Decision Trees for Expert Iteration: OXO Application

Michelle Alejandra Espinosa-Hernandez[1,2]

[1]School of Engineering and Sciences, Monterrey Campus, Tecnologico de Monterrey, Mexico
[2]School of Computer Science and Electronic Engineering, Colchester Campus, University of Essex, United Kingdom

Abstract—Monte Carlo Tree Search (MCTS) is among the most common reinforcement learning techniques used for determining winning moves in a game. OXO is no different, hence the reason for the application of such algorithm to a popular game. With the idea of simplifying the approach, decision trees will be used for expert iteration within MCTS. This, given the complexity of the tree that could be formed due to all the possible combinations resulting from the moves of both players. The ultimate goal of the investigation is to develop an algorithm which perorms the least moves, is constantly learning and applying better combiantions, and is highly victorious. The dataset to be used as input for the tree will be obtained from an array including the board, next player and next best move of each play. The number of times this will be run will vary until the accuracy is unaffected. Literature states that although ties would be expected from such a system, the player who moves first is 80% more likely to win and will never lose. Aditionally to the percentage of won games and the number of moves required to win will be used to evaluate the performance of the developed system.

Index Terms—Decision tree classifier, Monte Carlo Tree Search, OXO game, Reinforcement learning.

1 Introduction

Artificial intelligence aims to mimic human behavior especially in the interaction with others and the environment [1]. This unknowing and constant adaptation of computers to inputs, has become to be known as reinforcement learning (RL). This is the machine learning approach where no guidance is provided but rather are trained based on experience [2]. A common application for RL is games. For example, in 1997, Deep Blue, IBM's computer chess program [3], defeated the world chess champion [1]. Since then, algorithms alike have been developed for many other games, including OXO.

Monte Carlo Tree Search (MCTS) is a frequently used approach for identifying optimal decisions [4]. This becomes helpful in applications where it can be modeled by trees of sequential decisions such as games [4]. Although MCTS can be improved by the use of neural networks, as is the case of many other reinforcement learning applications, for a simple game such as OXO, it becomes surplus. The developed trees will provide a visual and direct path for finding the best move in order to win the game.

The following investigation will develop a decision tree classifier based on the Monte Carlo Tree Search algorithm in order to improve speed, adaptation and accuracy of a player. First, the basics of the game and the Monte Carlo will be explained, followed by a presentation of previous models developed. Next, the methods and obtainment of the dataset will be provided. Then, the experiments to be carried out as well as the expected results will be analyzed. An evaluation of the results and a conclusion will finish the document.

• *M. A. Espinosa-Hernandez is with Tecnologico de Monterrey, Nuevo Leon, Mexico, 64849. E-mail: A01039799@itesm.mx. She is also with University of Essex, Colchester, United Kingdom, CO4 3SQ. E-mail: me19672@essex.ac.uk.)*

2 Background

2.1 The game

OXO, also known as tic-tac-toe or noughts and crosses, is an ancient Roman game involving two players alternatively placing *X* or *O* in one of the 9 squares formed in the 3x3 board as seen in Fig. 1(a) [5]. The game ends when one player has three figures in a row horizontally, vertically or diagonally as can be seen in Fig. 1(b), or all spaces of the board are occupied but neither player has three in a row as in Fig. 1(c) [5].

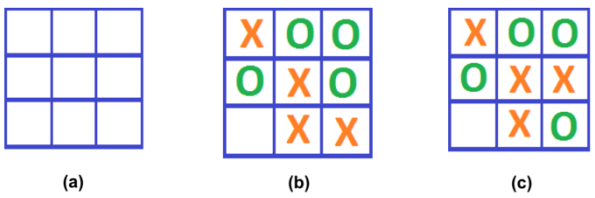

Fig. 1. (a) Empty board (b) Won game for player X (c) Tied game

2.2 Literature survey

Many approaches have been made in developing automated computer players such as OXO usually by means of neural networks or decision trees [6], [5]. Due to the fact that there is no labeled input nor ouput, but rather a continuous learning approach to identify the most rewarding actions is to be performed. The only way the system can learn is by interacting with the environment. This is reinforcement learning [7]. Among the approaches to onine automatized players of OXO, is the Monte Carlo

