Final Project Report – CS 6749

Yongyang Liu, Xiyang Wu, Zhenyu Jia

1. What we did (625 words):

MCTS:

This model is covered by Xiyang Wu and Yongyang Liu, where Xiyang establish a basic structure and Yongyang make modification to make it feasible for integration.

Implementation of the MCTS is on the early stage of the whole project. We first set up the vanilla MCTS model, which is the requirement of the undergraduate students. The idea for this model is: Once the game starts, the MCTS reads the real board and the list for the legal moves given by the Python Chess class, if the current node has been extended, the model will select the child node with the maximum UCB value, otherwise extend the node by adding all the new boards after implementing the legal moves on the current board to the child node list and randomly select one. Once the game ends or meets the maximum exploration threshold, which is 600 by default, the mcts tree updates the visit times and the UCB value via back propagation.

Later, we make modification to the action selection module and value estimation module. To be integrated with AlphaZero, we add a PUCT algorithm [1] to action selection, which used UCT value before, and estimate the winner value by Neural Network instead of a default function in vanilla MCTS.

AlphaZero:

This model is covered by Yongyang Liu and Xiyang Wu, where Yongyang establish the AlphaZero framework and Xiyang make modification to make it feasible for integration.

Different from a basic MCTS, which could read board state directly from Python Chess class, AlphaZero should consider the input and output of Neural Network. Inspired by [2], we design the input data as a probability distribution matrix of the board state with a size of 13x8x8, in which 8x8 represents the board and 13 refers to 13 different piece states, including 6 different pieces of my player, 6 different pieces of opposite player, and 1 empty square. The sum of probabilities over 13 pieces in each square is 1. For the output of Neural Network, we design it as the probability distribution over the action space. Because Chess represents actions as moves from one square to another square, instead of moves of pieces, we follow the same method, which is different from [2]. Then, we get an action space with 64x63x2=8064 moves, where 2 refers to with and without a queen promotion (consider promotion of only queen). To better connect the Chess class and AlphaZero, we also build a ChessRead function for the conversion between the Chess board state and action and AlphaZero state and action.

Because AlphaZero is fully observed model, while the proposed environment is a partially observed one. In this case, we make modifications to the AlphaZero to incorporate the Particle Filter. We first separate the node extension part into two functions for the current player and the opponent respectively, and then create a training agent with the stockfish engine.

Particle Flter:

This model is covered by Zhenyu Jia and Xiyang Wu, where Zhenyu establish the basic model and Xiyang make modification to make it feasible for integration.

To handle the uncertainty, we build two approaches, a particle filter [3] and a naive target tracker. Particle Filter tries to best estimate the entire board under the assumption that the opponent acts at random. The target Tracker, on the other hand, does not aim to recover the entire board information, but tries to locate the target, ie. the king, and its surrounding area. Both approaches provide a probabilistic estimate of the board with limited scanning and encode the board to a directly usable matrix. We also provide a simple and easy-to-use public API that can be called by the agent.

Later, we make several modifications to the basic model, and integrate it into the AlphaZero framework.

2. What we learned (523 words):

MCTS:

Since MCTS has been introduced in the course, implementing it from scratch is much more different. Similar to the neural network, the MCTS has the forward exploration and backpropagation function, though follows the tree-like model. In this part, what we learned covers the way to convert our thought on regulating the environment into feasible functions, like the action selection, node extension and value backpropagation. Through this, our understanding on the structure of the MCTS as well as the tree-based model like the RRT has been greatly improved.

AlphaZero:

AlphaGo Zero is so famous for the Go competition with human players. The chance to build an AlphaZero model by ourselves is so great. The building of AlphaZero for Chess helps us better understand the mechanism of AlphaZero and obtain the capability to build another one for other games. During the process, the MCTS in AlphaZero remind us the Markov Decision Process (MDP), which is also used to model the interaction between players. Both MDP and MCTS measure current situation by a certain value function, reward and PUCT, and select actions to maximize a value, V/Q value and P+U value. A further study on the comparison to these two method would be interesting, and the comparison result can help us better understand the pros and cons of these two method.

