Simulation of the navigation of a mobile robot by the Q-Learning using artificial neuron networks

Mezaache Hatem and Abdessemed Foudil

University Hadj Lakhdar of Batna-Algeria-Faculty of Engineering Department of Electronics
whatem_2004@yahoo.fr

Abstract.

This paper presents a type of machine learning is reinforcement learning, this approach is often used in the field of robotics. It aims to determine a control law for a mobile robot in an unknown environment. This kind of technique applies when one assumes that the only information on the quality of actions performed by the mobile robot is a scalar signal which has a reward or punishment, the process of learning is to improve the choice of actions to maximize rewards. One of the most used algorithms for solving this problem is learning the Q-learning algorithm which is based on the Q-function, and to ensure the generation of this latter function and the proper functioning of the apprenticeship system using an artificial neural network as the statements of changing environments where mobile robots have wide open spaces, the action performed by the mobile robot in its environment is ensured by using a selection function, this action is evaluated by a scalar signal which is -1, 0 and 1.

Key words: Reinforcement learning, Q-Learning, Q-Function, Artificial Neural Networks, Mobile Robot.

1 Introduction

Learning is a process to improve the performance of a system based on its past experiences. This method occurs when the problem seems too complicated to solve in real time, or when it seems impossible to solve the problem in a classical and rigorous. An example of learning methods cited reinforcement learning.

The reinforcement learning is a technique which is to acquire the agent executor behavior desired by methods based on the concept of reward or punishment. The optimal behavior of an agent is often difficult to implement given the large number of variables that may play a role. In the framework of reinforcement learning, the agent can learn to behave in the same way as we learn to ride a bicycle from a signal known as reinforcement. One of the fundamental parts of the system of reinforcement learning is the Q-function; it allows the agent to learn how to choose good actions and how to measure their utility.

One of the goals for the navigation of autonomous mobile robots is the avoidance of obstacles, several techniques and methods are used for this reason [1], [2]. The algorithm, which allows the mobile robot starting from a stop position and a final position following a pattern set meadows. If the robot encounters obstacles in its path, the algorithm of obstacle avoidance takes control of mobile robot. Once the path of the robot is free shipping to the destination is taken [3]. In this article we solve the problem of navigation with obstacle avoidance by a method of learning. For this purpose the Q-Learning with the Q-function is generated by a network of neurons

2 Reinforcement learning

2.1 Principle

The reinforcement learning is a learning technique based on trial and error, and the interaction between the agent and its environment [4], [5], where from a state or situation ***s*** in the environment, the agent selects and performs an action that causes a transition to state ***s'***. He receives in return a reinforcement signal ***r***, which can be a reward or punishment. The purpose of this learning is to maximize future rewards [4]. Fig. 1 shows a state of interaction between the agent and the environment

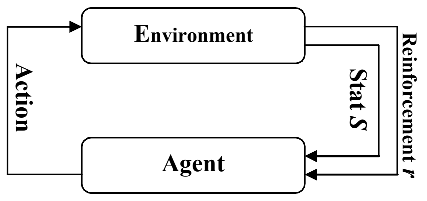

Fig. 1. Interaction between the agent and the environment

2.2 Q-Learning

The Q-Learning is a famous algorithm that is used for solving problems of reinforcement learning, it was proposed in 1989 by C. J. Watkins [6], [7]. This algorithm is based on three main functions, an evaluation function, a function of strengthening and reinforcing function. Fig. 2 shows the general model for the algorithms of reinforcement learning, by Q-Learning.

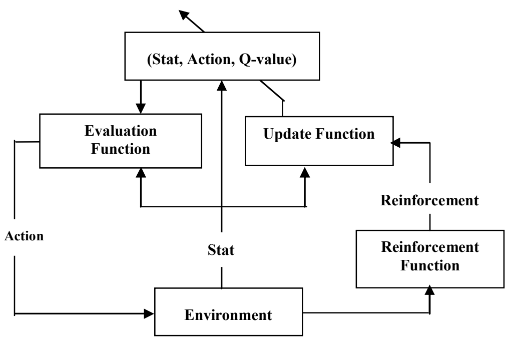

Fig. 2. The general model for the algorithms of reinforcement learning, by Q-Learning.