Tree Search (MCTS) [8]. This algorithm consists of 4 steps: selection, expansion, simulation, and backpropagation as can be seen in Fig. 2 [6]. The third step of the simulation stage is where the MC simulation takes place. The average of the outcomes can be applied to the evaluation value for every child node in the current phase [6]. The child node with the highest value is selected from the average value of each in order to undergo game actions [6].

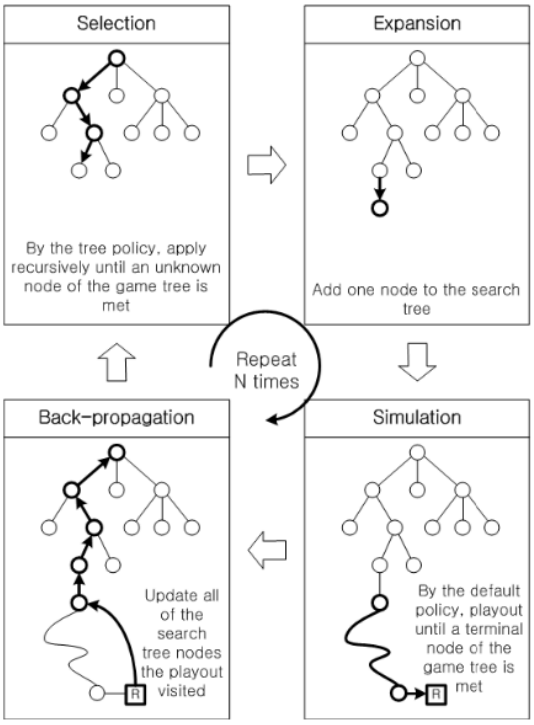

Fig. 2. Schematic of Monte Carlo Tree Search steps

In order to apply the MCTS, it requires (1) a game tree which records actions taken during the game, and (2) a search tree with necessary information for the simulation step [6]. Similarly, Expert Iteration (ExIt) uses a Tree Search to aid the training of the algorithm [8]. After each iteration, an *expert improvement step* [8] is done where the apprentice policy is bootstrapped in order to increase the performance of the expert. An apprentice, implemented in a deep neural network, is trained to similate the expert, which is implemented by a tree search algorithm [8].

3 Methodology

By means of this research, a fast, constantly-learning, and winning classifier will result. First, the basic Monte Carlo Tree Search code will be modified to record play-by-play the occupied positions on the board, next player, and next best move. This will be done *n* times to produce a dataset from which the decision tree may learn. Next, the decision tree machine learning approach will be implemented. It will branch every move and obtain the fastest way to win based on previous games. Then, this learned information will be input to the MCTS code for generating the next best move, rather than a more random approach.

The initial dataset will be a *panda data frame* that includes 11 columns: the first nine for the corresponding positions on the board, one for the player whose move is next, and one for the best obtained move that is next to occur. Given the last two columns are future-based, the very last play, where a winner or tie is determined, the tenth column will be given the value of the winning player plus five. This, in order to distinguish the plays of a game as 5, 6, or 7 is taken as the final play of a game, while a 0, 1, or 2 is just a play. Additionally, in the last play, the last column is left blank, again, to separate the winning play from the others.

Through the use of a decision tree as the classifier, the time taken to play a game oought to be minimized as the wining branch will be calculated and updated based on the player and opponent's move. Hence, the decision tree will update every play to find the shortest winning route. Since this approach will be applied to both players, the outcomes should tend to be draws. However, since the first-moving player will play offensively, it would be able to calculate and apply the winning branch easier rather than the defensive player.

