Target Following for an Autonomous Underwater Vehicle Using Regularized ELM-based Reinforcement Learning

Tingting Sun, Bo He*, Rui Nian
School of Information Science and Engineering
Ocean University of China, OUC
Qingdao, China
tts_612@163.com, bhe@ouc.edu.cn,
nianrui_80@163.com

Tianhong Yan
School of Mechatronic Engineering
China Jiliang University
Hangzhou, China
yanth@163.com

***Abstract*—In order to achieve autonomy and intelligence of autonomous underwater vehicle (AUV), dynamic target following in the unknown environment is one of the important problems to solve. Reinforcement learning (RL) offers the possibility of learning a policy to solve a particular task without manual intervention and previous experience. However, RL algorithms are not competent for continuous space problems existing universally in underwater environment. In this paper, the method of an AUV using regularized extreme learning machine (RELM) based RL is proposed for following the moving target. The most typical and common method of RL used in applications is Q learning, which can generate the proper policy for AUV control. And RELM is proposed as a modification of ELM, which can guarantee the generalization performance and work with continuous states and actions. The simulation results have shown that AUV successfully tracks and follows the moving target by the proposed method.**

Keywords—reinforcement learning; extreme learning machine; autonomous underwater vehicle; target following; machine learning

I. INTRODUCTION

Autonomous underwater vehicles (AUVs) play an important role in oceanic natural resources exploration and marine biology research [1]. The control of an AUV is still a challenge in the complicated and unknown environments. Reinforcement learning (RL) is a kind of Machine Learning algorithm with great intelligence and wide applications in the field of control and optimization [4], [5]. RL offers the possibility of learning an optimal policy, mapping the state of environment to an action, which can solve a particular task without manual intervention and previous experience. The essence of the learning process is the interaction between an agent and the environment. The RL algorithms do not need the datasets of input and output samples, but a reinforcement function determined by an operator is the only required information.

The Q learning algorithm [6], [7] is one of the typical RL algorithms that uses state-action mapping to find an optimal policy. However, when confronted with real-world problems, Q learning seems to be incompetent for the tasks in large-scale or continuous spaces. That is, Q learning should have the ability of generalization for AUV to solve control tasks. In order to figure out the generalization problem, a lot of generalization methods have been applied to Q learning. Takahashi et al. proposed a memory based implementation for vision-guided behavior acquisition [8] and Takeda et al. showed state-action quantification by a set of triangular patches [9]. Also, many proposals combined Q learning and neural networks (NN) [10] and the advantage of using NN enabled the RL to search for optimal policies more efficiently in several real-life applications [11], [12].

One of the typical NNs is back propagation (BP) [13], which combines the forward propagation of information with back propagation of error. Besides, support vector machine (SVM) [14] based on statistical learning also has the ability of nonlinear mapping. However, both BP and SVM cannot overcome the problems, such as slow training speed, trivial human intervention. Fortunately, there exists a novel learning algorithm that effectively solves the mentioned problems. That is extreme learning machine (ELM) proposed by Huang et al., which has been proved to have good generalization performance at a faster learning speed [15], [16]. Therefore, regularized extreme learning machine based Q Learning (RELMQL) is proposed in this paper. ELM has been proved to achieve with simple structure. But, it may be stuck in the problems of over-fitting and the empirical risk minimization. Therefore, the modified ELM with $L_{1/2}$ regularization [17] is proposed to overcome these problems. And RELM as a function approximator can replace the look-up table method of the original Q learning to store the state-action mapping and save storage space. In this paper, we focus on the behavior of target following [18], whose goal is to generate the control action, according to the position of the target, and make AUV follow the target. The realization process of target following is to minimize the relative distance and heading tangential error between AUV and the target [19].

The rest paper is organized as follows. In section II, we briefly describe previous work on the preliminary ELM. In section III, we describe the proposed method regularized

978-0-933957-43-5 ©2015 MTS

Extreme Learning Machine based Q Learning. In section IV, we report simulation results on moving target following by our algorithm. In section V, conclusions are summarized on our recent research and future work.

