**Deep Reinforcement Learning**

**for Othello (Reversi)**

Final Project

DATA130011.01  
Neural Network and Deep Learning  
School of Data Sciences  
Fudan University

**Rong Ye**

[14300690041@fudan.edu.cn](mailto:14300690041@fudan.edu.cn)  
Statistics  
School of Management

**Yuqing Xie**   
[14307130361@fudan.edu.cn](mailto:14307130361@fudan.edu.cn)  
Mathematics and Applied Mathematics  
School of Mathematical Sciences

**Abstract**

**Introduction**

**The Othello Game**

Rule

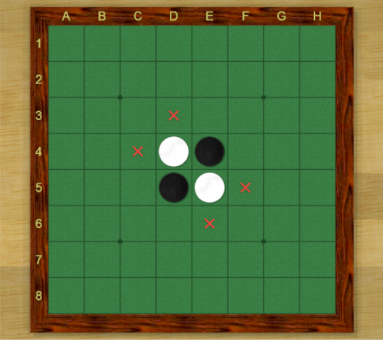
Othello is a two-player game on an 8 \* 8 board. There are 64 identical pieces which are white on one side and black on the other. The game begins with each player having two pieces placed diagonally in the center of the board **(Figure 1)**. The black player moves first. A move is legal if the newly placed piece is adjacent to an opponent’s piece and causes one or more of the opponent’s pieces to become enclosed from both sides on a horizontal, vertical or diagonal line. The enclosed pieces are then flipped. The black player's first move can be any one of four red-crosses locations, in **(Figure 1)**. Players alternate placing pieces on the board. If and only if a player does not have any legal move the player passes. The game ends when neither player has a legal move (of course when the board is filled with pieces), and the player with more pieces on the board wins. If both players have the same number of pieces, the game ends in a draw.

Figure 1 Othello play board and initial state.

Features & Difficulty

Othello is a perfect information, zero-sum, two-player strategy game. Despite simple rules, the game is far from trivial. The number of legal positions is approximately and the game tree has in the order of nodes [5], which precludes any exhaustive search method. Othello is also characterized by a high temporal volatility: a high number of pieces can be flipped in a single move, dramatically changing the board state[要给个例子吗，看篇幅，如果短的话，给个app游戏截图].

We can also reduce states according to the mirror and rotation property of the board. [if necessary，这个好！]

**Related work**

[1] Introduced CNN and using professional game records for training.

This paper verifies whether CNN-based move predictors prove effective for Othello. They compare several CNN architectures and board encodings, augment them with state-of-the-art extensions, train on an extensive database of experts’ moves, and examine them with respect to move prediction accuracy and playing strength.

[2] Mainly compares different reinforcement learning methods' performance

This paper compares three strategies: Learning by self-play, learning from playing against a fixed opponent, and learning from playing against a fixed opponent while learning from the opponent’s moves as well in using reinforcement learning algorithms, to let an artificial agent learn to play the game of Othello, using algorithms Q-learning, SARSA and TD-learning. It is found that the best strategy of learning differs per algorithm. Q-learning and SARSA perform best when trained against the fixed opponent they are also tested against, whereas TD-learning performs best when trained through self-play. Learning from the opponent’s moves leads to worse results compared to learning only from the learning agent’s own moves.

[3] Adversarial advantage for task-compeletion dialogue policy learning

This paper presents adversarial advantage actor-critic (Adversarial A2C), high significantly improves the efficiency of dialogue policy learning in task completion dialogue systems. They train a discriminator to differentiate responses/actions generated by dialogue agents from responses/actions by experts. Then, they incorporate the discriminator as another critic into the advantage actor-critic (A2C) framework, to encourage the dialogue agent to explore state-action within the regions where the agent takes actions similar to those of the experts.

[其实这部分我觉得不是很用的上]

[4] The best known Othello playing program is LOGISTELLO.

This paper surveys the evaluation and **search techniques** utilized by the strongest Othello programs of their time during the past twenty years. In this time span computer Othello has experienced considerable progress which culminated in the convincing 6-0 victory of LOGISTELLO against the then World-champion Takeshi Murakami in 1997. The focus of this article is the evolution of Othello evaluation functions and **heuristic search techniques** which quite nicely reflect the general Artificial Intelligence trend of replacing slow and error-prone manual tuning by automated machine learning approaches.

