Volume 8, No. 3, March – April 2017
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Comparative Analysis of Reinforcement Learning Methods for Optimal Solution of Maze Problems

Savita Kumari Sheoran
Associate Professor & Chairperson
Department of Computer Science & Applications
Ch. Ranbir Singh University, Jind (Haryana) - India

Poonam
M.Phil. Scholar
Department of Computer Science & Applications
Ch. Ranbir Singh University, Jind (Haryana) - India

Abstract: Reinforcement learning is popular machine learning techniques for optimal planning in complex environment. The maze is a complex environment which has a grid made of an arbitrary number of squares of width and length where finding optimal path, which converge in minimum time, is always a challenging task. There are various reinforcement learning methods where agent learn from environment to find optimal path in maze problems viz. discrete Q-Learning, Dyna-CA Learning and FRIQ-Learning (Fuzzy Rule Interpolation-based Q-Learning). This research intends to carry out a comparative study of these three methods to locate a method with best convergence time. The algorithms pertaining to these methods are tested on MATLAB computational platform for different obstacles configurations of maze to compare their real time parameter of convergence time. The performance results were analyzed and presented. The final result reveals that FRIQ-Learning outperforms the others under all conditions.

Keywords: Reinforcement learning, maze environment, Q-Learning, Dyna-CA Learning and FRIQ-Learning.

1. INTRODUCTION

Learning is the act of acquiring new knowledge or modifying and reinforcing, existing knowledge, behaviors, skills, values or preferences which may involve different types of information. The ability to learn is possessed by humans, animals, plants and machines. Learning means change in behavior occurs as a result of experience. According to Psychologists learning process often occurs in three stages: acquisition, retention and recall. Gagne divided learning into eight categories: Signal Learning, Stimulus-Response Learning, Chain–Learning, Verbal Association Learning, Multiple Discrimination, Concept Learning, Learning of principles and Problem solving [1-3]. Machine learning is a subfield of Computer Science which means to train machines like a human or better than human to perform tasks. It involves the study of pattern recognition and computational learning theory in Artificial Intelligence. The machine learning is of three types: Supervised Learning, Unsupervised Learning and Reinforcement Learning [2-3].

Reinforcement Learning (RL) is a learning theory that came from animal theory and now applied on machines to work like a human being. This learning is used to train machines by trial and error. The main strength of Reinforcement Learning is that it does not specify how to solve a particular problem rather only need is to define final goal. So this learning focuses on what to do not how to do. The primary idea of Reinforcement Learning technique was inherited from optimal control, Markov decision process and dynamic programming. RL techniques are a kind of trial-and-error style techniques adapting to dynamic environment through incremental iterations, without need for training the system with pre-fabricated examples like supervised learning methods. Basic components of Reinforcement Learning are: states, actions, rewards and policy [4-8].

2. REINFORCEMENT LEANING METHODS

The sub sections 2.1, 2.2 and 2.3 of this section respectively represents the three popular reinforcement learning methods viz. discrete Q-Learning, Dyna-CA Learning and FRIQ-Learning (Fuzzy Rule Interpolation-based Q-Learning) in details.

2.1. Q-Learning

Q-learning is the most popular Reinforcement Learning method and hence it is plausible to choose it for further study and extension possibilities. In its naïve, the Q-Learning method was developed by Watkins and others. The purpose behind development of this method was to find the fixed-point solution 'Q' of the Bellman Equation through iterations. This method is also known as 'method of Asynchronous Dynamic Programming (ADP) [8] [9].

Q- Learning is a model free RL algorithm that uses last experience to update its policy. In this learning updated Q-function Q_{t+1} depends only on the previous function Q_t combined with the experience s_t , a_t and r_t . This feature makes the algorithm more computationally and memory efficient. This technique is different from other techniques such as Experience Replay, where experience is stored for later use. In this learning, an agent`s experience consists of a sequence of distinct stages. In the n^{th} stage the agent [10] [11]:

- Observes the current s_t state.
- Selects and performs a_t action.
- Observes the subsequent s_{t+1} state.
- Receives an immediate reward r_t .

