Proceedings of the IEEE
International Conference on Automation and Logistics
August 18 - 21, 2007, Jinan, China

Mobile Robot Path Planning Based on Q-ANN

Hairong Xiao

Department of Information Engineering
Shandong Jiaotong University
JiaoXiaoLu 23# 250023 Jinan, Shandong Province, China
hairong.xiao@163.com

Li Liao and Fengyu Zhou

School of Control Science and Engineering
Shandong University
Jinan, Shandong Province, China
zhoufengyu@sdu.edu.cn

Abstract - **Path planning is a difficult part of the navigation task for the mobile robot under dynamic and unknown environment. It needs to solve a mapping relationship between the sensing space and the action space. The relationship can be achieved through different ways. But it is difficult to be expressed by an accurate equation. This paper uses multi-layer feed forward artificial neural network (ANN) to construct a path-planning controller by its powerful nonlinear functional approximation. Then the path planning task is simplified to a classified problem which are five state-action mapping relationship. One reinforcement learning method, Q-learning, is used to collect training samples for the ANN controller. At last the trained controller runs in the simulation environment and retrains itself furthermore combining the reinforcement signal during the interaction with the environment. Strategy based on the Combination of ANN and Q-learning, Q-ANN, is better than using only one of the two methods. The simulation result also shows that the strategy can find the optimal path than using Q-learning only.**

Index Terms - ANN, Q-learning, Q-ANN, path planning, mobile robot

I. Introduction

The goal of robotics is to design and build machines capable of executing complex tasks in a robust manner and minimal human supervision. The robot navigation issue is that the robot can choose the correct action and accomplish the task of reaching the destination without collision, according to the environment information perceived by the sensor of robot. Path-planning problem is a fundamental task of autonomous mobile robots and proves to be difficult to solve using a machine-learning approach beforehand in an unknown and dynamic environment, and, as such, much of the work in the past few decades concentrates on the use of artificial intelligence (AI) techniques because the dynamics of the environment is unknown. Consequently, the autonomous mobile robot must be able to adapt its skills in order to react adequately in complex, unknown and dynamic environments [1]. This indicates the robot should accumulate behaviour knowledge through the life-time learning [2]. Therefore, various leaning methods have been proposed so far for the mobile robot, such as fuzzy logic [3], evolutionary algorithm [4], ANN [5] and reinforcement learning algorithm [6]. Because ANN has the good generalization performance and can approximate any functions in any accuracy, it is natural that valuing function-approximation based on ANN is one of the effective approaches for solving the problem of nonlinear mapping. Reinforcement learning (RL) is different from ANN supervised learning. In reinforcement learning, agent perceives states of environment, then, learns the optimal mapping from the state to action. RL does not need teacher signal to guide action. It is able to evaluate the action through the reinforcement signal provided by environment. Therefore, the RL is suitable for the application of the control of the intelligent robot. But it has some drawbacks, such as the problems of the delayed reward and temporal credit assignment. With the integration of ANN and RL, the system may achieve the advantage of both without many of the limitations of either.

This paper utilizes multi-layer feed forward ANN to construct a path-planning controller for the mobile robot. The task of planning is simplified to a classified problem which are five state-action mapping relationship，namely running forward, turning left, turning right, rotating 180^0 and stopping. The feed forward ANN structure needs some training samples because it is a supervised learning method. This paper uses Q-learning, a reinforcement learning method, to collect the training samples. Because the stopping action needs only a simple condition, when the distance between the robot and the target is shorter than some threshold, whatever other conditions are, we all conceive that the robot has achieved the task. So this action does not need Q-learning. Only four groups of state-actions left to be learned. At first begin Q-learning for the task of path planning. Utilize four Q lookup-tables to record the predicting Q values of the state-action pairs respectively. When the Q values come to convergence, stop Q-learning and begin the second phase, ANN training. Use values of Q table as the training data. In Q-learning procedure there are some hidden states and it is impossible to visit all of the state-action pairs, so using Q-learning simply can't always find the optimal strategy. And ANN learning happens to have the ability to compensate the limitation due to its generalization. First train the connection weights of ANN using the sample data from Q tables. Then，trains ANN by itself farther based on the reinforcement signals. At last bring the learned ANN controller to the path planning. Because the actions designed are basically motions of motors and can be executed directly, there are no coordination problem existed in the behaviour-based robots. Furthermore the motor can execute only one action at every moment so at each time only one output action of ANN is available. This changes the learning task to a classified problem. Then deduce the learning space and increase the learning speed of convergence.

