Stochastic Planning in Games

Peter Müller
Why Games?

0 0 0
0 O X
X X

0 0
0 O
X X

- Chess board
- Tic Tac Toe
- Maze game
Tic-Tac-Toe
Tic-Tac-Toe
Tic-Tac-Toe
Tic-Tac-Toe
Tic-Tac-Toe
Tic-Tac-Toe
Tic-Tac-Toe
Tic-Tac-Toe
Tic-Tac-Toe
Tic-Tac-Toe

1 1 0 0 0
1 0 1 -1 0 -1 -1 0 0 1 1 -1 -1
1 0 1 -1 0 -1 -1 0 0 1 1 -1 -1
Tic-Tac-Toe
COMPLETE MAP OF OPTIMAL TIC-TAC-TOE MOVES

YOUR MOVE IS GIVEN BY THE POSITION OF THE LARGEST RED SYMBOL ON THE GRID. WHEN YOUR OPPONENT PICKS A MOVE, ZOOM IN ON THE REGION OF THE GRID WHERE THEY WENT. REPEAT.
Chess

- Branching Factor: 35
- Game Length: 80
Evaluation Function

White: 2x Bishop (3 points) + 5x Pawn (1 point) = 11 points
Evaluation Function

White: 2x Bishop (3 points) + 5x Pawn (1 point) = 11 points

Black: 1x Rook (5 points) + 1x Pawn (1 point) = 6 points
Evaluation Function

White: 2x Bishop (3 points) + 5x Pawn (1 point) = 11 points

Black: 1x Rook (5 points) + 1x Pawn (1 point) = 6 points

Eval: 11 - 6 = 5 points
Evaluation Function

White: 2x Bishop (3 points) + 5x Pawn (1 point) = 11 points

Black: 1x Rook (5 points) + 1x Pawn (1 point) = 6 points

Eval: 11 - 6 = 5 points

Equal

Black advantage (min player)  Equal  White advantage (max player)
Evaluation Function

White: 2x Bishop (3 points) + 5x Pawn (1 point) = 11 points

Black: 1x Rook (5 points) + 1x Pawn (1 point) = 6 points

Eval: 11 - 6 = 5 points

-2 -1 0 +1 +2

Black advantage (min player) Equal White advantage (max player)
Max

```
3
/  
-/  3  5  1
```

This is a binary tree diagram with nodes labeled with the values -1, 3, 5, and 1.
Min

1. 3
2. 3 5 1
3. -1 3 5 1

Max

1. 3
2. 5
3. -6 -4
Min

3

Max

3

3

5

≥5

-1

3

5

Gray
\[ \min \geq 5 \]
Min

Max

\[ \text{Max: } \geq 5 \]

\[ \text{Min: } \geq 5 \]
Go

- Branching Factor: 250
- Game Length: 150
Monte Carlo Tree Search
Monte Carlo Tree Search

1. Selection (Tree Traversal)

\[ UCB1(s_i) = \frac{w_i}{n_i} + C \sqrt{\frac{\ln N}{n_i}} \]

2. Expansion

3. Simulation (Rollout)

4. Backpropagation
Monte Carlo Tree Search

Exploitation vs. Exploration

• Exploit promising actions
• Explore little known actions
Monte Carlo Tree Search

Exploitation vs. Exploration

• Exploit promising actions
• Explore little known actions

\[ UCB1(s_i) = \frac{w_i}{n_i} + C \sqrt{\frac{\ln N}{n_i}} \]
Monte Carlo Tree Search

1. Selection (Tree Traversal)

\[ UCB1(s_i) = \frac{w_i}{n_i} + C \sqrt{\frac{\ln N}{n_i}} \]

2. Expansion

3. Simulation (Rollout)

4. Backpropagation
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}} \]

- \( s_0 \)
  - \( w_0 = 0 \)
  - \( n_0 = 0 \)
- \( s_1 \)
  - \( w_1 = 0 \)
  - \( n_1 = 0 \)
- \( s_2 \)
  - \( w_2 = 0 \)
  - \( n_2 = 0 \)
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]

- \( s_0 \) with \( w_0 = 0 \) and \( n_0 = 0 \)
- \( s_1 \) with \( w_1 = 0 \) and \( n_1 = 0 \)
- \( s_2 \) with \( w_2 = 0 \) and \( n_2 = 0 \)
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}} \]

- \( s_0 \) with 1 child:
  - \( s_1 \):
    - \( w_1 = 1 \)
    - \( n_1 = 1 \)
  - \( s_2 \):
    - \( w_2 = 0 \)
    - \( n_2 = 0 \)
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCBI(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]

1. \( s_0 \)
   - \( w_0 = 1 \)
   - \( n_0 = 2 \)

2. \( s_1 \)
   - \( w_1 = 1 \)
   - \( n_1 = 1 \)

3. \( s_2 \)
   - \( w_2 = 0 \)
   - \( n_2 = 1 \)
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]

- \( s_0 \):
  - \( w_0 = 1 \)
  - \( n_0 = 2 \)

- \( s_1 \):
  - \( w_1 = 1 \)
  - \( n_1 = 1 \)

- \( s_2 \):
  - \( w_2 = 0 \)
  - \( n_2 = 1 \)

- \( s_3 \):
  - \( w_3 = 0 \)
  - \( n_3 = 0 \)

- \( s_4 \):
  - \( w_4 = 0 \)
  - \( n_4 = 0 \)