In discrete environment, Q- Learning algorithm estimates the action-value function 'Q' iteratively by the following updating rule:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t) * (r_{t+1}) + \gamma max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \ldots\ldots\ldots\ldots\ldots\ldots \text{(i)}$$

Where 't' is discrete time step, s_t is state at time 't', a_t is the action at time 't', r_{t+1} is reinforcement or reward after executing action a_t in state s_t, $\alpha_t(s_t, a_t)$ is the step size parameter or learning rate, range belongs to (0<α<=1). 'γ' is discount factor, $Q_t(s_t, a_t)$ is the estimated state–action–value of action a_t in state s_t at time 't'. $max_a Q_t(s_{t+1}, a)$ is the estimated optimal future value [11] [12].

The learning rate or step size parameter determines the extent to which the newly acquired information will override the old ones. If the value factor set to '0' means agent does not learn anything and if factor is set to '1' would make the agent consider only the most recent information. In practice, maximum constant learning rate is used, such as $\alpha_{t.}$ =0.01 for all 't' [8] [13].

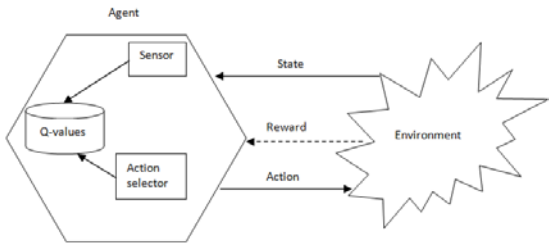

Figure 1: Basic Framework of Q-Learning

Discount factor 'γ' determines the future rewards importance. A value of '0' will make the agent short-sighted by only considering current rewards or counts the immediate reward only while a value approaching '1' will make it strive for a long-term high reward. Figure1 shows the basic framework of Q-Learning [8] [11].

2.2. Dyna-CA Learning

Dyna-CA Learning is a fast and effective learning algorithm. It is a Reinforcement Learning algorithm that is made by making some editing in Dyna Q- Learning algorithm. In this algorithm first of all the set of 'k' states will be observed around the current state. Then by using functional mapping, probability distribution of 'k' states on current state is calculated. On the current state, selection of action is made by recommending the weighted sum of best actions taken in the neighbor states to guarantee the learning with continuous actions. After that agent gets continuous actions and learns from experiences. Q values can be updated by using Dyna algorithm. These values are not only based on current neighbor state`s values but also the priori neighbor state`s experience [14].

Dyna-CA Learning is different from discrete Q- learning in the sense that in Dyna-CA Learning, updating Q values is the set of the 'k' states at the same time. Modeling the experience of the previous transitions from 'k' states to the next set of 'k' states, improve the convergence speed in comparison to discrete Q-Learning which is inspired by Dyna-Q- Learning [14].

Dyna-CA Learning is a new Reinforcement Learning algorithm that makes the agents to learn more effectively with continuous actions. This learning has two learning processes; one process learns its nearest actions according to nearest states while second process models the environment. Thus in this learning, agent chooses the best action.

Following algorithm describes the process of Dyna-CA algorithm.

Initialize A, S, k, γ, ε, Q(s, a) ⟵ 0, mode = []
Repeat {for each stage}
Initialize s
K ⟵ is set of k- nearest states of state s and get the probability P (K)
J ⟵ $argmax_A$ Q (K, A) (probability 1-ε)
(J ⟵ $randam_A$ Q (K, A) (probability ε))
a ⟵ <J, probability P (K) >
Q (K, J) =< max_A Q (K, A), P (K)>
Repeat {for each step of stage}
Take action a, observe state s` and reward r
K` ⟵ is set of k- nearest states of s` and get probability P (K`)
J` ⟵ $argmax_A$ Q (K`, A) (probability 1-ε)
(J` ⟵ $randam_A$ Q (K`, A) (probability ε))
a ⟵ <J`, probability P (K`) >
Q (K`, J`) =< max_A Q (K`, A), probability P (K`)>
Update Q and mode:
Q ⟵ Q + α(r + γ Q (K`, I`) – Q (K, I))
Model = (K, J, r, K`, J`), Mode ⟵ [Mode; model]
For I = 1 to N do
K, I ⟵ random previously observed set of k nearest states and actions
Get (K, J, r, K`, J`) from Mode
Q ⟵ Q + α(r + γ Q (K`, I`) – Q (K, I))
end for
a ⟵ a`, s ⟵ s`, K ⟵ K`, J ⟵ J`
until s is terminal
until learning ends