II. THE PRELIMINARY ELM

ELM [15] is designed based on the generalized signal hidden layer feedforward networks (SLFNs) [20], which have been proved to achieve good generalization performance at fast learning speed. The weights and bias between the input and hidden nodes can be generated randomly, and then the output weights can be calculated by solving a linear system analytically without iteration. Thus, the preliminary ELM can be implemented easily with few steps.

Given N distinct pairs of training samples $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$, where $\boldsymbol{x}_i = [x_{i1}, \cdots, x_{in}]^T \in \mathbb{R}^n$ is the input data and $\boldsymbol{y}_i = [y_{i1}, \cdots, y_{im}]^T \in \mathbb{R}^m$ is the output target. The standard model of ELM with L hidden neurons and activation function $g(\boldsymbol{w}_j, b_j, \boldsymbol{x}_i)$ can be mathematically expressed as

$$f_L(\boldsymbol{x}_i) = \sum_{j=1}^{L} \boldsymbol{\beta}_j g(\boldsymbol{w}_j, b_j, \boldsymbol{x}_i) = \boldsymbol{o}_i, \quad i = 1, 2, \cdots, N \tag{1}$$

Where the weight vector $\boldsymbol{w}_j = [w_{j1}, \cdots, w_{jn}]^T \in \mathbb{R}^n$ connects the j_th hidden nodes and the input nodes, and bias b_j is of the j_th hidden nodes. The weight vector $\boldsymbol{\beta}_j = [\beta_{j1}, \cdots, \beta_{jm}]^T \in \mathbb{R}^m$ connects the j_th hidden nodes and the output nodes.

ELM can approximate these N samples with zero error means that

$$\sum_{i=1}^{N} \|\boldsymbol{o}_i - \boldsymbol{y}_i\| = 0 \tag{2}$$

Then, equation (1) can be modified as

$$\sum_{j=1}^{L} \boldsymbol{\beta}_j g(\boldsymbol{w}_j, b_j, \boldsymbol{x}_i) = \boldsymbol{y}_i, \quad i = 1, 2, \cdots, N \tag{3}$$

The above N equations can be written compactly as

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{Y} \tag{4}$$

Where $\mathbf{H}$ is called the hidden layer output matrix of the neural network

$$\mathbf{H} = \begin{bmatrix} g(\boldsymbol{w}_1, b_1, \boldsymbol{x}_1) & \cdots & g(\boldsymbol{w}_L, b_L, \boldsymbol{x}_1) \\ & \vdots & \\ g(\boldsymbol{w}_1, b_1, \boldsymbol{x}_N) & \cdots & g(\boldsymbol{w}_L, b_L, \boldsymbol{x}_N) \end{bmatrix}_{N \times L} \tag{5}$$

$$\boldsymbol{\beta} = [\beta_1^T, \cdots, \beta_L^T]_{L \times m}^T \tag{6}$$

$$\mathbf{Y} = [\boldsymbol{y}_1^T, \cdots, \boldsymbol{y}_N^T]_{N \times m}^T \tag{7}$$

$\mathbf{H}$ can be obtained. And then the output weights $\boldsymbol{\beta}$ can be generated as

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{Y} \tag{8}$$

Where $\mathbf{H}^{\dagger}$ denotes the Moore-Penrose generalized inverse of matrix $\mathbf{H}$.

III. THR REGULARIZED ELM BASED Q LEARNING

A. The Regularized ELM

In order to overcome the empirical risk minimization principle of the preliminary ELM and prevent over-fitting in the learning procedure, we propose a structure named regularized extreme learning machine (RELM), which can guarantee the sparse solutions with less computational cost.

The important prerequisite of RELM is that assuming the errors between the output of the network and the target are not zero. We address a problem of a minimum squared-error with sparse solution and then define the error δ as the following equation according to (2).

$$\delta = \sum_{i=1}^{N} \|\boldsymbol{o}_i - \boldsymbol{y}_i\| \tag{9}$$

It can also be expressed as

$$\delta = \|\mathbf{Y} - \mathbf{H}\boldsymbol{\beta}\| \tag{10}$$

The general form of the regularization methods can be modeled by

$$\min\left\{\frac{1}{n}\sum_{i=1}^{n} l(y_i, f(x_i)) + \lambda \|f\|_k\right\} \tag{11}$$