- \( s_t \): win
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

$U_{CB1}(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$

- $s_0$: $w_0 = 1$, $n_0 = 2$
- $s_1$: $w_1 = 2$, $n_1 = 2$
- $s_2$: $w_2 = 0$, $n_2 = 1$
- $s_3$: $w_3 = 1$, $n_3 = 1$
- $s_4$: $w_4 = 0$, $n_4 = 0$
- $s_t$: win
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

$$UCB1(s_i) = \frac{w_i}{n_i} + 2\sqrt{\frac{\ln N}{n_i}}$$
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]
MCTS

1. Selection
2. Expansion
3. Simulation
4. Backpropagation

\[ UCB1(s_i) = \frac{w_i}{n_i} + 2 \sqrt{\frac{\ln N}{n_i}} \]
Value Network

How well are we doing?
Policy Network

What are the most likely actions?
Policy Network

What are the most likely actions?
**AlphaGo**

A rollout policy, SL policy network, RL policy network, and value network are shown. The rollout policy (\( p_\pi \)) is connected to the SL policy network (\( p_\alpha \)) via a policy gradient. The RL policy network (\( p_\rho \)) is connected to the value network (\( v_\theta \)).

**Data**
- Human expert positions
- Self-play positions

**Neural network**
- Policy network (\( p_{\alpha|\rho}(a|s) \))
- Value network (\( v_\theta(s') \))
AlphaGo MCTS

\[ u(a) = v(a) + p(a) \cdot p_b c \]
\[ u(a) = v(a) + p(a) \cdot pb_c \]
\[ u(a) = v(a) + p(a) \cdot pb_c \]
AlphaGo MCTS

\[ u(a) = v(a) + p(a) \cdot p b_c \]
AlphaGo MCTS

\[ u(a) = v(a) + p(a) \cdot pb_c \]
AlphaGo MCTS

\[ u(a) = v(a) + p(a) \cdot pb_c \]
**Domains**

**Knowledge**

---

**AlphaGo** becomes the first program to master Go using neural networks and tree search
(Jan 2016, Nature)

**AlphaGo Zero** learns to play completely on its own, without human knowledge
(Oct 2017, Nature)

**AlphaZero** masters three perfect information games using a single algorithm for all games
(Dec 2018, Science)
AlphaGo Zero MCTS
AlphaZero Training

\[(p, v) = f_\theta(s), \quad l = (z - v)^2 - \pi^T \log p + c ||\theta||^2\]
AlphaGo Zero Results
Atari

- Image Input
- No Access to rules
MuZero Planning
MuZero Planning

representation $h_\theta(o_1, \ldots, o_t) = s^0$
MuZero Planning

representation \[ h_\theta(o_1, \ldots, o_t) = s^0 \]

prediction \[ f_\theta(s^k) = p^k, v^k \]
MuZero Planning

representation: \( h_\theta(o_1, \ldots, o_t) = s^0 \)

prediction: \( f_\theta(s^k) = p^k, v^k \)

dynamics: \( g_\theta(s^{k-1}, a^k) = r^k, s^k \)
MuZero Planning

representation \[ h_\theta(o_1, \ldots, o_t) = s^0 \]
prediction \[ f_\theta(s^k) = p^k, v^k \]
dynamics \[ g_\theta(s^{k-1}, a^k) = r^k, s^k \]
MuZero Training Data Generation
MuZero Training

\[ l_t(\theta) = \sum_{k=0}^{K} l^p(\pi_{t+k}, p^k_t) + \sum_{k=0}^{K} l^v(z_{t+k}, v^k_t) + \sum_{k=1}^{K} l^r(u_{t+k}, r^k_t) + c\|\theta\|^2 \]
MuZero Training

\[
l_t(\theta) = \sum_{k=0}^{K} l^p(\pi_{t+k}, p_t^k) + \sum_{k=0}^{K} l^v(z_{t+k}, v_t^k) + \sum_{k=1}^{K} l^r(u_{t+k}, r_t^k) + c\|\theta\|^2
\]
MuZero Training

$$l_t(\theta) = \sum_{k=0}^{K} l^p(\pi_{t+k}, p^k_t) + \sum_{k=0}^{K} l^v(z_{t+k}, v^k_t) + \sum_{k=1}^{K} l^r(u_{t+k}, r^k_t) + c\|\theta\|^2$$
MuZero Training

\[ l_t(\theta) = \sum_{k=0}^{K} l^p(\pi_{t+k}, p^k_t) + \sum_{k=0}^{K} l^v(z_{t+k}, v^k_t) + \sum_{k=1}^{K} l^r(u_{t+k}, r^k_t) + c\|\theta\|^2 \]
MuZero Training

\[ l_t(\theta) = \sum_{k=0}^{K} l^p(\pi_{t+k}, p_t^k) + \sum_{k=0}^{K} l^v(z_{t+k}, v_t^k) + \sum_{k=1}^{K} l^r(u_{t+k}, r_t^k) + c||\theta||^2 \]
MuZero Training

\[
l_t(\theta) = \sum_{k=0}^{K} l^p(\pi_{t+k}, p_t^k) + \sum_{k=0}^{K} l^v(z_{t+k}, v_t^k) + \sum_{k=1}^{K} l^r(u_{t+k}, r_t^k) + c\|\theta\|^2
\]
MuZero Results
Summary
References

- 2020-10-22 MuZero ICAPS talk.pdf
- AlphaZero: Shedding new light on the grand games of chess, shogi and Go
- MuZero: Mastering Go, chess, shogi and Atari without rules
- https://www.nature.com/articles/nature16961.pdf
- https://www.nature.com/articles/nature24270.epdf?author_access_token=VJXbVjaSHxFoctQ4p2k4tRgN0jAjWel9jnR3ZoTv0PVW4gB86EEpGqTRDtpIz-2rmo8-KG06qqVobU5NSCFeHILHcVFUEmsbvWS-IxjqQGg98faovwjsxETUgZAUMnRQ