J Inf Process Syst, Vol.16, No.5, pp.1223~1230, October 2020
https://doi.org/10.3745/JIPS.02.0146

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Dynamic Action Space Handling Method for Reinforcement Learning Models

Sangchul Woo* and Yunsick Sung*

Abstract

Recently, extensive studies have been conducted to apply deep learning to reinforcement learning to solve the state-space problem. If the state-space problem was solved, reinforcement learning would become applicable in various fields. For example, users can utilize dance-tutorial systems to learn how to dance by watching and imitating a virtual instructor. The instructor can perform the optimal dance to the music, to which reinforcement learning is applied. In this study, we propose a method of reinforcement learning in which the action space is dynamically adjusted. Because actions that are not performed or are unlikely to be optimal are not learned, and the state space is not allocated, the learning time can be shortened, and the state space can be reduced. In an experiment, the proposed method shows results similar to those of traditional Q-learning even when the state space of the proposed method is reduced to approximately 0.33% of that of Q-learning. Consequently, the proposed method reduces the cost and time required for learning. Traditional Q-learning requires 6 million state spaces for learning 100,000 times. In contrast, the proposed method requires only 20,000 state spaces. A higher winning rate can be achieved in a shorter period of time by retrieving 20,000 state spaces instead of 6 million.

Keywords

Dance Tutorial System, Q-Learning, Reinforcement Learning, Virtual Tutor

1. Introduction

The recent emergence of deep reinforcement learning [1] has enhanced the performance of conventional reinforcement learning, facilitating the application of reinforcement learning in various fields. For example, dance-tutorial systems [2] enable users to learn dances while watching virtual instructors. Reinforcement learning can be applied to such a system to enable an instructor to perform an optimal dance to the music.

Typically, reinforcement learning has a state-space problem [3], for which a solution must be found to apply various reinforcement learning algorithms in the real world. The state space can be reduced as follows: first, the state space itself can be directly decreased [3]. Various approaches, such as estimating the state, have been studied to decrease the state space. Second, the action space of the model can be decreased. Typically, learning is achieved in reinforcement learning by fixing the number of actions that can be performed. However, if there are many possible actions, a state-space problem can occur because the state space considers both actions that can and cannot be performed given certain contexts. Thus, a method is required to reduce the state space by removing actions that are suboptimal or cannot be

Manuscript received July 28, 2020; first revision August 24, 2020; accepted August 24, 2020.
Corresponding Author: Yunsick Sung (sung@dongguk.edu)
* Dept. of Multimedia Engineering, Dongguk University, Seoul, Korea (woo.si@dgu.ac.kr, sung@dongguk.edu)

performed, depending on the environment, from the action space.

This paper proposes a reinforcement learning algorithm in which learning is performed by dynamically adjusting the action space. Because actions that are not performed or are unlikely to be optimal are not learned, and the state space is not allocated, the learning time can be shortened owing to the reduced state space. The remainder of this paper is organized as follows. Section 2 introduces a method to reduce the state space by adjusting the action space of a reinforcement learning algorithm. Section 3 details the application of the proposed method to Tic-Tac-Toe games [4] to verify the series of processes to be performed. Section 4 outlines the conclusions of this study.

2. Action Space Reduction

This section introduces a method to reduce the action space to solve the state-space problem of reinforcement learning. For this purpose, a virtual dance-tutorial system, to which reinforcement learning is applied, is described along with a series of processes.

2.1 Overview

In a virtual dance-tutorial system, a virtual instructor dances to music, and a user can learn how to dance by imitating the actions of the virtual instructor. The virtual instructor can analyze the music in real time and select an optimal dancing behavior. Optimal behavior can be identified by applying reinforcement learning. However, because the environment of the virtual instructor in the virtual dance-tutorial system is complex, a state-space problem inevitably occurs when applying reinforcement learning.

We propose a method to solve such a state-space problem by reducing the action space. The proposed method is applicable to various algorithms of reinforcement learning, such as the Monte Carlo method, Sarsa, and Q-learning [5], to enable their use in real time.

We trained the model by applying a Q-learning algorithm, which is a time-difference-based learning system that utilizes the merits of the Monte Carlo method and the dynamic planning method. Q-learning learns an optimal policy via an action value function used to calculate the cost incurred when a specific action is performed in the current state. Q-learning defines all possible state information in the form of a Q-table and defines action values for the actions that may occur in all possible states. Theoretically, reinforcement learning can be performed if all the states and action values of possible actions for the states are defined in the form of a Q-table. However, if a Q-table is defined for all states, a memory problem occurs. Therefore, we must limit the state space and action space to allow for the learning of the minimum number of Q-tables. Fig. 1 is an example Q-table that defines an action for a possible state and saves the corresponding action values.

