Deep learning with Othello

Application and analysis of deep neural networks on the evaluation function with Othello

Nian Xiaodong (3035087112)
Sun Peigen (3035084548)
Xu Chaoyi (3035084328)

Under supervision of Prof. Kwok-Ping Chan

Department of Computer Science
University of Hong Kong

Submission Date: 2017 Jan 20

Contact Information:

u3508711@hku.hk
sunbacon@hku.hk
nature@hku.hk
Abstract

Recently, deep learning is becoming prevalent in the AI field. However, currently most of the game AI are still using the manually extracted features. What if we apply the technology of deep learning to game AI? This report is going to discover the potential of deep neural network (DNN) to be the evaluation functions of the game Othello and describe the design, implementation as well as evaluation of our program. There are 6 hidden layers in our DNN and the neural network is trained by supervised learning. This report also introduces the way we measure the strength of the evaluation function. If the potential of DNN in game AI is proved, the way of building AI will have a evolution. By comparing the the different AI based on DNN and other methods, the applicability of using DNN as evaluation function has been verified. This finding may have a huge impact on game AI design.
Acknowledgement

We would like to express our special thanks of gratitude to our supervisor Prof. Kwok-Ping Chan as well as our principal Peter Mathieson who gave us the golden opportunity to do this wonderful project on the topic deep learning, which also helped us in doing a lot of Research and we came to know about so many new things we are really grateful to them.
Abbreviations

AI: Artificial Intelligence
DNN: Deep Neural Network
GPU: Graphic Processing Unit
GUI: Graphic User Interface
JSON: JavaScript Object Notation
tanh: hyperbolic tangent function
ASE: Average Squared Error (variance)
Application and analysis of deep neural networks on the evaluation function with Othello

Introduction

About half a year ago, AlphaGo became extremely famous due to a victory against the professional Go player, Mr. Lee Se-dol. Motivated by AlphaGo, the objective of our project is to develop a similar program, applying the same technologies as AlphaGo to play a similar game, Othello. The main technique used in our project is deep learning neural network. As a field which is developing astonishingly, deep learning benefits from the huge improvement of computational capability of modern processors and becomes the most popular research topic of artificial intelligence.

The basic rule of Othello is that the bounded disks will become the opponent's ones as illustrated in Graph 1.

(Graph 1: Illustration of Othello rules. After putting a new white disk, the bounded black one will become white)

It is obvious that the board of Othello is quite small and the number of legal moves during each step is limited. As a result, in average, both the total number of steps and the number of possible moves during one step are much smaller than Go. That is why we choose Othello: a simpler game will be easier for us to handle.

However, Othello still has the number of legal positions of at most $10^{28}$, and a game-tree complexity of approximately $10^{58}$ (Allis, 1994). So mathematically, Othello still remains unsolved. (A “solved game” is a game whose outcome can be correctly predicted from any position, assuming that both players play perfectly.)

In the following part, we will introduce the previous works in this field, the theories and algorithms we used, the progress and assessment of our project, and the future works we will accomplish.
Previous works

Despite the fact that Othello is unsolved, nothing can stop computer scientists from developing stronger and stronger Othello programs. *Iago* developed by Paul S. Rosenbloom in 1981 became the first program which beat the human world champion. But later in 1986, it was defeated “consistently” by *Bill*, which was developed by Kai-Fu Lee and Sanjoy Mahajan, adopting the concept of machine learning (quite shallow, though). However *Bill*, of course, is also far surpassed now. Nevertheless, the main ideas behind both *Iago* and *Bill* are worth studying and both of them have been patient teachers and qualified opponents of our program. In our future research and development, they will still be of great help.

Theoretical background

Since the final objective of our project is to develop a program which plays Othello, it can be simply categorized as a game AI development. When it comes to game AI, there is one thing which can not be avoided: the evaluation function.

An evaluation function is a method to evaluate actions based on a certain game board. If we define it as $E$, the features of gameboard as $G$, the next move as $a$ and the evaluated score as $S$, the equation can be written as

$$E(G, a) = S$$

Based on the scores of game boards provided by this function after potential moves, AI program can choose the move that will lead to the most beneficial situation for it. Without an effective evaluation function, the AI program cannot find the actual advantageous move and will lose step by step.

For example in Graph 2, if place A, B and C are all valid moves, the evaluation function should give different scores after different moves are taken. Based on experience, usually the corner is the best choice, followed by the edge and the middle. Thus, an ideal evaluation function should give the score as:

$$E(G, A) > E(G, B) > E(G, C)$$
For traditional methods of developing Othello AI including Iago and Bill mentioned above, features of the game board were focused on constructing a more precise evaluation function by programmers themselves manually. Logistello, which is one of the most famous Othello AI engines, used human-defined features to abstract useful information from the gameboard (Buro, 1998). It turned out that Logistello was able to beat the greatest human player and achieved great success.

However, in our DNN model, we did not use those traditional methods. No features were manually selected in our program. Instead, a deep neural network was trained to learn how to evaluate the game board by itself. Thus, we can use this neural network as the evaluation function to develop the AI program.

