International Journal of Control, Automation and Systems 15(4) (2017) 1790-1798
http://dx.doi.org/10.1007/s12555-015-0392-5
ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

Cyclic Error Correction based Q-learning for Mobile Robots Navigation

Rongkuan Tang and Hongliang Yuan*

Abstract: Similar to control systems, reinforcement learning can capture notions of optimal behavior using natural interaction experience. In the context of reinforcement learning, the temporal difference error of the generated experience measures how well the learner responds to the system. Specially sequential difference of accumulated temporal difference error can indicate the learning performance. In this paper, we fully utilize the error correction in closed-loop peculiarity by mapping a representation error to the step-size component. The proposed cyclic step-size could better control how new estimates are iteratively blended together over time, and the new estimates guide the action selection process which in turn influence the value distribution. To guide more promising action decision, an ensemble action selector is proposed which incorporates the idea of ensemble wisdom of the weak. Experimental results conducted under gridworld mobile robot navigation task demonstrate the validity, capacity of fast learning and easy-plugged implementation of the derived algorithm, leading to increasing applicability to real-life problems.

Keywords: Cyclic step-size, ensemble wisdom, mobile robots navigation, reinforcement learning.

1. INTRODUCTION

Reinforcement learning (RL) provides a sound framework for autonomous agents to acquire adaptive behaviors based on delayed reward feedback. And it has been explored extensively in area of machine learning research [1], the temporal difference algorithm proposed by Sutton [2] and the Q-learning of Watkins [3] are the two popular learning methodologies. Partly driven by the increasing availability of rich data, recent years have seen exciting advances in the theory and practice of reinforcement learning, including developments in fundamental technical areas such as control [4], robotics [5–7], exploration and empirical methodology, leading to increasing applicability to real-life problems.

In the soil of reinforcement learning, meta critical components such as step-size that controls how new estimates are iteratively blended together over time, and action selection which evaluates the tradeoff quality between exploration and exploitation [8] should always be carefully dealt with by the designer to tackle diverse tasks, e.g., mobile robot navigation. Nevertheless that's a considerable work tagged time consuming and efficiency cost. Concerned with step-size [9–12], which undergone three stages of evolution. Classically the constant step-size $\alpha \in [0,1]$ is concise but lack of flexibility. It couldn't grasp the data trend with inefficient coordinate between variance and bias. Empirically larger value at early learning stage and smaller at the latter could in some sense slack the flexibility of parameter tuning while assuring convergence. Without loss of generality, such mechanism resembles formula: $\alpha_t = C(\beta,t)/D(\gamma,t)$. Generally $D(\gamma,t)$ is linear function combination of the learning episode index. Noticeably this is error-independent leading to open loop between entire learning cycle particularly the pipeline among Q-value and td-error. Therefore resorting to sampling method could be a straightforward way to bridge the gap. And sampling directly from neither Q-value nor td-error could deteriorate the learning performance due to correlated influence among them.

Additionally when getting down to implement the Q-learning algorithm given specific task, those simple but rather not so much efficient action selection [13] strategies favored extensively emerge in mind. Pure exploitation causes the agent to reach the locally optimal policies quickly, whereas excessive exploration degrades the performance of the Q-learning algorithm even if it may accelerate the learning process and allow avoiding the locally optimal policies. Specifically under popularity, ε-*greedy* [14] selector injects a probability $\varepsilon \in (0,1)$ that generates random behavior so as to encourage more exploration. However, the tricky parameter ε is empirically fixed leading to complex tuning, what's more, as learning evolves to mature level, the random behavior should be

Manuscript received October 28, 2015; revised August 27, 2016; accepted October 1, 2016. Recommended by Associate Editor Huaping Liu under the direction of Editor Fuchun Sun. This work is supported by the National Natural Science Foundation of China under Grant No. 61273327.

Rongkuan Tang is with the College of Electronics and Information, Tongji University, Shanghai 201804, China (e-mail: stone8oy@gmail.com). Hongliang Yuan is with the College of Electronics and Information, Tongji University, Shanghai 201804, China (e-mail: hyuan@tongji.edu.cn).
* Corresponding author.

be pruned accordingly to exploit gained experience. To slack such 'greedy' effect, randomized methods like boltzmann distribution were adopted in more recent works. Boltzmann method samples from $Q(s,a)$ with softmax criteria: $a^* = argmax_{a' \in A} \exp Q(s,a') / \sum_{a' \in A} \exp Q(s,a')$. Compared with ε-*greedy*, the softmax [14] method made a second sampling over Q-value space. Nevertheless, there's a common leak here: with always optimal action a^* that satisfy the max criteria, this procedure would degenerate to the straightforward max-Q version : $a^* = argmax_{a' \in A} Q(s,a')$. As a result, there's no substantial discrepancy here. A well integrated selector was introduced in [15] called simulated-annealing based on Metropolis criterion. The Metropolis criterion derives from the importance sampling which originated from MCMC sampling method and has an effective approximation. This selector simulates an action transition probability : $P_{p \to r} = \exp((Q(s,a_r) - Q(s,a_p))/T)$ and a random number $\mu \in [0,1]$ sampled from uniform distribution where action a_r was accepted if $\mu < P_{p \to r}$. Unexpectedly, low value term $P_{p \to r}$ would result in amounts of invalid action transitions in turn.

