AlphaGo

Shusen Wang

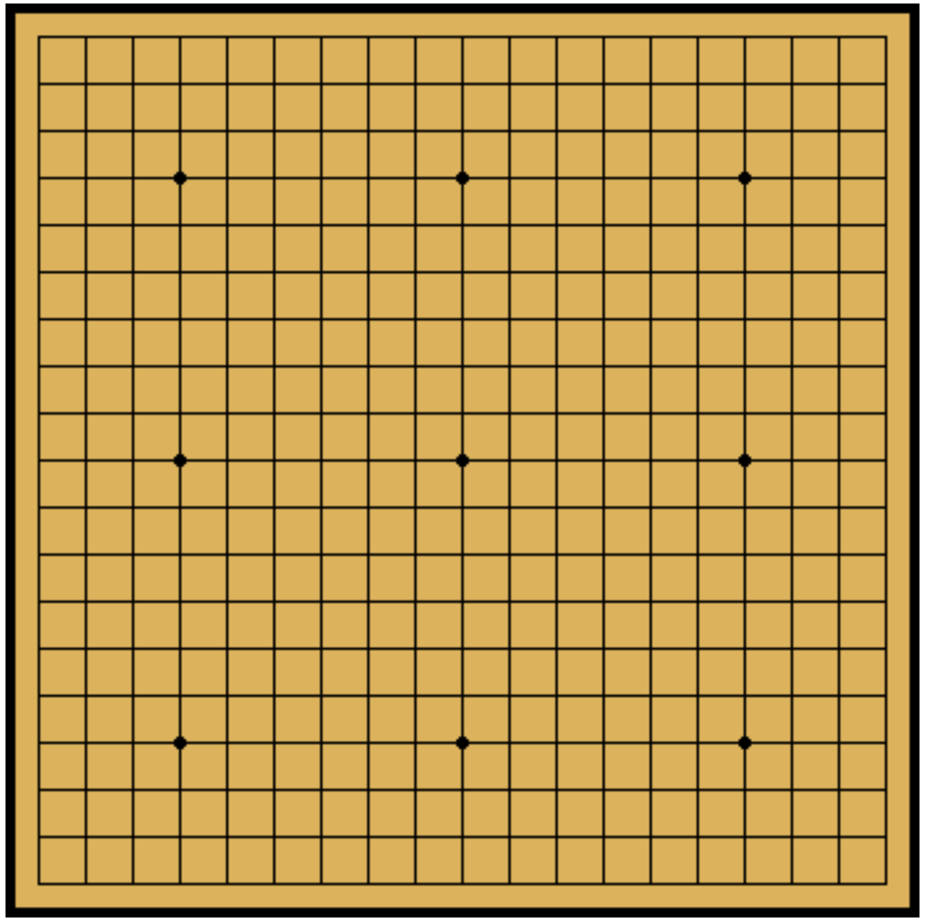
Disclaimer

- What taught in this lecture is not exactly the same to the original AlphaGo papers [1,2].
- There are simplifications here.

