The University of Hong Kong

Department of Computer Science

Final Year Project

Playing Othello by Deep Learning Neural Network

PROJECT PLAN

Supervisor

Dr. K.P. Chan

Team members

Yip Tsz Kwan : 3035068685

Chan Lok Wang : 3035072777

30 September, 2016

Table of Contents

1. Summary

This document aims to define the purpose and the management of the project. The background and limitations of the classical approach in developing a computer game program will be first provided, followed by an introduction of a modern approach using the technique of machine learning. Detailed objectives of the project and the methodology used to achieve the goal will be introduced in the next section. Division of work and the project schedule will be presented in the final section.

2. Introduction

Over the years, people have been trying to exploit the abilities of a computer to create a better world. One of such abilities is to make decision as if human do. A way to show the capability of a computer making good decision is by playing board games as it involves the evaluation of the current board settings and selecting the next best moves constantly throughout the game. It is also a good way to evaluate how good a computer can perform when compared with human.

There are two major components in a computer game program – the game search tree and the evaluation function.

Traditionally, a computer plays a board game by searching through the game tree - a tree containing all the possible moves with the corresponding weights, which indicate how likely one can win the game with those moves. The values of the weights are being assigned according to an evaluation function. [1]

Due to the limited storage and the complexity of the game search tree, the computer has to prune the branches with less weight for decision-making. [2] However, the problem of how well the computer can make such decisions arises, and this is where (deep learning) neural network proves to be useful.

Inspired by the human brain, neural network is a way of information processing. Several layers of notes in the neural network are connected together and change as the system is trained. A large amount of training data are used to fine-tune the connections during the training process so that the network can produce a specific output corresponds to the given input. A deep learning neural network is a network with many layers. [2] This deep learning neural network can help increase the accuracy of the results generated by the evaluation function throughout a series of learning processes, and hence, can help the computer make better decision of which branches to prune when searching for the next best move.

This project aims to demonstrate how powerful deep learning neural network can be for game-playing. The game Othello is chosen as the technique of deep learning neural network can be applied in this game. Meanwhile, the size and the complexity of this game is suitable for a one-year long project with limited resources. It is worth noting that when both sides play the game perfectly, it appears that the game will very likely end with a draw. [3] In additions, due to the limited resources, including the time and hardware specification, for training the neural network, our project will be considered as successful when it plays equally well against a strong (computer)

opponent. Therefore, in this project, the target winning rate of the computer Othello we will develop is set to be 50%.

3. Objectives

The objective of this project to develop a computer Othello program with the following attributes:

- The game board configuration being the size of 8x8;
- A winning rate of 50% when playing against a moderate (computer) opponent;
- A winning rate of 35% when playing against a strong (computer) opponent;
- A user-friendly UI for easy playing

While the ultimate outcome is to deliver the above program, the key of this project is to allow the evaluation function of the program being constructed by the program itself without human logic. Under this circumstance, it is expected that the first workable outcome will not play the game very well. Therefore, our target is to achieve a winning rate of 5% against moderate computer opponent at the end of the second phase of the project, i.e. the completion of the preliminary implementation of the program.

4. Scope

In this project, we plan to develop a program playing Othello with the following specifications:

- The board size of the game being 8 x 8;
- Machine learning algorithms, in particular, deep learning neural network, will be used to modify the evaluation function to increase its accuracy;
- Selection of the next best move will be done by the traditional game search tree;
- Central processing units (CPUs), instead of graphics processing units (GPUs), will be used for the deep learning process.

The following features will be included if time is allowed:

- A graphical user interface (GUI) for the game;
- Selection of the next best move being implemented with machine learning algorithms.

5. Deliverables

1. A program with a user interface (UI) that plays the game Othello with the following features will be delivered:
    - Game of board size being 8x8;
    - Winning rate of 50% when playing against a moderate opponent;
    - Winning rate of 35% when playing against a strong opponent;
    - Evaluation function being updated with the result of the deep learning neural network training;
    - Next best move being selected from the game search tree.
2. A report of the results obtained with different parameters and the corresponding analysis will be presented.

