\phi_t^{\mathrm{T}} \theta_t - \gamma \phi_{t+1}^{\mathrm{T}} \theta_t) z_{t+1} &= \frac{1}{n} \sum_{t=0}^{n-1} r_{t+1} z_{t+1} \\ \sum_{t=0}^{n-1} (\phi_t^{\mathrm{T}} \theta_t - \gamma \phi_{t+1}^{\mathrm{T}} \theta_t) z_{t+1} &= \sum_{t=0}^{n-1} r_{t+1} z_{t+1} \\ \sum_{t=0}^{n-1} (\phi_t^{\mathrm{T}} - \gamma \phi_{t+1}^{\mathrm{T}}) z_{t+1} \theta_t &= \sum_{t=0}^{n-1} r_{t+1} z_{t+1} \end{aligned} \tag{9}$$

where z_{t+1} is the eligibility traces defined as

$$z_{t+1} = \sum_{k=0}^{t} (\gamma\lambda)^{t-k} \phi_{k+1} = \lambda\gamma z_t + \phi_{t+1} \tag{10}$$

Obviously, Eq. (9) is linear with respect to θ. The left hand side and right hand side of Eq. (12) can be denoted as $\hat{A}_n$ and $\hat{b}_n$, respectively as

$$\hat{A}_n = \sum_{t=0}^{n-1} (\phi_t^{\mathrm{T}} - \gamma\phi_{t+1}^{\mathrm{T}}) z_{t+1} \theta_t = \hat{A}_{n-1} + z_t (\phi_t^{\mathrm{T}} - \gamma\phi_{t+1}^{\mathrm{T}})^{\mathrm{T}} \tag{11}$$

$$\hat{b}_n = \sum_{t=0}^{n-1} r_{t+1} z_{t+1} = \hat{b}_{n-1} + r_{t+1} \tag{12}$$

If the matrix $\hat{A}_n$ is non-singular, the solution of LSTD(λ) is:

$$\theta_n = \hat{A}_n^{-1} \hat{b}_n \tag{13}$$

In TD (λ), $\lambda \in [0,1]$ is a parameter allowing to unify Monte-Carlo and TD(0) method. Actually, LSTD(λ) algorithm converges to the same solution as TD(λ). Previous studies have shown that LSTD(λ) converges faster than TD(λ). Furthermore, LSTD (λ) avoids the tuning work of step-size and the choice of the initial value θ. However, the price of these merits is the increased computational complexity of LSTD(λ).

3.2 LSTD (λ) based on Regularized ELM

In this part, RELM-LSTD(λ) is proposed to approximate value function. Similarly as [15], the proposed method combines the LSTD(λ) with RELM to avoid the tuning work of parameters of feature functions instead of LSTD with radial basis functions. That is, the operation of randomly initializing the parameters of hidden layer simplifies the training process of RELM and then the output layer weights can be calculated analytically. Furthermore, the introduction of regularization improves the robust performance of ELM. The training approach of RELM can be viewed as the process of mapping the input space into feature space through a nonlinear transformation in the hidden nodes, and then computing the output layer weights by using ridge regression. The training process of RELM can be summarized as the following steps:

1. Randomly initialize weights ω and biases b ; initialize g.
2. Calculate the output matrix H of the hidden layer using Eq.(3).
3. Calculate H^* as in Eq.(6)
4. Calculate the weight vector β of output layer using Eq.(5).

Compared to ELM, the target of RELM is to minimize the objective function:

$$J_2(\beta) = \min_{\beta} \left\{ \left\| H\beta - T \right\|_2^2 + g \left\| \beta \right\|_2^2 \right\} \tag{14}$$

The extra penalty term $g\left\|\beta\right\|_2^2$ makes the values of output weights smaller to achieve a better generalization ability.

