Reinforcement Fuzzy Tree: A Method extracting Rules from Reinforcement Learning Models

1st Wenda Wu
Institute of Software, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, P.R. China
wenda2016@iscas.ac.cn

2nd Mingxue Liao
Institute of Software, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, P.R. China
mingxue@iscas.ac.cn

***Abstract*—Reinforcement learning methods are more and more popular in machine learning fields, and have achieved outstanding breakthroughs in various applications combined with deep convolution neural networks. Deep convolution neural networks have excellent fitting abilities and promote the prosperity of machine learning. However, we want to look inside the model and get some rules sometime, but it is a hard task for deep convolution neural networks and reinforcement learning methods now. Our reinforcement fuzzy tree model unites reinforcement learning methods, deep convolution neural networks and decision tree, could extract explicit rules and keep excellent performance at the same time. Our paper show that models based on reinforcement learning and deep convolution neural networks could be redundancy sometime, but the extracted rules could be simpler than the model we constructing and keep the same performances.**

***Index Terms*—Reinforcement Learning, Convolution Neural Network, Machine Learning, Decision Tree**

I. Introduction

Reinforcement Learning [21] method is inspired by the process that human learn something from the environment we living in. We learn knowledge by interacting with the environment, and adjust our own behaviors with the feedbacks of the environment. The feedbacks we received consists of information that would be helpful to our decisions and achieving our goals.

Mostly, we apply reinforcement learning method for getting highest rewards in an specific environment. There are some agents interacting with the environment by taking actions, and learning how to make the best decisions in different situations. Different from other methods, reinforcement learning methods do not rely on artificial labels but trialing, and get delayed rewards after several trial steps.

Reinforcement learning methods have made marvelous progress in recent years. There are many papers received by Nature, Deep Q-network [22], AlphaGo [19] and DNC [5] and so on. The methods and their extension have been used in many areas, including Bionic robots [11], Machine Translation [1], Autonomous vehicles [13] and so on. However, neither traditional reinforcement learning methods nor deep reinforcement learning methods could not present explicit rules that they have learned. They only tell you what actions you should take in certain situation but do not explain why. In this paper, we present a new model which could maintain the high performance and reveal the rules behind the working model simultaneously.

II. Related work

Reinforcement Learning researchers focused on traditional board games and video games to promote development of reinforcement learning methods.

Go game have attracted researchers all around the world and made excellent achievements. Maddison [12] constructed Go model to play against amateur players but it could not reach professional level. AlphaGo [19], an intelligent Go player which caused a hit, defeated Lee Sedol, an 18-time world champion Go player, with 4:0 in 2016. AlphaGo is constructed by deep convolution neural network [10], reinforcement learning, supervised learning [3] and Monte-Carlo tree search method [6]. It absorbed the experience of thousands of professional Go players and could calculate short-term rewards and long-term rewards of every single step it choosing.

AlphaGo Zero [20] had set an milestone in reinforcement learning development, which learned Go from tabula rasa to superhuman master. It took residual network [8] structure and combined policy network and value network together. AlphaGo Zero improved its own policy through self-play, and thrashed AlphaGo with 100:0. AlphaGo Zero never referred to human knowledge, and learned Go just from rules. Reinforcement learning method show its strong learning ability for complicated environment combined with deep convolution neural networks and Monte-Carlo tree search.

Video games is another platform for artificial intelligence training and testing. Atari [15] is an video games compilation platform including a bunch of classical games designed for human. DeepMind presented their paper [22] in 2015, which was on the cover of Nature. In this paper, DeepMind taught computer playing video games by catching the screen shots as the inputs of deep convolution neural network. DeepMind applied this model called DQN to 49 different games and the model performed better than human in more than half of games.