In this work, a distinct derived learning method termed as natural Q-learning aka NaQL that seamlessly blends together learning speed and generalized action decision module made by various 'weak experts' is proposed to boost the comprehensive learning performance. The remainder of the paper is organized as follows. Section 2 covers the preliminaries of the elements of reinforcement learning. In Section 3, the original flesh and blood of the NaQL algorithm is presented including the cyclic step-size and ensemble action selector. Subsequently the learning performance is manifested through considerable and comparable experiments and ensuing results are statistically visualized. Conclusions and insights of future research step will follow in Section 5 finally.

2. PRELIMINARIES

Every living organism interacts with its environment and uses those interactions to improve its own actions in order to survive and increase. Reinforcement learning refers to an actor or agent that interacts with its environment and modifies its actions or control policies, based on stimuli received in response to its actions. This is based on evaluative information from the environment and could be called action-based learning. RL implies a cause and effect relationship between actions and reward or punishment [16]. It implies goal directed behavior at least insofar as the agent has an understanding of reward versus lack of reward or punishment.

2.1. Markov decision process

Reinforcement learning is characterized as an interaction between a learner and an environment that provides

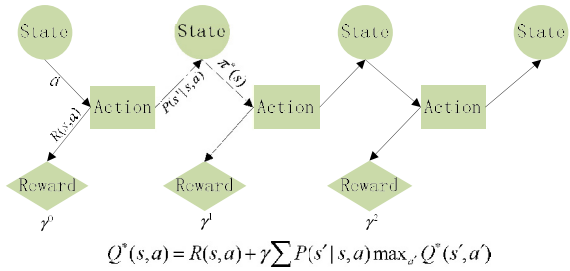


Fig. 1. Bellman equation illustration.

evaluative feedback. The environment is often conceptualized as a Markov decision process [17] - a formal mathematical model that dates back to the 1950s. A Markov decision process is defined by a set of states S(situations in which a decision can be made), and actions A(the decisions or interventions the decision maker can select). These quantities can be taken to be finite, but continuous state and action spaces are often valuable for capturing interactions in important reinforcement learning applications such as robotic control. A transition function $P(s'|s,a)$ defines the probability of the state changing from s to s' under the influence of action a. It specifies the 'physics' or 'dynamics' of the environment.

The decision maker's task is defined by a reward function $R(s,a)$ and discount factor $\gamma \in [0,1]$. Rewards are delivered to the decision maker with each transition and maximizing the cumulative discounted expected reward is its objective. Specifically, the decision maker seeks a behaviour π^* mapping states to actions generating a sequence of rewards r_0,r_1,r_2,...such that $E_{r_0,r_1,}...[r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + ...]$ as large as possible. The relation between the environmental interaction (state, action, reward, state, action, reward,...) and the cumulative discounted expected reward is captured by the Bellman equation (Fig. 1) for the optimal state-action value function Q^*. The solution to the Bellman equation can be used to define optimal behaviour by $\pi^*(s) = argmax_a Q^*(s,a)$. The cumulative discounted expected reward for the policy that takes action a from state s and then behaving optimally thereafter is the immediate reward received plus the discounted expected value of the cumulative discounted expected reward from the resulting state s' given that the best action is chosen [18].

Planning methods use knowledge of P and R to compute a good policy π. Reinforcement learning methods have access to P and R only through their ongoing interactions with the environment and must learn good behaviour. Note that the Markov decision process model captures both sequential feedback and the more specific one-shot feedback (when $P(s'|s,a)$ is independent of s and a). It also captures both exhaustive feedback and the more general sampled feedback (when states are represented by features that allow $R(s,a)$ to be represented compactly).

2.2. Q-learning

Tabular reinforcement learning problems are those in which the state-action space can be explored exhaustively. Although challenging, there are a number of relatively simple algorithms [19] that can be used effectively in this setting. Q-learning is a model-free algorithm that uses transition experience of the form $< s,a,r,s' >$ (from state s, action a resulted in reward rand next state s') to improve an estimate $\hat{Q}$ of the optimal state-action value function Q^* . In particular, since $r+\gamma \max_{a'} \hat{Q}(s',a')$ is an unbiased estimator of $Q^*(s,a)$ as defined by the Bellman equation and $\hat{Q}$ is an estimate of Q^*, the estimate can be updated as:

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha * \delta_t, \tag{1}$$

where α is a step-size parameter that controls how new estimates are iteratively blended together over time,δ is the td-error which equals to $r+\gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a)$. Indeed, if the step-size is decayed at an appropriate rate,

$$\hat{Q}(s,a) \to Q^*(s,a) \tag{2}$$

for all state-action pairs. A simple model-based approach to tabular reinforcement learning uses the observed transition experience to estimate$R(s,a)$ (by averaging all rewards received from state sand action a) and $P(s'|s,a)$ (by the fraction of transitions from state sand action a that transition to state s').