In RELM-LSTD(λ), the feature vector is chosen as

$$\begin{cases} \phi(s_t) = [f(s_t \mid \omega_1, b_1), f(s_t \mid \omega_2, b_2), ..., f(s_t \mid \omega_k, b_k)] \\ f(s_t \mid \omega_i, b_i) = \dfrac{1}{1 + \exp(-(\omega_i s_t + b_i))} \end{cases} \tag{15}$$

where ω_k and b_k are the weights and bias of the k th hidden node. The matrix $\hat{A}$ and vector $\hat{b}$ are calculated from equations (14) and (15) respectively. Therefore, the output weights β can be computed as

$$\beta = (\hat{A} + gI)^{-1} \hat{b} \tag{16}$$

Clearly, RELM-LSTD(λ) aims at minimizing:

$$J_3(\beta) = \min_{\beta} \left\{ \left\| \hat{A}\beta - \hat{b} \right\|_2^2 + g \left\| \beta \right\|_2^2 \right\} \tag{17}$$

The output of RELM is the estimated state-value:

$$V^{\pi}(s) = \phi(s)^{\mathrm{T}} \beta \tag{18}$$

Thus, the algorithm steps of RELM-LSTD(λ) is shown in Table 1.

Table1. The algorithm steps of RELM-LSTD(λ)

Algorithm 1. LSTD(λ) learning based on RELM.

Given: Policy π, discount factor γ, reward function R, sources of data samples $\{(s_t, r_2, s_{t+1})\}$ generated by an MRP under the policy π.

1: Randomly initialize weights ω and biases b of the hidden nodes.
2: Define the activation function f and the number of hidden nodes M.
3: Initialize λ, g; $\hat{A} = 0$; $\hat{b} = 0$; t=0.
4: **repeat**
5: Select a start state
6: **while** $s_t \neq s_{end}$ **do**
7: Compute $\hat{A}$, $\hat{b}$, z_{t+1} as in (11), (12), (10);
8: $t = t + 1$
9: **end while**
10: **until** reaching the desired number of episodes
11: Compute β as in (16);
12: return V^{π} as in (18).

RELM-LSTD(λ) avoids feature selection. The learning prediction problem is solved by VFA in off-line manner and the output layer weights are trained through ridge regression. Feature selection (also called basis functions selection) is an active area in RL recently, and ELM provides an alternative method. Previous works focus on avoiding overfitting. Relative works include the uses of regression trees [25], kernel method [26, 27], least angle regression with l^1 -regularization [28], etc. Here, the proposed algorithm makes feature vectors very convenient that RELM-LSTD(λ) only needs to initial feature function randomly.

4 EMPIRICAL RESULTS

In this section, two experiments are given to show the effectiveness of the proposed method. First, the performance of RELM-LSTD(λ) is compared with those of LSTD and ELM-LSTD on the learning prediction problem

in a finite MRP. Then the effectiveness of these two algorithms are shown in a continuous MRP. The first experiment is a chain walk problem called Hop-World problem [29-31], and the second experiment is a continuous-state random-walk problem. Note that tuning the feature vector of LSTD is hard when the feature space is large, because each basis function has two parameters: centre and width. Thus the same tuning method as LSTD is adopted according to the previous work [13, 15]. The parameters of the feature vector in ELM-LSTD and RELM-LSTD(λ) are initialized randomly as mentioned above.

In our experiment, the true value functions are calculated by:

$$V = R + \gamma PV = (I - \gamma P)^{-1} R \tag{19}$$

where R is the vector with components $\mathrm{E}\{\mathrm{r}_{t+1} \mid s_t = s\}$ and P is the state-transition probabilities matrix.

4.1 Hop-World problem

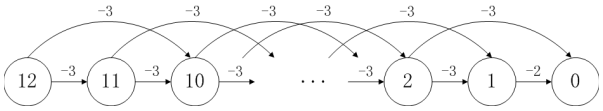

Fig 1. The Hop-World problem.