Based on DQN, there are several new models presented in recent years. Prioritized experience replay method [18] calculated the importance of experiences and replayed the

978-1-7281-0801-8/19/$31.00 ©2019 IEEE
ICIS 2019, June 17-19, 2019, China

most important experiences for training. Dueling-DQN [7] branched off the output of convolution layers into two different tributaries. The results of first tributary are state values and the other results are action advantages. Distributional perspective method on reinforcement learning [2] constructed model for value distribution rather than average value common used and surpassed DQN in stability and convergence. A3C [14] adopted Actor-Critic algorithm. Model made several couples of Actor and Critic, and trained them in different computer cores. Every Actor-Critic pair referred to the weights of other pairs and updated its own weights. Noisy networks [4] added parametric noise in its model and got higher scores in Atari games.

III. Methods

Although both reinforcement learning methods and deep reinforcement learning methods have effective performance in varied problems, they might encounter bottlenecks in practical applications. On one hand, either reinforcement learning tables or deep convolution neural networks are lack of interpretability to present what have happened during training, and we can not learn experience from artificial intelligence model directly in some area that models perform better than human; On the other hand, models may be stuck into enormous amount of parameters in some simple environments. We constructed a new model to jump out the traps above.

A. Game environment

We started our experiments in game environment. We set two forces(Red force and Blue force) in limited grid playground. Each force controls the same numbers of agents. These agents can move freely in playground. The state of a single agent is showed in Tab. 1.

TABLE I
THE STATE OF A SINGLE AGENT

HP(hit points)	Location	Ammunition
6	(1-map weight, 1-map height)	12

We located two blue-force agents in the center and two red-force agents in the corner. Blue-force agents can support each other because they could fire to the same enemy which are trapped into the common area of blue-force agents. Two red-force agents are far away from each other so they must learn to band together for winning. The basic layout is showed in Fig. 1.

We use a white grid map describes our playground. Red-force agents are presented by red squares and blue-force agents are presented by blue squares. Red-force agents turn to green and blue-force agents turns to black when they are destroyed. Below the white grid map is an information text block. It shows every agent's HP, action and location.

There are two types of reward systems we can choose in our game environment. First reward system does not pay attention

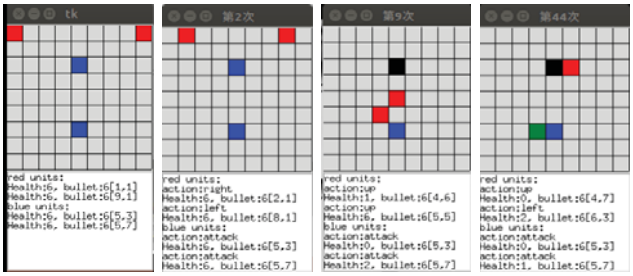

Fig. 1. Basic layout

to game process but the final results.The other reward system is more complicated. We want to encourage red-force agents to fire to enemies rather than escape from damages. The details of reward system are showed in Tab. 2 and Tab. 3. We found that reward system one is better than reward sysytem two during our experiments, so we adopt system two in our experiments when we constructed our new model.

TABLE II
THE DETAILS OF REWARD SYSTEM ONE

	Red agent lose HP	Blue agent lose HP	Red agent is destoryed	Blue agent is desto-ryed
reward	-0.2	0.31	-4	+6

TABLE III
THE DETAILS OF REWARD SYSTEM TWO

	Red Force	Blue Force	Reward
agents left	0	2	-3
	0	1	-2
	1	2	-1
	0	0	0
	1	1	0
	2	2	0
	2	1	1
	1	0	2
	2	0	3

B. Basic models

We constructed reinforcement learning models as the backbones of our model. Deep Q-network(DQN), which catch the image of current situation as input and traditional reinforcement learning, which create table for every possible situation are used in our game environment.

Different from standard DQN model, we construct state matrices as the inputs of DQN. Comparing to images of screen shots, state matrices are better input data which directly present current situation. State matrices have the same scale as our game playground which take less calculation power and do not need extra image preprocessing. We constructed deep convolution neural network with 3*3 convolution layers, max-pooling layers and full connected layers. We chose Relu [16] as our activate function and Adam [9] as our optimization

method. The outputs of our model are combinations of actions that agents choosing in the same force. The actions including move up, down, left and right. Our DQN model is showed in Figure 2.