3. NATURAL Q-LEARNING

Motivated by closed-loop peculiarity and the ensemble wisdom [20, 21] of the weak, cyclic step-size as well as ensemble action selector are proposed respectively to naturally boost the algorithm performance as well as meta learning design.

3.1. Cyclic step-size

Error is measured and injected in common closed-loop dynamic systems for signal feedback and decision support. This could enhance the stability and learning performance of the system to be modeled. Systems such as RL that incorporate td-error are always isolated from the whole system which exists as an independent component. Whereas there exists some interaction behavior between two variables: td-error and step-size as illustrated in Fig. 2.

Common RL implementations establish a direct link from td-error to system update, while in NaQL td-error is streamed down to the error correction block and cycles through system update. The whole interaction is depicted as a cycle and the general description is given as follow:

$$eM : \alpha_t = \Psi(d_t), \quad \alpha_t \in (0,1], \tag{3}$$

where d_t is a modified version of td error and Ψ is an error-mining function that exploits information from error. Besides the eM is continuous overall the axis horizon.

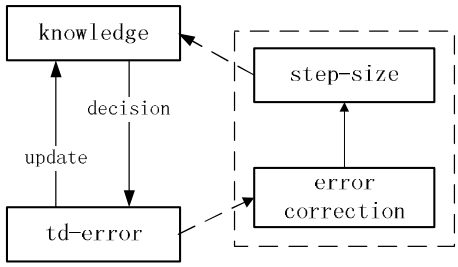


Fig. 2. Cyclic block.

To elaborate the tricks behind (3), mathematical interpretation would be considered. Define $\bar{e}_i$ as the average td-error value summed over an episode indicating how much error the system make in one learning cycle or how well the learner behaves in that episode. Subsequently, the instant td-error generates out the learner would evaluates how much distance it drifts from the $\bar{e}_i$ compared with last scenario, in which the drift variable is defined as d_t could be expressed by the following equations:

$$\bar{e}_i = \frac{1}{N}\{\underbrace{\delta_t^1 + \delta_t^2 + ...\delta_t^{N-m}}_{N-m} + \underbrace{\delta_t^{N-m+1} + ... + \delta_t^N}_{m}\} \tag{4}$$

$$= \frac{1}{N}\{\sum_{t=1}^{N-m} \delta_t^i + \sum_{t=N-m+1}^{N} \delta_t^i\} \tag{5}$$

$$= \frac{1}{N}\{E_{N-m} + E_m\}, \tag{6}$$

$$d_t^i = \Phi(\delta_t^i - \bar{e}_{i-1}). \tag{7}$$

The summed error is split as two parts: the history $N-m$ experiences $\to E_{N-m}$ and the recently m experiences $\to E_m$. The separation is inspired by the fact that the recently learning experience weights much more importance than the history one. Therefore if the history part is discarded, (5) could be intuitively reformulated as $\frac{1}{m}\sum_{t=N-m+1}^{N} \delta_t^i$ by maintaining a m-length queue.

Additionally (7) builds an error feedback bridge among consequential learning episode in which Φ is a cleaning or feedback function. Here by subtracting least k mean td-error from current td-error for confidence metric, the derived d_t^i populates down to the cyclic block. By chaining (4) through (7), the derived cyclic step-size could capture the inner regulation trick in the long term while guarantee learning speed.

3.2. Ensemble action selector

A tiresome tradeoff in q-learning is the exploration and exploitation. Pure exploitation causes the agent to reach the locally optimal policies quickly, whereas excessive exploration degrades the performance of the Q-learning algorithm even if it may accelerate the learning process and allow avoiding the locally optimal policies. This section gives a straightforward solution that can relax such dilemma by ensemble method. Ensemble is a bootstrap

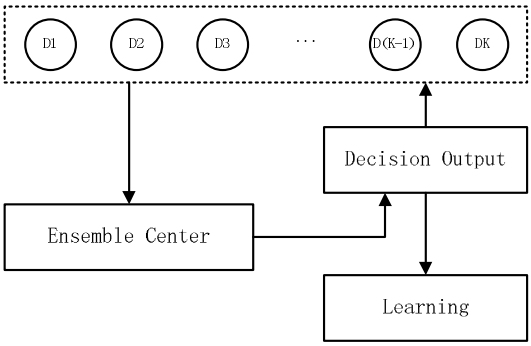


Fig. 3. Ensemble decision block.

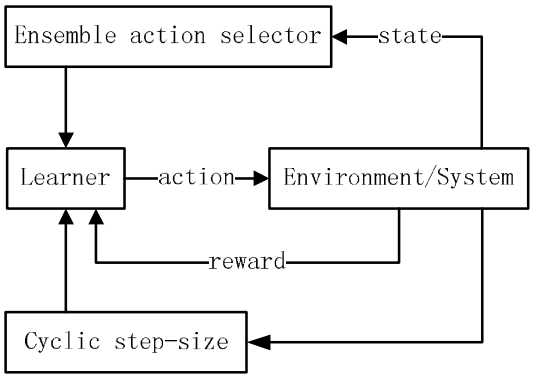


Fig. 4. NaQL algorithm block.

method that can improve decision accuracy and reduce the decision bias. A general ensemble decision block could be captured by Fig. 3, where $D1 \sim DK$ are the K same base weak decision makers. The final decision is achieved by fusing all the K base decision outputs characterized by the ensemble center.