Agent selects the best action from its nearest states and updates the 'q' value. 'j' is the set of best actions and 'K' is the set of nearest neighbor states. $P(K)$ is the probability of 'K' nearest states. 's' is the current state, 'a' is the action on state 's' and 's``' is the next observed state. '$K$' is the set of $k$- nearest states of '$s$``' and $P(K`)$ is the probability of set 'k``'. In the Algorithm of Dyna-CA the ε-greedy policy had been referred to make the action-selection. The main characteristics of DYNA- CA algorithm are that this algorithm works on the problem that has continuous actions. It has faster convergence of learning than discrete Q-Leaning and DYNA Q- Learning [14].

2.3. FRIQ- Learning

In Fuzzy Rule Interpolation-Based Q- Learning, fuzzy rules are used to represent action-value function with the capability of sparse rule base. FRIQ is an extension of discrete Q- Learning with fuzzy rule interpolation method. Hence this method could handle continuous state and action space because of fuzzy rule based knowledge representation. FRIQ Learning algorithm is same as standard Q- Learning algorithm except of making derivation of Q-values using fuzzy inference system and updating of the Q- function (fuzzy rules). FQ- Learning has drawback of having same rule base for all number of state - action pairs. So, if there is a problem with large states, then it will be very complicated to handle all states with their state–action–value function using FQ- Learning. To eliminate this drawback, Fuzzy

Rule Interpolation (FRI) methods have been applied with FQ- Learning. By using FRI methods the rule-base can be simplified as well as the number of the rules representing the Q-function can be optimized for reducing the size of the state - action space representation because it can handle sparse rule base. There exist numerous FRI methods in the literature with the application of Q- Learning [12] [15].

The most important of them is FIVE (Fuzzy rule Interpolation based on Vague Environment) FRI method applied with Q- Learning (FRIQ- Learning). FRIQ-Learning method is application oriented method and serves crisp conclusions directly, without requirement any additional defuzzification steps. Most of the applications require crisp observations and crisp conclusions from the controller. The main idea of the FIVE is based on the fact that FIVE turn the fuzzy interpolation to crisp interpolation. The idea of a Vague Environment (VE) is based on the similarity of rules in the rule base or considered elements. In VE, the fuzzy membership function μA(x) is indicating the level of similarity, equivalently, or the degree to which 'x' is indistinguishable from specific element $'a'$ that is a representative or prototypical element of the fuzzy set $\mu A(x)$ [8] [16].

FQ-learning allows the omission of fuzzy rules or action - state values from the rule base and it has potentiality of applying in higher state dimensions with a reduced action - state space. By using the 0-order Takagi-Sugeno fuzzy model in FQ- Learning with the FIVE the FRI convert in to FRIQ- Learning [8] [16] [17].

In FRIQ- Learning method, Q-values are derived using fuzzy rule interpolation method by fuzzy inference system. Suppose a learning agent receives a vector of continuous states S= (s_1 ,s_2,….s_n) with 'n' dimensions and 'a' is the action on the state 's'. Fuzzy rules that derive Q values of state - action pair are given below:

R_i *: If* s_1 *is* V_{is_1} *and* ….s_n *is* V_{is_n} *and 'a' is* V_{ia} *then Q=* f_i (s_1 ,s_2,….s_n) .....(ii)
were V_{is_1},…., V_{ia} are fuzzy sets on variables. 'R' is vector of reward and $'R_i'$ is the i^{th} reward.