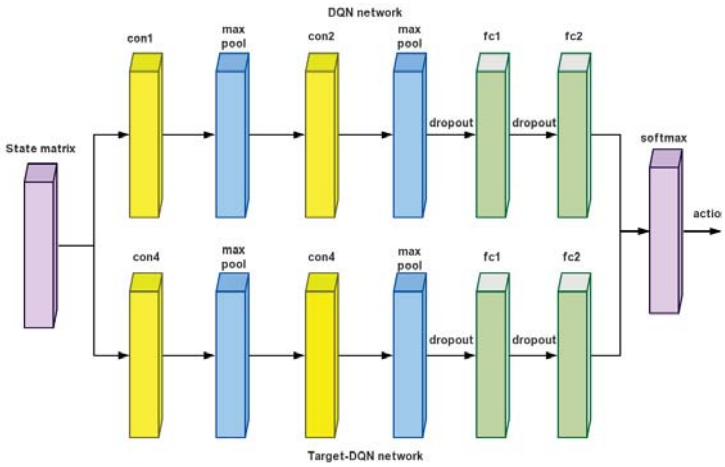

Fig. 2. Our DQN model structure

In the reinforcement learning method that creating Q table for every possible situation, model listed all crucial features that describe the situation including HP, location and so on. The Q table got random initial data when it was created and updated its data while gaming. Combinations of actions are popped as the output just like DQN mentioned above.

C. Data filter

The reason that reinforcement learning methods could learn about acting rules of complicated environment, from reward system, is that agents would trial all kinds of possible actions with different possibilities. There is one thing we should ask about–are all the trials agents taking are essential for modeling?

The answer is NO. Let us think about a simple question: what is the result of $1+1+1$. We could not use any other answers except 2 when we have calculated the first part $1+1$ in formulation $1+1+1$. In other words, there are many situations existing only in thoughts but realities, and we must believe that we should make decisions according to the possibilities.

Our models are also facing this problem. To solve this problem, we create a data filter giving up useless data. Reinforcement learning models set a mark in data that environment feeding back, marking the number of appearance of data. According to the characteristic of reinforcement learning methods, the appearance of a data which describing a specific situation, is more and more rely on the appearance possibility marked on this situation during the model running.

We filtered out the data which hardly appearing and got smaller-size, more effective and less-computation-needed data. The filtered data is fed into a tree structure which is more suitable than others since the data scale getting much smaller. Firstly, tree structure could present explicit rules in smaller scale data. Secondly, tree structure is more friendly for extracting rules than other structures.

D. Reinforcement Fuzzy Tree

Our reinforcement fuzzy tree included reinforcement learning part, data filter part and decision tree part. The whole model is showed in Fig. 3.

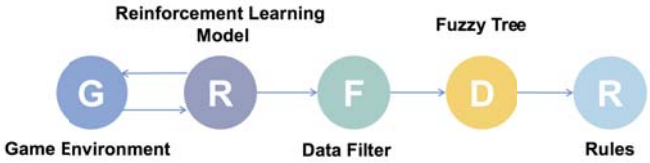

Fig. 3. The whole model structure

The first part major in analyzing the environment and collecting data of all possible situations. We applied DQN in complicated environment where Q-table would occupy enormous space, but Q-table reinforcement learning method in simpler environment where DQN would cost more time to train.

The data collected from reinforcement learning methods went through the data filter and threw off the unessential data. Here is a hyper-parameter needed for limiting the minimum appearance number of data.

Decision tree received the outputs of data filter and structured its branches and leaves. The branch nodes of decision tree are the features which would be used to split data. The leaf nodes are the actions agents finally choosing. We could read rules from the path that root node guiding to leaf node, and mark it down for knowledge graph [17] complement and knowledge base completion of human beings in the future.

IV. Experiments

Our experiments started from a static game scenario. In this scenario, there are two Blue force agents guarding in the centre of playground, and two Red force agents which are controlled by model. Red force agents would try to defeat all Blue force agents and save their own HP.