4 Results

For the evaluation of the algorithm, the original dataset will consist of plays from only 10 games. Other datasets will be created with increasing number of games to then test the accuracy and effect on the victory. Many initial plays in the dataset will allow more combinations to become evident and hence reach a win faster. The accuracy of the algorithm is reflected on the number of branches and the depth of the game trees [6]. MCTS is used when the quantity and extension is large [6], hence it is expected a complex tree will result. Nonetheless, the speed of the algorithm is to be kept minimum. This involves the number of plays required to finish the game victoriously.

Although constant ties may be expected from an algorithm that works efficiently as both pkayers are being input the same data and trees, it is bound for the player that moves first to win more often. This player would be developing a strategy while the other player will be more focused on blocking the first player's moves rather than win. In [6], 10,000 games were played of which 6,069 were won and 3,931 were tied by the first-moving player. In other words, the initial player will never lose, making evident the advantage of moving first.

5 Discussion

The aspects that will be considered for the evaluation of the results are the number of moves made before winning, the calculation of the optimal move, and the percentage of won games of the player who plays first.

First, the number of moves. For it to be optimal, it must be made in less than the 9 turns so as to not tie. Ideally, it should be 5 moves as that implies that the three

turns resulted in a direct win and the opponent did not block. The importance of this value is to observe it as time progresses as the algorithm should keep learning and improving. After a certain number of games, it should stabilize and come close to 5.

Next, the calculation of the optimal move. This imples that the next play is truly the best decision in the short and long term. Ultimately, the goal is to ensure a win, hence the importance of this statistic.

Lastly, the percentage of games won. This is the most evident statistic of the accuracy of the algorithm. Through this value, the development can be compared to the literature to fully evaluate the importance of moving first and the advantage it provides.

6 Conclusion

Through this investigation, the application of decision trees to reinforcement learning will be a first. The simplicity of the classifier will allow the understanding and inititation to RL. Having a direct visualization and effect of the application on the game allows a clearer explanantion. This, especially since the background knowledge (what OXO is and how it is played, what is reinforcement learning, what are computer players) is present. Additionally, having understood it fully, it can then be applied to other, more complex applications or games.

References

[1] D. Silver *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science,* vol. 362, no. 6419, pp. 1140-1144, 2018.

[2] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature,* vol. 550, no. 7676, pp. 354-359, 2017.

[3] F.-h. Hsu, "IBM's deep blue chess grandmaster chips," *IEEE Micro,* vol. 19, no. 2, pp. 70-81, 1999.

[4] C. B. Browne *et al.*, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games,* vol. 4, no. 1, pp. 1-43, 2012.

[5] M. Abu Dalffa, B. S. Abu-Nasser, and S. S. Abu-Naser, "Tic-Tac-Toe Learning Using Artificial Neural Networks," 2019.

[6] B.-D. Lee, "Enhanced strategic Monte-Carlo Tree Search algorithm to play the game of Tic-Tac-Toe," *Journal of Korea Game Society,* vol. 16, no. 4, pp. 79-86, 2016.

[7] V. François-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.

[8] T. Anthony, Z. Tian, and D. Barber, "Thinking fast and slow with deep learning and tree search," in *Advances in Neural Information Processing Systems*, 2017, pp. 5360-5370.

Plan: Gantt Chart

	Week 1	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**	**Week 7**	**Week 8**	**Week 9**
	Jan 20 – Jan 26	Jan 27 - Feb 2	Feb 3 – Feb 9	Feb 10 – Feb 16	Feb 17 – Feb 23	Feb 24 – Mar 1	Mar 2 – Mar 8	Mar 9 – Mar 15	Mar 16 - Mar 29
Assignment of project									
Literature survey									
Code for generation of dataset									
Obtain dataset									
Writing of report									
Delivery of Assignment 1									
Coding of decision tree									
Code learned classifier not random									
Collect data									
Writing of report									
Neural network (if time allows)									
Delivery of Assignment 2									