The Hop-World problem consists of a chain with 13 states (numbered from 12 to 0), shown in Fig.1. Each episode starts from the initial state 12 and ends in the absorbing state 0. In each non-absorbing state, two actions are available: taking one step toward the right side (denoted as '0'), or taking two steps toward the right side (denoted as '1'). In state 2-12, either action to be adopted shares the same probability of 0.5. The reward function is -2 when the transition from state 1 to state 0 occurs, and -3 at remaining transitions.

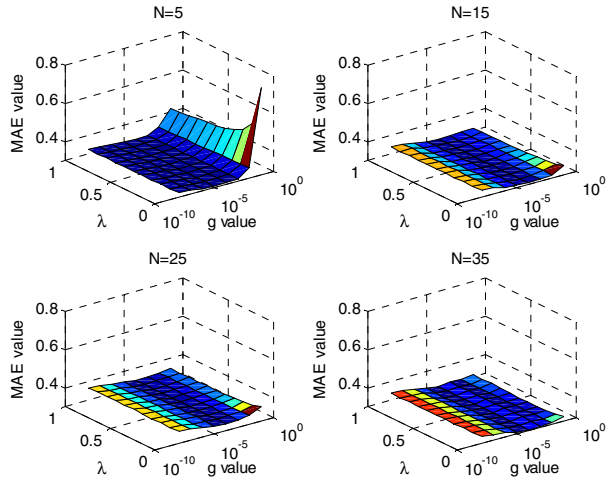

Fig 2. MAE error of RELM-LSTD(λ) on the Hop-World problem, for different values of λ and g. Each subpanel shows the MAE error at different number of features. The number of features in the subtitle refers to dimension of basis function for RBF and hidden nodes number for RELM.

This experiment employs the standard LSTD, ELM-LSTD and RELM-LSTD(λ) to construct VFA. The task is to approximate the true value function under a fixed policy: choosing the action "0" or "1" with the same probability 0.5. Under this policy, 20 episodes as a sample set are generated and recorded with the format $\{s_k, r_{k+1}, s_{k+1}\}$. Then, LSTD, ELM-LSTD and RELM-LSTD(λ) construct the VFA from the identical sample set as a test run. The performances of both methods are measured by the mean absolute error (MAE) of 20 episodes averaged over 100 independent runs. According to (19), the true value function for state $i(0 \le i \le 12)$ is $\mathrm{V}(i) = -2i$. The performances of all the algorithms are compared under the same feature number, which is noted as N.

For LSTD, the feature vector consists of some RBF basis. The center of RBF is distributed uniformly in the state space, and the widths are fixed as [13，15].

$$\sigma = \frac{d_{\max}}{\sqrt{2M}} \tag{20}$$

where d_{max} denotes the maximum among the distance between any two center positions, and M denotes the number of centers. The regularization parameter g in RELM-LSTD(λ) is chosen from the set of $g \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$.

Fig.2 shows the performance of RELM-LSTD(λ) under different values of g, λ and N. The results show that g value has strong effect on the performance of RELM-LSTD(λ). When g=10^{-4}, RELM-LSTD(λ) achieves the best performance. When g is relatively large, the effect of λ becomes large. The results also show that λ value has slightly effect on RELM-LSTD(λ)'s performance. Fig.3 compares RELM-LSTD(λ) (at the best g value and λ value)

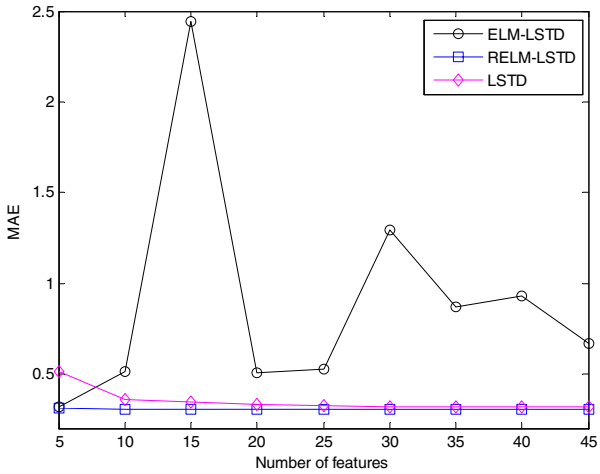

Fig 3. MAE error of RELM-LSTD(λ), LSTD and ELM-LSTDon the Hop-World problem, for different values of feature number. The number of features in the x-axis refers to dimension of basis function for RBF and hidden nodes number for RELM.