2.2.1 Evaluation function

The state of the environment and the action performed by the agent are evaluated by this function is called Q-Function

From a current state s of the environment that is observed by the agent, an action is performed based on knowledge available within the internal memory (This knowledge is stored in the form of utility value associated with a pair of "State Action"). [8]

2.2.2 Reinforcement Function

After executing the action by the agent in its environment; Reinforcement function of ***R***, provides for each new state ***s'***, ***r*** a signal that can be a reward or punishment, this signal usually takes a single value, 1, -1 or 0, which is used by the Update function to adjust the Q-value function associated with the pair "State Action" [8].

2.2.3 Update Function

This function uses the value of reinforcement to adjust the value associated with the pair "State, Action" which has just been completed [8].

The Q-Learning as a principle estimating the Q-function noted by: Q^*, and is defined by:

$$Q^*(s,a) = E\left[r_{t+1} + \gamma V^*(s)\right] = E\left[r_{t+1} + \gamma \max_{a'} Q^*(s',a')\right] \quad \textbf{(1)}$$

Using the equation for updating one finds that:

$$Q^{\pi}_{t+1}(s_t,a_t) = (1-\alpha_t)Q^{\pi}_t(s_t,a_t) + \alpha_t \left[r_{t+1} + \gamma \max_{a'}\left(Q^{\pi}_t(s',a')\right)\right] \quad \textbf{(2)}$$

Or:

r_{t+1} Is the reinforcement received when the agent selected the "***a***" action in the state "***s***" to move to the state "***s'***".

α_t Is a real positive: $\alpha_t \in \left]0,1\right[$

In principle it should be randomly ***explore*** the environment for a large number of iterations for the Q-Learning converges to the optimal Q - function [8], then we can use the optimal policy defined by:

$$\pi'(s) = \arg\max_{a \in A} Q^*(s,a) \tag{3}$$

3 System architecture of reinforcement learning

The objective of this proposition is to have a learning system that allows a robot that moved in an environment that is totally unknown in avoiding obstacles.

The generation of the Q-function is performed by an artificial neural network-type MLP, where the inputs are reading ***sensors*** associated with the robot on its three sides, giving the perception of its ***environment***, and the vector of possible actions (Turn Left, Forward and Turn Right). The choice of action that the agent must perform is a function called by ***function selection***.

Fig. 3 shows the structure of reinforcement learning based on an Artificial Neural Network

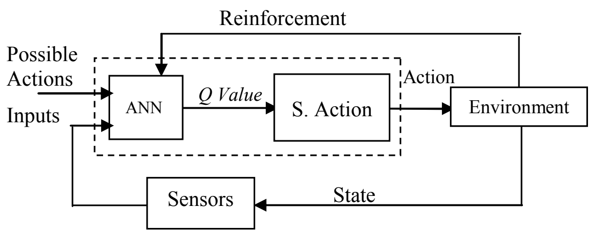

Fig. 3. The structure of reinforcement learning based on an Artificial Neural Network.

4 Generating the Q-function with an Artificial Neural Network

The generation of the Q-function can be made by a table in which each cell corresponds to an approximation of the Q-function for a configuration of the pair state / action. This severely limits the size of problems we can solve; in fact, many real-world problems such as robotics have a large space. An innovative approach to the generation of the Q-function in the case of large spaces is to use Artificial Neural Networks. The approximation of the Q-function is obtained, using Artificial Neural Networks with the learning algorithm Back propagation [9], [10]

In this implementation, the Artificial Neural Networks chooses is a Multi Layer Perceptron, which entered as the state of the environment and the possible actions, a layer containing n_h hidden neurons and one output neuron that max of the Q-function [11]. The activation function of all neurons is the sigmoid function. Fig. 4 shows the generation of the Q-function with Artificial Neural Network-type MLP for Q-Learning.

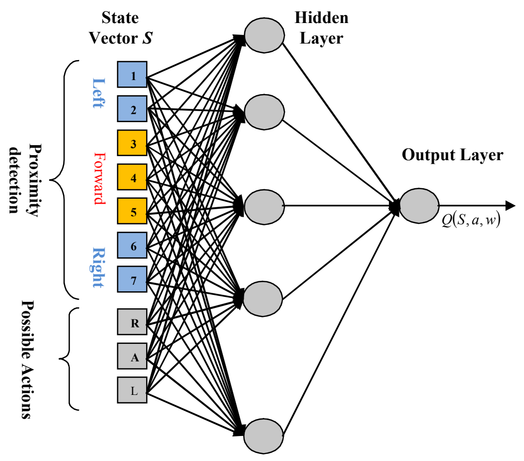

Fig. 4. Q-function with Artificial Neural Network-type MLP for Q-Learning

4.1 Learning of Artificial Neural Networks

The learning Network is based on updating or adjusting the weight matrices of the Neural Network using the equation of the update of the classic algorithm of Q-Learning and the algorithm Back propagation