	$Action_1$	$Action_2$	$Action_3$	•••	$Action_N$
$State_1$	Q_{11}	Q_{12}	Q_{13}	•••	Q_{1M}
$State_2$	Q_{21}	Q_{22}	Q_{23}	•••	Q_{2M}
$State_3$	Q_{31}	Q_{32}	Q_{33}	•••	Q_{3M}
⋮	⋮	⋮	⋮	⋱	⋮
$State_N$	Q_{N1}	Q_{N2}	Q_{N3}	•••	Q_{NM}

Fig. 1. Q-table.

2.2 State Space Reduction

Defining and accurately expressing the state space is important. If the state space cannot be expressed, or is expressed incorrectly, problem solving in reinforcement learning becomes difficult. A 5×5 Tic-Tac-Toe state space can be expressed as follows: the total number of state spaces that can be expressed in this manner is approximately 160 billion. Fig. 2 shows how to express the state space according to the number of places in the state space.

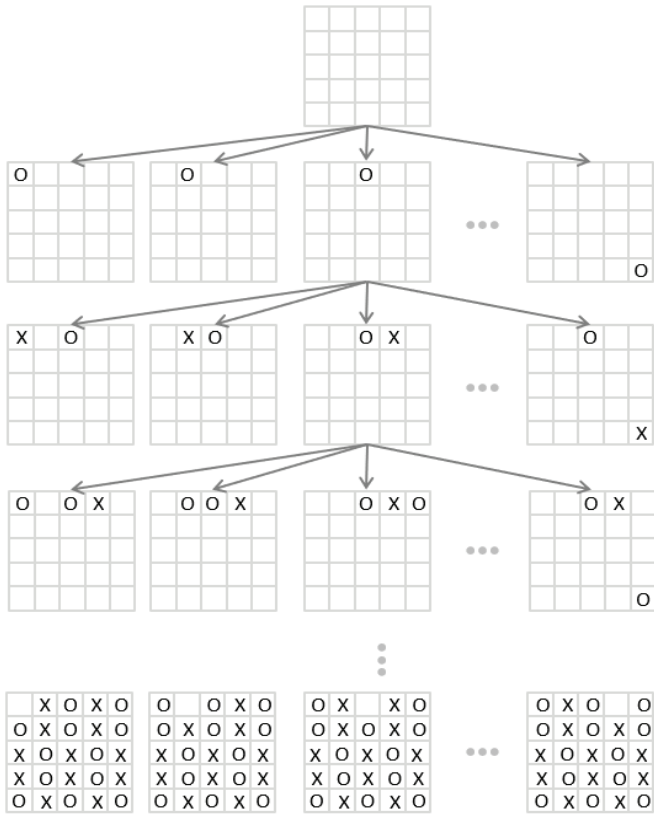


Fig. 2. State space.

Although there are 160 billion state spaces, we do not have to consider the state spaces that occur after 5 turns because it is possible for a player to win on turn 5. Therefore, we simply calculate the state space that can occur for the first 4 turns, giving a size of 83,425.

Table 1 shows how the state space size was calculated, and summing all the state space values gives us 83,425. Table 1 gives the number of state spaces along as the game progresses in turns. In the first turn, the first player places their mark. In the second turn, the second player does the same. If this is repeated up to the 4th turn and the first player places their mark on a winning position during the 5th turn, the game is over. It is possible to immediately know where the placement occurs without the state space information of the 5th turn if the state space up to the 4th turn is determined. Therefore, we trained the first strike model by only saving the state space up to the 4th turn to reduce the state space.

Table 1. State space number

Turn	1	2	3	4
Number of state space	$\frac{25!}{1!\times0!\times24!}=25$	$\frac{25!}{1!\times1!\times23!}=600$	$\frac{25!}{2!\times1!\times22!}=6{,}900$	$\frac{25!}{2!\times2!\times21!}=75{,}900$

If we define status information for 83,425 state spaces and calculate the status and action values for each status, we can create an agent that can win under any situation. We trained the first strike model by saving all state information and policy repetition.

If the first placement location is set among the 83,425 that occur in the first strike, the state spaces for the remaining three moves can be reduced to 6,650.