Where $l(\cdot)$ represents a loss function, $\{x_i, y_i\}_{i=1}^{N}$ is a dataset and $f(x_i)$ is the function with regard to x_i. λ is the regularization parameter, and $\|\cdot\|_k$ denotes *k-norm*. The equation is the general form of L_k regularizer. The former research has proven that the solutions of L_2 and L_∞ regularizers do not possess the ability of sparsity. In order to maintain approximate calculating time compared with the preliminary ELM, the regularization methods with sparse solutions should be taken into consideration. To our knowledge, both L_0 and L_1 regularizers are competent to obtain the sparse solutions. However, it is difficult to solve NP-hard problem for L_0 regularizer and L_1 regularizer cannot generate the sparsest solutions as L_0 regularizer does. Fortunately, Xu et al. [17] proposed $L_{1/2}$ regularizer and proved that $L_{1/2}$ regularization can produce sparser solutions compared with L_1 regularization. Therefore, we adopt $L_{1/2}$ regularizer in the regularization method, and calculate the minimum parameter $\boldsymbol{\beta}$ in (10) as follows.

$$\boldsymbol{\beta} = \arg\min_{\beta}\left\{\left\|\mathbf{Y}-\mathbf{H}\boldsymbol{\beta}\right\|_2^2 + \lambda\left\|\boldsymbol{\beta}\right\|_{1/2}\right\} \tag{12}$$

Note that $\left\|\mathbf{Y}-\mathbf{H}\boldsymbol{\beta}\right\|_2^2$ is the loss function and there is no need to calculate the Moore-Penrose generalized inverse or SVD as the preliminary ELM does.

Although the solutions of $L_{1/2}$ regularization are proved to be much sparser than those of L_1 regularization, the solving process of $L_{1/2}$ regularization seems to be more difficult. In order to solve the L_1 regularization problem, we adopt the *l1_ls* toolbox proposed by Koh [21]. Then an iterative method is proposed to solve the $L_{1/2}$ regularization problem. The main principle of the method is to transform $L_{1/2}$ regularization into a series of weighted of L_1 regularization, which is still solved by the existing method.

Algorithm 1 The RELMQL algorithm

Input The dataset $\{x_i, Q_i\}_{i=1}^{N}$, initial hidden nodes number L, the activation function $g(\cdot)$, the parameter λ.

Output Select the action a_{t+1} corresponding to the maximum Q value for each state.

***Step* 1.** Execute the Q learning algorithm, from which the state-action mapping can be derived. AUV can receive the current state s_{t+1} from the environment. The reward r_t can be calculated by the reinforcement function, and the last state s_t is obtained from a delay unit.

***Step* 2.** In the RELM network, randomly generate the weight $\boldsymbol{w}_j$ and bias b_j between the input and hidden nodes. And take states and actions as input and the Q values as output.

***Step* 3.** Calculate the hidden layer output matrix $\mathbf{H}$. And then calculate the coefficients $\boldsymbol{\beta}$ of the $L_{1/2}$ regularized RELM network by (12).

***Step* 4.** The new action a_{t+1} can be calculated by maximizing the Q values in the current state s_{t+1}.

***Step* 5.** Update the dataset by removing the old sample that similar to the new one. And then update the weights of RELM according to the new samples.

B. The Structure of RELMQL

The Q learning algorithm is a temporal difference (TD) algorithm that uses the state-action mapping to find an optimal policy when the environment is dynamic and unknown. Q learning algorithm can be fulfilled as follows. In each iteration t, AUV observes the current state s_t, receives the reward r_t and generates an action a_t. And then the environment reacts to the action and changes to next state s_{t+1} and receives a new reward r_{t+1}. Q value can be calculated according to the following expression.

$$Q(s_t,a_t) = (1-\alpha)Q(s_t,a_t) + \alpha\left[r_t + \gamma\max_{a_{t+1}} Q(s_{t+1},a_{t+1})\right] \tag{13}$$

Where $0<\alpha<1$ is a learning rate, $0<\gamma<1$ is the discount factor.

