Decision Trees for Expert Iteration (Reinforcement learning) (revised February 2020)

Irshad Shahzadi Mahaa

Abstract—This work investigates the optimal characteristics of a supervised prediction model implemented with the Monte Carlo tree search (MCTS). We exploit the random rollout feature of MCTS to generate a dataset with potential states and optimal moves, solely based on the reinforcement learner, without any expert human's data. This dataset trains a classic supervised learning model, decision tree. We also use a neural network to replace manual input of the dataset to the classifier to test performance improvements. We demonstrate the effect of varying the tradeoff between exploration and exploitation of MCTS. Moreover, we explore the impact on the learner when trained further with a self-playing agent.

Index Terms—MCTS, supervised learning, reinforcement learning, neural networks

1 Introduction

Researchers and computer scientists have been working on developing game playing abilities in computers since 1949 when the focus was on strategy games like chess [1]. Machine learning, a subset of artificial intelligence, creates a model based on statistical datasets, omitting the need to explicitly program computers for a specific task. It is widely adopted to learn game playing strategies in computer systems. The interest in this area has been gradually increasing for 60 years, due to the fact that games prove to be a considerable source to stimulate computational intelligence [1]. In the past, a common perception regarding game playing computers was that it can never outperform humans. However, several types of machine learning systems in recent years have shown exceptional performance that outclassed even expert human players. A well-known software AlphaGo, developed by Alphabet in 2015, was the first computer to defeat a world champion in the board game, Go. The following years it proved its potential again by winning victory over; Lee Sedol, a professional Go player and the number one ranking player at that time, Ke Jie. Moreover, several machine learning methods have been employed for various board games including Backgammon, Othello, Tic-tac-toe, and Ludo. Increasing demand for artificially intelligent (AI) agents in video games are evident and set to expand dramatically in the near future. Popular video games such as StarCraft series, Dota2, planetary annihilation have employed AI agents to implement real-time game strategies.

• *S. M. Irshad is a MSc. In Intelligent Systems and Robotics student at the University of Essex, Colchester, CO4 3SQ, UK. E-mail: si19783@essex.ac.uk*

2 Background

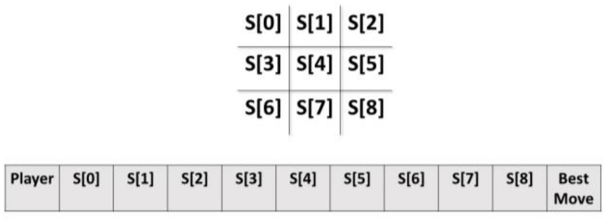
The main components of a machine learning algorithm are obtaining a dataset, creation of an evaluation method to monitor performance loss or improvement and establishment of an optimization procedure [2]. Numerous methodologies in machine learning exist to implement these components. Machine learning is categorized mainly as supervised learning, unsupervised learning and reinforcement learning [2]. The work of several researchers exploiting different categories and methodologies to attain the same ability of learning game strategies is discussed in this section.

The AlphaGo, Go-playing system, utilizes a supervised learning algorithm to learn from human expert's moves and a self- playing reinforcement learning algorithm. Using this information, deep neural networks trained to predict moves. The Monte Carlo tree search (MCTS) is used simultaneously with the two neural networks to evaluate positions and select moves [5]-[7]. These double neural networks have a significant enough impact on the performance of the learner to beat a human player. [8] uses a slightly different approach by creating a learning dataset only using reinforcement learning. The system starts training using only reinforcement learning by self-playing with random moves. It is followed by a single neural network that predicts the moves. Finally, MCTS evaluates positions and sample moves. A similar approach is adopted by [9]'s work as they employ MCTS for optimal action policy in the Hearthstone game. Moreover, [10] uses neural networks to improve the performance of the MCTS-based learner for the same game.

The type of a game can differ on the basis of knowledge on complete game and uncertainty of game events. Hearthstone game is an example of a game with partially hidden information. Maciej employs a custom MCTS, using an iterative approach to train a neural network by playing itself. This provides guidance to the game state search heuristic and improves the win-rate.