2.3 Action Space Adjustment Approach

An analysis of the action space is required to reduce it. Action space can be categorized into space used for learning and space not used. For example, any action that is never performed requires no space because that action is not to be learned. This type of space should be removed from the action space to solve the state-space problem. Any additional space can also be reduced if it is not related to the optimal behavior. Thus, a method is required for applying reinforcement learning and configuring rules by considering only the actions related to the optimal behavior.

To derive the action to be performed, the following procedure is applied: in the entire action set A, the set of actions that can be performed at time t is defined as A_t, and the action set A_t is defined every hour: $A_t \subset A$. Function $f(A)$ takes a set, A, and returns an action, A_t, which can be performed. Each action in the action set A_t satisfies Eq. (1). The position where an agent is located at time t is defined as s_t, and a_t is the action the agent performs at time t. In the Tic-Tac-Toe example, each placement position shown in Fig. 1 denotes a state

$$a_t \in A_t\,, \tag{1}$$

where $A_t \subset A, s_t \in S,$ and $g(a_t) \in h(s_t)$. Function $g(a_t)$ returns the spatial data of action a_t. Function $h(s_t)$ returns a set of possible spatial data on the state s_t. Action a_t is an element of set A and set A_t, and is an action that can be used for learning when the spatial data of action a_t is an element of the possible spatial data set in state s_t. The agent determines the action a_t to be performed from the dynamically defined action set A_t during reinforcement learning. For example, Q-learning is updated as expressed in Eq. (2)

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\big(r_t + \gamma maxQ(s_{t+1}, a_{t+1})\big), \tag{2}$$

where $a_t \in A_t$. $Q(s_t, a_t)$ is an action value when an action is a_t for the state s_t at time t, $maxQ(s_{t+1}, a_{t+1})$ is an action value that is the largest among all possible actions in the next state. r_t is the reward for action a_t. α and γ are constants; α is the step size in the incremental mean, and γ is the depreciation rate. The proposed method has the advantage of being applied to various algorithms of reinforcement learning without modifications, as it reduces the state space by decreasing the number of selectable actions.

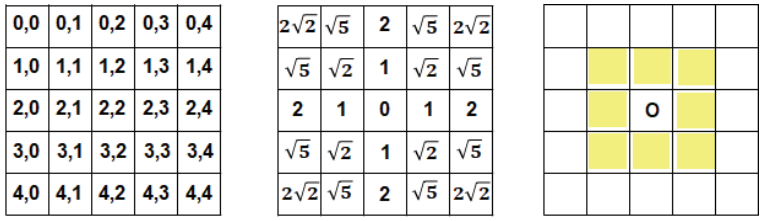
3. Experiments

This section introduces a series of processes to verify the proposed method. The proposed method is applied to Tic-Tac-Toe games for verification. This section presents the Tic-Tac-Toe game and details the application process and experimental results of the proposed method. Furthermore, the experimental results are analyzed.

3.1 Environment Introduction

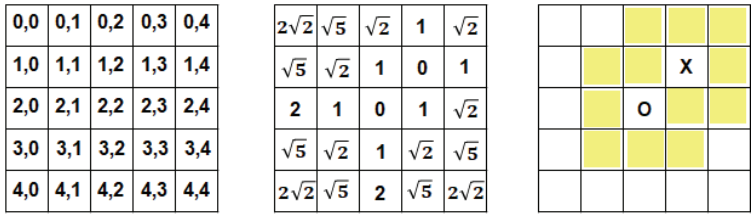
Tic-Tac-Toe is a game for two players, X and O, who take turns marking the spaces, traditionally in a 3×3 square-grid environment. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is determined the winner. Despite the simple rules of the game, the possible number of state spaces is 5,920. The proposed method was applied to 5×5 Tic-Tac-Toe to create a situation where the state-space problem exists. The size of the state space that can occur in a 5×5 Tic-Tac-Toe environment is approximately 160 billion.

The proposed method was applied to solve the state-space problem. The action space of the Tic-Tac-Toe game was reduced as follows to remove the state-space in relation to actions that were not performed or were unlikely to be performed. In the 5×5 environment, each marked space was processed as spatial data. Thus, when marking a space at a specific position was defined as an action; the function g(a) was used to process that position as spatial data. As shown in Fig. 3, function h(s) returns a set of positions of the empty cells within a distance $\sqrt{2}$ from the existing marks as a set of possible spaces.



(a) (b) (c)

Fig. 3. Case 1 of the action space: (a) state expression, (b) distances from last position, and (c) state space within $\sqrt{2}$.