Algorithm: Fuzzy Rule Interpolation Based Q- Learning (FRIQ) [13]:

Step 1: s_t ***is the current state at time t.***
Step 2: Derive Q-Values for each state-action pair (s_t***,*** a_t***) using fuzzy inference system.***
Step 3: Choose an action a_t ***on state*** s_t***.***
Step 4: Carry out action a ***and transit the state*** s_t ***to next state*** s_{t+1} ***.***
Step 5: Derive Q-Values for each state-action pair (s_{t+1}***,***a_{t+1} ***) using fuzzy inference system.***
Step 6: Calculate update amount of Q-Value (ΔQ).
Step 7: Update Q-Function or fuzzy rules.
Step 8: Now go back to step 1.

As explained above, FRIQ has fuzzy rule base in the place of state-action-value function as was in traditional Q-Learning. In fuzzy rule base, k^t fuzzy rule has the following form [8] [12] [17] [18]:

Suppose s_1 =$S_{k,1}$, s_2= $S_{k,2}$…….., s_N= $S_{k,N}$ *and* a= A_k *Then* $\tilde{Q}(s,a)$= q_k.....(iii)

Where $'s_1'$ is observational state with k^{th} rule in the set of the discrete possible states 'S' and 'a' is the action at k^t rule in the set of the discrete possible actions 'A'. Also q_k is the observed value of the k^{th} rule.

According to state – action - value function $\tilde{Q}(s,a)$ updating rule, with fuzzy rule consequents ($q_{i_1 i_{2\ldots,} i_N u}$) is the following form:

$q^{k+1}_{\boldsymbol{i_1,i_2,\ldots,i_n,u}}$= $q^{k+1}_{\boldsymbol{i_1,i_2,\ldots,i_n,u}}$ + = $\Delta\tilde{Q}^{k+1}_{i,u}$ + $\alpha^k_{i,u}$.($g_{i,u,j}$ + $\gamma.max_{v\epsilon U}\tilde{Q}^{k+1}_{j,v} - \tilde{Q}^k_{i,u}$) ..... *(iv)*

Otherwise,

$q^{k+1}_{\boldsymbol{i_1,i_2,\ldots,i_n,u}}$ = $q^{k+1}_{\boldsymbol{i_1,i_2,\ldots,i_n,u}}$ + $\prod_{n=1}^{N}(1/\delta^{\lambda}_{s,k})/(\sum_{k=1}^{r} 1/\delta^{\lambda}_{s,k}).\Delta\tilde{Q}^{k+1}_{i,u}$

= $q^{k+1}_{\boldsymbol{i_1,i_2,\ldots,i_n,u}}$ + $\prod_{n=1}^{N}(1/\delta^{\lambda}_{s,k})/(\sum_{k=1}^{r} 1/\delta^{\lambda}_{s,k}).\alpha^k_{i,u}.($ $g_{i,u,j}$ +

$\gamma.max_{v\epsilon U}\tilde{Q}^{k+1}_{j,v} - \tilde{Q}^k_{i,u}$) ..... *(v)*

Where $q^{k+1}_{\boldsymbol{i_1,i_2,\ldots,i_n,u}}$ is the $(k+1)^t$ iteration of fuzzy rule consequents ($q_{i_1 i_{2\ldots,} i_N u}$) with action '$A_u$'. The factor $g_{i,u,j}$ is the observed reward from state $S_i \rightarrow S_j$ with actionA_u. $\alpha^k_{i,u}$ is the step size parameter between range [0, 1] and 'γ' is the discount factor.

3. MAZE ENVIRONMENT

A Maze is a path or network of paths that has a goal. It has many obstacles in the path, so one has to cover these obstacles to find goal. Main motive is to find goal with shortest path with less time and more accuracy from these puzzle network of paths. In Figure 2 and 3, the Green Square is an initial point and Black Square is goal point. There are two paths for find the goal. But the path in later figure is lengthy and time consuming in comparison to later [19].

There is a need of better algorithms and techniques for solving this puzzle network of paths (solving maze) for arriving at the goal. The solution to a Maze problem, mean finding a route through the maze to arrive at the goal. The Maze can be solved manually or by computing machines. For such problems there is need is to solve these puzzle paths with minimum time, high speed and with high accuracy, which can be only be possible with computer [18].