The principle of selecting the K basis is randomness and simplicity. Randomness means the random parameter initialization if necessary and simplicity signifies the nutshell goods. In q-learning, the $\varepsilon - greedy$ action selector is preferred by the designers due to its simplicity and relative efficiency. While implementing the action selector in *NaQL* by the ensemble method, the base K components are chosen as *greedy* selectors. Through K basis, the parameter ε could be removed so that the meta design is enhanced to a high level. And the key component in ensemble center is the fusion function that could be generalized by follow:

$$k - basis = base\{gs_1, gs_2, ..., gs_k\}, \tag{8}$$

$$k - outputs = decision\{k - basis\}, \tag{9}$$

$$f - decision = voted\{k - outputs\}. \tag{10}$$

3.3. NaQL algorithm

In previous subsection, we discuss how each NaQL component is elicited. Furthermore, materials presented above lay a foundation for our NaQL algorithm to be interpreted. It's worth pointing out that the essence of NaQL lies in that the agent could adjust its step-size evaluated by td-error cycle throughout its learning life. The full block of NaQL algorithm is diagramed as Fig. 4.

The learner is equipped with two critical 'weapons': cyclic step-size and ensemble selector, the cyclic step-size block receives the system error and feeds the processed step-size to the learner while the ensemble selector makes current decision which action would interact with the system. Noticeably inner the cyclic step-size block, there is an error-mining function Ψ needs being fixed before learning. To satisfy the range and continuity constraints, here in later experiments the sigmoid function [22] is chosen to connect error and step-size. It is ubiquitously used in neural networks for layer output activation function by

Algorithm 1: *NaQL*

1. **Initialize** Q - table to zero
2. **Repeat** (for each episode)
3. $td_{sum} = 0, s = s_{start}$,steps=0
4. **while** not terminal
5. **a** $\leftarrow$ **ensemble action selection**
6. observe next state s'
7. $\delta \leftarrow$ error compute
8. $td_{sum} \leftarrow$ error collection
9. $\alpha \leftarrow$ **cyclic step-size update**
10. Q-knowledge update
11. s $\leftarrow s'$
12. $steps \leftarrow steps + 1$
13. prepare for next training

researchers in machine learning field. Hence α_t can be gained using follow equations:

$$g(x) = \frac{1}{1 + exp(-x)}, \tag{11}$$

$$\alpha_t = \beta * [g(d_t/l) - b_c], \tag{12}$$

where l is a positive scaling parameter,it controls the quantity of error by either squashing or amplifying; $\beta \in [0,2]$ is the gain parameter controlling the upper bound of α_t. $b_c = g(0)$ is a constant offset parameter which assures the validation of α_t together with β; Meanwhile, the innovative ensemble block is embedded for action selection. The only parameter or the one you need to think for a while is the base decision makers and its total numbers k, however the larger, the better. Therefore it's really a pretty selector for common tasks.

After analyzing above two blocks, the derived algorithm termed as NaQL block is presented in Algorithm 1. In the *NaQL* algorithm (Algorithm 1), line 5 takes an ensemble action selection procedure illustrated in Function 2.

In ensemble selector, fist initialize k base selectors(greedy selectors here), and the straightforward way to do such initialization is to generate k random floats

Function 2: Ensemble action selector.

1. **Init** k-base greedy selectors:$\{gs_1, gs_2, ..., gs_k\}$
2. **For** $i \leftarrow [1,k]$:
3. $\quad bout_i = gs_i(s,a)$
4. $a \leftarrow \mathbf{max}_i(voted(a_i)), a_i \in A$

Function 3: Cyclic step-size update.

1. $d_t = \delta - \bar{e}$
2. $\alpha_t = \Psi(d_t) = \beta * [g(d_t/l) - b_c]$

without repetition as the greedy parameter. Then get the vote statistics of each action $a_i \in A$ which formatted like a map structure: $vstat : \{(a_1,c1),(a_2,c2),...,(a_n,cn)\}$, and c_i is the number of selectors that vote for action i. Finally the first action by sorting *vstat* in descending order is fed to the learner.

After interacting with the environment with action selected by the ensemble block, the cyclic step-size is periodic updated by Function 3. Afterwards feeding the tuned value into the Q-value iteration step. Trained with an amount of training episodes, the step-size tends to approximate a stationary value in that the Q-value would approximate the optimal one.

In practical experiments, we incline to initialize the Q-values a larger value, e.g., 5, to make the agent more explorative. Furthermore, we tend to smooth $\bar{e}$ in each training episode via queuing up the historical $\bar{e}$ values as well as averaging them. As a result, this can gain a mild learning process.

4. EXPERIMENTAL RESULTS

To validate the tractability and evaluate the performance of the proposed NaQL algorithm for mobile robot, we take the classic experimental scenario: indoor navigation control of an autonomous mobile robot with settings figured in (Fig. 5).