(a) (b) (c)

Fig. 4. Case 2 of the action space: (a) state expression, (b) distances from last position, and (c) state space within $\sqrt{2}$.

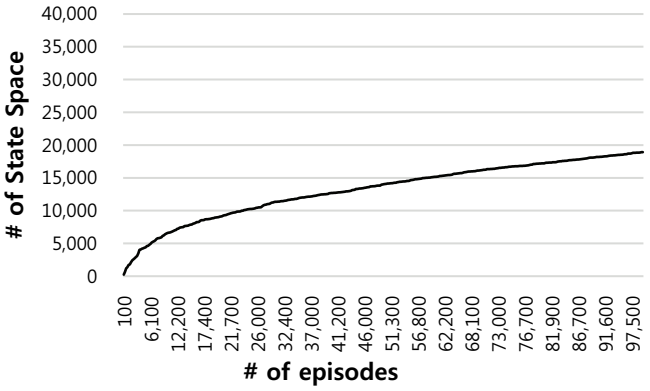
In a 5×5 Tic-Tac-Toe game, the strategies of the first and second players are different. The first player has a relatively high chance of winning. Therefore, the proposed method was applied to the second player. The second player was trained with 100,000 episodes using the proposed method, and the learning time

was 10 minutes. Learning was performed by exploring approximately 20,000 state spaces.

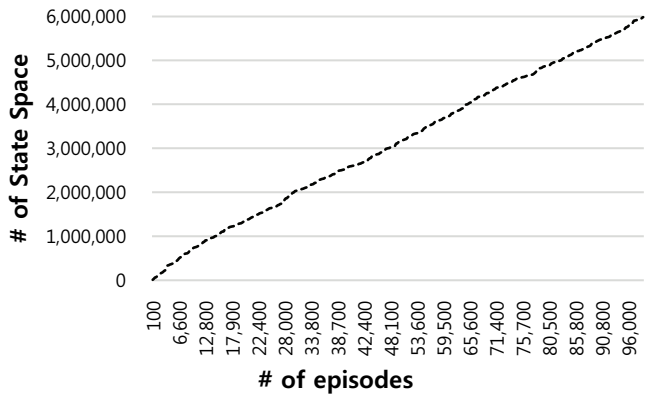
The number of wins with the proposed method was 44,683 times that of traditional Q-learning. The proposed method was far more efficient as it used only 20,000 state spaces, which is approximately 0.33% of the 6 million state spaces used in traditional Q-learning.

Figs. 3 and 4 show the process of limiting the action space to within $\sqrt{2}$ of the placed marks. There are coordinates corresponding to each position, and each marked position is mapped to these coordinates. These coordinates allow for a distance calculation. Fig. 3 shows the process of reducing the 24 action spaces available to the second player to 8, and Fig. 4 shows the process of reducing the 23 action spaces available to the first player to 12.

3.2 Experimental Results



(a)



(b)

Fig. 5. The comparison of action space search space: (a) the proposed method and (b) traditional Q-learning.

In this experiment, the proposed method and traditional Q-learning were compared according to the number of action spaces retrieved, and the count of accumulated wins. As shown in Fig. 5, the proposed method retrieved fewer than 20,000 state spaces for learning approximately 90,000 episodes. Traditional Q-learning searched approximately 6 million state spaces in learning the same number of episodes. Thus, the number of state spaces searched by the proposed method was approximately 0.33% of that by

traditional Q-learning, indicating that the time and state space can be significantly reduced by decreasing the search space.

Fig. 6 shows a comparison of the two algorithms according to their accumulated wins. The proposed method resulted in approximately 45,000 wins in training with 100,000 episodes, which was similar to the count of wins for traditional Q-learning. Although the win count differed according to learning type, it did not significantly affect the performance analysis.