4.1.1 The output layer

- For the weight matrix

$$w_2(t) = w_2(t-1) + \alpha[(r(s_t,a_t) + \gamma a(S)) - Q(s_t,a_t,w_t)]f(S)\frac{\partial Q}{\partial w_2}(a(S) - Q_{target}) \quad \textbf{(4)}$$

- For the vector of means we have

$$b_2(t) = b_2(t-1) + \alpha[(r(s_t,a_t) + \gamma a(S)) - Q(s_t,a_t,w_t)]f(S)\frac{\partial Q}{\partial w_2}(a(S) - Q_{target}) \quad \textbf{(5)}$$

Where:
$\boldsymbol{Q}_{target}$ is simplifying the equation of optimality BELLMAN which is given by the following equation

$$Q_{target} = r(s_t, a_t) + \gamma \max Q(S, a_t, w) \quad \textbf{(6)}$$

f : Activation functions of neurons in the output layer.

$Q(s_t, a_t, w)$: Function value state / action corresponding to the action performed.

S : State of the environment.

4.1.2 The hidden layer

- For the weight matrix

$$w_1(t) = w_1(t-1) + \alpha[(r(s_t, a_t) + \gamma a(S)) - Q(s_t, a_t, w_t)]S \frac{\partial f}{\partial w_1} \sum w_2 S \tag{7}$$

- For the vector of means we have:

$$b_1(t) = b_1(t-1) + \alpha[(r(s_t, a_t) + \gamma a(S)) - Q(s_t, a_t, w_t)]S \frac{\partial f}{\partial w_1} \sum w_2 S \tag{8}$$

With:

f : Activation functions of neurons in the hidden layer.

S : State of the environment.

$Q(s_t, a_t, w)$: Function value state / action corresponding to the action performed.

These changes in values for the weight matrix and vector of bias are present if the reinforcement signal is: -1 or 0.

5 Function Selection Action

The neural network allows us to generate the Q-function. The set of possible actions is given by where:

• a_1: Turn Left Action.
• a_2: Forward Action.
• a_3: Turn Right Action.

The selection of the action that the robot must execute is based on the policy Exploration / Exploitation (**EEP**) [12], for this reason we used the greedy method (ε -greedy) which consists of choosing the greedy action with probability ε and to choose a random action with a probability, $1-\varepsilon$.

• $p \in [0,1]$ a random number.

• If $p < \varepsilon$ we chooses a random action a "***Exploration***", where $a \in A$ of the all actions possible.

• If $p > \varepsilon$ you selected

$$a(S) = Arg \max_{b \in A} Q(S, b, w) \text{ “\textbf{\textit{Exploitation}}”} \tag{9}$$

6 Environment

Reading the state of the environment is done through sensors that are placed on three sides of the robot. Two on the left, two on the right and three in front. The sensors can be used type of proximity detection. The opening angle of each sensor varies between -π/12 and π/12. The state vector ***S*** is chosen so as to obtain information on the existence of obstacles on three sides of the robot.

This vector is composed of seven binary variables s_i, i = 1,.. 7. The choice of these variables is made in order to have any information or anything. i.e., for example if $s_i = 1$ then there is an obstacle near the robot, if $s_i = 0$ no obstacle near the robot.

Fig. 5 shows a state of the environment that gives the value of state vector ***S***, such as $S = [1,1,0,0,0,0,0]$.

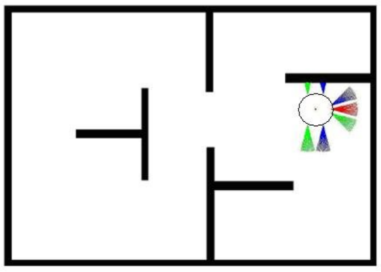

Fig. 5. State of the Environment.

7 Function of reinforcement

For each state in which the robot is found, an evaluation of the action done is found by a signal known as signal reinforcement. This function of reinforcement allows the robot to explore its environment; this signal is related to the values of sensor measurements that indicate the presence or absence of obstacles in three directions, left, front and right, which represent the state of the environment. The value of this function or the signal reinforcement is given by:

$$r = \begin{cases} -1 & \text{le robot percute un obstacle} \\ 0 & \text{les autres cas.} \quad . \\ +1 & \text{le robot avance tout droit} \end{cases} \qquad \textbf{(10)}$$

8 Model of the Robot

The type of robot that we have chosen for the application is circular of radius R. This robot is operated by two independent wheels separated by a distance L. Fig. 6 shows this type of robot.