**Reinforcement learning model**

Basic reinforcement learning model:

In reinforcement learning, the learner is a decision making agent that takes actions in an environment and receives a reward (or penalty) for its actions in trying to solve a problem. After a set of trial-and-error runs it should learn the best policy, which is the sequence of actions that maximize the total reward.

We define our problem as a Markov decision process.  
(1) A finite set of states ; In Othello, s means a certain broad state and S contains all possible states of the game, approximately states.  
(2) A finite set of actions ; In Othello, a means a legal position to place piece.  
(3) The transition function is simply to place a piece chosen by action a after state s.  
(4) A reward function , providing the reward the agent will receive for executing action a in state s, where denotes the reward obtained at time t; In Othello, a reward is received only when the whole game comes to an end. Also in our project, official judgement does not consider how many pieces are wined, but only care whether black player win the game. We will try this reward function and also consider other choice to improve the agent's performance.  
(5) A discount factor which discounts later rewards compared to immediate rewards.

In a reinforcement learning model, we also have to define value functions:   
We want our agent to learn an optimal policy for mapping states to actions. The action to be taken in any state s is. The value of a policy , , is the expected cummulative reward that will be received when the agent follows the policy starting at state s. It is defined as: . The optimal policy is to achieve largest state value in all states. We sometimes also use Q-value () instead of value function.

Learning algorithms

During the process when our agent is playing the game, we could learning the Q-value as

, where*α* is the learning rate

TD- network  
1) Observe the current state st  
2) For all afterstates s0 t reachable from st use NN to compute V (s0 t)  
3) Select an action leading to afterstate sa t using a policy π  
4) According to (10) compute the target value of the previous afterstate V new(sa t−1)  
5) Use NN to compute the current value of the previous afterstate V (sa t−1)  
6) Adjust the NN by backpropagating the error V new(sa t−1)− V (sa t−1)  
7) sa t−1 sa t  
8) Execute action resulting in afterstate sa t

Q-learning newtwork  
1) Observe the current state st  
2) For all possible actions a0 t in st use NN to compute Q ^(st; a0 t)  
3) Select an action at using a policy π  
4) According to (8) compute the target value of the previous state-action pair Q ^new(st−1; at−1)1) Observe the current state st  
5) Use NN to compute the current estimate of the value of the previous state-action pair Q ^(st−1; at−1)  
6) Adjust the NN by backpropagating the error Q ^new(st−1; at−1) − Q ^(st−1; at−1)  
7) st−1 st, at−1 at  
8) Execute action at

Here are some points that we should focus on:  
1) According to the huge amount of states, we have to learn the policy using statistic methods and network structure may affect learning proficiency greatly.  
2) The reword function is also critical for learning ability  
3) The algorithms' performance also vary when the opponent change. We have fixed policy opponents, experts playing records, best players in the world and we also tried self-play opponent inspired by Alpha-Go.

**Experiment**

**Dataset (if used the professional database)**

**Learning& test – results**

**Compare& analysis**

**Reference**

[1] Learning toPlay Othello with Deep Neural Networks（CNN的那篇）  
[2] ReinforcementLearning in the Game of Othello: Learning Against a Fixed Opponent and Learningfrom Self-Play（主要参考程序章结构）  
[3] ADVERSARIALADVANTAGE ACTOR-CRITIC MODEL FOR TASK-COMPLETION DIALOGUE POLICY LEARNING（最后修改value学习部分可以参考这个）  
[4]M. Buro, “The evolution of strong othello programs,” in Entertainment Computing - Technology and Applications, R. Nakatsu and J. Hoshino,Eds. Kluwer, 2003, pp. 81–88. <https://skatgame.net/mburo/>  
<https://skatgame.net/mburo/log.html>  
[5]Victor L Allis. *Searching for solutions in games and artificial intelligence*. PhD thesis, University of Limburg, Maastricht, The Netherlands,1994