A Design Method of Fuzzy Logic Controller by Using Q Learning Algorithm

Xin Zhou*
College of Aerospace Science and Engineering,
National University of Defense Technology,
Changsha, 410073, P. R. China
inksci@outlook.com

Cai zhi Fan
College of Aerospace Science and Engineering,
National University of Defense Technology,
Changsha, 410073, P. R. China
caizhifan@nudt.edu.cn

Jun Wu
College of Aerospace Science and Engineering,
National University of Defense Technology,
Changsha, 410073, P. R. China
woshiwumingjun@163.com

ABSTRACT
As one of Reinforcement Learning (RL), Q learning algorithm has been applied in many fields of dynamic programming, and its Q Neural Network (Q-NN) can map the states of the environment to the corresponding control actions. Q learning algorithm can reach or exceed human level in some video games[1]. Nevertheless, the function of a neural network is similar to a black box. When the network fails in some control tasks, the adjustment of the network by the people is very difficult to achieve[2]. In contrast, due to the semantic control rules of the Fuzzy Logic Controller (FLC), incorrect rules can be artificially adjusted, however, a completely manual control rules design is a burdensome task. Therefore, in this paper, we propose a new method for the FLC to learn rules from well-trained neural networks and then fine-tune incorrect rules based on human knowledge. Finally, we experimented with navigational task and showed that the FLC we designed is able to accomplish the task, which achieved all of the reaching-target goals in 100,000 tests. (all the experiments results and relative codes are available on https://github.com/inksci/logic-controller).

Keywords
Q learning; fuzzy logic controller; neural networks; genetic algorithm; navigation.

1. INTRODUCTION
Reinforcement Learning (RL) can learn rules from a new environment independently without models and knowledge provided by human. In Q-learning, the Q neural network is trained over multiple learning cycles and the state action function $Q(s,a)$ is fitted using the following Bellman iteration rule:

$$Q(s,a) \leftarrow r + \gamma \max_{a'} Q(s',a') \ . \qquad (1.1)$$

In (1.1), s , a and r are the current state, action and reward respectively; s' and a' are the next state and action; γ is the discount parameter with a value between 0 and 1. The combination of deep learning and RL has been applied to play Atari games to achieve some excellent performance and reach the human level[1]. RL has shown great potential in control tasks, but robustness, as a key factor in the field of control, is hard to be guaranteed[3]. The role of the neural network is similar to a black box. We can train and use the neural network, but it's hard to fine-tune its rules[4].

In contrast, the fuzzy logic controller (FLC) uses fuzzy set to describe the environment with semantic description that can be understood by people, for example, describing distance as "close, near, far, or very far ". Its corresponding control rules are also semantic, such as "move to the left when the target is on the left" and Table 1 illustrates a type of FLC in tabular form[5]. Since the FLC is transparent, the rules for fuzzy logic controllers can be artificially designed[6]. However, more complex control tasks require many more control rules. It is hard to design control rules based entirely on human knowledge, and errors will probably appear.

Table 1 A type of fuzzy logic controller.

Target	**Obstacle**			
	far left	**close left**	**close right**	**far right**
far left	left small	right very big	left very big	left big
close left	left small	right big	left big	left small
close right	right small	right big	left big	right small
far right	right big	right very big	left very big	right small

In this paper, we used the Q learning algorithm to learn the control task in the first place. After sufficient training of the Q neural network, we let the FLC learn the control rules from the trained Q neural network and then fine-tune the incorrect control rules in the FLC. Our experiments showed that the performance improvement of network is not obvious in the latter part of Q learning algorithm training. Eventually, Q-NN experienced a failure of about 1.93% after 250,000 learning cycles. Logic controllers that learn rules from the neural network can completely overcome the same navigation task after manual tuning.

The structure of this paper is as follows: section 2 explained the method for learning the rules of fuzzy logic controller from the well-trained network and gave the relevant algorithms; section 3 describes how to optimize the design of fuzzy sets by using genetic algorithm; in section 4, the simulation of the navigation task under the obstacle avoidance conditions further explained the application of this method and verified the effectiveness of the method. Finally, section 5 summarized the research results of this paper and analyzed the problems in the method and the future research work. Related symbol definitions:

s A state that used in the Q learning algorithm.

a An action, which not only can be used within the Q learning algorithm, but also can be used within the FLC.