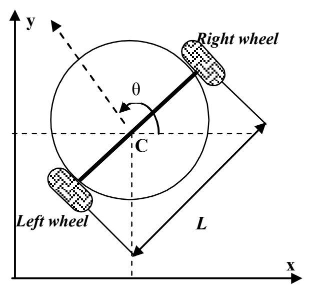

Fig. 6. The robot Type.

This robot is characterized by the following cinematic equations:

8.1.1 The position of the robot

$$x_r(k+1) = x_r(k) + v(k) * \cos\left(\theta(k) + \frac{\Delta\theta(k)}{2}\right) \quad \textbf{(11)}$$

$$y_r(k+1) = y_r(k) + v(k) * \sin\left(\theta(k) + \frac{\Delta\theta(k)}{2}\right) \quad \textbf{(12)}$$

x_r (k) and y_r (k) are the x and y of the robot in the landmark (Ox, Oy).

8.1.2 The orientation of the robot

$$\theta(k+1) = \theta(k) + \Delta\theta(k) \quad \textbf{(13)}$$

With:
$\theta(k)$: Is the angular position of the robot in the landmark (Ox, Oy)

8.1.3 The speed of the robot

$$v(k) = \frac{1}{2}(\theta_r(k) + \theta_l(k)) \quad \textbf{(14)}$$

With:
$\theta_r(k)$: The angular velocity of the right wheel of the robot.
$\theta_l(k)$: The angular velocity of the left wheel of the robot.

8.1.4 The change in theta

$$\Delta\theta(k) = \frac{1}{2*L}\left(\theta_r(k) - \theta_l(k)\right) \tag{15}$$

Where:
L: is the distance between the right wheel and left wheel.

9 Algorithm Q-Learning with Artificial Neural Networks

```
(1)-Initialize random weights W of the neural network;

(2) Give the initial position of the robot [($X_r(0)$,
$Y_r(0)$, $\theta_r(0)$)];

**For** k = 1 to iteration k

(3)-Reading the $S_t$ state of the environment by the seven
sensors (L, A, R);

(4) For a fixed state, calculation of the Q-function by
the neural network for the three possible actions (TL,
Ar, TR);

(5)-Determination of the optimal action is action which
is the maximum value of Q (optimal_Action $\rightarrow$ Max (Q));

(6)-Run the optimal with probability **ε** ;or random action
with probability 1-**ε**

(7) Reading the new state $S_{t+1}$ of the environment
through sensors the Seven (G, A, D);

(8)-Reinforcement;

(9)-Test of Reinforcement

    If r =- 1 (there is an obstacle)

          (1) - Update weights and biases of the neural
network with the formula of Q-Learning.

          (2) - Return to the original state.
```

```
        End if
    (10) - $\varepsilon = \varepsilon * 0.99$ to decrease gradually;
End For
```

10 Simulation results

The proposed algorithm is implemented for a simulation for a mobile robot in a scene or two obstacles are deferred. The use of the algorithm Q-Learning with Artificial Neural Networks for MLP-type obstacle avoidance is the major objective of our simulation.

The generation of the Q-function is based on the use of Artificial Neural Network-type MLP, in which learning takes place by the algorithm of back-propagation. The Artificial Neural Networks is characterized by ten neurons as input cells where seven neurons present state of the environment and the three other neurons present possible actions. The values of its input neurons are binary types (0, 1), a hidden layer which contains seven neurons whose activation function is the sigmoid function, and the output layer contains one neuron with activation function sigmoid. The adjustment of weights and biases of this neurons network, that fact in a collision of the robot with an obstacle, as it returns to its original state. The action performed by the robot is based on the use of the greedy method. The state of the environment is obtained by the simulation of a proximity sensor, placed on three sides of the robot. Fig. 7 shows the environment evolution of the robot, where the reinforcement is given by a scalar signal which is set to -1 when the robot hits an obstacle, the robot 1 when straight ahead and 0 in all other cases.

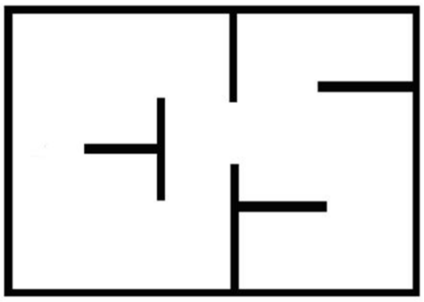
Fig. 7. Evolution Environment of the robot.