Scope

To complete this project, both software and hardware resources are essential.

There are a large number of programming languages supporting deep neural network, for example, C++, R, Matlab, etc. Among these languages we have selected Python to be our developing language. There are two reasons. Firstly, all our teammates are quite familiar with Python so that no time will be wasted on learning a new language. Due to the time limit of this project, we have to choose a language which can shorten the time spent on coding and debugging. Secondly, Python has the highest cost-effective value and the widest supporting packages related to deep neural network. In a word, Python has a rapid development cycle, which can spare us to focus more on the architecture of our models and expedite our entire development process.

What’s more, it was still necessary to make sure which learning package is to be used. After thorough consideration, we determined that Keras would be chosen as our framework. Keras as a new learning package is specially designed for deep learning and its main feature is the simplicity to build new models. Using Theano and TensorFlow (Theano is a widely used deep learning package and TensorFlow is a machine learning package developed by Google) as
computing backend, the speed of Keras is guaranteed. Moreover, we do not need to touch the complex inner model-build process when using Keras due to its high level integration.

In addition to software support mentioned above, we also need hardware equipment for this project. The hardware to be involved in the project is the graphics card. Graphics cards can accelerate the training speed of the neural networks to a great extent and shorten the development cycle, which can free us from keep waiting for results and spare us more time to redesign models. Currently, the graphic card we are going to use is Nvidia GT 640, as it is cheap and has sufficient computing ability. The possibility that we switch to another more powerful graphic cards still can not be ruled out depending on the actual scale of our model.

**Methodology**

In our project, two AI programs were built and both of them used game-tree search. The game-tree search algorithm used in our program is called alpha-beta search, which is widely used in AI design. By implementing the alpha-beta search into our program, the AI bot can predict some further steps in the future, guess the most possible counter-strategy of the opponent and choose its move which will lead the game to the most advantageous situation for it.

The difference between these two AI programs is the distinct evaluation function. The first AI program only served as a baseline, so the evaluation function of it was based on weighted square strategy (Rosenbloom, 1982), which is a rather simple evaluation strategy. This strategy is developed from the observations that occupying different places on the Othello game board has distinct influences to the game result. From earlier experience, the outer places such as the four sides, play much more important roles than those at the inner board. Especially, the corners are the most influential places as once been taken, they cannot be reoccupied by the opponent, thus they provide unimpeachable stability for the player who occupies them and can help to possess the sides and the inner board afterwards. According to the theory of this strategy, a scoring matrix storing the different importance of places is needed to evaluate the board. If we denote the scoring matrix as $M$, the evaluation function should be

$$E(B, a) = \sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij} \times B'_{ij}$$

where $B$ is the current game board and $B'$ is the board after taking action $a$. Here, a three-way representation is used to encode the game board. $B'_{ij}$ is $1$ if the place at $i^{th}$ row $j^{th}$ column is occupied by the current player, and is $-1$ if that place is occupied by the opposite player. If that place is not occupied by either, $B'_{ij}$ is $0$. As the game board of Othello has the size $8 \times 8$, $n = 8$ in this function.

The evaluation function of the second AI program is implicit. Instead of manually selecting features or defining strategy, a deep neural network is constructed to automatically learn how to evaluate the game board. At the current stage, the neural network with 3 convolutional layers and a fully-connected layer with a $tanh$ activation function is used to predict the evaluated score of the game board. We did not use max-pooling layers because the gameboard is relatively too small. The graphic illustration of the neural network is below (Graph 3).
This neural network was trained by supervised learning. The training data of the network were from the self-playing games of another Othello AI program - WZebra, which is one of the strongest Othello AIs in the world. This AI provides different levels of search depth, and we generated training games with six search steps, considering the balance of search strength and generating efficiency. Currently, over 4000 self-playing games with evaluation scores of each step were recorded as the training set. The scores provided by WZebra are generally within a range between -8 to +8 (as a reference of the result).

Deliverables

At the current stage, there are three main deliverables in this project according to the methodology mentioned above.

The first one is a game engine, which can interact with players’ moves, automatically mark the valid moves and change the game board if a valid move is made. It has two versions: Python and Javascript. The Python version is for faster calculation of AI program and only supports crude display function. The Javascript version can communicate with Python backend using JSON and has a beautiful and scrutable GUI. This GUI serves as a graphical input panel for us to tune the models and test their powerfulness.

The second one is a simple AI based on the traditional methods of feature extractions. This AI can play moves based on a weak evaluation function whose details have been discussed above. This AI bot is a simple sketch for us to judge the strength of our other models.

The third one is the AI based on deep neural networks. The way to construct it has also been illustrated above. At this stage, we only compared several simple models and no deeper explorations had been conducted. This AI needs further improvement and retuning to achieve higher performance.