6. Approach and Methodology

6.1 Software development model

A combination of the waterfall and agile model will be adopted for this project. (See Figure 1) Requirements will be collected and thorough study and analysis on the current solution and strategy will be done at the inception phase of the project. 100 sets of training data will also be collected for the future use in deep learning. We will then design the game search tree for selecting the next best move, implement the evaluation function for deep learning, and implement the game. After constructing the program, testing will be started with different sets of parameters. Each set of parameters will be used for the deep learning training for one week. Results will be evaluated and, if necessary, the values of the parameters will be re-chosen to improve the performance.

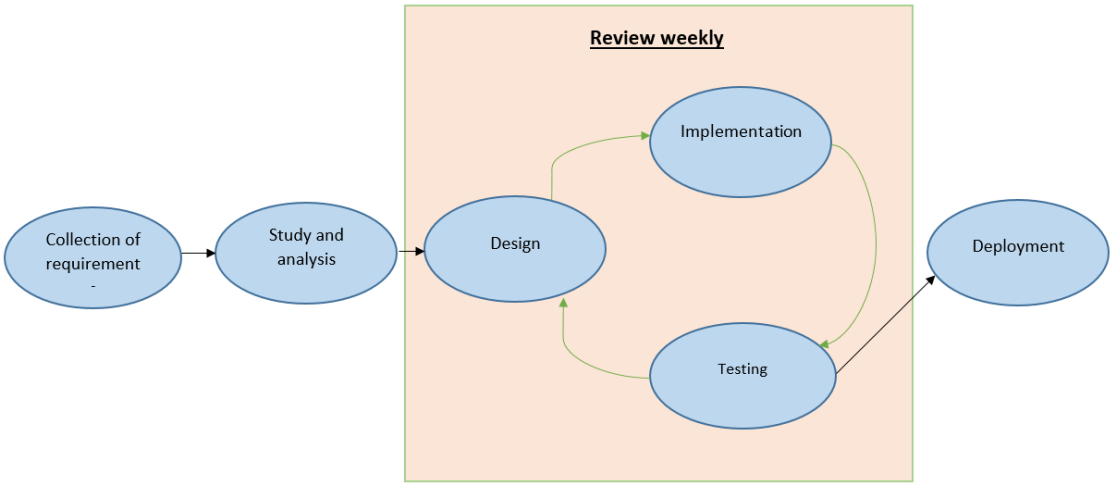

Figure 1: Software development life cycle of the project. At the inception phase, requirements for the project will be collected. Thorough study and analysis of the current solutions will also be done. The game search tree, evaluation and the game will then be designed and implemented in the second phase of the project. Testing will be done with different sets of parameters and the results will be reviewed weekly. Adjustment of the parameters will be made when necessary and testing will be done again until the winning rate of 50% is achieved with the program. The program will then be considered as completed and report on the result and for the project will be delivered.

6.2 Algorithms

6.2.1 Evaluation Function

The simple flow of building the evaluation function is demonstrated in the figure 2 below. The set of board configurations of different game will be pre-processed and used as the training data for the neural network. Once a learning model is built and configured, it can learn from the training data during the learning process and the evaluation function will be updated accordingly. An updated evaluation function for the board configuration will be obtained at the end of the learning process.

