AlphaGo: The Program Ahead of Its Time

Lucas Buccafusca
IE 598

November 29th, 2016
Overview

- Introduction
- Monte-Carlo Tree Search
- Policy and Value Networks
  - Policy Network
    - Deep Reinforcement Learning
  - Value Network
  - Convolutional Neural Networks
- Results
Introduction

- Go originated 2,500+ years ago
- Currently over 40 million players
- “Success at Go requires the tactic of the soldier, the exactness of the mathematician, the imagination of the artist, the inspiration of the poet, the calm of the philosopher, and the greatest intelligence.” - Zhang Yunqi
Introduction

- Rules of Go

Capture

Territory
Introduction

- **Game Trees:**
  - A game tree is a directed graph whose nodes are positions in a game and whose edges are moves.
  - Fully exploring this tree allows for optimal play for simple games.
  - Complexity of tree: $O(b^d)$, where $b$ is the breadth (number of legal moves per position) and $d$ is its depth (the length of the game).
Introduction

- **Game Trees:**
  - Chess: $b \approx 35$, $d \approx 80$
  - Go: $b \leq 361$, $d \approx 150$
  - Size of tree for Go is more than the number of atoms of the universe! ($10^{170} \text{ vs } 10^{80}$)

- **Brute force intractable**
Monte-Carlo Tree Search

- Heuristic search algorithm for decision trees
- 4 phases:
  - Selection
  - Expansion
  - Simulation
  - Backpropagation
Monte-Carlo Tree Search

- Note that at each step, you implement policies based on some heuristic. Typically some form of optimization
- Many possible choices for each step
Policy and Value Networks

- Exhaustive Search
Policy and Value Networks

- Monte-Carlo Tree Search
Policy and Value Networks

- **Goal:** Reduce both breath and depth of search
- **How?**
  - Use policy network to explore better (and fewer) moves
    - How?
  - Use value network to estimate lower branches of tree
    - How?
Policy and Value Networks

- Reducing Breath: Policy Network
Policy and Value Networks

- **Policy Network**
  - Take a traditional approach of move analysis and instead change the problem to a classification problem
  - For a given board state and opponent play, try to build a set of probabilities to estimate best moves
Policy and Value Networks
Policy and Value Networks: Deep Reinforcement Learning

- **Supervised Learning**

  - **Training Data:** 30 million positions from human expert games
  - **Training Algorithm:** maximize log-likelihood of the action $\Delta \sigma$
  - **Training Time:** 4 weeks
  - **Results:** 57% accuracy of classification

\[ \Delta \sigma \propto \frac{\partial \log p_{\sigma}(a|s)}{\partial \sigma} \]
Policy and Value Networks: Deep Reinforcement Learning

- **Reinforced Learning**

- **Training Data:** 128,000+ games of self-play using policy network in 2 stages
- **Training Algorithm:** maximize wins of the action $\Delta \sigma$
- **Training Time:** 1 week
- **Results:** 80% accuracy vs. supervised learning

$$\Delta \sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial \sigma} \cdot z$$
Policy and Value Networks

- Reducing Depth: Value Network
Policy and Value Networks

- **Value Network**
  - Biggest issue is overfitting to strongly correlated positions within games
  - Given board states, estimate probability of victory
Policy and Value Networks
Policy and Value Networks: Deep Reinforcement Learning

- Reinforced Learning

- Training Data: 30 million games of self-play
- Training Algorithm: minimize mean-squared error by stochastic gradient descent
- Training Time: 1 week
- Results: AlphaGo ready for play against pros
Policy and Value Networks: Improvement
Monte-Carlo Tree Search in AlphaGo

- **Selection:**

\[
Q + u(P) \quad \text{max} \quad Q + u(P)
\]

\[
Q + u(P) \quad \text{max} \quad Q + u(P)
\]

- \( P \) prior probability
- \( Q \) action value
- \( u(P) \propto P/N \)
Monte-Carlo Tree Search in AlphaGo

- Expansion:
Monte-Carlo Tree Search in AlphaGo

- Simulation:
Monte-Carlo Tree Search in AlphaGo

- Backtracking:
Convolutional Neural Networks

- In practice, properly evaluating board states to build the policy and value networks is very complex.
- Many features to capture (color of pieces, time since a move was played, number of opponent pieces captured, etc.)
Convolutional Neural Networks

- For both the policy and value networks, a multilayer convolutional neural network was built.
- You take your matrix and convolve it with a filter to get a newer, smaller matrix with features of the filter included.
Convolutional Neural Networks

- For example:

<table>
<thead>
<tr>
<th></th>
<th>1 x0</th>
<th>1 x1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 x0</td>
<td>1 x1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0 x0</td>
<td>1 x1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1 x0</td>
<td>1 x1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1 x1</td>
<td>1 x1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

![Convolved Feature](image)

*Image*
Convolutional Neural Networks
## Convolutional Neural Networks

- For both the value and policy networks, there were 12 layer convolutional neural networks built for both, each with 4 feature planes.

<table>
<thead>
<tr>
<th>Feature</th>
<th># of planes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stone colour</td>
<td>3</td>
<td>Player stone / opponent stone / empty</td>
</tr>
<tr>
<td>Ones</td>
<td>1</td>
<td>A constant plane filled with 1</td>
</tr>
<tr>
<td>Turns since</td>
<td>8</td>
<td>How many turns since a move was played</td>
</tr>
<tr>
<td>Liberties</td>
<td>8</td>
<td>Number of liberties (empty adjacent points)</td>
</tr>
<tr>
<td>Capture size</td>
<td>8</td>
<td>How many opponent stones would be captured</td>
</tr>
<tr>
<td>Self-atari size</td>
<td>8</td>
<td>How many of own stones would be captured</td>
</tr>
<tr>
<td>Liberties after move</td>
<td>8</td>
<td>Number of liberties after this move is played</td>
</tr>
<tr>
<td>Ladder capture</td>
<td>1</td>
<td>Whether a move at this point is a successful ladder capture</td>
</tr>
<tr>
<td>Ladder escape</td>
<td>1</td>
<td>Whether a move at this point is a successful ladder escape</td>
</tr>
<tr>
<td>Sensibleness</td>
<td>1</td>
<td>Whether a move is legal and does not fill its own eyes</td>
</tr>
<tr>
<td>Zeros</td>
<td>1</td>
<td>A constant plane filled with 0</td>
</tr>
<tr>
<td>Player color</td>
<td>1</td>
<td>Whether current player is black</td>
</tr>
</tbody>
</table>
Results

- AlphaGo covered here faced Fan Hui in October 2015
- Won 5-0
- Was the first time a program beat a professional on a full 19x19 board with no handicaps
Results

- AlphaGo was then updated in Seoul to play against top professional players
- Played against Lee Sedol in March 2016
- Won 4-1
- Lee Sedol essentially winning one match on move 78 with “God’s Move”
- World was shocked, this result not expected for at least 10 more years
Questions?
THANK YOU
References/Image Pulls

- “Mastering the game of Go with deep neural networks and tree search” D. Silver, A. Huang, C.J. Maddison, et. al. Nature 529 (7587), 484-489
- “An Intuitive Explanation of Convolutional Neural Networks” Blog Post, Ujjwal Karn, https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
- “Deep Reinforcement Learning/AlphaGo presentation”. David Silver IJCAI 2016 Invited Talk