Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005

REINFORCEMENT LEARNING NEURAL NETWORK TO THE PROBLEM OF AUTONOMOUS MOBILE ROBOT OBSTACLE AVOIDANCE

BING-QIANG HUANG[1], GUANG-YI CAO[1], MIN GUO[2]

[1] Department of Automation, Shanghai Jiaotong University, Shanghai 200030, China
[2] Department of Net Building, Jiaxing Telecom, Jiaxing, 314000, China
E-MAIL: bingqiang@sjtu.edu.cn

Abstract:

An approach to the problem of autonomous mobile robot obstacle avoidance using reinforcement learning neural network is proposed in this paper. Q-learning is one kind of reinforcement learning method that is similar to dynamic programming and the neural network has a powerful ability to store the Q values. We integrate these two methods with the aim to ensure autonomous robot behavior in complicated unpredictable environment. The simulation results show that the simulated robot using the reinforcement learning neural network can enhance its learning ability obviously and can finish the given task in a complex environment.

Keywords:

Reinforcement learning; Reinforcement learning neural network; Obstacle avoidance

1. Introduction

Research on autonomous mobile robots aims at both understanding intelligent behavior and building systems that exhibit intelligent behavior [1][2]. Obstacle avoidance is a fundamental task of autonomous mobile robots. The autonomous mobile robot must be able to adapt its skills in order to react adequately in complex, unknown and dynamic environments. A good method for achieving this goal is reinforcement learning (RL) because a complete knowledge of the environment is not necessary and also permits the robot to learn online.

Reinforcement learning is a term borrowed from the study of animal learning [3], and is also related to theory of learning automata in control engineering [4][5]. The reinforcement learning can be used to learn the unknown desired outputs by providing autonomous mobile robots with suitable evaluation of their performances and also can be used to realize the robots online learning. Online learning and adaptation are desirable traits for any robot learning algorithm operating in changing and unstructured environments where the robot explores its environment to collect sufficient samples of the necessary experience [6]. In a complex environment, it is required perform online learning through interact with the real environment. In our work, we will focus on learning the obstacle avoidance behavior. This is an example of a behavior that would receive reinforcement when used in autonomous mobile robot.

This paper reports on the development of a neural network-based reinforcement learning architecture, and its application to the learning of robotic obstacle avoidance. The paper is organized as follows. The essential concept behind reinforcement learning technique is first discussed. This is followed by a description of the reinforcement learning neural network (RLNN) architecture. The problem of obstacle avoidance task based on the RLNN is introduced in detail in Section 3. The simulation and its result are discussed in Section 4, and the conclusions are outlined in Section5.

2. Reinforcement learning

Reinforcement learning is a learning technique based on trial and error. The basic idea behind reinforcement is that the learning system shown in Figure 1 can learn how to solve a complex task through repeated interaction with the environment.

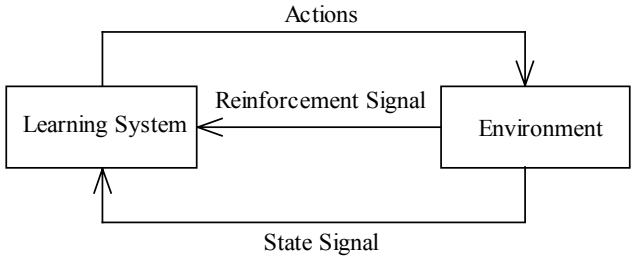


Figure 1. Reinforcement learning system

The learning system receives state information about

the environment by means of its sensors, and this state information is used by a reasoning process to determine the action to be given in the given state. After the action is implemented, the learning system receives a reinforcement signal from the environment to indicate the consequences of its actions. The goal of the learning is to maxmize the amount of reward it receives in the long term [7].

Watkins introduced the method of reinforcement learning called Q-learning in 1989 [8]. It is a reinforcement learning algorithm that attempts to learn a the state-action value $Q(x,a)$,whose value is the maximum discounted reward that can be achieved by starting in state x , taking an action a , and following the optimal policy thereafter. The action space is discrete and a separate $Q(x,a)$ value exists for each action a . We implement the Q-learning with a neural network to solve the obstacle avoidance prolem of autonomous mobile robot.