II. Experimental Platform

The platform on which we undertake our navigation experiments is a mobile robot named Pioneer 3. It adopts the popular design of a semicircular platform with differential steering. To provide information about the robot's environment a passive sonar array is chosen. The system consists of eight ultrasonic sensors which form a sonar array mounted in a semicircle on the exterior of the robot's body. This sensing system is inexpensive but has the flexibility to allow reasonably complex navigation experiments. It forms a coarse one-dimensional facet eye with a 180^0 field of view in front of the robot. The backward action is not designed so need no back sensors and replace it with the action of 180^0 rotation. Distances between the robot and the obstacles are measured by the sensors and are labels as $d_{roi}(i = 0,1,...7)$. The sonar array is shown in Fig.1.

The current position (x_r, y_r) of the robot is detected by a encoder. Given the target position (x_g, y_g), the distance between the target and the robot is $d_{rg} = \sqrt{(x_r - x_g)^2 + (y_r - y_g)^2}$, and the relative angle between them is $\theta_{rg} = \arg\tan\dfrac{y_r - y_g}{x_r - x_g}$. Using an action space to express the five executable discrete actions, labelled as $M=\{m_f, m_l, m_r, m_o, m_s)$. Then the task of path planning can be changed to find a directly mapping relationship, namely $\{d_{roi}, d_{rg}, \theta_{rg}\} \rightarrow M$. This relationship can also be seen as a classified problem. Divide the whole state-action mapping relationship into five categories. This can deduce the learning space evidently and increase the convergent speed. Conditions needed for the optimal path and other restrictions are realized by the reinforcement function.

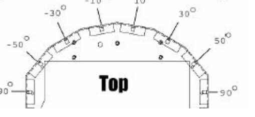

Fig. 1 Sonar array of Pioneer 3

III. Design The Structure Of ANN

ANN is an information processing approach simulating the nerve of the biology. The merit of it is that it can deal with the processes and systems that are difficult to be described by models or rules. ANN method owns a uniform description for the nonlinear systems. It has the powerful ability of information fusion and fault tolerance. This paper adopts a multi-layer feed forward ANN consisting of an input layer, a hidden layer, and an output layer. Because a structure owning single layer can only solve the linear classified problem, to solve the nonlinear classified matters, we have to use the multi-layer ANN structure.

There are no effective ways to determine the number of the hidden neurons. How to determine it is also an important problem. Too little hidden number can cause deficiency adaptation problem of the network, while too much can cause excessive adaptation. A two layer feed ward ANN can realize any nonlinear functional approximation if it has infinite hidden neural. But to the finite limited input there are no needs to have the infinite number. It is generally believed that the number is related with the requirement of the task, input number and the output number. According to the experience by many people, the number can be determined by formula (1).

$$n = \sqrt{n_i + n_o} + a \tag{1}$$

Where n is the hidden neuron number, n_i is the input neuron number and n_o is the output number. A is a Constant limited between one and ten.

The inputs for ANN are the readings coming from eight ultrasonic sensors, and the relatively distance and angle with regard to the current position of the robot which are counted at each sampling time, expressed with $\{d_{roi}(i = 0 \sim 7), d_{rg}, \theta_{rg}\}$. The outputs are the five discrete actions of the motors. So the ANN needs ten input neurons and five output neurons. According to formula (1) and across validation, the hidden layer selects eight neurons. The structure is shown in Fig.2.

Sigmoid function is used as the activation function for the hidden layer and denotes W_{ij} as the input synaptic weight values. At each time only one output is allowed to be 1 and others to be 0, so select competitive function as the output activation function. The training samples come from Q-learning. Extend the mapping relationship to the hidden states unvisited by Q-learning based on the good generalization performance of ANN.