Reference

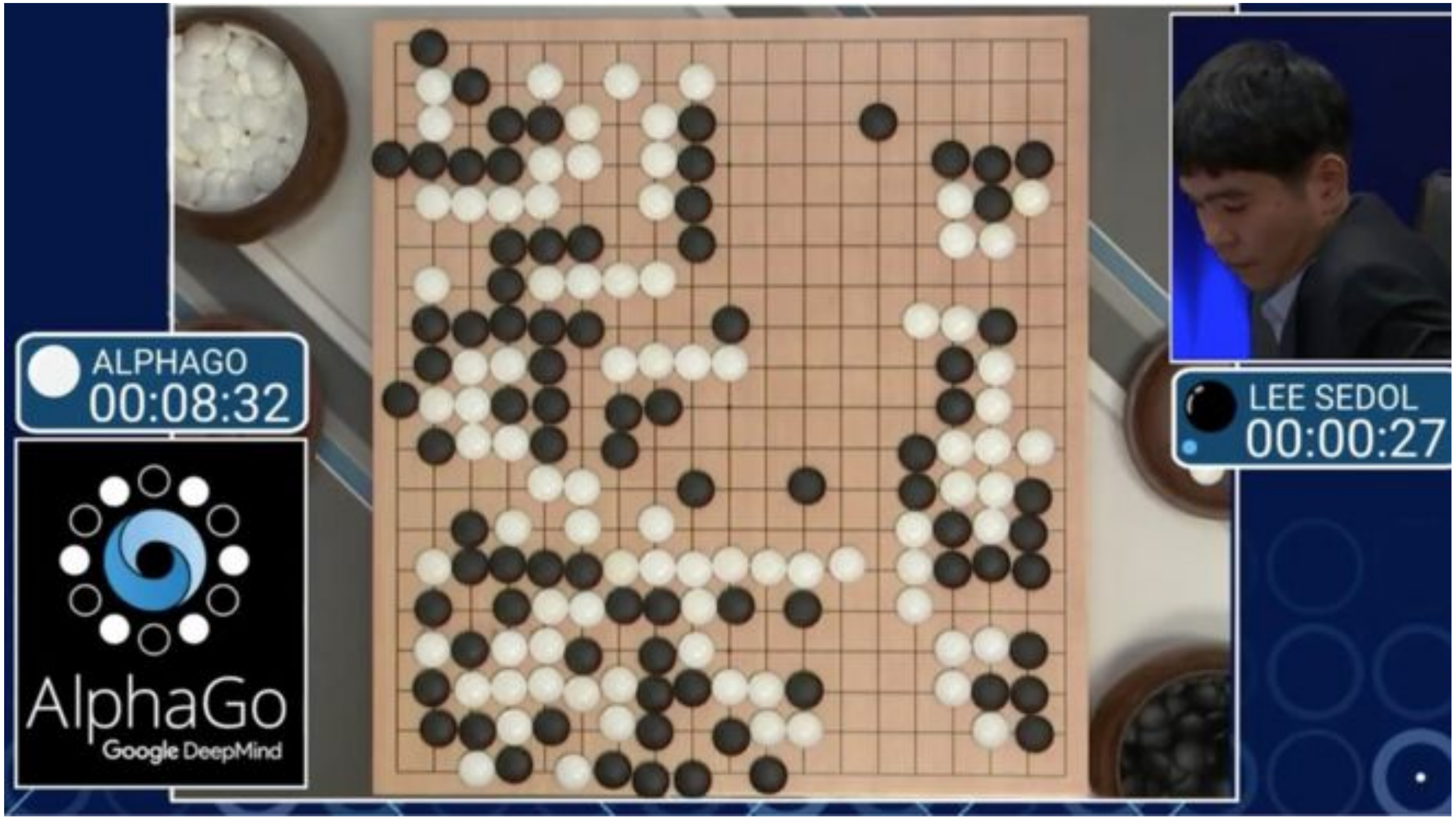
1. Silver and others: Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.
2. Silver and others: Mastering the game of Go without human knowledge. *Nature*, 2017.

Go Game



- The standard Go board has a 19×19 grid of lines, containing 361 points.
- State: arrangement of black, white, and space.
  - State s can be a $19 \times 19 \times 2$ tensor of 0 or 1.
  - (AlphaGo actually uses a $19 \times 19 \times 48$ tensor to store other information.)
- Action: place a stone on a vacant point.
  - Action space: $\mathcal{A} \subset \{1, 2, 3, \cdots, 361\}$.
- Go is very complex.
  - Number of possible sequence of actions is $\boxed{10^{170}}$.

AlphaGo



High-Level Ideas

Training and Execution

Training in 3 steps:

1. Initialize policy network using behavior cloning.监督学习 (Supervised learning from human experience.)
2. Train the policy network using policy gradient. (Two policy networks play against each other.)
3. After training the policy network, use it to train a value network.

Training and Execution

Training in 3 steps:

1. Initialize policy network using behavior cloning. (Supervised learning from human experience.)
2. Train the policy network using policy gradient. (Two policy networks play against each other.)
3. After training the policy network, use it to train a value network.

Execution (actually play Go games):

- Do Monte Carlo Tree Search (MCTS) using the policy and value networks.