with ELM-LSTD and the original LSTD. The number of features in the x-axis refers to dimension of RBF in LSTD and hidden nodes number of ELM or RELM. The results

show that LSTD depends on the number of features. The MAE of LSTD is decreasing with the growing number of features. The MAE curve of ELM-LSTD fluctuates significantly over different numbers of feature. This is due to the random initialization of parameters. When number of features is five, ELM-LSTD achieves equal approximation accuracy as RELM-LSTD(λ) does. With increasing number of features, ELM-LSTD performs worse than the other two methods. Besides, RELM-LSTD(λ) performs steadily over different number of features. This result demonstrates that the proposed method can approximate value functions with few features, and the introduction of regularization in ELM can overcome the fluctuating performance of the original ELM.

4.2 Continuous-state random-walk task

In experiment 2, a more difficult random-walk task with continuous state space [32] is studied to compare the generalization abilities of LSTD, ELM-LSTD and RELM-LSTD(λ). In this task, the state space is over the continuous interval [0, 1]. Two boundaries states of the chain are the absorbing states. Each episode starts from state 0.5 with random step [-0.2, 0.2] and ends when going beyond either boundary. The reward function is as follows: r equals 0 when the transition between two non-absorbing states occurs, and equals the value of terminal position when the transition from a non-absorbing state to an absorbing state occurs. The true value function for state $i(i \in [0,1])$ is $V(i) = i$. 50 points uniformly distributed in [0, 1] are used to compute the prediction errors. Other settings are the same as the first experiment.

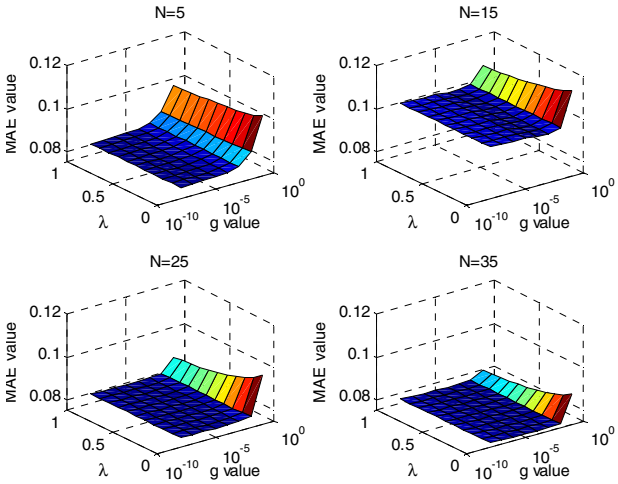

Fig 4. MAE error of RELM-LSTD(λ) on the continuous-state random-walk task, for different values of λ and g. Each subpanel shows the MAE error at different number of features. The number of features in the subtitle refers to dimension of basis function for RBF and hidden nodes number for RELM.

Fig.4 and Fig.5 show the results of this experiment. In Fig.4, the performances under different values of g, λ and N are shown. In Fig.5, the MAEs for RELM- LSTD (λ) (at the best g value and λ value) with ELM-LSTD and the original LSTD is compared. The results show that different g value (in Fig.4) has strongly effect on performance of RELM-LSTD(λ). RELM-LSTD(λ) achieves the best performance at about $g=10^{-4}$. Results also show that different λ value has slightly effect on RELM-LSTD(λ)'s performance. Clearly, all of the results in the second experiment have the identical trends with those in the first experiment.