TABLE IV
SCORE POSSIBILITY IN $1 * 10^8$ GAMING

Red Force win	Red Force left	Blue Force left	Rounds	Possibility
No	0	2	302834	0.3
No	0	1	6342454	6.3
No	1	2	0	0
No	others	others	57511594	57.5
Yes	2	1	697	0
Yes	1	0	34146705	34.1
Yes	2	0	1695716	1.8

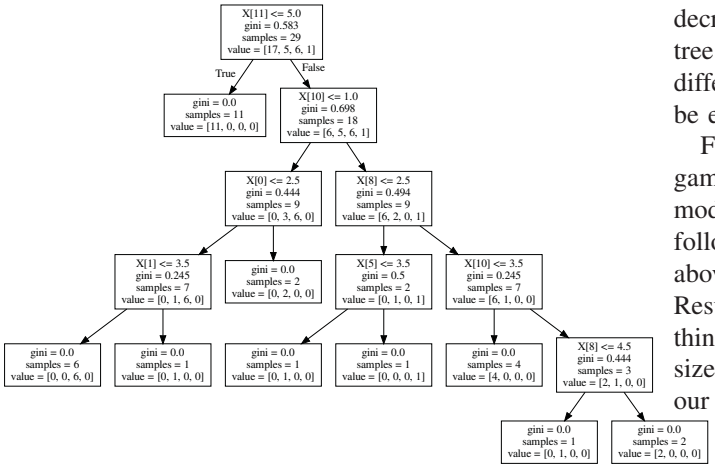

Fig. 4. Decision tree structure

We set reward rules and win-lose standard. We determine red-force is lost when two forces have the same number of agents left or the number of red-force agents left is less than Blue force has. It is showed in Tab. 4 and tell us Blue force agents get advantage over Red force agents.

Over one million turns training, Red force defeated Blue force easily using reinforcement learning method, and saved all own agents. In the training process, we set data pools for both DQN and Q-table based reinforcement learning method for data collection on game environment. The final data we get includes HP, location and ammunition of every agent.

In data filtering stage, we chose different thresholds to purify data. Decision tree accepted purified data and modified its construction. It is worth to be mentioned that, because of the decrease of data size, we used all data to train the decision tree instead of dividing data into training set and validation set. The different thresholds of data appearance time we had chosen and the performance of decision trees are showed in Tab 5. The decision tree is presented in Fig. 4.

TABLE V
EXPERIMENTS IN STATIC GAME SCENARIO

Thresholds of appearance	Data size	Percentage of original data(%)	Red force win	Score
No Thresholds	15752961	100	Yes	3
0	164130	1.0419	Yes	3
1	111960	0.7107	Yes	3
5	51397	0.3263	Yes	3
10	27563	0.1750	Yes	3
50	4249	0.0270	Yes	3
100	2301	0.0146	Yes	3
500	772	0.0049	Yes	3
1000	512	0.0033	Yes	3
5000	153	0.0010	Yes	3
10000	80	0.0005	Yes	3
50000	22	0.0002	Yes	1

As we can see from Tab. 5 and Fig. 3, data size is hugely decreased but has no influence on game results. Our decision tree part fuzzy input into several parts and divide data into different branches according to their feature values. Rules can be extracted from the paths from root node to leaf node.

Furthermore, we also took our experiments in dynamic game scenario. Our experiments had two turns. In turn one, model controlled Blue force agents, while Red force agents followed trained reinforcement learning model part mentioned above. Our model control Red force agents again in turn two. Results are showed in Tab. 6 after one million trainings. One thing should be mentioned here is that because of the huge size of data, we only recorded the data had appeared during our experiments in dynamic game scenario.

TABLE VI
EXPERIMENTS IN DYNAMIC GAME SCENARIO

	Thresholds of appearance	Data size	Percentage of original data(%)	Blue force win	Score
Turn One	0	513320	100	Yes	3
	1	221259	43.1035	Yes	3
	5	88086	17.1601	Yes	3
	10	48005	9.3519	Yes	3
	50	4087	0.7962	Yes	3
	100	1826	0.3557	Yes	3
	500	523	0.1019	Yes	3
	1000	409	0.0797	Yes	3
	5000	112	0.0218	Yes	3
	10000	48	0.0094	Yes	1
	50000	10	0.0019	No	0
Turn Two	0	1115402	100	Yes	3
	1	513158	46.0066	Yes	3
	5	206374	18.5022	Yes	3
	10	114698	10.2831	Yes	3
	50	14606	1.3095	Yes	3
	100	6534	0.5858	Yes	3
	500	1172	0.1051	Yes	3
	1000	691	0.0619	Yes	3
	5000	307	0.0275	Yes	1
	10000	220	0.0197	No	0
	50000	29	0.0025	No	0

It is clearly that even in dynamic game scenario, our model could find the best way to win and reduce the data size apparently.