3 Methodology

This work aims to explore MCTS for data generation, a supervised learning algorithm like decision tree as learner and modification methods on the MCTS function. MCTS is a heuristic driven algorithm that is a combination of the classic tree search algorithm and machine learning principles of reinforcement learning. The tree search provides all the potential states of a given problem by the actions that would achieve them and the reinforcement learning aims to find the optimal action in a given problem by maintaining a reasonable balance between exploration and exploitation. MCTS is preferred over other tree search algorithms like minimax because it omits out the branching factor in large problem space.
The code provided by [4] is used to play the OXO game employing MCTS. To create the dataset, it is modified by running the game 500 times. In an OXO game, the best result for a player is to win, followed by a game draw. On each iteration of the game, in the case of a winner, the moves of the respective player are stored in an array and in the case of a draw, the moves taken by both players are stored. Whereas, the moves of the losing player are ignored. As shown in Fig. 1, a combined array with 11 columns and rows depending on the number of best moves taken are stored as a dataset to train the supervised learner. This dataset is fed into a decision tree classifier that learns the best moves for a player, depending on the state of the game.



Player	S[0]	S[1]	S[2]	S[3]	S[4]	S[5]	S[6]	S[7]	S[8]	Best Move

Figure.1: Index of dataset generated

A comparatively better approach to store the data will be explored using a neural network. Based on the current state of the board, the neural network would create a hidden layer containing information on the predicted move. This information would be fed into the classifier as before. Furthermore, the tradeoff between exploration and exploitation would be explored by tuning the ratios to check for any improvements in the performance of the learner. The Upper Confidence Bound (UCB1) parameters will be tuned for this purpose. Finally, the most commonly used self-playing agent would be engaged with the trained classifier to investigate its effect.

4 Results

As discussed above, this work compares four types of algorithms: a simple MCTS feeding dataset to decision tree directly, a neural network used to send data to the decision tree classifier, MCTS with different exploitation and exploration values and, a classifier with a self-playing agent. The performance of these four different algorithms would be compared by evaluating the Elo rating, prediction accuracy and mean-squared error (MSE) for each approach. These parameters would be evaluated by running the classifiers on the test data consisting of a number of states, which will either be a partition from the training data or competition with an expert OXO player. The action of the player would be best rewarded if it wins and partially rewarded in case of a match draw.

5 Discussions

A comparison of the evaluation parameters on the four players will be conducted. As Elo rating finds the relative skill levels of players, the player with the highest value will be preferred as the players with the best skills. The accuracy parameter provides information about the frequency of correct predictions by a player. The player with the highest prediction accuracy will be identified as the player with the best moves. Finally, the player with the lowest MSE will be declared as the player with the lowest chances of losing.

6 Conclusion

Because IEEE The scope of this project extends to performance enhancements that can be carried out on MCTS followed by a decision tree classifier. Several evaluation methods are carried out to compare the different approaches but the best method would be declared based on the overall performance of each player.
YZ.

References

[1] S. Lucas, "Ms Pac-Man competition", ACM SIGEVOlution, vol. 2, no. 4, pp. 37-38, 2007. Available: 10.1145/1399962.1399969.

[2] L. Yuxi. "Deep reinforcement learning: An overview." arXiv preprint arXiv:1701.07274, 2017.

[3] Świechowski, Maciej, Tomasz Tajmajer, and Andrzej Janusz. "Improving hearthstone ai by combining mcts and supervised learning algorithms." In 2018 IEEE Conference on Computational Intelligence and Games (CIG), pp. 1-8. IEEE, 2018.

[4] https://drive.google.com/file/d/1aeBVSFMlwVUqqnqta3wU-5tG5iiV-kpI8/view

[5] Coulom, R. Efficient selectivity and backup operators in Monte-Carlo tree search. In 5th Int. Conf. Computers and Games (eds Ciancarini, P. & van den Herik, H. J.) 72–83 (2006).