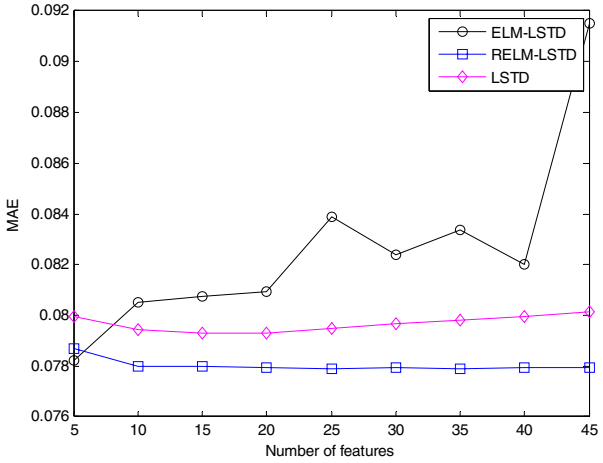

Fig 5. MAE error of RELM-LSTD(λ), LSTD and ELM-LSTD on the continuous-state random-walk task, for different values of feature number. The number of features in the x-axis refers to dimension of basis function for RBF and hidden nodes number for RELM.

5 CONCLUSIONS AND FUTURE WORKS

In this paper, LSTD(λ) based on RELM was proposed to approach learning prediction problem with continue state space. The proposed algorithm combined the LSTD(λ) algorithm with RELM. Although ELM has fast learning speed and avoid tuning issues of traditional ANN, the randomness of the ELM parameters would result in fluctuating performance. The proposed method overcomes these problems caused by ELM in RL problem. First of all, regularized ELM was introduced to achieve stable performance. Then, aiming at further increasing the usage of data, RELM-LSTD was extended to RELM-LSTD(λ). At last, the capability of the RELM-LSTD(λ) to solve the learning prediction problem were verified through examples. In the experiments, the comparisons of the proposed algorithm with LSTD and ELM-LSTD show that the RELM-LSTD(λ) can achieve a more stable and better performance in approximating the value function of a fixed policy.

Although RELM-LSTD(λ) performed better than the standard LSTD and ELM-LSTD, it still has several issues to be addressed in the future. Firstly, the matrix inversion is needed in this algorithm, which rises a computational complexity $O(n^3)$. A lower computational complexity algorithm is needed. Secondly, it's also interesting to study other regularized methods like l^1 regularization in ELM. Thirdly, RELM-LSTD(λ) has two more tuning parameters than ELM-LSTD, which will result in more difficult tuning

work. Automatic regularization method will be a reasonable solution to this problem.

REFERENCES

[1] X. Xu, L. Zuo, Z. Huang, Reinforcement learning algorithms with function approximation: recent advances and applications, Information Sciences, 261(2014) 1-31.

[2] N. Rezzoug, P. Gorce, A reinforcement learning based neural network architecture for obstacle avoidance in multi-fingered grasp synthesis, Neurocomputing 72(2009) 1229-1241.

[3] F.L. Lewis, D. Vrabie, Reinforcement learning and adaptive dynamic programming for feedback control, IEEE Circuits Syst. Mag. 9 (3) (2009) 32–50.

[4] L. A. Prashanth, S. Bhatnagar, Reinforcement learning with function approximation for traffic signal control, IEEE Trans. Intelligent Transportation Systems, 12(2011) 412-421.

[5] P.J. Werbos, Intelligence in the brain: a theory of how it works and how to build it, Neural Networks (2009) 200–212.

[6] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, The MIT Press, Cambridge, MA, USA, 1998.

[7] Cs. Szepesvári, Algorithms for Reinforcement Learning, Morgan and Claypool, 2010.

[8] H.Yu, D.P.Bertsekas, New error bounds for approximations from projected linear equations. Technical Report C-2008-43, Dept. Computer Science, Univ. of Helsinki, July 2008.

[9] H. R. Maei, R. S. Sutton, GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces, in: Proceedings of the Third Conference on Artificial General Intelligence, 2010, pp. 91-96.

[10] H.Yu, Convergence of least-squares temporal difference methods under general conditions. In International Conference on Machine Learning (*ICML*), 2010.