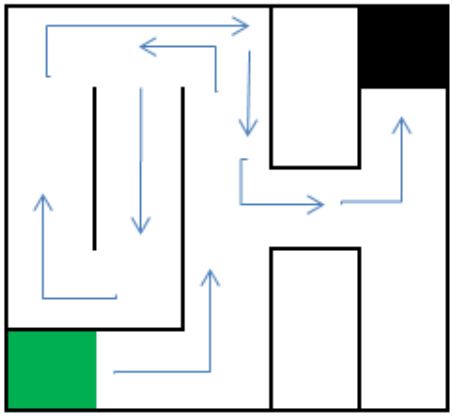

Figure 2: Not the Most Efficient Short Path in this Maze [19]

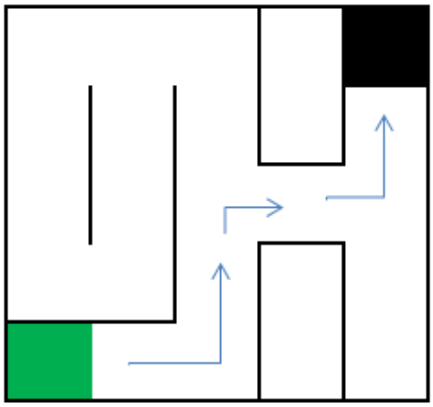

Figure 3: A Better Result of Shortest Path in this Maze [19]

Reinforcement Learning techniques are applied on maze problem with different obstacle delays. As the states increases in any problems, time also increases to find goal. Discrete Q- Learning is often considered a slow procedure. However, in the case of other Reinforcement Learning methods such as Dyna-CA- Learning and FRIQ- Learning are fast, incremental and scalable. These states of art Reinforcement Learning algorithms are applied on maze problem to analyze the performance. FRIQ- Learning algorithm has recently received much attention because it reduces size of state – action – value function. It highly reduced state representations by varying these dimensions in maze environment. We have achieved better search efficiency per simulation. Furthermore, the advantage of FRIQ- Learning increased with larger state dimensions.

Maze problem is a simple benchmark for comparing the effectiveness of different RL algorithms. So this environment is chosen for comparison of these techniques. Maze itself is *n*m* matrix, where 'n' represents number of rows and 'm' represents number of columns. This play area has a start point, a goal point and some obstacles that make problems to achieve the goal. The agent has to start from the start point and goes to the goal point as well as avoiding the obstacles. The main objective of this agent is to find the shortest path from start position to goal position by avoiding the obstacles. To find out goal with shortest path, here start position, obstacles and goal position are denoted by variables [12].

4. PERFORMANCE ANALYSIS

The discrete Q- Learning, Dyna-CA Learning and FRIQ Learning are implemented on the basis of time taken from start position to goal position in different maze environments. For implementing these reinforcement learning techniques in maze environment with time convergence, MATLAB GUI tool has been used. Four maze environments have been taken without and with obstacles delay. Four edit buttons are used for denoting maze matrix as 9×6 and 18×12. The convergence time is calculated in traditional unit of "second". The command button name as "compare" is used to compare these four maze environments:

- 9×6 (without obstacle).
- 9×6 WO (with obstacle).
- 18×12 (without obstacle).
- 18×12 WO (with obstacle).

Figure 4 depicts the graphical representation of convergence times for these three methods without obstacle delay in maze environments. Figures 5, 6, 7 and figure 8 graphically analyses the convergence times for these three methods with obstacle delays 0.001, 0.002, 0.01 and 0.02 respectively. The cohesive intra comparison of convergence time for three learning methods is presented below:

As revealed from plots presented in Figures 4, 5, 6, 7 and 8 with difference of times on different maze environments with varying obstacles and increasing state-action-value function, it is clear that discrete Q- Learning effected more because Q- Learning uses action-value function of each state-action. Therefore, with increase in states, state-action-value functions also increases. Hence for large state space, it is difficult to handle large state - action- value function because it takes more time with large state space. FRIQ-Learning affected less than other methods because, in FRIQ Learning state-action-value function is used as rule base. In rule base, fuzzy rule interpolation method has been applied, because of that rules those not have more importance or rules those are beyond the upper or lower limit, they not fired.