Then, the key to improving the generalization performance is to design a function approximator. Therefore, we combine RELM with Q learning, and the RELM network can realize the mapping from a state-action pair to the Q value. In the concrete implementation process, the states and actions are taken as input and Q values are taken as output. And after training the RELM network, Q value can be estimated when given the state and action.

Given a sequence of states $s_i = [s_{i1}, s_{i2}, \cdots, s_{im}]^T \in \mathbb{R}^m$ and action $a_i \in \mathbb{R}$. Then the input of the RELM is a state-action pair, represented by $x_i = (s_i, a_i)^T = [s_{i1}, s_{i2}, \cdots, s_{im}, a_i]^T \in \mathbb{R}^{m+1}$. And the output of ELM is the Q value $Q(s_i, a_i)$, which has been obtained by the Q learning. The concrete process of RELMQL can be described as Algorithm 1.

Note that in the step 4, before the new action is selected, a ε-*greedy* policy is applied to the target following process. The ε-*greedy* policy selects random actions with ε probability and greedy actions with $(1-\varepsilon)$ probability. And the structure of RELMQL algorithm is presented in Fig. 1.

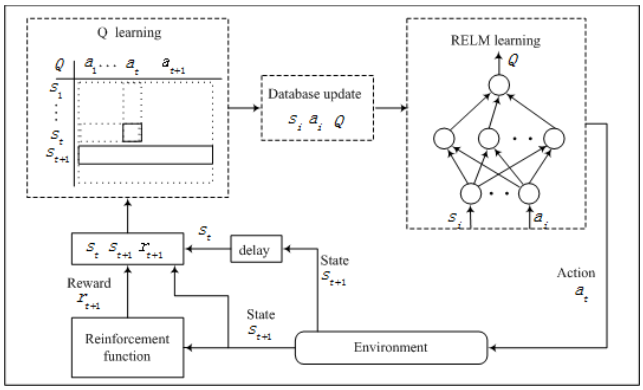


Fig. 1. Structure of RELMQL.

IV. Simulation Results

An exhaustive set of simulation experiments were implemented in order to demonstrate the performance of RELMQL method.

A. Experimental setup

All the simulations for the algorithms were carried out in Matlab R2010b environment running in an Intel (R) Core (TM) i5-3470 3.20GHz CPU. In the experiments, the parameters were designed as the TABLE I shows. Among the parameters design, we focused on the setting of the reinforcement function, which related the state spaces to a reward value. As Fig. 2 described, we replaced the piecewise function (left) by

Gaussian function (right). For Gaussian function, when the states lie in the midpoint of ‘b’ and ‘c’, the reward reaches the maximum, and then the value gradually decreases between ‘a’ and ‘b’ or between ‘c’ and ‘d’. However, for piecewise function, the reward value stays constant in every interval of state. Therefore, Gaussian function can precisely reflect the real situation of the environment, from which the AUV can observe the states, and generate an action by our method.

TABLE I. PARAMETERS SETTING

algorithms	*Parameter*	*Symbol*	*Value*
Q learning	Learning rate	α	0.1
	Discount factor	γ	0.9
	Exploration probability	ε	0.3
	Density	t	0.5
ELM	Activation function	g	‘sig’
	Hidden nodes	n	1000

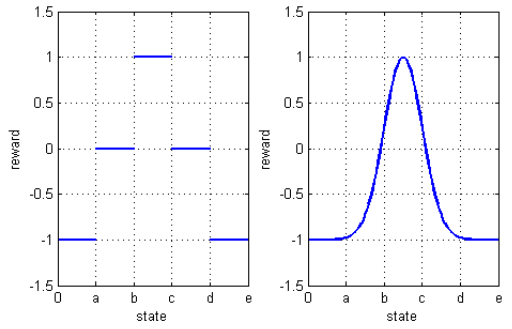


Fig. 2. Reinforcement functions comparison.