4.1. Scenario description

In Fig. 5, each cell of the gridworld represents an individual state of the environment. In any free states, the robot can perform one of four primary actions: up, down, left, right. As their effect the agent moves one cell forward in the corresponding direction. When encountering the wall, the agent has no possibility of moving in the given direction - the resulting state after such action is the original one. Each atom operation has cost 1 and the received immediate reward can be assumed to be the negative cost value, i.e., -1. When the agent reaches the goal state, it receives reward 100 while receiving reward -100 once hitting the wall. The goal of the algorithm is to learn to seek an optimal policy which would make the robot move from S to G with minimized cost (or maximized rewards), and

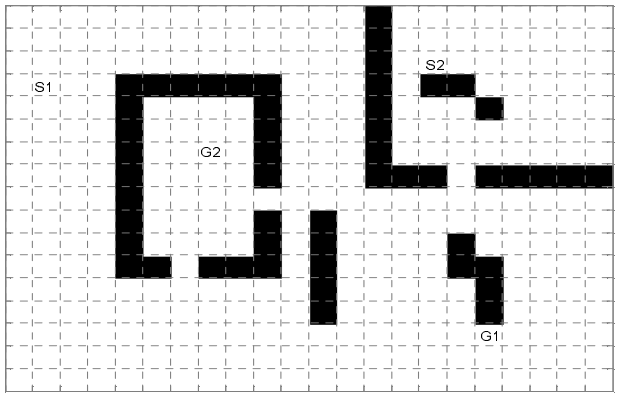


Fig. 5. The gridworld environment.

Table 1. Learning parameter table.

Parameter Type	Value	Domain
gamma	0.98	ALL
k-basis	50	NaQL
α list	{0.15,0.35,0.55,0.75}	CQL
gain β	2	NaQL
factor ω list	{0.2,0.4,0.6,0.8}	DQL
episode N	8000	ALL

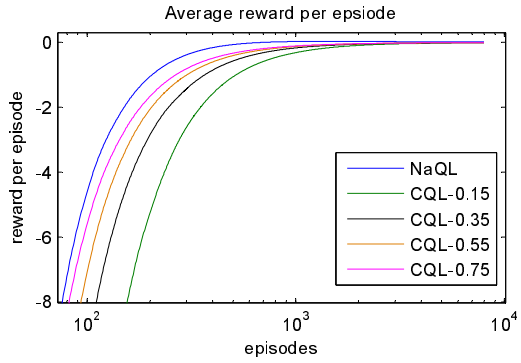
we defined two tasks for the simulation: **task1** : $S1 \rightarrow G1$, **task2** : $S2 \rightarrow G2$. The simulation program is implemented with MATLAB R2012b under the operating system of Windows 7.

4.2. Parameter table

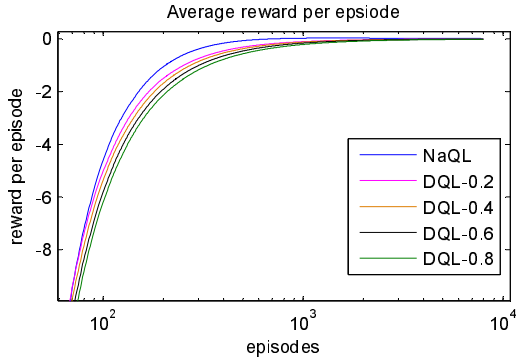
The algorithms to be evaluated and parameter settings are given in Table 1. **NaQL**: the proposed natural q-learning algorithm, the specified parameter is the number of base decision makers - k in the ensemble block and the gain factor β in the cyclic block. **CQL**: the q-learning with multiple constant step-size α settings. **DQL**: the q-learning with a decreased step-size with principle such as $\alpha_n = \frac{1}{(n/10+1)^{\omega}}$ where n is the episode index. Here **ALL** represents the parameters all the three algorithm share. Additionally the discount factor for all the three algorithms implemented is $\gamma = 0.98$ and the overall epoches used to train is $N = 8000$. And the navigation tasks.

4.3. Results

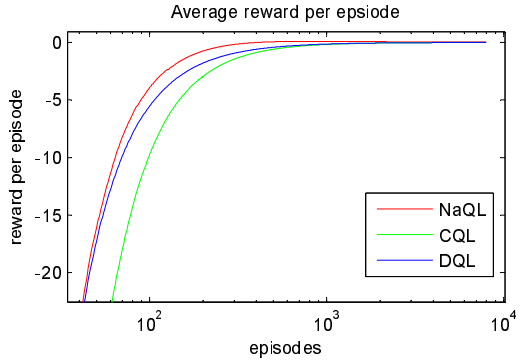
An important evaluation metric in reinforcement learning is the average reward per episode (ARPE) which represent the learning speed and the ratio of maximizing reward. The higher the better. And high value characterizes that the learner could acquire the optimal policy faster with a better approximation Q-value knowledge. Under this metric, as illustrated in Fig. 6(c) and Fig. 7(c), NaQL algorithm surpasses both CQL and DQL to an considerable extent.



(a) Rewards of NaQL, CQL with parameter list: α = {0.15,0.35,0.55,0.75}.