[6] Kocsis, L. & Szepesvári, C. Bandit based Monte-Carlo planning. In 15th Eu. Conf. Mach. Learn. 282–293 (2006).

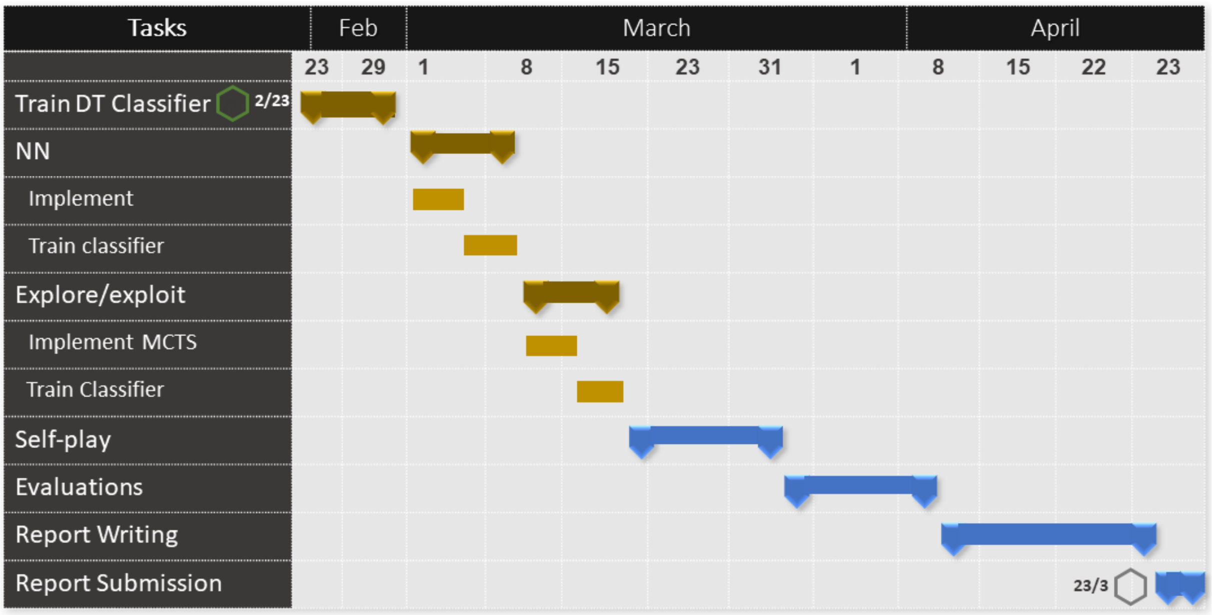
[7] Browne, C. et al. A survey of Monte Carlo tree search methods. IEEE Trans. Comput. Intell. AI Games 4, 1–49 (2012).

[8] D. Silver et al., "Mastering the game of Go without human knowledge", Nature, vol. 550, no. 7676, pp. 354-359, 2017. Available: 10.1038/nature24270.

[9] A. Santos, P. A. Santos, and F. S. Melo, "Monte carlo tree search experiments in hearthstone," in IEEE Conference on Computational Intelligence and Games, CIG 2017, New York, NY, USA, August 22-25, 2017, 2017, pp. 272–279. [Online]. Available: https://doi.org/10.1109/CIG.2017.8080446

[10] S. Zhang and M. Buro, "Improving hearthstone AI by learning high-level rollout policies and bucketing chance node events," in IEEE Conference on Computational Intelligence and Games, CIG 2017, New York, NY, USA, August 22-25, 2017, 2017, pp. 309–316. [Online]. Available: https://doi.org/10.1109/CIG.2017.8080452

Appendix A
Project Plan


Tasks
Feb
March
April
Train DT Classifier
2/23
NN
Implement
Train classifier
Explore/exploit
Implement MCTS
Train Classifier
Self-play
Evaluations
Report Writing
Report Submission
23/3