International Journal of Computing and Network Technology
ISSN 2210-1519
Int. J. Com. Net. Tech. 3, No. 3 (Sept. 2015)

Speeding Up the Learning in A Robot Simulator

Mostafa Al-Emran[1]

[1]Al Buraimi University College, Al Buraimi, Oman

Received: 10 May 2015, Revised: 25 July 2015, Accepted: 10 August 2015, Published: 1 (September) 2015

Abstract: Q-learning is a one of the well-known Reinforcement Learning algorithms that has been widely used in various problems. The main contribution of this work is how to speed up the learning in a single agent environment (e.g. the robot). In this work, an attempt to optimize the traditional Q-learning algorithm has been done via using the Repeated Update Q-learning (RUQL) algorithm (the recent state-of-the-art) in a robot simulator. The robot simulator should learn how to move from one state into another in order to reach the end of screen as faster as possible. An experiment has been conducted in order test the effectiveness of the RUQL algorithm versus the traditional Q-learning algorithm by comparing both algorithms through using similar parameters' values for several trials. Experiment results revealed that the RUQL algorithm has outperforms the traditional Q-learning algorithm in all the trials.

Keywords: Robot, Simulator, Q-Learning.

1. Introduction

Q-learning has proved its effectiveness as one of the Reinforcement Learning algorithms as to be used in a wide range of problems. Recently, researchers try to optimize the traditional Q-learning performance such as: Q-learning Influence Map (QIM) [1], Transfer Learning (TL) [2], Frequency Adjusted Q-learning (FAQL) [3] and the Repeated Update Q-learning (RUQL) [4].

In this paper, RUQL has been used to optimize the traditional Q-learning performance in a robot simulator. The robot simulator has been programmed in Java. The robot should learn how to move from one state into another in order to reach the end of the screen as faster as possible. An experiment has been conducted in order to compare the RUQL algorithm and the traditional Q-learning algorithm by trying similar parameters' values for both algorithms for several trials.

The other paper sections are organized as follows: section 2 will provide a background about Q-learning; section 3 will demonstrate the related research papers in the field; section 4 will describe the methodology and techniques used; evaluation and analysis will take place under section 5; discussion of results will be demonstrated in section 6; conclusion and future work will be presented in section 7.

2. Q-Learning

Q-learning is one of the Reinforcement Learning algorithms [1], [4], [6] that has been widely used in various domains such as: simple toys, face recognition and games [5]. The Q-learning update equation is described as follows:

$$Q^{t+1}(s,a) = Q^{t}(s,a) + \propto (r + \gamma\, maxQ^{t}(s,a') - Q^{t}(s,a))$$

Q-learning attempts to find an optimal action policy by calculating the function $Q(s,a)$ where s represents the state from the possible set of states$S$, and a represents the action from the possible set of actions A. The parameters: α represents the learning rate and γ represents the discount factor.

3. Related Work

Different research papers have been worked on different optimization techniques in order to enhance the traditional Q-learning performance. Cho [1] stated that the larger number of environment's states where the agent interacts with, can cause the Q-learning to take more time to learn these states as much as the number of states become larger. Q-learning technique using an influence map (QIM) has been proposed in order to reduce the amount of time required for learning.

Celiberto [2] utilizes the Transfer Learning (TL) among agents that permits the use of cases as heuristics in order to speed up the regular Q-learning. Q-learning shows artifacts in non-stationary environments (e.g. the probability of playing the best action might be reduced if the Q-values diverge considerably from the true values; this could occur in the initial phase along with the changes in the environment. Kaisers [3] had resolved the

E-mail address: malemran@buc.edu.om

mentioned artifacts through Frequency Adjusted Q-learning (FAQL). FAQL on the other hand suffered from the practical concerns that limit the policy sub-space for which the behavior was enhanced.