A A fuzzy set (or a fuzzy state), which is used within the FLC.

π The value function for every combination of a fuzzy set A and an action a, and $\pi(A,a)$ is used within the FLC.

ϕ The function which maps every state s to a fuzzy set A.

$\arg\max_x f(x)$ Operation: choose a value of x that maximizes the value of f.

2. LEARN THE RULES FROM THE FULLY TRAINED NEURAL NETWORK

Reinforcement learning algorithms can learn the knowledge of environment autonomously and gain the ability to solve problems without the need to model the task[7]. Especially, the Q learning algorithm can adjust the parameters of the neural network through numerous action tempo. After sufficient training, the Q neural network (Q-NN) used can map each state to an optimal action accordingly; similarly, the FLC is also able to match each fuzzy set with an optimal action[8], as shown in Figure 1.

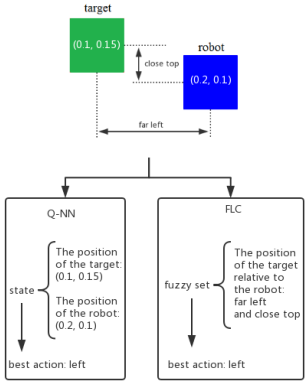

Figure 1 The comparison of Q-NN and FLC in the applying of control.

Fuzzy sets can be regarded as a combination of states obtained by clustering the states, that is, each state S can be classified into a fuzzy set A:

$$s \in A \tag{2.1}$$

Under the control of a neural network, each state is mapped to one action a:

$$s \to a \tag{2.2}$$

If the neural network is well-trained, the mapping rules, as shown is (2.2), are reasonable, so it can be deduced that in the fuzzy logic controller, there is the following mapping:

$$A \to a \tag{2.3}$$

If there are states s_1, s_2 and actions a_1, a_2, the following conditions are met:

$$s_1 \neq s_2 \tag{2.4}$$

$$a_1 \neq a_2 \tag{2.5}$$

The states s_1, s_2 belongs to the same fuzzy set A:

$$s_1, s_2 \in A \tag{2.6}$$

And based on neural network mapping rules:

$$s_1 \to a_1 \tag{2.7}$$

$$s_2 \to a_2 \tag{2.8}$$

For this situation, there will be two different actions a_1, a_2 for the same one fuzzy set A. However, the mapping rules in the fuzzy logic controller must be uniquely determined. In order to solve this problem, we define the function of FLC π and initialize it with zeros:

$$\pi(A,a) = 0 \quad \forall A, a\,. \tag{2.9}$$

The corresponding value in the function π is updated according to the following formula:

$$\pi(A,a_1) \leftarrow \pi(A,a_1) + 1 \tag{2.10}$$

$$\pi(A,a_2) \leftarrow \pi(A,a_2) + 1 \tag{2.11}$$

Finally, for each fuzzy set A, the fuzzy logic controller will select the action which can let the value of π be the largest one as the best action, and the corresponding control rules are as follows:

$$\arg\max_a \pi(A) \to a \tag{2.12}$$

In this way, we use a neural network to perform a task and constantly update the function π according to the corresponding mapping rules, resulting in an FLC that has almost the same function as the neural network. Not only that, in order to make the rules that FLC learned from neural network as correct as possible, in the actual algorithm, we initialize a state s_0 first. If the task can be completed under the control of neural network finally, reset the initial state with s_0 again; repeating the control process, and let the function π be updated, or otherwise initialize another new state. The whole algorithm for updating the function π is shown in Table 2.

Table 2. The algorithm used for learning rules from networks.