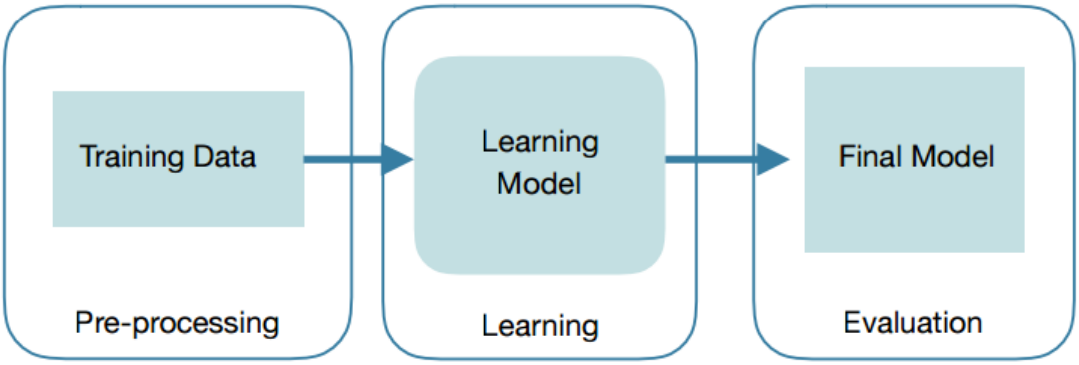

Figure 2: The flow of constructing evaluating function.

6.2.2 Search Algorithm

Brute-force approach of minimax algorithm will be applied in searching the game tree. It will also be supplemented with alpha-beta pruning, which eliminates the less favorable moves for the player to win, and hence, can help narrow down the number of possible searches.

6.3 Programming language

Python is chosen as the programming language for this project. Python, first released in 1991, is a general-purposed, interpreted, dynamic programming language. It supports different programming paradigms, including object-oriented, imperative and functional programming, which facilitate both the GUI design and the functional design of the game. [4] Besides, there are a number of machine learning libraries available in Python, which help implement the algorithms required in the project. In particular, Theano, a python library, can help writing the deep learning

model easily. [5] Together with the simplicity of Python, it can help simplify our code and make it clear and easy to understand.

6.4 Testing

The computer Othello developed will be tested against two types of opponents, moderate computer opponents and strong computer opponents. The choice of opponents at both levels for testing shall be discussed and agreed with the supervisor during the second phase of the project.

The two types of opponents will be tested with 50 games each. When playing against a moderate computer opponent, the targeted winning rate is 50%, while the targeted winning rate for playing against a strong computer opponent is 35%.

There are two reasons for the decision on the winning rate.

Firstly, it is believed that when both sides play the game perfectly, the game will very likely end with a draw. [3] Hence, when the winning rate of 50% is achieved, the computer Othello we developed can be said to be comparable with the existing computer Othello developed with the traditional method. Therefore, the winning rate of 50% when playing against a moderate opponent is chosen.

Secondly, the neural network may not be trained very well due to the limited time and other resources. Besides, the number of different board configurations we obtained as the training data may not be large enough for constructing a good evaluation function. These factors will affect the performance of the computer Othello we are going to develop, and hence, a lower winning rate of 35% is set when our computer Othello plays against a strong computer opponent.

7. Risk, Challenges & Mitigation

The time consumed for the following processes may be longer than expected, which may cause slippage of our schedule:

1. Assigning values to the board configurations.

   At the initial stage of deep learning process, data, in specific, different board configurations are required for the deep learning. Values need to be assigned to the board configurations to determine the chance of winning the game. The study of assigning such values may take up more time than expected, depending on the number of the board configurations we obtained in the data collection process and the how well we know the game. This may cause delay in schedule. Thorough study on the game will be done before assigning the values to the board configurations, which hopes to ease the problem.

2. Optimization of the deep learning process with different sets of parameters

   Parameters are set for the deep learning process to determine the rate of learning and adjustments. Regardless of the default setting of the rate of learning and the update of the parameters, the time taken for each learning process with a set of pre-defined parameters may take longer than expected due to the limited computational power of an ordinary central processing unit (CPU). In practice, graphics processing units (GPUs) are used for deep learning as it favors matrix multiplication, which is one of the most essential and important operations in deep learning algorithms. Compared with a CPU, the training time required for a GPU can be 8.5 to 9.0 times faster. [6] However, due to limited resources, we may not be able to use a GPU for the project, but instead, less powerful CPUs will be used. Hence, the time taken for each deep learning process will be lengthened, and hence the overall optimization process, which may result in the slippage of schedule. To address this problem, the evaluation function and the game will be implemented in December, and more time will be reserved for optimization.