Robust validation also be taken during our experiments. In this series of experiments, we changed the position information of agents in the game environment. We added 1 on X-axis and Y-axis of every agent and fed the changed data to our model. The performance of our model showed in Tab. 7 after one million trainings.

Our model performs effectively either in static game scenario or dynamic game scenario, and have excellent robust character. Explicit rules also could be extracted, which could be learned easily by human beings.

V. CONCLUSION

In this paper, we proposed a new model which could present explicit rules and keep excellent performance at the same time. We use reinforcement learning models as the backbone of our

TABLE VII
ROBUST VALIDATION RESULTS

Thresholds of appearance	Data size	Percentage of original data(%)	Red force win	Score
No Thresholds	15752961	100	Yes	3
0	164130	1.0419	Yes	3
1	111960	0.7107	Yes	3
5	51397	0.3263	Yes	1
10	27563	0.1750	Yes	1
50	4249	0.0270	Yes	1
100	2301	0.0146	Yes	1
500	772	0.0049	Yes	1
1000	512	0.0033	Yes	1
5000	153	0.0010	No	0
10000	80	0.0005	No	0
50000	22	0.0002	No	0

model and purify training data according to the times data appearing. The purified data is used to construct decision tree and represent the tree to the rule system furthermore.

During the static game scenario and dynamic game scenario testing, our model is proved to be effective and have excellent robust character. This new model provides an novel contribution to the black-box models, makes them easier understood by human beings and be helpful to construct expert system or design future models.

VI. FUTURE WORK

As we can see above, our reinforcement fuzzy tree is divided into three parts that each part has its own duties. Time would be cost during the data transmission process when the data is large enough. We would like to align the whole model into end-to-end mode in the future. It would be beneficial to space and calculation occupation.

Furthermore, we would like to consider extract rules directly from deep convolution neural network with the help of middle output of deep convolution neural networks. It would be challenging but exciting.

REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *Computer Science*, 2014.

[2] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. 2017.

[3] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *International Conference on Machine Learning*, 2006.

[4] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, and O. Pietquin. Noisy networks for exploration. 2017.

[5] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwi??Ska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, and J. Agapiou. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[6] X. Guo, S. Singh, H. Lee, R. Lewis, and X. Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *International Conference on Neural Information Processing Systems*, pages 3338–3346, 2014.

[7] H. V. Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. *Computer Science*, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *International Conference on Neural Information Processing Systems*, pages 1097–1105, 2012.

[11] L. Lin, H. Xie, D. Zhang, and L. Shen. Supervised neural q learning based motion control for bionic underwater robots. *Journal of Bionic Engineering*, 7(3):S177–S184, 2010.

[12] C. J. Maddison, A. Huang, I. Sutskever, and D. Silver. Move evaluation in go using deep convolutional neural networks. *Computer Science*, 2015.

[13] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Computer Vision and Pattern Recognition*, 2015.

[14] V. Mnih, A. P. Badia, M. Mirza, A. Graves, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. 2016.

[15] N. Montfort and I. Bogost. *Racing the Beam: The Atari Video Computer System*. 2009.

[16] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on International Conference on Machine Learning*, 2010.

[17] J. Pujara, M. Hui, L. Getoor, and W. Cohen. Knowledge graph identification. In *International Semantic Web Conference*, 2013.

[18] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *Computer Science*, 2015.

[19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[20] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, and A. Bolton. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[21] S. Thrun and M. L. Littman. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 16(1):285–286, 2005.

[22] M. Volodymyr, K. Koray, and David. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.