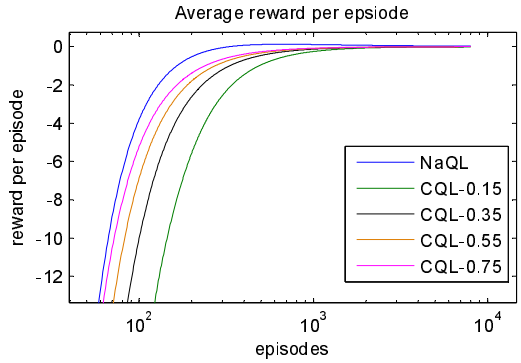


(b) Rewards of NaQL, DQL with parameter list: ω = {0.2,0.4,0.6,0.8}.

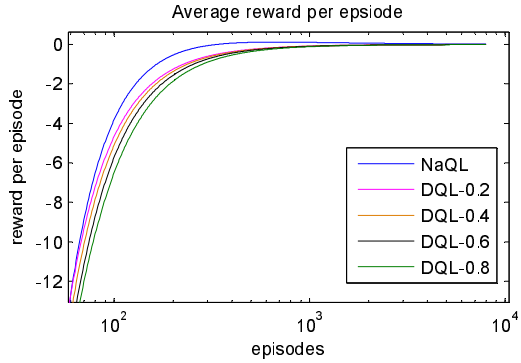


(c) Averaged rewards for NaQL, CQL and DQL, respectively.

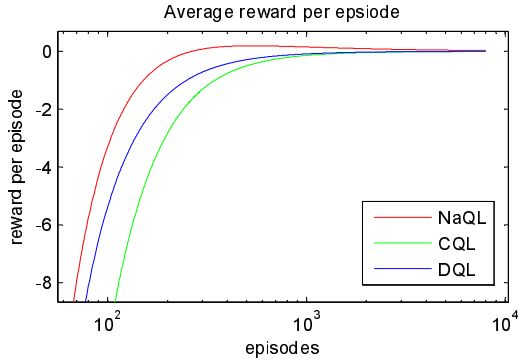
Fig. 6. Reward performance of task 1.



(a) Rewards of NaQL, CQL with parameter list: α = {0.15,0.35,0.55,0.75}.



(b) Rewards of NaQL, DQL with parameter list: ω = {0.2,0.4,0.6,0.8}.



(c) Averaged rewards for NaQL, CQL and DQL, respectively.

Fig. 7. Reward performance of task 2.

To remove the parameter factor in learning as much as possible,a set of learning parameters specifically for CQL with α list and DQL with ω list are taken for further evaluation. The resulting curves are pictured in Fig. 6 and Fig. 7. It's safe to say that the NaQL algorithm outperforms the other two and shows much space to improve. For how could it behave so well, we can treat the learning system as a dynamic system, using error-feedback in each iteration, the error gradient is decreasing. With high error offset, the cyclic function could check it for next update vice versa. [12] provides a better specified description which is not covered for detail discussion in this work.

Additionally, to make the statistical performance of NaQL more convincing, a separate specified experiment is conducted. In detail, the ensemble action selector module is removed from NaQL under the assumption that each module of NaQL contributes to the learning performance. And the assumption is validated via results captured in Fig. 9. The mechanism behind such phenomenon is that the two modules of NaQL shares no data dependence.

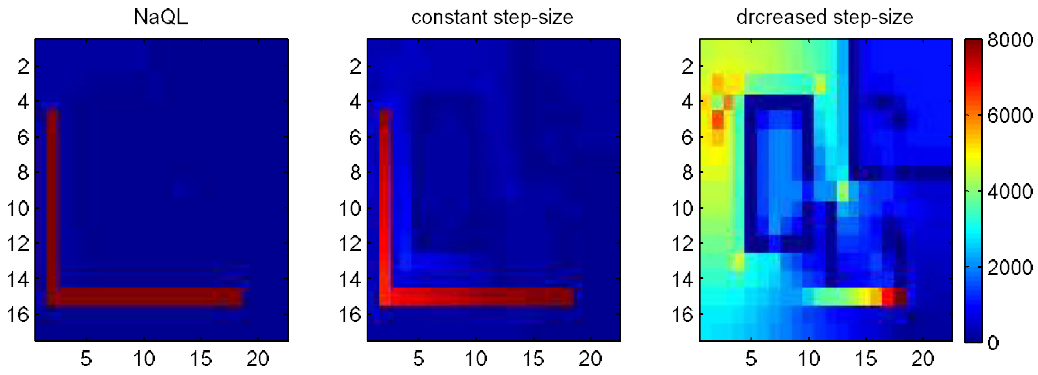


Fig. 8. The density map of visited states for task 1.