8. Division of Work

The responsibilities and the proportion of work are listed below:

Member	Responsibilities	Proportion
Chan Lok Wang	Preparation of the training data	50% (50 games played)
	Design and implementation of the model in building the evaluation function of the game tree	50%
	Design and implementation of the decision making algorithm	100%
Yip Tsz Kwan	Preparation of the training data	50% (50 games played)
	Design and implementation of the model in building the evaluation function of the game tree	50%
	Front-end development, including the UI design and the graphical user interface (GUI) gameplay environment	100%

9. Schedule of Tasks

Date	Task(s)	Deliverables	Remarks
Sep 2016	1 Self-study on 1.1 Game tree 1.2 Deep learning neural network 1.3 Strategies of playing Othello 2 Preparation of training data		
2 Oct 2016		1. Detailed project plan 2. Webpage	
Mid-Oct 2016		1. 100 sets of training data	
Late-Oct 2016	1. Assignments of values for the board configurations obtained from training data		
16 Nov 2016		1. Completion of implementation on the game tree	
21 Dec 2016		1. Completion of implementation on the evaluation function and the game	
28 Dec 2016		1. Report on the result of the first set of pre-defined parameters for deep learning process	
9-13 Jan 2017	1. Interim presentation		
22 Jan 2017		1. Delivery of the interim report	
08 Feb 2017		1. Delivery of the UI of the game	
Mid-Mar 2017	1. Finalized optimization		
16 Apr 2017		1. Delivery of the final report	
02 May 2017		1. Project exhibition and presentation	

10. Conclusion

With deep learning neural network, the computer Othello we are going to develop can select the next best move better by updating the evaluation function itself throughout the learning process from playing the games over time, which can help improve the performance of the program. It also demonstrates the 'learning ability' of the computer, which may be helpful in the application of artificial intelligence.

Our target for this project is to deliver a computer Othello program with a winning rate of 50% when playing against a moderate component, and 35% when playing against a strong opponent.

However, there are some risks and challenges which may hinder the development and the achievement of our project, which should be well-noted.

Finally, we would like to thank our advisor, Dr. K.P. Chan, for his guidance and advice.

11. References

1. David Eppstein, ‘Minimax and negamax search’, David Eppstein. 17-Apr-1997. Available: https://www.ics.uci.edu/~eppstein/180a/970417.html. [Accessed 15-Sep-2016]
2. Emerging Technology from the arXiv, ‘Deep Learning Machine Teaches Itself Chess in 72 Hours, Plays at International Master Level’, MIT Technology Review, [14-Sep-2015] Available: https://www.technologyreview.com/s/541276/deep-learning-machine-teaches-itself-chess-in-72-hours-plays-at-international-master/. [Accessed 15-Sep-2026]
3. Othello Sky, ‘Chapter 11 Book openings’, Othello Sky. Available: http://www.soongsky.com/en/strategy2/ch11.php [Accessed: 17-Sep-2016]
4. A.M. Kuchling, Functional Programming HOWTO – Python 3.5.2 Documentation’, A. M. Kuchling. [Online] Available: https://docs.python.org/3/howto/functional.html. [Accessed 13-Sep-2016]
5. LISA lab, University of Montreal, ‘Deep Learning Tutorial’, LISA lab, University of Montreal. 01-Sep-2015 [Online]. [Accessed: 03-Sep-2016]
6. Larry Brown, ‘Deep Learning with GPUs’, Larry Brown. Jun-2015 [Online]. Available: http://www.nvidia.com/content/events/geoInt2015/LBrown_DL.pdf. [Accessed: 18-Sep-2016]