1.	**Initialize The Fuzzy Logic Controller** π :
2.	set π (set, action)=0 for all sets, actions
3.	**Learning The Control Rules from The Networks:**
4.	**for** n epsilons:
5.	state S ← reset environment
6.	**for** taking m steps:
7.	action ← networks(state)
8.	state ← taking one step by action

```
    end for
    if the agent reaching the target:
        reset environment with state S
        for taking m steps:
            state, action ← taking one step
            S' ← classify this state into fuzzy set
            π (S', action) += 1
        end for
    end if
end for
Map Sets into Actions using The Controller π :
action ← argmax( π (set))
```

3. DESIGN AND OPTIMIZATION OF THE FUZZY SETS

In this section, we mainly study how to classify the states into corresponding fuzzy sets, that is, the design of fuzzy sets[9]. We can use function $\phi(s)$ to represent the mapping of states to fuzzy sets:

$$\phi(s) \to A \tag{3.1}$$

In this paper, the design workflow of the FLC is shown in Figure 2 (left). The quality of the fuzzy set will affect the final performance of the FLC. In order to make the performance of the controller good enough, the parameters of the function $\phi(s)$ can be constantly adjusted so as to continuously improve the performance of the controller. This is a non-linear optimization problem that can be solved using Genetic Algorithm (GA). The performance of the controller is defined as the objective function. With the evolution of the "population", we can finally make a well design of the fuzzy sets. The corresponding optimization process is shown in Figure 2 (right).

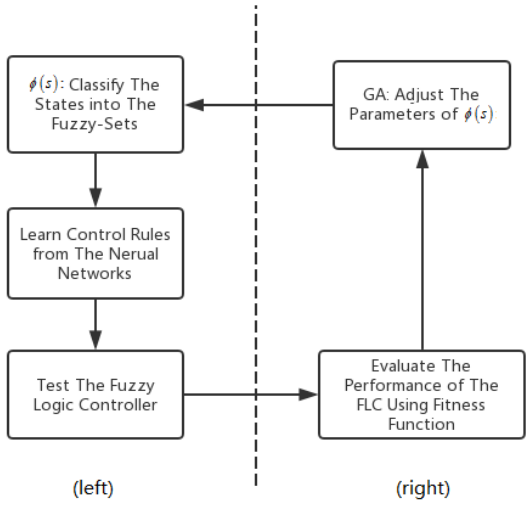

Figure 2 The design of fuzzy-sets using genetic algorithm.

In the experimental section, we will show how to encode the parameters of $\phi(s)$ accordingly. The definition of the objective function is given here: In the FLC performance test, let the FLC control the task with 1000 rounds and get the final success rate. The success rate increases with the number of learning from the neural network, and after a considerable number of learning, the increase in success rate becomes slow, i.e., tends to be saturated. It can be considered that when the rate of increase of the success rate is below a certain threshold, it is deemed to be saturated, and the success rate multiplied by a proportional constant can be used as the value of the corresponding objective function.

4. EXPERIMENTS

4.1 Simulation Environment

In order to validate the method proposed in this paper, we design a navigation task, as shown in Figure 3, where the obstacle is need to be avoided.

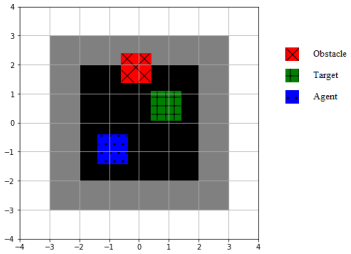

Figure 3 The navigation task used for experiments.

As shown in Figure 3, the obstacle, target, and agent are all squares with size of 1×1. When the environment is reset, the center positions of the obstacle, target and the agent are randomly set within a range of 4×4 and they are not in contact with each other. The scope of activity of an agent, that is, the center of an agent, is constrained to a range of 6×6. At each step, the agent can move upwards, downwards, to the right or to the left by a distance of 0.2, to reward 1.0 if the agent touches the target, to reward -1.0 if it collides with an obstacle, or else gets -0.1 reward. The number of steps of the Agent can move in each mission turn is a finite constant.

4.2 Gene Coding

We need to design a function that implements the mapping of states to fuzzy sets. For this navigation environment, we use 5 cut points, dividing each distance between agent and obstacle/target into 6 parts (namely closest, closer, close, far, farther and furthest respectively). Adding the information of the orientation of obstacle/target, we get the fuzzy sets as partly illustrated in Table 3.