Each time, the agent takes an action a from state x at time t , the current state-action pair value estimate from a and x donated by $Q_t(x,a)$ is updated as follow:

$$Q_{t+1}(x,a) \leftarrow Q_t(x,a) + \alpha[r_{t+1} + \gamma \max_{b \in A} Q_t(y,b) - Q_t(x,a)] \quad (1)$$

y is the actual next state. $0 < \gamma < 1$ is the discount factor and α is a learning rate. A is the set of possible actions and r_{t+1} is the payoff the agent receives when action a is taken in state x . The state-action value estimates for other states and actions remain unchanged.

3. Reinforcement learning neural network

3.1. Reinforcement learning neural network architecture

To handle the large number of state-action pairs in autonomous mobile robot obstacle avoidance problem, we implemented Q-learning using a neural network to store Q . Figure 2 shows the structure of the robot learning system with a neural network. The neural network gives a more compact representation of Q than a lookup table and also allows us to interpolate $Q(x,a)$ for state-action pairs that have not been visited [9].

At no point do we know the true function Q—our neural networks only estimate Q . Because this estimate feeds into the update for the neural network, our updates are noisy. Given more and more interactions with the world, the estimate will converge to the true value and this noise will be eliminated. To speed this process, we record our experiences and at each iteration retrain on them, using the new value of Q to generate our training update.

Here we use a three-layer Back-Propagation network. Figure 3 shows the Q-learning with BP neural network. It has n neurons in input layer, h neurons in hidden layer and m neurons in output layer [10]. Every output corresponds with an action Q value. The output units of the network have a linear activation function to be able to produce Q-values of arbitrary magnitude. For the hidden units, the sigmoid function is used. In this study, we used the error back propagation algorithm to train the network.

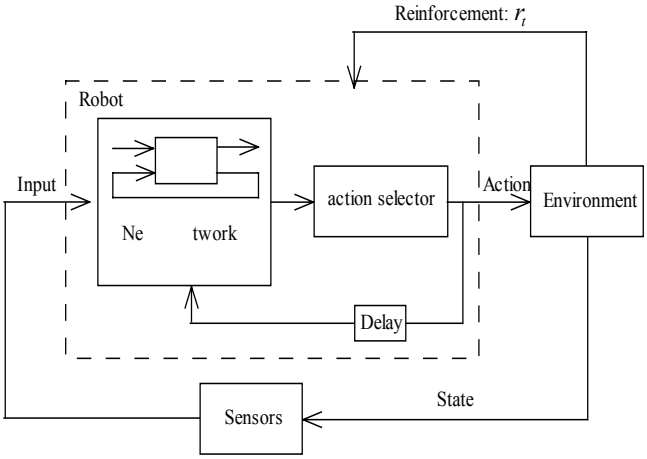


Figure 2. Structure of the robot learning system based on the neural network

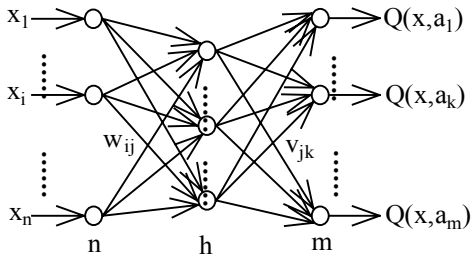


Figure 3. Q-learning with BP neural network

3.2. Action Selection

We use greedy action selection. When $Q(x,a)$ has converged to the true state-action values $Q^*(x,a)$, then the greedy policy that selects actions according to the following criterion is optimal:

$$a^*(x) = \arg\max_{b \in A} Q(x,b) \quad (2)$$

However, during learning, the agent has to explore by performing different actions. One way is select actions according to the Boltzmann probability distribution [11].

$$P(a \mid x) = \frac{e^{Q(x,a)/T}}{\sum_{b \in A} e^{Q(x,b)/T}} \quad (3)$$

Where T is a parameter that determines the probability of selecting non-greedy actions. It is slowly decreased during learning to make the policy greedier.