Particle Flter:

There are two major challenges of using a particle filter in our problem. The first is that we are to estimate a definite discrete space instead of a probabilistic continuous state space. This means that we need to have all possible state particles in our candidate space because we would otherwise lose all information if we accidentally delete the actual state. Also we could not recover a state if it's lost because we cannot use a continuous buzz to recover such loss. The second challenge is that scan only provides very limited information that could not reduce the particle space by many. It means that, over time, the state space will expand to a huge space. Some smart improvements like entropy measures could be introduced here to better select a scan that would help most efficiently reduce the state space.

Integration:

Another important experience we learned is to incorporate the partial observation model with the MCTS and AlphaZero. The initial implementation of AlphaZero is much different from the partial observation requirement of the environment. Since the truth board is unknown, some necessary changes on the initial needs to be made, like separating the nodes in the MCTS model into the nodes generated by the opponent and the current player and setting up different updating policies for them. Based on our experience in this part, we think an essential thing in the project is to plan the general structure of the project carefully at the very early stage and then leave the interface for modules that have not been implemented yet. This could help the integration of the modules in the end, otherwise the module incorporation could be difficult due to the matching problem between the modules, which forces me to re-write the agent and the training and testing framework in the end.

3. Contribution (544 words):

Xiyang Wu:

My major contribution covers two parts: The first is to establish the vanilla MCTS model for the chess player agent. Though this model is based on the fully-observed environment without specifying the action and state space, the action selection, node extension and backpropagation function are of importance for the later implementation, since this model sets up the framework for the whole project that helps the successor modules to implement. Another contribution of mine is to incorporate different modules into the

same single agent and write the training and testing framework. For this part, since Yongyang and I accomplished the AlphaZero model that combines two players together rather than separates them into the agent that reads the board state and takes action, I first thoroughly re-write the AlphaZero model to incorporate it with the particle filter interface implemented by Zhenyu, and then set up the training and testing framework for the new agent. In each turn, the modified AlphaZero agent first reads the opponent's move and updates board estimation, then takes action based on the estimated board and the AlphaZero policy and updates the board again.

Yongyang Liu:

In this project, the main contribution of mine is to modify a MCTS developed by Xiyang Wu and build the AlphaZero framework, which provides a fundamental structure for the project.

First, I design the state space and action space for AlphaZero model and build a Chess-Read function for the conversion between the Chess class state & action and AlphaZero state & action. This function successfully connects the Chess class and our AlphZero model.

Second, since the vanilla MCTS has different mechanism to AlphaZero one, I thoroughly rewrite the MCTS model and incorporate it into the AlphZero framework.

Third, I build a Reinforcement Learning framework, i.e. AlphaZero framework, which includes Chess-read module, Neural Network module, modified MCTS model, self-play module, AlphaZero Training module, and game play module.

In total, my contribution is developing a AlphaZero framework for Chess, which could train itself and play Chess against other agents successfully. However, the built AlphaZero framework is based on ground truth board state, which means it needs the truth of the board and it cannot play the Recon Chess due to uncertainty.

Zhenyu Jia:

My major contribution is developing, researching, and testing of the different tracker methods to handle uncertainty. I actually came up with three models which turned out to be two usable versions. I started with understanding the uncertainty as a whole and estimating the board from a simple and easy approach as a naive king tracker which only focuses on the king. I then improved the method by recovering more information without losing the king's position. That came up with the current version of king tracker which also recovers a 5X5 board area plus the king location and also re-capture the king position when king is lost. On top of the naive king tracker, I upgraded the method to a more advanced approach: the particle filter. This particle filter handles the accurate estimate of the board state and manages to reduce the board state space very efficiently. I also researched and designed the board state encoding to most efficiently. I also handled the miscellaneous stuff including some general util functions and handling the repository management.

Reference

[1] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A. and Chen, Y., 2017. Mastering the game of go without human knowledge. *nature*, *550*(7676), pp.354-359.

[2] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T. and Lillicrap, T., 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

[3] Seminar f¨ur Statistik, D-Math, ETH Zurich, CH-8092 Zurich, Switzerland 2013, Particle filters. *arXiv: 1309.7807v1*