[11] T. Song, D. Li, L. Cao, K. Hirasawa, Kernel-based least squares temporal difference with gradient correction. IEEE Trans. Neural Networks Learn. Sys., In Press.

[12] M. Geist, B. Scherrer, Off-policy learning with eligibility traces: A survey. The Journal of Machine Learning Research, 15(Jan):289–333, 2014.

[13] S.O. Haykin, Neural Networks and Learning Machines, Prentice Hall, Upper Saddle River, NJ, USA, 2008.

[14] C.M. Bishop, Neural Networks for Pattern Recognition, 1st ed., Oxford University Press, USA, 1995.

[15] Pablo Escandell-MonteroJosé, M.Martínez-Martínez, José D. Martín-Guerrero, Emilio Soria-Olivas, Juan Gómez-Sanchis. Least-squares temporal difference learning based on an extreme learning machine, Neurocomputing 141 (2014) 37–45.

[16] G. Huang, Q. Zhu, C. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.

[17] J. Pan, X.-S. Wang, Y.-H. Cheng, G. Cao, Multi-source transfer ELM-based Q learning, Neurocomputing 137 (2014) 57–64.

[18] J. M. Lopez-Guede, B. Fernandez-Gauna, J. A. Ramos-Hernanz, A L-MCRS dynamics approximation by ELM for Reinforcement Learning, Neurocomputing 150(Part A) (2015) 116–123.

[19] Y. Lan, Y. Soh, G.-B. Huang, Ensemble of online sequential extreme learning machine, Neurocomputing 72 (2009) 3391–3395.

[20] Z. Sun, T. Choi, K. Au, Y. Yu, Sales forecasting using extreme learning machine with applications in fashion retailing, Decis. Support Syst. 46 (2008) 411–419.

[21] G. Zhao, Z. Shen, C. Miao, Z. Man, On Improving the Conditioning of Extreme Learning Machine: A Linear Case.

[22] S. Suresh, S. Saraswathi, N. Sundararajan, Performance enhancement of extreme learning machine for multi-category sparse data classification problems, Eng. Appl. Artif. Intell. 23 (2010) 1149–1157.

[23] Z.-F Shao, M. J. Er, N. Wang. An effective semi-cross-validation model selection method for extreme learning machine with ridge regression, Neurocomputing 151 (2015) 933–942.

[24] S.J. Bradtke, A.G. Barto, Linear least-squares algorithms for temporal difference learning, Mach. Learn. 22 (1996) 33–57.

[25] D. Ernst, P. Geurts, L. Wehenkel. Tree-based batch mode reinforcement learning, Journal of Machine Learning Research. 6(2005) 503-556.

[26] X. Xu, D. Hu, X. Lu, Kernel-based least squares policy iteration for reinforcement learning, IEEE Trans. Neural Netw. 18 (2007) 973–992.

[27] Y. Engel, S. Mannor, R. Meir, Reinforcement learning with Gaussian processes. In De Raedt and Wrobel. 2005, pp. 201-208.

[28] J.Z. Kolter, A.Y. Ng, Regularization and feature selection in least-squares temporal difference learning, in: Proceedings of the Twenty-Sixth Annual International Conference on Machine Learning, ICML '09, ACM, New York, NY, USA, 2009, pp. 521–528.

[29] R.S. Sutton, H.R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvri, E. Wiewiora, Fast gradient-descent methods for temporal-difference learning with linear function approximation, in: Proceedings of the Twenty-Sixth Annual International Conference on Machine Learning, ACM, Montreal, Quebec, Canada, 2009, pp. 993–1000.

[30] J.A. Boyan, Technical update: least-squares temporal difference learning, Mach. Learn. 49 (2002) 233–246.

[31] X. Xu, H. He, D. Hu, Efficient reinforcement learning using recursive least squares methods, J. Artif. Intell. Res. (JAIR) 16 (2002) 259–292.

[32] X. Xu, Reinforcement Learning and Approximate Dynamic Programming, Science Press, Beijing, 2010.