Policy Network

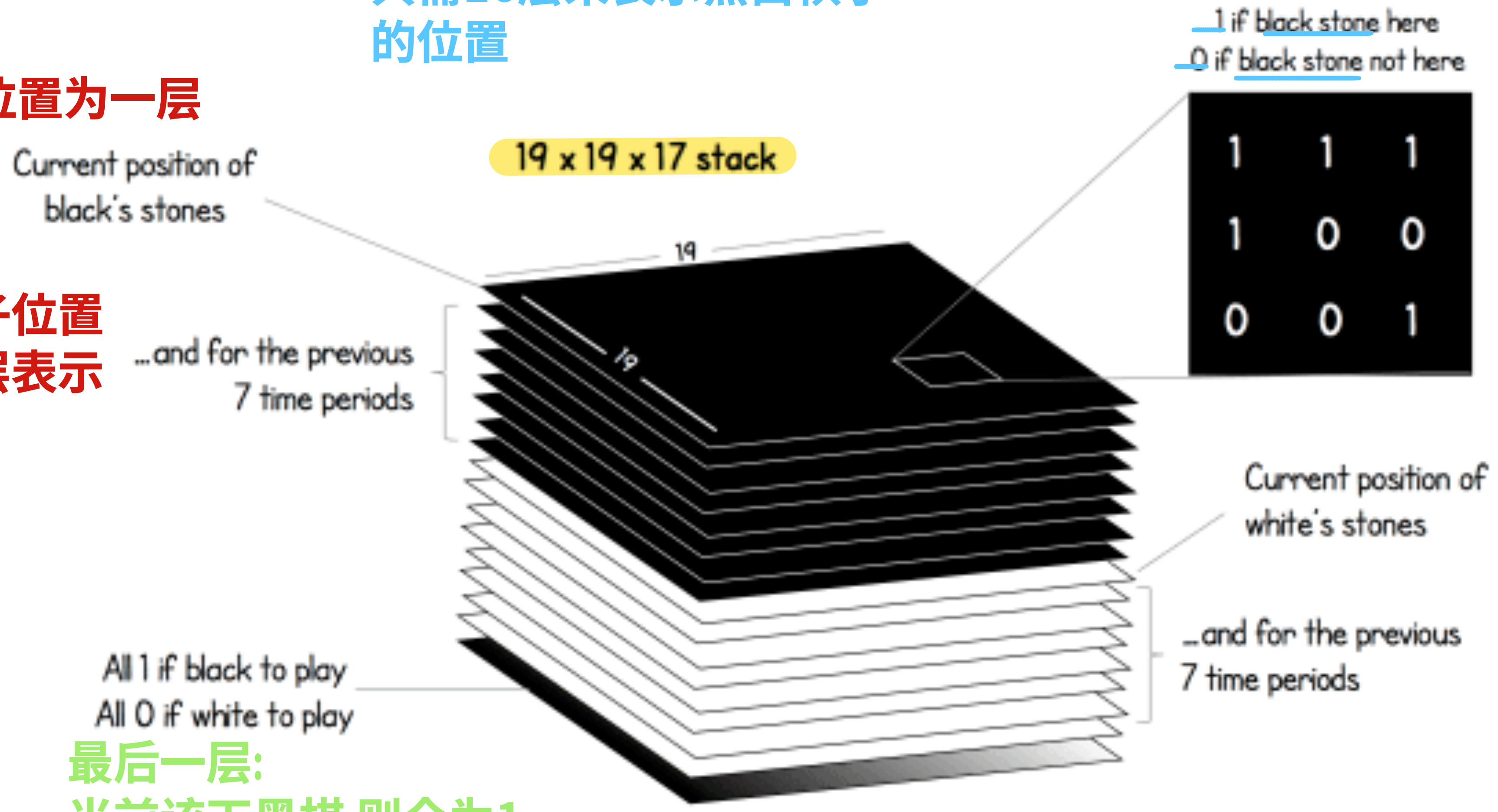
State (of AlphaGo Zero)