Current status

So far, the first deliverable of the whole process has been accomplished. In detail, we have successfully made the graphical UI of the program, where users can already play Othello with another player just by clicking, and the whole front-end part has been deployed to the webpage distributed by the department (The GUI can be viewed and experienced in our
Now our team is working on the second deliverable. The prototypes of two AI programs based on weighted square strategy and DNN has been built already. The current network we have trained has a training ASE of 110.7027 and a validation ASE of 73.9561. Additionally, to measure the strength of the evaluation functions directly was difficult, so an alternative way was used in our assessment: let the AI based on the different evaluation functions play against each other and a higher winning rate would indicate a better evaluation function. Based on this idea, we designed the experiments letting these two AI programs play with each other.

In the first experiment, we let the AI based on neural networks compete with a random choice AI. The random choice AI simply chose from all possible valid moves randomly. We recorded the winning rate of different programs based on which took the first move. The results and winning rates are recorded in table 1 and 2 as below.

<table>
<thead>
<tr>
<th></th>
<th>DNN first</th>
<th>RC first</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random choice wins</td>
<td>104</td>
<td>83</td>
<td>187</td>
</tr>
<tr>
<td>Neural network wins</td>
<td>135</td>
<td>157</td>
<td>292</td>
</tr>
<tr>
<td>draw</td>
<td>11</td>
<td>10</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 1: Summary of games between Neural networks and Random choice

<table>
<thead>
<tr>
<th></th>
<th>Winning rate when moving first</th>
<th>Winning rate when moving later</th>
<th>Total winning rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random choice</td>
<td>33.20%</td>
<td>41.60%</td>
<td>37.40%</td>
</tr>
<tr>
<td>Neural network</td>
<td>54.00%</td>
<td>62.80%</td>
<td>58.40%</td>
</tr>
</tbody>
</table>

Table 2: Winning rate of Neural networks and Random choice

The AI based on neural network won considerably more games than the other one based on random choice and the winning rate of the AI based on neural network was much higher than that of traditional method. This experiment shows that the AI based on neural network do have "intelligence" and is stronger than random choice.

In the second experiment, we let our AI based on neural networks compete with the AI based on weighted square strategy. The results and winning rates are presented below in table 3 and 4.

<table>
<thead>
<tr>
<th></th>
<th>DNN first</th>
<th>TM first</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional method wins</td>
<td>111</td>
<td>167</td>
<td>278</td>
</tr>
<tr>
<td>Neural network wins</td>
<td>137</td>
<td>75</td>
<td>212</td>
</tr>
<tr>
<td>draw</td>
<td>2</td>
<td>8</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 3: Summary of games between Neural networks and Weighted Square Strategy

<table>
<thead>
<tr>
<th></th>
<th>Winning rate when moving first</th>
<th>Winning rate when moving later</th>
<th>Total winning rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighted Square Strategy</td>
<td>66.80%</td>
<td>44.40%</td>
<td>55.60%</td>
</tr>
<tr>
<td>Neural network</td>
<td>54.80%</td>
<td>30.00%</td>
<td>42.40%</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
</tbody>
</table>

Table 4: Winning rate of Neural networks and Weighted Square Strategy

In this experiment, the AI based on neural networks showed significant weakness when playing later. The strength of DNN AI is evaluated as "weaker than the weighted square strategy".

Combining the results from two experiments, we can conclude that DNN does help in analyzing the game board. However, finding ways to use DNN to construct a powerful AI on Othello still calls for further research.

**Future works**

Considering the current performance and the difficulties encountered, in the next step, there are two objectives.

One is to generate more training data. In practice we discovered that the training data that we have are still far from enough for training the DNN. Thus, we decide to make use of a mobile app “The Othello”, which can record the moves automatically and export them in one file. We are going to play with the AI by ourselves, thus generating more manuals as training data.

The other one is to improve the performance of our DNN model. After analyzing our models, three solutions have been come up with. First, the input and output of the DNN are supposed to be changed. By providing more information of the game board to the DNN as input, our model will have a clearer view of the game; by changing the distribution and activation function of the output, the DNN will adapt a different way of self-learning. Second, more layers will be added into our DNN. As more layers a DNN has, usually stronger its analytic power will be. Tentatively our DNN only have several layers, which restrict its power greatly. Last but most important, the application of reinforcement learning. With reinforcement learning, the model will be able to learn from the games that it play against itself. As our development moves forward, other tricks may also be included in our model to improve its strength. For example, some human-defined features such as "stable disks" may also be added as new input layers.

**Conclusion**

This report has described the idea and implementation of our project, whose objective is to adopt the technology of deep learning neural network to play Othello. Our expectation is that with the help of new technologies, the program developed by us can achieve, if not transcend, the level of traditional algorithms. Regrettably, this aim has not been accomplished yet. However, the overwhelming victory of our DNN AI against our traditional AI is quite inspiring. It gives us the confidence that our trained deep neural network is indeed intelligent and does have a strong (although still not tremendous enough) power in playing Othello as expected. Certainly, there is still something not enough in our current program, such as lack of training data and absence of reinforcement learning. In the future, some targeted improvement about these limitations will certainly be made, and many other steps which will improve the performance can also be taken, such as increasing the dimension of input, adding more layers to the network, etc. Nowadays the ability of AI in games is having a closer and closer relationship with user experience, hopefully our research can also cast some light over other
game AI engines.
References