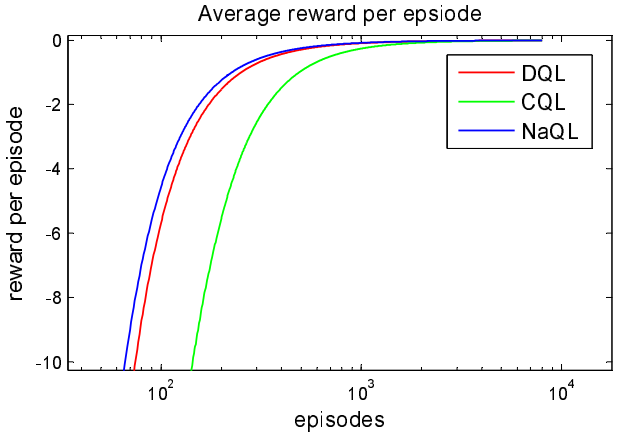
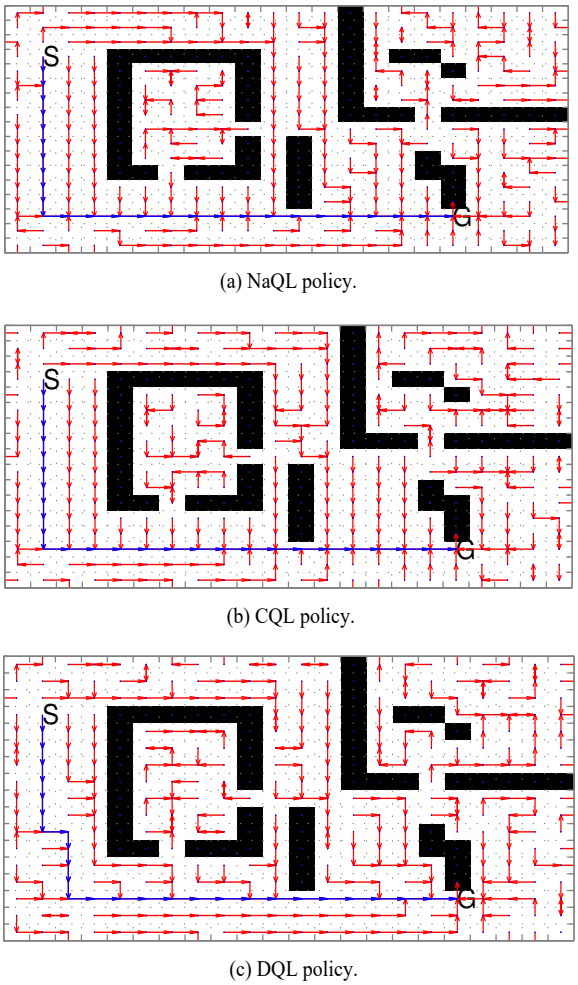


Fig. 9. Averaged rewards (under task1) with CQL, DQL, NaQL (which do not use the ensemble action selector).



(a) NaQL policy.

(b) CQL policy.

(c) DQL policy.

Fig. 10. Policies learned for task 1.

Another key metric for evaluation is the counts of each states visited or the density map of the states visited. This is a metric statistically measuring the frequency visits of each corresponding state. Dispersive density distribution reflects the learner could not behave its real learning process with respect the system, to the contrary, the centralized one means a better tradeoff between exploration and exploitation. The density maps (dmap) of task1 for evaluated algorithms is diagramed in Fig. 8. The vertical and horizontal coordinates are the location index illustrated in the gridworld environment, and the value at location (x,y) is the frequency of the state(location). It's obvious to see the dmap of DQL is dispersive due to its fixed numeric change. Further to say, each time the gained knowledge in later process is not always better than the history one while rigidly decreasing the step-size.

Compared with DQL, the dmap of CQL is relatively centralized due to its zero mean of step-size. However, this would result in high error variance latently. Turn to the dmap of NaQL, the distribution is well-blocked highlighting the better tradeoff and learning speed to some biological mechanism. To conclude, the proposed NaQL approach is tractable, comparable, efficient and promising for further mining.

Last but not the least, when learning process completed, the policy learned in the navigation task1 also the optimal path is visualized in Fig. 10. The notion of the figure is : blue arrows $\rightarrow$ the optimal policy, red arrows $\rightarrow$ the policy acquired in corresponding state s. Consistent with density

map metric, NaQL acquired the best policy. The reason for elaborating this neat result is twofold:

Error direction In NaQL, the cyclic step-size could more grasp the error direction through interaction with td-error. Hence in each iteration, the Q-value would go follow the approximated path.

Ensemble wisdom Base decision maker is weak and unstable with big bias and variance. While ensemble wisdom of the weak can beat the base one to an considerable extent. Additionally it could reduce bias and variance for better numeric calculation or update.

As learning evolves, the twofold inducements could compensate each other, namely have some variable interactions during learning while boosting the algorithm performance.

5. CONCLUSIONS

We herein investigated the notion that how step-size could adapt the learning process and what selector can grasp the trend of state transition. In this work a fresh algorithm called NaQL is proposed in the context of reinforcement learning for mobile robot navigation. Considerable experiments and pertinent metrics are employed. Furthermore, extensive statistical and mechanism analysis are elaborated in terms of simulation results.