(A) Comparative Analysis of Q-Learning, Dyna-CA, FRIQ- Learning without Obstacle Delay

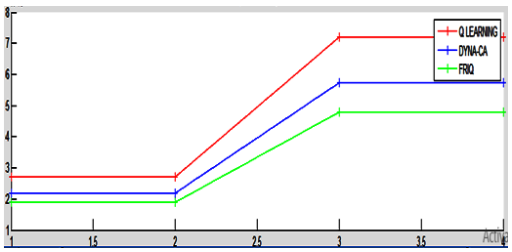

Figure 4: Convergence Time Comparison without Obstacle Delay

(B) Comparative Analysis of Q-Learning, Dyna-CA-Learning, and FRIQ- Learning with Obstacle Delay = 0.001

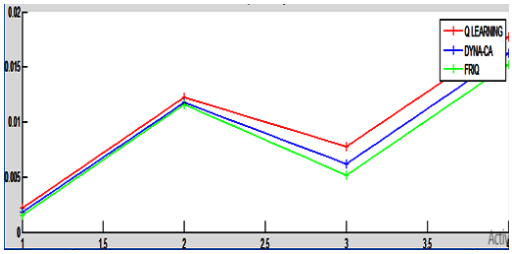

Figure 5: Convergence Time Comparison with Obstacle Delay = 0.001

(C) Comparative Analysis of Q-Learning, Dyna-CA-Learning, FRIQ- Learning with Obstacle Delay = 0.002:

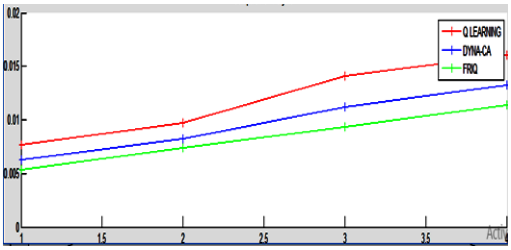

Figure 6: Convergence Time Comparison with Obstacle Delay = 0.002

(D) Comparative Analysis of Q-Learning, Dyna-CA-Learning and FRIQ- Learning with Obstacle Delay = 0.01:

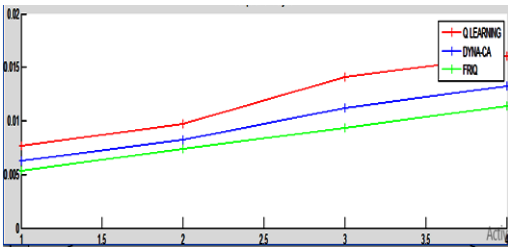

Figure 7: Convergence Time Comparison with Obstacle Delay = 0.01

(E) Comparative Analysis of Q-Learning, Dyna-CA-Learning and FRIQ- Learning with Obstacle Delay = 0.02:

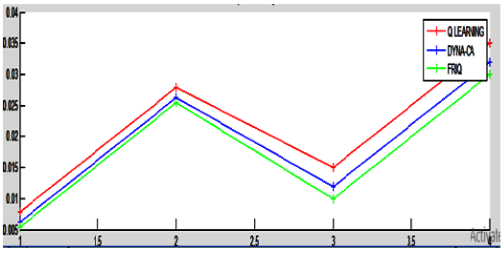

Figure 8: Convergence Time Comparison with Obstacle Delay = 0.02

It can be concluded from these plots that Q- Learning takes more time than both DYNA-CA- Learning and FRIQ-Learning while FRIQ- Learning converge in minimum time to find the goal.

5. CONCLUSION AND FUTURE DIRECTION OF RESEARCH

By observing graphs, it is clear that discrete Q- Learning affected more with increasing obstacles delays and states, because Q- Learning uses action – value function for each state-action. Therefore with increasing of states, state – action - value functions also increases. So in large state space, it is difficult to handle large state – action - value function, it takes more time with large state space. FRIQ-Learning is affected less in comparison to other methods. FRIQ- Learning has rule base with fuzzy rule interpolation method. In rule base, only the method within the upper or lower limit of system can fire. Q- Learning takes more time than both DYNA-CA- Learning and FRIQ- Learning while FRIQ- Learning takes minimum time to find the goal. Therefore, FRIQ- Learning emerge as fastest converging (in term of speed and time) Reinforcement Learning technique for maze environment with or without obstacle.