Table 3 The description of fuzzy sets.

	Target		Obstacle	
	Orientation	**Distance**	**Orientation**	**Distance**
Description 1	left	furthest	above	further
Description 2	below	far	right	closest
⋮	⋮	⋮	⋮	⋮

Taking into account that the sizes of the obstacle, target and agent are all 1×1, we let the position of the first cut point x_1 range from 0-1.5, the second cut point x_2 range from x_1 to $x_1+1.5$, and so on. In this way, the length of the value range for each cut point is 1.5, and combine five gene fragments of the same length into one chromosome. Such chromosome is the equal description of the positions of the five cut points. In the end, we realized that we could express the function $\phi(s)$ by gene coding, and optimize the function $\phi(s)$ in the process of evolution through the optimization of the gene.

4.3 Fine-tune The Control Rules

Within the FLC, control rules have specific semantics, which can be understood by people[10]. When a control rule is unreasonable, it can be modified artificially. While first of all, you need to find out these incorrect rules. In this navigation task, the incorrect rules will lead to the collision of the agent with the obstacle, or let the agent produce the "push-pull behavior" as shown in Figure 4.

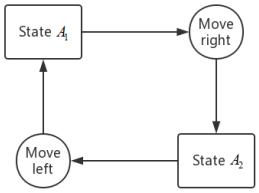

Figure 4 The "push-pull" behavior.

In Figure 4, when the fuzzy state is A_1, the agent generates a rightward movement, resulting in a state transition to A_2, according to the control rule, the state A_2 corresponds to an action that let the agent moves to the left, i.e., returns to the state A_1 again.

In each step, we let the corresponding state, action be printed out. When a collision occurs, it will cause the task to fail and exit the entire round. Obviously, the last control rule is wrong and the rule can be amended. When the "push and pull" occurs, the state and action will repeat every two times and ultimately the agent unable to achieve the goal until all the steps have been exhausted, so long as we just need make an analysis for the last two control rules.

4.4 Results

4.4.1 Comparison of Success Rates

We conducted three different experiments successively, as shown in Figure 5. In Experiment 1, Q learning algorithm was used to learn the navigation tasks and to adjust the parameters of the neural network through a lot of training. Finally, the neural network which has been fully trained (with 350,000 learning cycles) was used to control the navigation tasks. In Experiment 2, using the method described in section 2, let the fuzzy logic controller learn the rules from the neural network and then use the fuzzy logic controller to control the navigation tasks. In Experiment 3, the incorrect rules of the FLC used in Experiment 2 were fine-tuned by human knowledge, and the fine-tuning FLC was tested. The success rate of Experiment 1 is about 98%, indicating that neural network can learn the environment and has a good ability to solve the task. The neural network's training curve is shown in Figure 6, and after approximately 250,000 learning cycles, the average reward no longer increases significantly with the number of training sessions, but remains almost unchanged. In Experiment 3, a large number of tests for the FLC with fine-tuning are made, the results show that all tasks can be completed, proved the robustness of this controller.

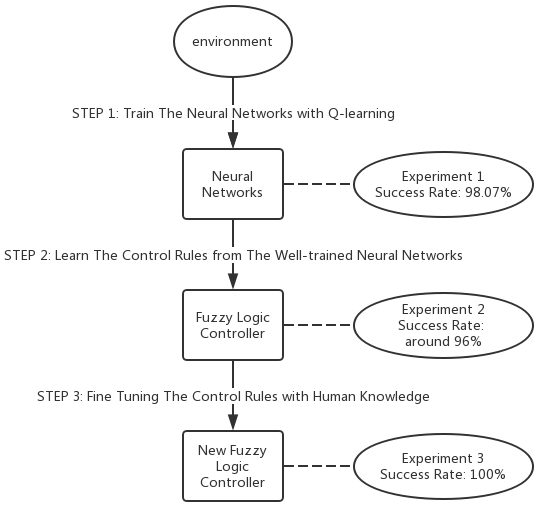

Figure 5 The success rate of 3 experiments.