3.3. Learning procedure for the Q-learning algorithm

(1) Initialize the neural network.
(2) Initialize the autonomous mobile robot system.
(3) Get current state.
(4) Obtain $Q(x,a)$ for each action by substituting current state and action into the neural network.
(5) Determine an action according to equation: $action = \max(Q(x,a))$.
(6) Robot moves and gets current state.
(7) If fail (collision occurred), reinforcement = -1 and back to the start position again.
(8) Generate Q^{target} according to equation:

$$Q^{\text{target}}(\text{x,a}) = g(x,a,y) + \gamma \max_{b \in A} Q(y,b) \quad (4)$$

use Q^{target} to train the network as Figure 4 shown.
(9) Repeat 3-8, until the robot learns it.

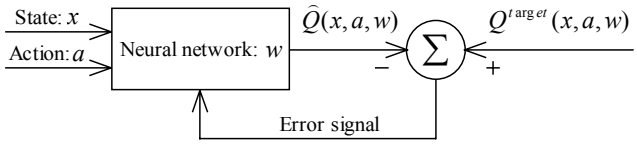
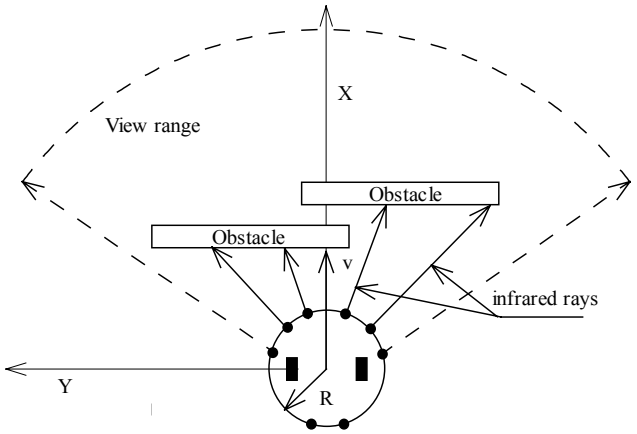


Figure 4. Neural network layout for approximating the target Q-value

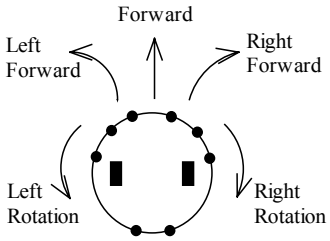
4. The simulated robot and simulation

4.1. The simulated robot and environment

The simulated robot has a cylinder-like shape as shown in Figure 5(a), similar to the well-known miniature Khepera [12]. It can observe the environment through eight infrared sensors marked as dots in the figure and the robot moves on two wheels. The infrared sensor determines the distance from obstacles, but its effective range is limited as shown by the dash line. R is the radius of the robot. The robot can act in five patterns: FORWARD, RIGHT FORWARD, LEFT FORWARD, RIGHT ROTATION and LEFT ROTATION (Figure 5(b)).



(a) Sensor Input



(b) Motor Output

Figure 5. The simulated robot

The obstacle avoidance problem is a task in which the robot wanders in an indoor environment without colliding with any obstacles in the absence of any prior knowledge about obstacle positions. Figure 6 is the environment we used which has many obstacles and four walls around. Thus, the robot has to use different resolutions in this environment.

As an evaluation of the required task, the reinforcement signal r is set as:

$$r = \begin{cases} -0.01 & for \quad rotation \quad action \\ 0.02 & for \quad forward \quad action \\ -1.00 & for \quad collision \end{cases} \quad (5)$$