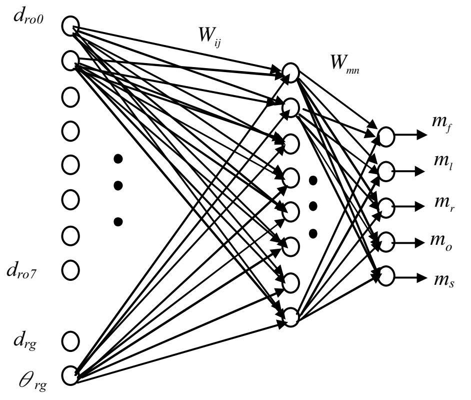

Fig. 2 Structure of multi-layer ANN

IV. Q-Learning

A. Q-learning procedure

Reinforcement learning is a learning technique based on trial and error. It 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. In a complex environment, it is required perform online learning through interaction with the real environment. In our work, we will focus on learning the path planning behaviour. Q-learning, one of the reinforcement learning algorithms has been widely used due to its simpleness and theory maturity.

Because the planning task has been partitioned into five action space, only four mapping relationship needs to be learned, namely $\{d_{roi}, d_{rg}, \theta_{rg}\} \to m_f$, $\{d_{roi}, d_{rg}, \theta_{rg}\} \to m_l$, $\{d_{roi}, d_{rg}, \theta_{rg}\} \to m_r$, $\{d_{roi}, d_{rg}, \theta_{rg}\} \to m_o$, while the stopping action mapping $\{d_{roi}, d_{rg}, \theta_{rg}\} \to m_s$ can be simply realized by one reason rule. The size of the learning space is determined by the number of the input variables and their quantified degree. When the sensors are settled, variables are quantified more finely, more states are required, and much bigger the learning space is. Here we only want to verify the feasibility of the algorithm. Furthermore the experimental robot is non-homonymic and the moving speed is low, so only roughly quantified degree can satisfy the path planning need. Two values, 0 and 1 which mean near and far are used to quantify the distances d_{roi} with regard to the robot. And the distance d_{rg} with regard to the target is quantified into three values, near, very near and far. Angle θ_{rg} with regard to the target is quantified into three values also, namely big, small and very small. Then the number of states needing to be learned are about 2000~2500. Only a *Q* lookup-table can store them. But when the values are quantified more finely, the number of the states can come to millions. So using a table can't satisfy the need. Other methods can be used to settle those problems [7]. Q-learning selects the optimal behaviour based on the experienced series actions for the robot in a Markov environment. Actually it is a transform behaviour of Markov decision processes. The robot observes the current state s_t, takes an action m_i, then comes to next state s_{t+1}, and gains a payoff r_t. The goal of Q-learning is to find the series state-action pairs which can make the *Q(s,m)* maximum. When a robot interacts with its environment, it can collect a data vector (s_t, m_i, r_t) at each time. A set of data vectors is collected after a learning trial is completed. The collected data is compressed into a lookup-table $\hat{Q}(s,m)$, in which each item is an estimated accumulative reward for the robot. $\hat{Q}(s,m)$ is the prediction value of *Q(s,m)* . The *Q* table can be updated during the learning when more trials are run. The *Q* values of all state-action pairs are updated by Bellman formula.

$$\hat{Q}_n(s_t, m_i) = (1-\alpha_n)\hat{Q}_{n-1}(s_t, m_i) + \alpha_n[r_{t+1} + \gamma \max_{m_i \in M}(s_{t+1}, m_i)] \quad (2)$$

Where α_n is the learning rate, γ is the discount rate between 0 and 1. In this paper we use the value 0.9. After the learning, the final optimized action m_i at the feature state s_t can be acquired by:

$$m_i(s_t)^* = \arg\max_{m_i} Q(s_t, m_i) \quad (3)$$

B. Reward function

In Q-learning, the rewards are the payoffs returned from environment to indicate the effect of robot's actions taken so far. The control algorithms are acquired automatically through learning guided by the rewards. Rewards can be singular values [8] to indicate if goals or sub-goals are achieved and usually can't be obtained immediately. These rewards are sparse in both temporal and spatial senses. For complex tasks, Q-learning would take long time to obtain sparse rewards. Selecting an appropriate reward can increase the learning speed.