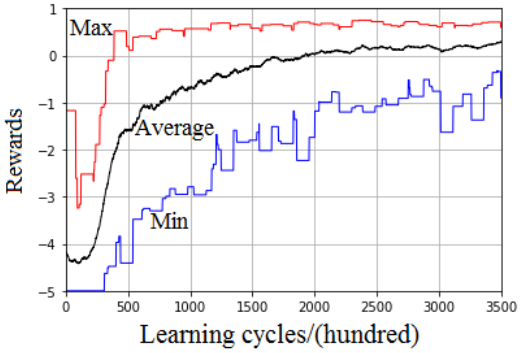

Figure 6 The learning curve of Q-learning.

In Figure 6, a "learning cycle" is defined as the process of an agent getting information from the environment and train its neural network start from an environmental reset to the next time when the environmental reset again.

4.4.2 Comparison of Control Performance

In Figure 7, (a) and (b) are the experimental results of using the Q-NN for control, and (c) and (d) are the results of using the FLC for control. The initial environment settings in (c) and (d) are the same as (a) and (b), respectively, that is, the initial positions of the agent, target and obstacle are the same. The experimental results show that both control methods can find a short path to avoid obstacle and finally reach the target. In contrast, the motion trajectories obtained by using the FLC have fewer turning points and show better smoothness control performance than the neural networks. The reason for this is that one fuzzy set corresponds to many states and maps to one action. Using fuzzy logic controller can make multiple different states adopt the same action. In neural network, each state

corresponds to its own action, and these actions may or may not be the same[11].

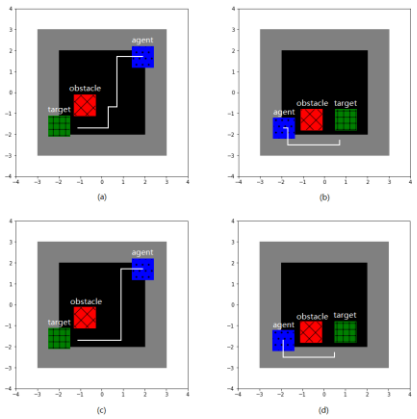

Figure 7 The performances of the FLC and the QNN.

5. CONCLUSION AND DISCUSSION

This paper presented a design method of fuzzy logic controller, which combined the advantages of the Q learning algorithm that can learn the control rules without relying on the task model and the advantages of the fuzzy logic controller that is transparent. Finally, a fuzzy logic controller is designed by using this method and applied to navigation tasks with obstacle avoidance. The 100% success rate of task completion in test is achieved through manual fine-tuning of the controller learned. In addition, the adjustment and optimization of fuzzy sets that based on genetic algorithm are also introduced. This shows that the method that realizes the design of the fuzzy logic controller using reinforcement learning algorithm can combine human knowledge with the autonomous exploration capabilities of machine learning, besides, it has unique potential in some areas of controlling.

In the Experiment 3 as described before, the fine-tuning FLC test results show that it can achieve navigation function with obstacle avoidance with 100% success rate when there is only one obstacle and one target. If there are two or more obstacles, we take only the nearest distance considered in each direction, as shown in Figure 8, so that the original FLC can adapt to the situation of multiple obstacles.

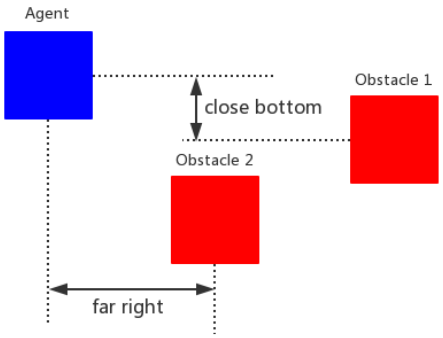

Figure 8 The situation with two obstacles.

For the case of two obstacles, the experimental results are shown in Figure 9. (A) and (b) are the cases where there is only one obstacle, and the two obstacles in (c) are the same as the obstacles in (a) (b) respectively. The FLC is still able to complete the task, showing some flexibility. However, it can be seen that the FLC failed in (f) although the FLC fulfilled its mission in the case of an obstacle (d) and (e). Because FLC can only grasp the surrounding local information, but cannot make a program on the global level, it is hard to overcome the problem of local minimum[12].