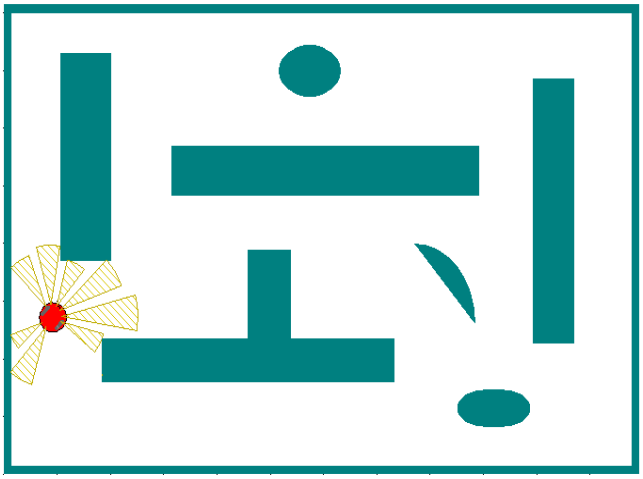
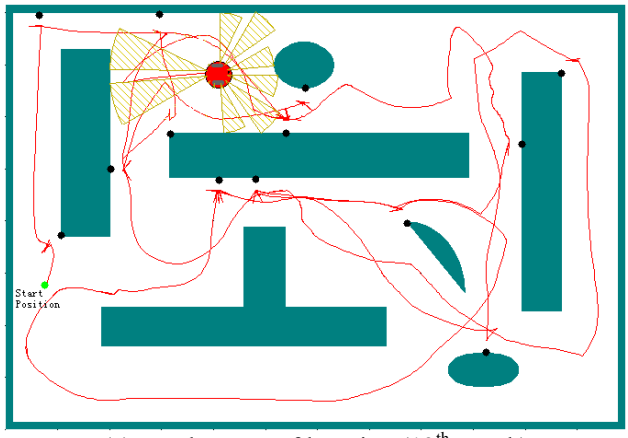


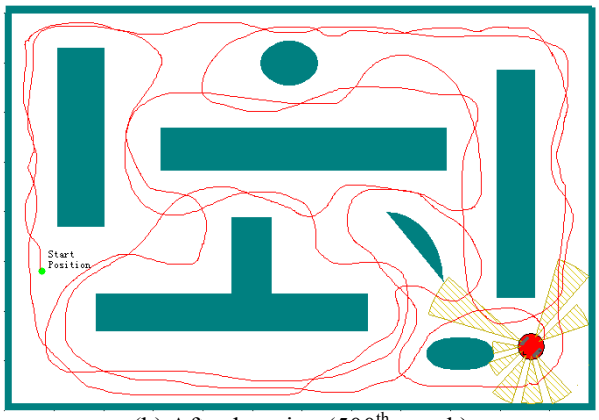
Figure 6. The environment for robot simulation

4.2. Simulation results

We let the robot move aimlessly in the environment so that it can learn the ability of obstacle avoidance. The robot geometrical parameters are set as follows: The robot diameter is 0.5m, the distance between two wheels is 0.35m, the wheel diameter is 0.2m, and the maxim distance that the infrared sensor could detect is 0.8m. We adopt a Back-Propagation network to store Q-value. The neural network has one linear input layer with eight nodes, one hidden sigmoid layer with sixteen nodes, and one linear output layer with five nodes. The net weight will be adjusted at once when the robot accomplishes an action. The robot performs learning for a once while occurring a collision. The robot returns to the initiative position when the collision occurs and adjusts the net weight again based on the former learning result. The robot trajectories in the early stages of learning and after learning are shown in Figure 7. Figure 7(a) shows the longest trajectory of the robot in each of the first 13 epochs. The collision positions with walls and obstacles are denoted by ●. Figure 7(b) shows the trajectory of the robot that learned 500 epochs. One epoch includes 1000 steps, and one step consists of one observation-action pair. From the result, we can see that the trajectory in the early stage is not smooth and it becomes smooth after long time learning. At last, the robot could move in the environment without undergoing any collision.