To conclude, two specific blocks features the NaQL algorithm, cyclic step-size and ensemble selector respectively. Cyclic step-size could well adapt the td-error and the ensemble wisdom could mitigate the tradeoff. They altogether push the performance of NaQL to a high level. Nevertheless, to the best of our knowledge, adaptors or selectors in this work and almost all literatures are still priori specified by the designer. That's still intervened. To further squeeze the performance of these features, our future investigations will step into networked multi-robot navigation [23–26] as well as turn to dynamic environment.

REFERENCES

[1] M. Hutter and S. Scanner, *Recent Advances in Reinforcement Learning*, Springer, New York, 2012.

[2] R. Sutton and A. Barto, *Reinforcement Learning: An introduction*, MIT Press, Cambridge, MA, 1998.

[3] C. Watkins and P. Dayan, "Q-learning," *Machine Learning,* vol. 8, no. 3, pp. 279-292, 1992. [click]

[4] R. Coulom, *Reinforcement Learning Using Neural Networks, with Applications to Motor Control,* Institut National Polytechnique de Grenoble-INPG, 2002.

[5] J. Kober, J. Bagnell, and J. Peters, "Reinforcement learning in robotics: a survey," *International Journal of Robotics Research,* vol. 32, no. 11, pp. 1238-1274, 2013.

[6] B. Zuo, J. Chen, L. Wang, and Y. Wang, "A reinforcement learning based robotic navigation system," *Proc. 2014 IEEE International Conference on. Systems, Man and Cybernetics (SMC)*, pp. 3452-3457, 2014.

[7] J. Millan and C. Torras, "Learning to avoid obstacles through reinforcement," *Proc. the 8th International Workshop on Machine Learning*, pp. 298-302, 2014.

[8] A. Gosavi, *Simulation-based Optimization: Parametric Optimization Techniques and Reinforcement Learning,* Springer, US, 2014.

[9] A. Gosavi, "On step sizes, stochastic shortest paths, and survival probabilities in reinforcement learning," *Proc. the 40th Conference on Winter Simulation,* pp. 525-531, 2008.

[10] K. Moriyama, "Learning-rate adjusting Q-learning for prisoner's dilemma games," *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT'08,* pp. 322-325, 2008.

[11] A. Mahmood, R. Sutton, T. Degris, and P. Pilarski, "Tuning-free step-size adaptation," *Proc. 2012 IEEE International Conference on Acoustics, Speech and Signal Processing,* pp. 2121-2124, 2012.

[12] E. Even-Dar and Y. Mansour, "Learning rates for Q-learning," *The Journal of Machine Learning Research*, vol. 5, pp. 1-25, 2004.

[13] S. Lee, I. Suh, and W. Kwon, "A motivation-based action-selection-mechanism involving reinforcement learning," *Int. J. of Control, Automation, and Systems*, vol. 6, no. 6, pp. 904-914, 2008.

[14] M. Tokic and G. Palm, "Value-difference based exploration: adaptive control between epsilon-greedy and softmax," *Advances in Artificial Intelligence,* Springer Berlin Heidelberg, pp. 335-346, 2011.

[15] M. Guo, Y. Liu, and J. Malec, "A new Q-learning algorithm based on the metropolis criterion," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* vol. 34, no. 5, pp. 2140-2143, 2004. [click]

[16] F. Lewis, D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine,* vol. 9, no. 3, pp. 32-50, 2009. [click]

[17] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming,* John Wiley & Sons, 2014.

[18] M. Littman, "Reinforcement learning improves behaviour from evaluative feedback," *Nature*, vol. 521, no. 7553, pp. 445-451, 2015. [click]

[19] C. Szepesvari, "Algorithms for reinforcement learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 4, no. 1, pp. 1-103, 2010.

[20] C. Zhang and Y. Ma, *Ensemble Machine Learning*, Springer, New York, 2012.

[21] M. Mendoza and A. Bazzan, "The wisdom of crowds in bioinformatics: what can we learn (and gain) from ensemble predictions?" *Proc. the 27th AAAI Conference on Artificial Intelligence,* pp. 1678-1679, 2013.

[22] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.

[23] D. Borrajo and L. Parker, "A reinforcement learning algorithm in cooperative multi-robot domains," *Journal of Intelligent and Robotic Systems*, vol. 43, no. 2-4, pp. 161-174, 2005.

[24] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387-434, 2005. [click]

[25] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics,* vol. 21, no. 3, pp. 376-386, 2005. [click]

[26] Y. Li, L. Chen L, K. Tee, and Q. Li, "Reinforcement learning control for coordinated manipulation of multi-robots," *Neurocomputing*, vol. 170, pp. 168-175, 2015. [click]

Rongkuan Tang received his B.S. degree in Automation from Changshu Institute of Technology in 2013. He received his M.S. degree in Control Science and Engineering from Tongji University in 2016. His research interests include reinforcement learning and its application to robots.

Hongliang Yuan received his B.S. degree in Automatic Control from University of Science and Technology of China in 2002. He received his M.S. and Ph.D degrees in Electrical Engineering from University of Central Florida, USA, in 2006 and 2009, respectively. He is currently an associate professor at the School of Electronics and Information Engineering, Tongji University, P.R. China. His research interests include robotics, intelligent control, optimizations, vehicle dynamics and control, etc.