Abdallah [4] presented a novel solution that solves the undesirable artifacts of Q-learning and without the practical concerns of FAQL through the Repeated Update Q-learning (RUQL). In order to test the efficiency of the RUQL, experiments have been conducted in different multi-agent environments (e.g. Prisoner's Dilemma Game, Multi-Armed Bandit, and social learning). Experiments results revealed that RUQL is much better than both FAQL and QL in non-stationary environments. The RUQL update rule is described as follows:

$$Q^{t+1}(s,a) = [1-\propto]^{\frac{1}{\pi(s,a)}} Q^t(s,a) + [1 - (1-\propto)^{\frac{1}{\pi(s,a)}}]\, [r + \gamma Q^t(s', a_{max})]$$

If an action is selected with low probability *π(s,a)* then instead on updating that action once, RUQL repeats the update (1/*π(s,a)*) times.

RUQL proved its efficiency in different multi-agents environments, however a single agent environment (e.g. robot) still not yet investigated; the reason that motivated us to conduct this research.

4. METHODOLOGY AND TECHNIQUES

A. Software and Language used

The robot simulator has been programmed using *Java* programming language and the software used for programming is *NetBeans IDE* (which extensively used for Java desktop applications).

B. MDP Model

The most widely used model in Reinforcement Learning that describes the agent environment in <S, A, P, R> format where S represents the states, A represents the Actions, P represents the transition probability and R represents the reward. In the robot simulator, the states are represented by the robot arm and hand directions, so there are four states: (Arm up, hand right), (Arm up, hand left), (Arm down, hand right) and (Arm down, hand left). Actions are represented by the robot arm and hand movements, so the actions are: (Arm up, Arm down, hand right, hand left). Reward is represented by the distance in which the robot is traveled to reach the end of the screen. Fig. 1 depicts the robot simulator where the yellow line represents the arm and the red line represents the hand.

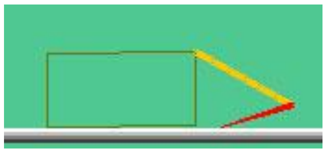

Figure 1. The Robot Simulator.

C. Parameters Description and Values

The optimal parameters' values that have been observed during the experiment (more in section 5) that allow the robot based on the RUQL to move as faster as possible up to the end of the screen is: γ= **0.9**, ***α*** **= 0.8** and ***Epsilon*** **= 0.4**. These parameters values make it easily for the robot to reach the end of the screen in **1** minute and **2** seconds.

D. Techniques and improvements

In order to optimize the traditional Q-learning performance and allow the simulated robot to reach the end of the screen as faster as possible, the *Repeated Update Q-learning (RUQL)* has been used in this work. An experiment has been conducted (section 5) in order to test the effectiveness of the RUQL algorithm and compare it to the traditional Q-learning algorithm in a robot simulator. A video had been recorded in order to show the race between the RUQL algorithm versus the traditional Q-learning algorithm via using the optimal parameters' values.

5. EVALUATION AND ANALYSIS

An experiment has been conducted in order to compare the RUQL algorithm (which is the main concern in this work) and the traditional Q-learning algorithm by trying the same parameters' values for both algorithms for each trial (more than 60 trials) and observes which one is reaching faster up to the end of the screen as in (Table 1). Table 1 show a sample from the entire number of these trials (where each trial is represented by a record) as follows:

TABLE I. RUQL VS. Q-LEARNING.

Epsilon	Gamma	Alpha	Q-learning (time in minutes)	RUQL (time in minutes)
0.6	0.8	0.8	**2:22**	**2:10**
0.4	0.6	0.1	**2:45**	**2:00**
0.6	0.9	0.8	**2:30**	**1:53**
0.6	0.7	0.9	**2:21**	**1:51**
0.6	0.6	0.8	**2:49**	**1:42**
0.5	0.8	0.8	**2:14**	**1:27**
0.4	0.6	0.3	**2:10**	**1:21**
0.4	0.6	0.9	**2:42**	**1:20**
0.4	0.8	0.8	**2:04**	**1:13**
0.4	0.6	0.7	**2:23**	**1:07**
0.4	0.9	0.8	**1:51**	**1:02**

The parameters' values in the above table are listed according to the time (from the largest to the smallest) in which the robot moves up to the end of the screen.