The learning task of robot is to arrive at the goal in the shortest time and avoid the obstacles. Obstacles information are provided by ultrasonic sensors. And the target approaching degree is weighed by counting values of d_{rg} and θ_{rg} . Suppose the current state of robot is s_t , and R_t is the reinforcement signal from the environment at the state. After an action m_t gain the next state s_{t+1} . Design the reinforcement function R_t as follows:

$$R_t = -\sum_{i=0}^{7} w_i d_{roi} + w_8 d_{rg} + w_9 \theta_{rg} \quad (4)$$

Where $w_i(i = 0 \sim 7)$, w_8 and w_9 are weigh parameters. The values of weighs should reflex the influence degree with regard to the corresponding modular. Suppose at next state s_{t+1}, the corresponding payoff is R_{t+1}. Solve the difference as follows.

$$r_t = \Delta R_t = R_{t+1} - R_t \quad (5)$$

This paper selects r_t as the reward instead of R_t [8] in order to provide an immediately payoff from the environment. The signal can reflect whether the planning ability has been ameliorated after executing the current action. This can decrease the influence of the delayed reward or temporal credit assignment to some degree and increase the learning speed. At last input r_t to the Q-learning procedure.

V. Learning Of Ann

A. Supervised learning

In Q-learning phase, when the value of $\hat{Q}(s,m)$ does not change evidently, namely Q-learning is convergent, then stop the updating of $\hat{Q}(s,m)$. Feed the state-action pairs in the *Q* lookup-table to ANN and adjust the connection weights of it. The trained network determines a correspondence between the sensory input and the action pattern and outputs the action.

Sigmoid function is used as the activation function for the hidden layer. Learning algorithm adopts the steepest descent back propagation (SDBP). Only one output neuron is allowed to be 1 at each time, others have to be 0, so select competitive function as the activation function for the output layer. Connection weights between the input and hidden layer which are labelled as W_{ij} are updated as follows.

$$W_{ij}(k+1) = W_{ij}(k) - a\frac{\partial E(k)}{\partial W_{ij}(k)} \quad (6)$$

A is a learning rate, suppose the value is 0.01. $E(k)$ is the total error performance function, selected as the mean square error.

The learning algorithm of the connection weighs between the hidden layer and the output are updated according to the Kohonen learning rule. If the ith neuron wins, then the ith row elements of the connection weights vector are adjusted using formula (7). The weights are labelled as W_{mn}.

$$W_{mi}(k+1) = W_{mi}(k) + a[p(k) - W_{mi}(k-1)] \quad (7)$$

Where a is learning rate, $p(k)$ is the input vector of the output layer. By learning, those weight values which are nearest to the input vector are updated and make them to near the input vector much more. The result is that in the next time when input the similar vectors the victorious neuron have more chances to win. But to the weights which are far away from the input vector will have little and little probability to win. After more and more training, input modes which have the similar characters will make the corresponding neuron outputs 1, others output 0.

B. Retraining of ANN

Using the trained connections weights of ANN as the initial values and put them to the task of path planning for another training period. The weights will have a micro-regulating only in this phase. Evaluate the reinforcement signal from the environment after executing the output action.

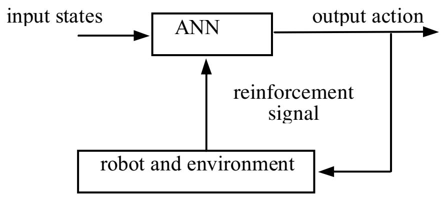

Fig. 3 Retraining procedure of ANN

If the signal R_t is positive, the current adjusted weights should be strengthened. Otherwise the values are weakened. In order to acquire the optimal control strategy, extend the mapping relationship to the hidden states and unvisited states in Q-learning based on the good generalization generation performance of ANN by training the neural network. The training sketch map is shown in Fig.3.

VI. Experiment And Simulation

Construct the simulation platform based on the Saphira/Aria agent structure provided by Pioneer 3. In the experiment, we assume that the autonomous mobile robot does not have knowledge of the obstacles in the working environment, such as their positions, numbers, and size. Give the start point and the goal randomly, begin Q-learning of the path planning for mobile robot in the simulation environment. Then, record *Q* data and the corresponding state-action pairs in the *Q* lookup-tables. In each environment, select 10 start point randomly and carry out 1000 groups of training and learning. The maximum training steps are 5000. When the robot can find the relatively optimal path without colliding with the obstacles and *Q* data have come to the convergence then stop Q-learning and begin the next learning phase. Input the state-action pairs in the *Q* table to the input layer of the designed ANN. After learning with the sampled data begin training ANN by itself with the reinforcement signal from interaction with the environment. At last bring the designed ANN controller of path planning to run in other environment. Contrast the trajectory planed by our strategy designed in this paper with using Q-learning only, we can find the former one can find the shorter path than the latter. This demonstrates that the method we design in this paper can find the optimal control strategy. The trajectories are shown in Fig.4 and Fig.5.