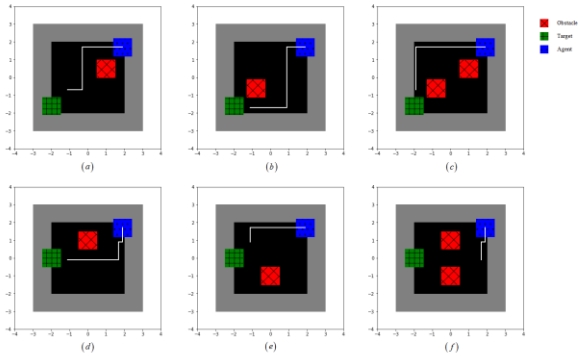

Figure 9 The experiments of the FLC for different numbers of obstacles.

For the navigation tasks in this paper, the fuzzy state of the FLC is represented by 4 relative orientations and 6 relative distances of the target and obstacle, corresponding to $(4\times6)^2 = 576$ kinds of fuzzy states and 576 control rules. When the fuzzy state has a larger dimension, more rules need to be stored. In this case, the classification neural network can be used to fit the control rules. The fuzzy state is used as the training sample and the corresponding action is used as the training label, so that less parameters are used to represent all the control rules.

6. REFERENCES

[1] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning.[J]. Nature, 2015, 518(7540):529.

[2] Althoefer K, Krekelberg B, Husmeier D, et al. Reinforcement learning in a rule-based navigator for robotic manipulators[J]. Neurocomputing, 2001, 37(1):51-70.

[3] Barto A G. Reinforcement learning[M]// Reinforcement Learning. Springer Berlin Heidelberg, 1998:665-685.

[4] Mitaim S, Kosko B. The shape of fuzzy sets in adaptive function approximation[J]. IEEE Transactions on Fuzzy Systems, 2001, 9(4):637-656.

[5] Jang J S R, Sun C T, Mizutani E. Neuro-Fuzzy and Soft Computing-A Computational Approach to Learning and Machine Intelligence [Book Review][J]. Automatic Control IEEE Transactions on, 2002, 42(10):1482-1484.

[6] Flores H, Srirama S. Adaptive code offloading for mobile cloud applications:exploiting fuzzy sets and evidence-based learning[C]// Proceeding of the fourth ACM workshop on Mobile cloud computing and services. ACM, 2013:9-16.

[7] Beom H R, Cho K S. A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning[J]. Systems Man & Cybernetics IEEE Transactions on, 1995, 25(3):464-477.

[8] Driankov D, Saffiotti A, Fahrzeugnavigation, et al. Fuzzy Logic Techniques for Autonomous Vehicle Navigation[J]. Studies in Fuzziness & Soft Computing, 2001, 61.

[9] Pradhan S K, Parhi D R, Panda A K. Fuzzy logic techniques for navigation of several mobile robots[J]. Applied Soft Computing, 2009, 9(1):290-304.

[10] Benyettou Y D A. Fuzzy Reinforcement Learning[J]. International Journal of Modern Physics C, 2002, 13(05):659-674.

[11] Juang C F, Lin J Y, Lin C T. Genetic reinforcement learning through symbiotic evolution for fuzzy controller design.[C]// IEEE International Conference on Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence. IEEE, 2000:1281-1285 vol.2.

[12] Ye C, Yung N H C, Wang D. A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance[J]. IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics A Publication of the IEEE Systems Man & Cybernetics Society, 2003, 33(1):17.

Columns on Last Page Should Be Made As Close As Possible to Equal Length

Authors' background

Your Name	Title*	Research Field	Personal website
Xin Zhou	master student	reinforcement learning and manipulator control	
Cai zhi Fan	associate professor	intelligent control and visual servo	
Jun Wu	lecturer	multi-agent reinforcement learning	

***This form helps us to understand your paper better, the form itself will not be published.**

***Title can be chosen from: master student, Phd candidate, assistant professor, lecture, senior lecture, associate professor, full professor**