6. Discussion

By analyzing the experiment trials that were described in (Table 1), we can conclude the following:

- RUQL outperforms the traditional Q-learning with all the parameters' values used.
- By reducing the epsilon value, the robot moves faster up to the end of the screen.
- The table has explicitly revealed that the optimal parameters' values for getting the optimal robot speed are: $\boldsymbol{\gamma} = \mathbf{0.9}$, $\boldsymbol{\alpha} = \mathbf{0.8}$ and ***Epsilon* = 0.4**.

7. Conclusion and Future work

The Repeated Update Q-learning (RUQL) algorithm has been used in this work in order to optimize/ improve the traditional Q-learning algorithm in a robot simulator (i.e. in a single agent environment). Experiment results revealed that the RUQL algorithm has outperformed the traditional Q-learning algorithm in all the trials. The robot has learned how to move from one state into another in order to reach the end of the screen in **1** minute and **2** seconds.

The current work focuses on speeding up the learning via using RUQL algorithm and compares it to the traditional Q-learning algorithm. As a future direction, we could work on programming the robot simulator using the FAQL algorithm and compare the observed results to the results of this work.

References

[1] Cho, K., Sung, Y., & Urn, K. (2007, October). A Production Technique for a Q-table with an Influence Map for Speeding up Q-learning. In *Intelligent Pervasive Computing, 2007. IPC. The 2007 International Conference on* (pp. 72-75). IEEE.

[2] Celiberto, L. A., Matsuura, J. P., López de Màntaras, R., & Bianchi, R. A. (2010, October). Using transfer learning to speed-up reinforcement learning: a cased-based approach. In *Robotics Symposium and Intelligent Robotic Meeting (LARS), 2010 Latin American* (pp. 55-60). IEEE.

[3] Kaisers, M., & Tuyls, K. (2010, May). Frequency adjusted multi-agent Q-learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1* (pp. 309-316). International Foundation for Autonomous Agents and Multiagent Systems.

[4] Abdallah, S., & Kaisers, M. (2013, May). Addressing the Policy-bias of Q-learning by Repeating Updates. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems* (pp. 1045-1052). International Foundation for Autonomous Agents and Multiagent Systems.

[5] Rodrigues Gomes, E., & Kowalczyk, R. (2009, June). Dynamic analysis of multiagent Q-learning with ε-greedy exploration. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 369-376. ACM.

[6] Al-Emran, M. (2015, April). Hierarchical Reinforcement Learning: A Survey. *International Journal of Computing and Digital Systems*, *4*(2).

[7] T. Hester, and P. Hester, "Intrinsically motivated model learning for developing curious robots," Artificial Intelligence, 2015.

[8] L. Lin, H. Xie, D. Zhang, and L. Shen, "Supervised neural Q learning based motion control for bionic underwater robots," Journal of Bionic Engineering, 7, S177-S184, 2010.

[9] J. Grizou, M. Lopes, and P. Y. Oudeyer, "Robot learning simultaneously a task and how to interpret human instructions," IEEE, Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference, pp. 1-8, August 2013.

[10] R. Pegoraro, A. H. R. Costa, and C. H. Ribeiro, "Experience generalization for multi-agent reinforcement learning," IEEE, Computer Science Society, SCCC'01, XXI International Conference of the Chilean, pp. 233-239, 2001.

[11] X. Wu, J. Kofman, and H. R. Tizhoosh, "Active exploratory Q-learning for large problems," IEEE, Systems, Man and Cybernetics, ISIC, pp. 4040-4045, October 2007.

[12] J. Li, and W. Liu, "A novel heuristic Q-learning algorithm for solving stochastic games," IEEE, Neural Networks, IJCNN, IEEE World Congress on Computational Intelligence, pp. 1135-1144, June 2008.

[13] J. Broekens, "Robot Learning from Feedback," Springer, Encyclopedia of the Sciences of Learning, pp. 2873-2875, 2012.

Mostafa Al-Emran is the Head of Technical Support / Admission & Registration Department at Al Buraimi University College. Al-Emran got his BSc in Computer Science from Al Buraimi University College with the first honor level. He got his MSc in Informatics from The British University in Dubai with a distinction level. Al-Emran has published some research papers and is currently working on different research areas in Computer Science.