The trajectory followed by the robot during the learning stage is presented in Fig. 8 for 2500 iterations. After learning the trajectory of the robot is presented in Fig. 9 for 2500 iterations.

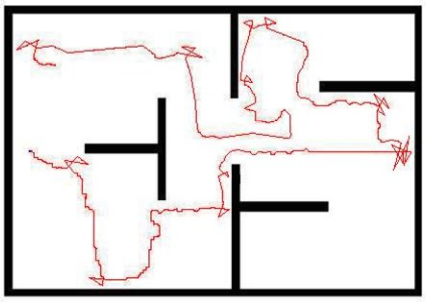

Fig. 8. The trajectory followed by the robot during the learning stage for 2500 iterations.

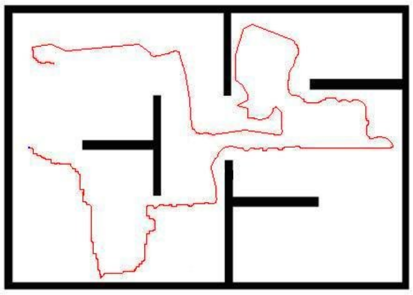

Fig. 9. The trajectory of the robot after learning for 2500 iterations.

To test the ability of our artificial neural network proposed in Fig. 10 shows a change of environment, where the robot moves through its environment while avoiding obstacles.

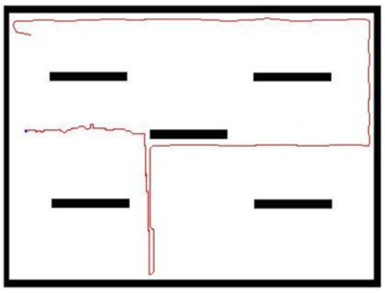

Fig. 10. Change of environment.

Conclusion

In this paper we presented the technique of reinforcement learning where we have chosen the Q-Learning algorithm by using artificial neural networks for the generation of Q-function this approach has been used for navigation of a mobile robot in an unknown environment while avoiding obstacles, the results are very satisfactory, and meet the target. This allows us to say that this approach can are used for the navigation of mobile robot in an unknown environment.

References

[1] J. Barraquand and J.C. Latombe, "Robot Motion Planning: A Distributed Representation Approach", The Int. Jour. of Robotics Research, Vol. 10, No. 6, 628-649 (1991).
[2] Hamid Boubertakh, Mohamed Tadjine, Pierre-Yves Glorennec," A Simple Goal Seeking Navigation Method for a Mobile Robot Using Human Sense, Fuzzy Logic and Reinforcement Learning", KES (1) 2008: 666-673
[3] S.T. Li and Y. C. Li, "Neuro Fuzzy Behavior-Based Control of a Mobile Robot in an Unknown Environments", Proc. Of the 3rd Int. Conf. On Machine Learning and Cyb., Shangai, 26-29, August, 2004.
[4] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction, Bradford Books, MIT, 1998.
[5] Bing-Oiang Huang. Guang- Yicao.Min Guo. Reinforcement Learning Neural Network to the Problem of Autonomous Mobile Robot Obstacle Avoidance. Août 2005
[6] F. Abdessemed, K. Benmahammed and E. Monacelli, " A Fuzzy-Based Reactive Controller For Non-holonomic Mobile Robot", Journal of Robotics and Autonomous Systems, 47 (2004) 31-46
[7] Watkins J. C. H, Learning from Delayed Rewards, PhD Thesis, University of combridge, England. 1989.
[8] Takanori Fukao, Takaaki Sumitomo, Norikatsu Ineyama and Norihiko Adachi. Departement of Aapplied Systems Science, Graduate School of Engineering, Kyoto Uuniversity.1998. Q-Learning Based on Regularization Theory to Treat the Continuous States and Actions.
[9] Claude F. Touzet. Q-Learning for Robots. 1997.
[10] Kristijan Macek, Ivan Petrovic and Nedjelko Peric. University of Zagreb A reiforcement learning approach to obstacle avoidance of mobile robots.
[11] M. Carreras. P. Ridao and El- Fakdi. Institute of Informatics and Aapplications. University of Girona. Spain Semi-Online Neuronal Q-Learning for real Time Robot Learning.
[12] Pierre Yves Glorennec Département d'informatique INSA de Rennes / IRISA 1999. Algorithmes d'apprentissage pour les systèmes d'inférence floue.