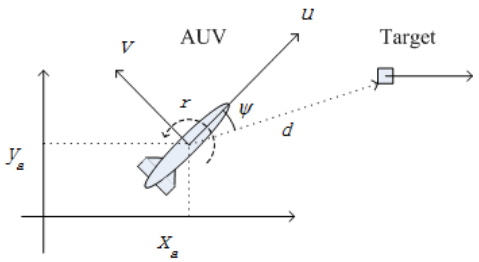
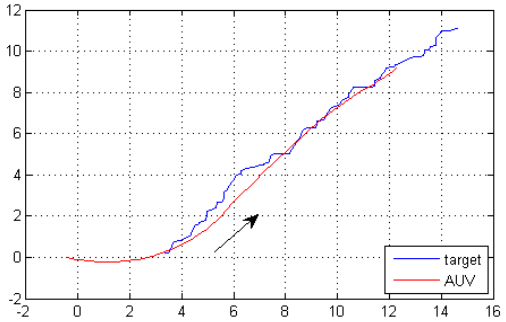
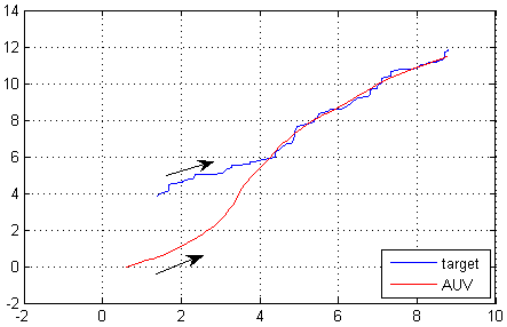


Fig. 3. A conceptual graph of target following.

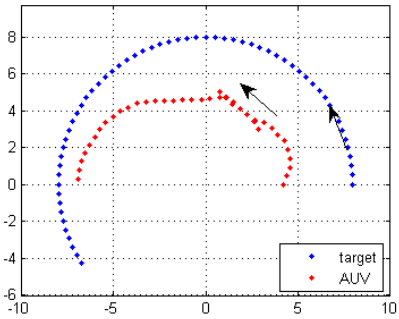
In order to implement the task of target following using AUV, three stages can be adopted. First, the position can be detected and calculated from the camera and side-scan sonar. Second, the desired trajectory and the surge and yaw velocities are determined based on the target position. Last, a nonlinear dynamic controller can be generated by our algorithm. Two RELMQL algorithms were used for two different states, the



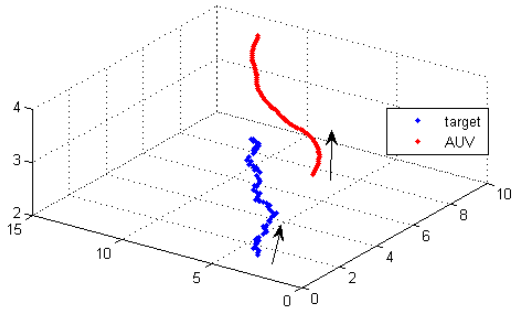
(a)



(b)



(c)



(d)

Fig. 4. Experimental results of the proposed method for target following. (a), (b) and (c) show two-dimensional representation for following the moving target, and (d) shows three-dimensional representation.

relative distance d and the relative rotation angle ψ between the AUV and target, respectively. More detailed information has been presented in Fig. 3, where u is the surge velocity, v is the sway velocity, and r is the yaw angular velocity. $[x_a, y_a]$ donates the position of the center of mass of the AUV.

B. Simulation results

Some of the simulation results have been presented in Fig. 4. At the beginning, there is no valid information for the Q function, so our algorithm behaves exploration actions. As more and more information about states and actions obtained by the algorithm, exploitation actions are more and more adopted by greedy actions strategy. In Fig. 4 (a) (b) and (c), it can be seen how the AUV followed the moving target at a different starting point in the two-dimension space. The target was moving toward to a rough direction at a random speed. The moving trajectory of AUV was close to a straight line in Fig. 4(a) and (b), and it was nearly circle in Fig. 4 (c), as the target does. Actually, when forming the figures, it can be clearly observed that the target moved a step, and then AUV interacted with it by generating an action. And Fig. 4 (d) was the three-dimension representation of the target following task, which showed the target could be followed at a certain distance.

V. CONCLUSIONS

This paper has presented the use of regularized ELM-based reinforcement learning for learning a reactive AUV behavior (target following). It is an important contribution to demonstrate the feasibility of RL methods in a real-world task, especially in the complex and unknown environment using AUV. Also, it can achieve autonomy and intelligence of AUV more easily than the popular control methods. As future work, the learning algorithm will be improved to work in more challenging tasks and tested in real scenarios instead of simulations.