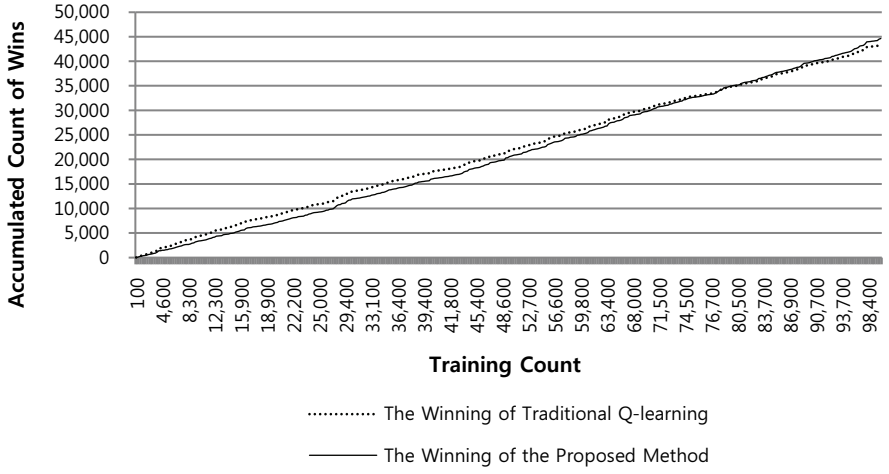


Fig. 6. The comparison of the accumulated wins.

Ultimately, the proposed method significantly reduced the learning time and state space in terms of the number of action spaces searched, while showing similar performance to Q-learning in the cumulative win count. Although the proposed method has the disadvantage of extracting spatial data from the action and state, this method is applicable in various fields because action is typically dependent on the location, and this method is expected to considerably improve performance.

In particular, although the impact of the state-space problem has recently been reduced through deep learning, the state space and training time limitations still exist; so, the proposed method can be applied to other models to address these limitations.

4. Conclusion

In this paper, a method of reinforcement learning that dynamically adjusts action space is proposed. Because actions that were not performed or were unlikely to be the optimal behaviors were not learned, and state space was not allocated to them, the learning time could be shortened, and the state space could be reduced. The proposed method was experimentally verified by applying it to a game of Tic-Tac-Toe. The proposed method showed results similar to those of traditional Q-learning even when the state space was reduced to approximately 0.33%, indicating that the proposed method could reduce the cost and time required for learning the same amount.

A virtual dance-tutorial system is used to learn how to dance with the help of a virtual instructor. If the proposed method is applied, reinforcement learning can be applied to such a system. In the future, the

proposed method will be combined with deep learning to study how the dance motions of virtual instructors can be improved.

Acknowledgement

This research was supported by the Ministry of Science, ICT (MSIT), Korea, under the High-Potential Individuals Global Training Program (No. 2019-0-01585) supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP).

References

[1] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends in Machine Learning*, vol. 11, no. 3-4, pp. 219-354, 2018.

[2] O. Alemi, J. Françoise, and P. Pasquier, "GrooveNet: real-time music-driven dance movement generation using artificial neural networks," in *Workshop on Machine Learning for Creativity in conjunction with the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Halifax, Canada, 2017.

[3] A. Raghu, M. Komorowski, L. A. Celi, P. Szolovits, and M. Ghassemi, "Continuous state-space models for optimal sepsis treatment-a deep reinforcement learning approach," in *Proceedings of the Machine Learning for Health Care Conference (MLHC)*, Boston, MA, 2017, pp. 147-163.

[4] R. Garg and D. P. Nayak, "Game of tic-tac-toe: Simulation using Min-Max algorithm," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 7, pp. 1074-1077, 2017.

[5] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is Q-learning provably efficient?," *Advances in Neural Information Processing Systems*, vol. 31, pp. 4863-4873, 2018.

Sangchul Woo **https://orcid.org/0000-0003-3158-6211**

He received B.S. degrees in Dept. of Computer Engineering at Shinhan University, Seoul, Korea in 2020. He has been interested in artificial intelligence since college. He conducted an artificial waste bin project using TensorFlow. Although he successfully completed the AI bin project, he felt the need to study due to poor recognition. His current research interests include artificial intelligence and immersive SW. Since 2020, he has studies in the Dept. of Multimedia Engineering from Dongguk University-Seoul as a M.E. student.

Yunsick Sung **https://orcid.org/0000-0003-3732-5346**

He received his B.S. degree in the Division of Electrical and Computer Engineering from Pusan National University, Busan, Republic of Korea, in 2004, the M.S. degree in Computer Engineering from Dongguk University-Seoul, Seoul, Republic of Korea, in 2006, and the Ph.D. degree in Game Engineering from Dongguk University, Seoul, Republic of Korea, in 2012. He was employed as a member of the Researcher at Samsung Electronics in the Republic of Korea between 2006 and 2009. He was a postdoctoral fellow at the University of Florida, Florida, USA, between 2012 and 2013. His research interests are focused on the areas of superintelligence in the fields of immersive softwares, UAVs, security, and music using demonstration-based learning and deep learning. He is currently an associate professor in the Department of Multimedia Engineering at Dongguk University-Seoul, Seoul, Republic of Korea.