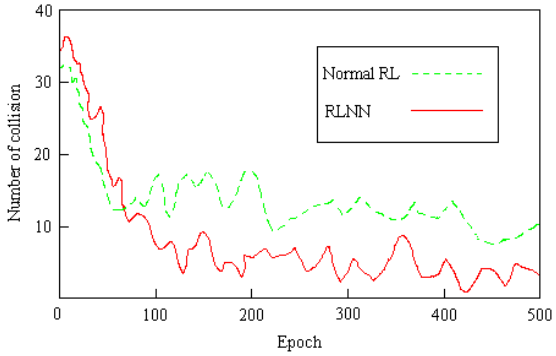


(a) Early stage of learning (13th epoch)

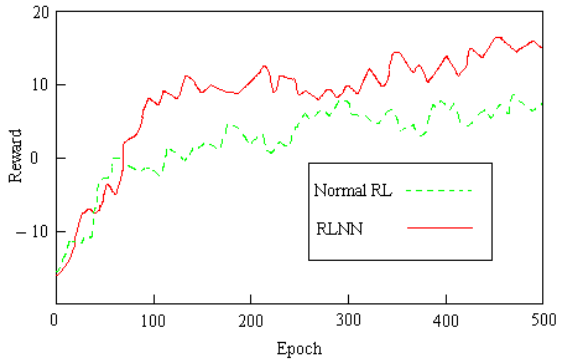


(b) After learning (500th epoch)

Figure 7. Trajectories of the robot learning



(a) Number of collision of the robot with obstacles



(b) Total reward in each epoch

Figure 8. A comparison with normal RL algorithm and RLNN algorithm

The comparison between normal RL and RLNN learning is shown in Figure 8. Figure 8 (a) shows the number of collision of the robot with walls and obstacles in each learning epoch. Figure 8 (b) shows the total reward obtained by the robot in each epoch. We can conclude that reinforcement learning neural network is better than the algorithm that adopts the normal reinforcement learning and its learning effect could be improved obviously.

5. Summary and conclusions

In this paper we have reported the reinforcement learning neural network application to the problems of in door autonomous robot obstacle avoidance. The result shows that a mobile robot that is learned using reinforcement is able to wander freely in a useful or "reasonably" manner and its learning rate is increased rapidly. The robot has an obvious ability of obstacle avoidance using this RLNN algorithm and this method can also be applied to the other automata systems.

References

[1] Pfeifer.R, Scheier.C. Understanding Intelligence. MIT Press, Cambridge, MA, 1999.

[2] Lichtensteiger.L, Salomon.R. The evolution of an artificial compound eye by using adaptive hardware. In: Processing of the 2000 congress on evolutionary computation, La Jolla Marriott, San Diego, CA, USA, 1144-1151, 16-19 July 2000.

[3] Mendel.J.M, Mclaren.R.W. Reinforcement-learning control and pattern recognition systems. In adaptive learning and pattern recognition systems: Theory and Application, eds J.M.Mendel, K.S.Fu, Academic Press, New York, NY. 287-318, 1970.

[4] Narendra.K, Thathachar.M.A.L. Learning automation: An Introduction. Prentice-Hall, Englewood Cliffs, NJ. 1989.

[5] Sameer.M.P, Devendra.P.G. Fuzzy-logic-based reinforcement learning of admittance control for autonomous robotic manufacturing. Engineering Application of Artificial Intelligence, Vol.11, 7-23, 1998.

[6] Hagras.H, Callaghan.V, Colley.M. Learning and adaptation of an intelligent mobile robot navigator operating in unstructured environment based on a novel online Fuzzy-Genetic system. Fuzzy sets and systems. Vol.141, 107-160, 2004.

[7] Boada.M.J.L, Barber.R, Salichs.M.A, Visual approach skill for a mobile robot using learning and fusion of simple skills, Robotics and Autonomous Systems, Vol.38, 157-170, 2002.

[8] Watkins.J.C.H, Dayan.P. Q-learning. Machine Learning, Vol.8: 279-292, 1992.

[9] Onat.A. Q-learning with recurrent neural networks as a controller for the inverted pendulum problem. The Fifth International Conference on Neural Information Processing, October 21-23, 837-840, 1998.

[10] Haykin.S. Neural networks. Second edition. 603-631, 1998.

[11] Cervera.E, A.P.del.Pobil. Sensor-based learning for practical planning of fine motions in robotics. Information Sciences. Vol.145, 147-168, 2002.

[12] Olive.M (1996),Khepera Simulator Package(version 2.0), Freeware mobile robot simulator download from the WWW at: http://diwww.epfl.ch/lami/team/michel/khep-sim