The FRIQ-Learning has smaller Q-function representation, leads to better convergence speed. But equal size of the Q-function rule bases in all the maze configurations raise concern in decision making, so rule-base reduction method needs further investigation in FRIQ- Learning for better result. Since this study is a based upon theoretical conceptualization. Therefore, the practical implementation of proposed objectives on real time maze environment may depict the desired outcome and can be applied to daily life problems. Also the whole study is carried out in 2-D only. The 3-D extension of study may be much fruitful to solve intricate geometrical problems in Euclidean and Non-Euclidian space.

6. REFERENCES

[1]. N. J. Nilsson, “Introduction to Machine Learning,” Mach. Learn., vol. 56, no. 2, pp. 387–99, 2005.

[2]. K. Wagstaff, “Machine Learning that Matters,” Proc. 29th Int. Conf. Mach. Learn., pp. 529–536, 2012.

[3]. Hal Daume, a book “A Course in Machine Learning” version 0.8 , August 2012

[4]. S. Mahajan, “Reinforcement Learning : A Review from a Machine Learning Perspective,” vol. 4, no. 8, pp. 799–806, 2014.

[5]. W. Qiang and Z. Zhongli, “Reinforcement learning model, algorithms and its application,” IEEE Int. Conf. Mechatron. Sci. Electr. Eng. Comput., no. 1, pp. 1143–1146, 2011.

[6]. Jos´e Antonio Mart´ın H., “An Effective Algorithm for Continuous Actions Reinforcement Learning Problems” IEEE, 2009.

[7]. R. S. Sutton and A. G. Barto, “Reinforcement learning,” Learning, vol. 3, no. 9, p. 322, 2012.

[8]. D. Vincze, university of Miskolc faculty of mechanical engineering and informatics, Ph. D. Dissertation on “Fuzzy Rule Interpolation-based Q-learning,” 2013.

[9]. M. Pieters and M. A. Wiering, “Q-learning with Experience Replay in a Dynamic Environment.”

[10]. Christopher Watkins and Peter Dayan, “Technical Note: Q-Learning” 1992.

[11]. C. H. C. Ribeiro, “A Tutorial on Reinforcement Learning Techniques,” Supervised Learn. track tutorials 1999 Int. Jt. Conf. Neural Networks, pp. 1–44, 1999.

[12]. T. Tompa and S. Kovács, "Q-learning vs. FRIQ-learning in the Maze problem," IEEE, pp. 1–6, 2015.

[13]. Tackishi Horiuchi, Akiiiori Fujino, Osaniu Kat ai and Tetsuo San-aragi, "Fuzzy Interpolation based Q-Learning with continuous states and actions," IEEE, 1996.

[14]. F. U. Bo, C. Xin, H. E. Yong, and W. U. Min, "An Efficient Reinforcement Learning Algorithm for Continuous Actions," IEEE, pp. 80–85, 2013.

[15]. D. Vincze and S. Kovacs, "Rule base reduction in Fuzzy Rule Interpolation based Q-Learning" vol. 2, no. 1, pp. 1–6, 2015.

[16]. D. Vincze and S. Kovács, "Fuzzy Rule Interpolation-based Q-learning," pp. 55–60, 2009 IEEE.

[17]. D. Vincze and S. Kovács, "Reduced Rule Base in Fuzzy Rule Interpolation- based Q-learning," pp. 533–544.

[18]. M. Ahuja, B. Homchaudhuri, K. Cohen, M. Kumar, S. Member, and A. Fellow, "Fuzzy Counter Ant Algorithm for Maze Problem," no. January, pp. 1–13, 2010.

[19]. A. Bakar and S. Saman, "Solving a Reconfigurable Maze using Hybrid Wall Follower Algorithm," Int. J. Comput. Appl., vol. 82, no. November, pp. 22–26, 2013.