ACKNOWLEDGMENT

This paper is partially supported by the Natural Science Foundation of China (41176076, 51075377, and 51379198), the High Technology Research and Development Program of China (2006AA09Z231, 2014AA093410).

REFERENCES

[1] C. Alt, "Autonomous underwater vehicles," Autonomous Underwater Lagrangian Platforms and Sensors Workshop, Vol. 3, Mar. 2003.

[2] E. Bovio, D. Cecchi, D. Baralli, "Autonomous underwater vehicles for scientific and naval operations, " Annual Reviews in Control, vol. 30, pp. 117-130, 2006.

[3] R. B. Wynn, V. Huvenne, T. P. Le Bas, et al. "Autonomous Underwater Vehicles (AUVs): Their past, present and future contributions to the advancement of marine geoscience," Marine Geology, vol. 352, pp. 451-468, 2014.

[4] R. S. Sutton, A. G. Barto, "Reinforcement learning: An introduction," Cambridge: MIT press, 1998.

[5] J. Kober, and J. Peters, "Reinforcement learning in robotics: A survey," Reinforcement Learning. Springer Berlin Heidelberg, vol. 12, pp. 579-610, 2012.

[6] P. Dayan, and C. Watkins, "Q-learning," Machine Learning, pp. 279-292, 1992.

[7] E. EM. Moodie, N. Dean, and Y. R. Sun, "Q-learning: flexible learning about useful utilities," Statistics in Biosciences, pp. 223-243, 2014.

[8] Y. Takahashi, M. Takeda, and M. Asada, "Continuous valued Q-learning for vision-guided behavior acquisition," Multisensor Fusion and Integration for Intelligent Systems, 1999. MFI'99. Proceedings. 1999 IEEE/SICE/RSJ International Conference on. IEEE, pp. 255-260, 1999.

[9] M. Takeda, T. Nakamura and T. Ogasawara, "Continuous valued Q-learning method able to incrementally refine state space," Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on. vol. 1, pp.265-271, 2001.

[10] D. F. Specht, "A general regression neural network, "Neural Networks, IEEE Transactions on, vol.2, pp.568-576, 1991.

[11] M. Carreras, J. Yuh, J. Batlle, P. Ridao, "Application of SONQL for real-time learning of robot behaviors," Robotics and Autonomous Systems, vol. 55, pp. 628-642, 2007.

[12] A. Ghanbari, Y. Vaghei, S. Noorani, "Reinforcement learning in neural networks: a survey," International journal of Advanced Biological and Biomedical Research, vol. 2, pp. 1398-1416, 2014.

[13] Y. Chauvin, and D. E. Rumelhart, "Backpropagation: theory, architectures, and applications," Psychology Press, 1995.

[14] J. A. Suykens, J. Vandewalle, "Least squares support vector machine classifiers," Neural processing letters, vol. 9, pp. 293-300, 1999.

[15] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," Neurocomputing, vol.70, pp. 489-501, 2006.

[16] G. B. Huang, H. Zhou, X. Dong, "Extreme learning machine for regression and multiclass classification," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 42, pp. 513-529, 2012.

[17] Z. Xu, X. Chang, H. Zhang, "L1/2 regularization: a thresholding representation theory and a fast solver," IEEE Transactions on neural networks and learning systems, vol. 23, pp. 1013-1027, 2012.

[18] Y. Hu, W. Zhao, L. Wang, Y. Jia, "Underwater target following with a vision-based autonomous robotic fish," American Control Conference, IEEE, 2009.

[19] P. Švec, A. Thakur, E. Raboin, B. C. Shan, S. K. Grupta, "Target following with motion prediction for unmanned surface vehicle operating in cluttered environments," Autonomous Robots, vol. 36, pp. 383-405, 2014.

[20] G. B. Huang, H. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," Neural Networks, IEEE Transactions on, vol. 9, pp. 224-229, 1998.

[21] K. Koh, S. J. Kim, and S. Boyd, l1_ls: Simple Matlab Solver for L1-regularized Least Squares Problems. Available: http://stanford.edu/~boyd/l1_ls/