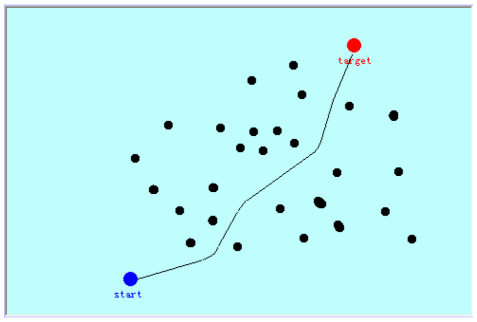

Fig. 4 Trajectory planed by the designed strategy

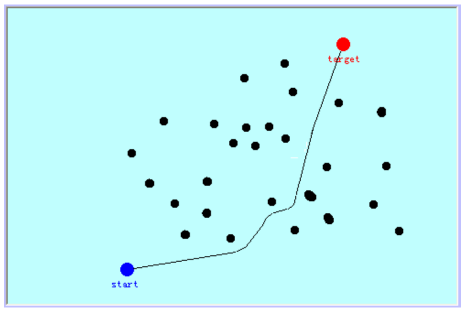

Fig. 5 Trajectory planed by the Q-learning only

VII. Conclusion

To make an autonomous mobile robot is the main research direction of the artificial intelligent. It requires the robot own learning and adaptive abilities. This can be achieved by various control methods. Each method has some merits and also shortcomings. Using only one method may not obtain better control results in some instances. Combination some of them can be likely to find optimal control strategies. In this paper we use ANN control method to construct a path planning controller under uncertainty environment. The training samples data are provided by Q-learning and the connection weights are updated in a roughly way. After this phase ANN trains itself in a minor manner on-line based on the reinforcement signal from the environment. Combining ANN with Q learning can overcome the disadvantages of low learning speed by only using ANN method and impossible abilities to learn all of states by Q-learning. The simulation results also demonstrate this. Integration of various control methods can provide a good resolve means for some intelligent control problems.

References

[1] Bing-qiang Huang, Guang-yi Cao, and Min Guo, "Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance," Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 pp. 85-89, August 2005.

[2] Naoyuki Kubota, "A spiking neural network for behavior learning of a mobile robot in a dynamic environment," 2004 IEEE International Conference on Systems, Man and Cybernetics, pp.5783-5788, 2004.

[3] Meng Wang, James N.K. Liu, "Fuzzy logic based robot path planning in unknown environment," Proceedings of the fourth international conference on machine learning and cybernetics, Guangzhou, 18-21, pp.813-818, August 2005.

[4] Lucidarme, Philippe, "An evolutionary algorithm for multi-robot unsupervised learning, " 2004 Congress on Evolutionary Computation, CEC2004, v 2, Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004, pp.2210-2215, 2004.

[5] Shou-tao Li, Yuan-Chun Li, "Neuro/Fuzzy behavior-based control of a mobile robot in unknown environments," Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, 26-29, pp.806-811, August 2004.

[6] I1 Hong Suh, Min Jo Kim, Sanghoon Lee and Byung Ju Yi, "A Novel Dynamic Priority-based Action-Selection-Mechanism Integrating a Reinforcement Learning," Proceedings of the 2004 IEEE International Conference on Robotics & Automation, New Orteans, LA, pp.2639-2646, April 2004..

[7] Yan-duo Zhang, Min Feng, "Application of reinforcement learning based on artificial neural network to robot soccer," Journal of Harbin Institute of Technology, Vol.36, No.7, pp859-861, July 2004.

[8] Lu Jun, Xu Li, and Zhou Xiao-ping, "Research on reinforcement learning and its application to mobile robot," Journal of Harbin Engineering University, Vol.25, No.2, pp.176-179, April 2004.