Intelligent Non-Player Character with Deep Learning

Meng Zhixiang, Zhang Haoze
Supervised by Prof. Michael Lyu
CUHK CSE FYP 2016 - 2017
Agenda

- Review
- Objective
  - Policy Network – Reinforcement Learning
  - Evaluation Network – Supervised Learning
- Results
- Discussion & Conclusion
Previously on our project...

- **NPC:**
  character not controlled by a player but by computer through artificial intelligence
Previously on our project...

- Policy Network – Supervised Learning

Game AI

- Policy Network
  - Predict probabilities of next moves

- Evaluation Network
  - Evaluate winning rate
Objective

- Policy Network – Reinforcement Learning
- Evaluation Network – Supervised Learning
- Selection Strategies
Intelligent Non-Player Character with Deep Learning
Version Iteration

Version 001: only trained by Supervised Learning

Version 018: Trained by Reinforcement Learning

Version 018.1: Add Evaluation Model

Version 018.2: Add Minimax Search
Piece Selector & Move Selector

Piece Selector NN Structure

- Input layer: 9 x 10 x 8
- First hidden layer: 9 x 10 x 32
- Second hidden layer: 9 x 10 x 128
- Output layer: 90

Move Selector NN Structure

- Input layer: 9 x 10 x 9
- First hidden layer: 9 x 10 x 32
- Second hidden layer: 9 x 10 x 128
- Output layer: 90
## Feature Channels

<table>
<thead>
<tr>
<th>Feature Channel</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Channel 1</td>
<td>Pieces belonging to different sides</td>
</tr>
<tr>
<td>Feature Channel 2</td>
<td>Pieces of Advisor type</td>
</tr>
<tr>
<td>Feature Channel 3</td>
<td>Pieces of Bishop type</td>
</tr>
<tr>
<td>Feature Channel 4</td>
<td>Pieces of Cannon type</td>
</tr>
<tr>
<td>Feature Channel 5</td>
<td>Pieces of King type</td>
</tr>
<tr>
<td>Feature Channel 6</td>
<td>Pieces of Knight type</td>
</tr>
<tr>
<td>Feature Channel 7</td>
<td>Pieces of Pawn type</td>
</tr>
<tr>
<td>Feature Channel 8</td>
<td>Pieces of Rock type</td>
</tr>
<tr>
<td>Feature Channel 9 (only for MoveSelector)</td>
<td>Valid moves for the selected piece</td>
</tr>
</tbody>
</table>
Output Sample

Current chessboard (black cannon move from green to red)

Output from piece selector (green circle on chessboard)

Output from move selector (red circle on chessboard)
Reinforcement Learning

- inspired by behaviour psychology
- exploration vs exploitation
- how to take actions
- to maximize the reward
Reinforcement Learning

- Assigning Rewards
  - Positive Reward: 1 for moves from winning side
  - Negative Reward: not used

- Compete with different middle version models
  - to avoid overfitting
Training

- change the opposite model roughly every 4,000 games
- in total around 40,000 games, over 2,000,000 moves
Testing

Intelligent Non-Player Character with Deep Learning
Results

Intelligent Non-Player Character with Deep Learning
Results

![Graph showing the winning rate vs previous version number. The graph indicates that the winning rate increases from version 1 to version 5, then decreases slightly to version 7, and then sharply decreases to version 9.]

Intelligent Non-Player Character with Deep Learning
Results

![Graph showing winning rate vs version number.]

Intelligent Non-Player Character with Deep Learning
Evaluation Network

- Why?
  - Only Policy Network is not enough
  - Need to evaluate the winning rate of a chessboard status

- Supervised Learning
  - Regression Problem
Evaluation Network
## Network Input

<table>
<thead>
<tr>
<th>Feature</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player Side</td>
<td>1</td>
</tr>
<tr>
<td>The Number of Pieces of Each Type</td>
<td>14</td>
</tr>
<tr>
<td>Pieces List (alive or not, xy-coordinates)</td>
<td>32 * 3</td>
</tr>
<tr>
<td>The number of valid moves for Rock, Cannon and Knight</td>
<td>12</td>
</tr>
<tr>
<td>Attack and Defend Map</td>
<td>90</td>
</tr>
</tbody>
</table>
Training

- How to get the target values?
  - one evaluation function from an open-source API
  - do some mapping, shrink the range

- Trained over 1,900,000 chessboard statuses

- Testing Results: loss -> ~ 20

| 0 ~ ±100 | 0 ~ ±100 |
| ±101 ~ ±700 | ±101 ~ ±170 |
| over ±9000 | ±200 |
How to use the models?

Feature Extractor -> Piece Selector -> Selection Strategy

Move Selector -> Decision Maker -> Evaluation Model

Selection Strategy -> Final Choice
Minimax Searching

- Select minimum and maximum value in turn
Selection Strategy

Piece Selector Prediction

Move Selector Prediction

Evaluation Network

MiniMax Searching

Intelligent Non-Player Character with Deep Learning
Selection Strategy Enhancement

- Eliminate moves with probabilities below a certain threshold at first

- Different max breadth for different layers
  - 24 -> 12 -> 12

- Different quota for different piece types

<table>
<thead>
<tr>
<th>Piece type</th>
<th>Quota 1</th>
<th>Quota 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Advisor</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Bishop</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Rock</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Cannon</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Knight</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Pawn</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Testing

- Aliyun Server
- Socket.io
- Multiple login
Results

- The winning rate is 76%, won 19 out of 25 games
- On average, it takes 26.3 moves to win

<table>
<thead>
<tr>
<th>Number of Games</th>
<th>Average Number of Moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win 19</td>
<td>26.3</td>
</tr>
<tr>
<td>Lose 6</td>
<td>37.5</td>
</tr>
</tbody>
</table>
Demo
Discussion & Conclusion

- Policy Network and Evaluation Network
- Supervised Learning and Reinforcement Learning
- performed much better than the model in Term 1
- can compete with ordinary people now

need further improvement:

- negative reward is not working in Reinforcement Learning
- continue to train the model
- try different model structures
Q&A