AlphaGo Zero & AlphaZero

Mastering Go, Chess and Shogi without human knowledge
Silver et al. 2017-2018

Presenter: Philipp Wimmer
Outline

• Timeline
• AlphaGo Zero
  – Training Pipeline
  – Modified MCTS
  – Reasons for better performance
• AlphaZero
  – Generalization to Chess/Shogi
  – AlphaZero vs Stockfish
• Conclusion
• 2012: Crazy Stone
  - MCTS search with handcrafted heuristic
  - Professional level play

• 2015: AlphaGo Fan
  - MCTS with Value and Policy Network
  - Defeated European Go champion Fan Hui (2-dan)

• 2016: AlphaGo Lee
  - Larger networks than AlphaGo Fan
  - Added selfplay of policy network
  - Won 3/1 against Lee Sedol (9-dan)

• 2017: AlphaGo Master
  - Use single network for both policy and value
  - Using ResNet instead of Convolutional NN
  - Won 60/0 against team of professional players

• 2018 AlphaGo Zero
  - Trained from scratch

• 2018 AlphaZero
  - Generalized to Chess & Shogi
AlphaGo Zero: learning from scratch

• No human knowledge
  – Trained by self-play reinforcement learning from scratch
  – Only raw board as input

• Single neural network
  – Policy and value networks are combined into single NN

• Simpler (cheaper) search during gameplay
  – Instead of Monte-Carlo rollouts, only uses NN to evaluate

Less complex and more general => AlphaZero (also plays Chess, Shogi, ...)
Learning Pipeline

1. Selfplay workers write states to the replay buffer (500,000 states).
2. Training worker reads states from the replay buffer.
3. Training worker copies the current model to create a new model.
4. Training worker updates the current model based on the new model.
5. Evaluator evaluates the current model every 1000 iterations.
6. If the evaluator considers the current model to be improved, the new model becomes the current model.
Learning Pipeline

- **Replay Buffer** (500,000 states)
  - Read states
  - Write states

- **Selfplay Workers**
  - Copy

- **Current Model**
  - Update
  - Play

- **Training Worker**
  - Copy

- **Evaluator**
  - Every 1000 iterations
  - New Model
Policy/Value-Network

Current position of black’s stones

19 x 19 x 17 stack

... and for the previous 7 time periods

Current position of white’s stones

1 if black stone here
0 if black stone not here

1 1 1
1 0 0
0 0 1

All 1 if black to play
All 0 if white to play

... and for the previous 7 time periods

https://medium.com/applied-data-science/alphago-zero-explained-in-one-diagram-365f5abf67e0
Learning Pipeline

- replay buffer (500,000 states)
  - write states
  - read states
- training worker
  - training
  - copy
- new model
  - play
- current model
  - update
  - copy
- selfplay workers
- evaluator
  - every 1000 iterations
Modified MCTS

Each edge stores: \{ N(s, a), W(s, a), Q(s, a), P(s, a) \}

- visit count
- total action value
- mean action value
- prior probability
Select

- Select action according to PUCT

\[
U(s,a) = c_{puct} P(s,a) \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)}
\]

\[
a_t = \arg \max_a (Q(s_t, a) + U(s_t, a))
\]

$c_{puct}$ : level of exploration
Expand

- Evaluate NN at leaf node:
  \[(d(p), v) = f_\theta(d(s_L))\]
- Insert new edge:
  \[\{N(s_L, a) = 0, W(s_L, a) = 0, Q(s_L, a) = 0, P(s_L, a) = p_a\}\]
- Backup value
Backup

• Increment visit counts

\[ N(s_t, a_t) = N(s_t, a_t) + 1 \]

• Add value to action value

\[ W(s_t, a_t) = W(s_t, a_t) + v \]

• Update Mean action value

\[ Q(s_t, a_t) = \frac{W(s_t, a_t)}{N(s_t, a_t)} \]
Play

\[
\pi(a|s_t) = \frac{N(s_t, a)^{1/\tau}}{\sum_b N(s_t, b)^{1/\tau}}
\]
Policy iteration

- **replay buffer (500,000 states)**
- **selfplay workers**
- **training worker**
- **current model**
- **new model**
- **evaluator every 1000 iterations**

- **write states**
- **read states**
- **copy**
- **training**
- **update**
- **play**
$(p,v) = f_\theta(s)$

- $p$: policy
- $\pi$: MCTS probabilities
- $v$: value
- $z$: outcome of game

Loss-function:

$$l = (z - v)^2 - \pi^T \log p + c \|\theta\|^2$$

- $\ell^2$-loss
- Cross-entropy
- $\ell^2$-weight normalization
Why is it better?

- MCTS search in training loop provides stable gradient for training
  - Augmented policy is always better at predicting the best move
Supervised vs Reinforcement learning
Why is it better?

• MCTS search in training loop provides stable gradient for training
  - Augmented policy is always better at predicting the best move

• ResNets instead of ConvNets
  - Ability to train even deeper models

• Same network for Policy and Value
  - Multi-task learning with hard parameter sharing regularizes training and prevents overfitting
• 2012: Crazy Stone
  – MCTS search with handcrafted heuristic
  – Professional level play

• 2015: AlphaGo Fan
  – MCTS with Value and Policy Network
  – Defeated European Go champion Fan Hui (2-dan)

• 2016: AlphaGo Lee
  – Larger networks than AlphaGo Fan
  – Added selfplay of policy network
  – Won 3/1 against Lee Sedol (9-dan)

• 2017: AlphaGo Master
  – Use single network for both policy and value
  – Using ResNet instead of Convolutional NN
  – Won 60/0 against team of professional players

• 2018 AlphaGo Zero
  – Trained from scratch

• 2018 AlphaZero
  – Generalized to Chess & Shogi
Comparison to human play

- Superhuman performance
- Learned to play human Joseki
AlphaGo Zero vs AlphaZero

• Absence of human knowledge made transfer to Shogi and Chess very easy
• No change to NN architecture
• Only raw board states as input
• No evaluator → Continuous update of NN
# Go vs Chess/Shogi

<table>
<thead>
<tr>
<th>Rules</th>
<th>Go</th>
<th>Chess/Shogi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation invariance</td>
<td>Yes</td>
<td>Partial</td>
</tr>
<tr>
<td>Locality</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Symmetry</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Action space</td>
<td>Simple</td>
<td>Compound</td>
</tr>
<tr>
<td>Game outcomes</td>
<td>Probability of winning</td>
<td>Win / Loss / Draw</td>
</tr>
</tbody>
</table>
Performance of AlphaZero

A

Chess

Elo vs Thousands of Steps

AlphaZero

Stockfish

B

Shogi

Elo vs Thousands of Steps

AlphaZero

Elmo

C

Go

Elo vs Thousands of Steps

AlphaZero

AlphaGo Zero

AlphaGo Lee
AlphaZero vs Stockfish

- Stockfish: Alpha Beta Pruning with handcrafted heuristics, endplay-tables, opening book, etc…
- Stockfish 60’000’000 Moves/Second
- AlphaZero 60’000 Moves/Second
AlphaZero vs. Stockfish

1/100 time
1/30 time
1/10 time
1/3 time
same time

W: 29.0%  D: 70.6%  L: 0.4%

W: 2.0%  D: 97.2%  L: 0.8%
Conclusion

● Playing smart is better than brute-force
● Generality is better than handcrafting features
● Not injecting human knowledge promotes generality
● Multitask learning prevents overfitting
References

- AlphaGo Zero Cheat Sheet: https://medium.com/applied-data-science/alphago-zero-explained-in-one-diagram-365f5abf67e0