共需16层来表示黑白棋子的位置

当前的黑子位置为一层

前7步黑子位置分别用1层表示共七层

1 黑
0 白

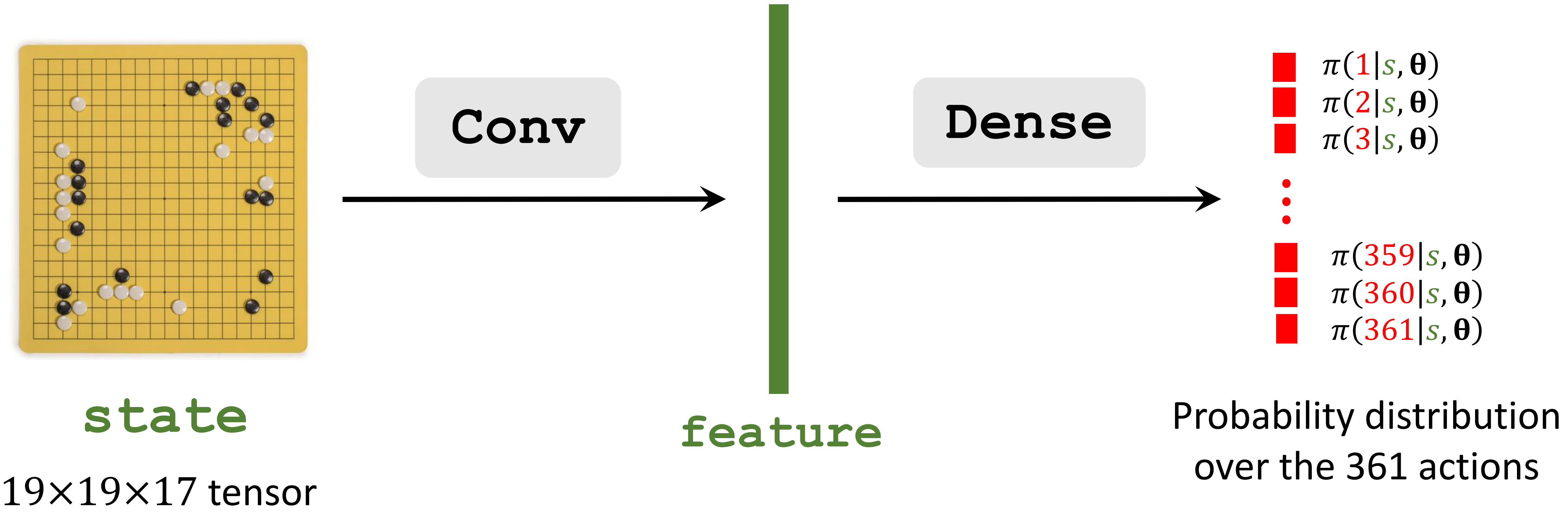


1 if black stone here
0 if black stone not here
19 x 19 x 17 stack
Current position of black's stones
19
19
1 1 1
1 0 0
0 0 1
...and for the previous 7 time periods
Current position of white's stones
...and for the previous 7 time periods
All 1 if black to play
All 0 if white to play

最后一层:
当前该下黑棋,则全为1
该下白色棋子,全为0

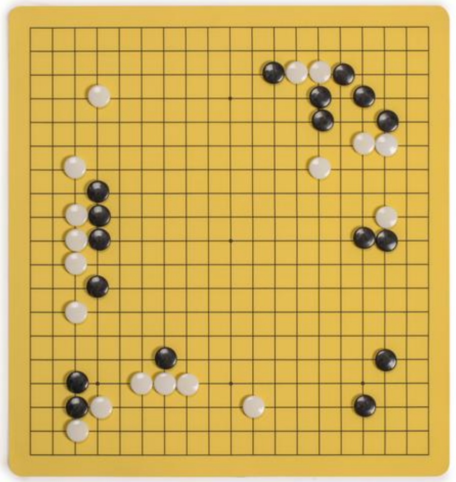
Policy Network (of AlphaGo Zero)

最新版


Conv
Dense
$\pi(1|s,\boldsymbol{\theta})$
$\pi(2|s,\boldsymbol{\theta})$
$\pi(3|s,\boldsymbol{\theta})$
$\pi(359|s,\boldsymbol{\theta})$
$\pi(360|s,\boldsymbol{\theta})$
$\pi(361|s,\boldsymbol{\theta})$
state
19×19×17 tensor
feature
Probability distribution over the 361 actions

Policy Network (of AlphaGo)

2016版



state

19×19×48 tensor

Conv

$$\xrightarrow{\hspace{3cm}}$$

🟥 $\pi(1|s,\boldsymbol{\theta})$

🟥 $\pi(2|s,\boldsymbol{\theta})$

🟥 $\pi(3|s,\boldsymbol{\theta})$

⋮

🟥 $\pi(359|s,\boldsymbol{\theta})$

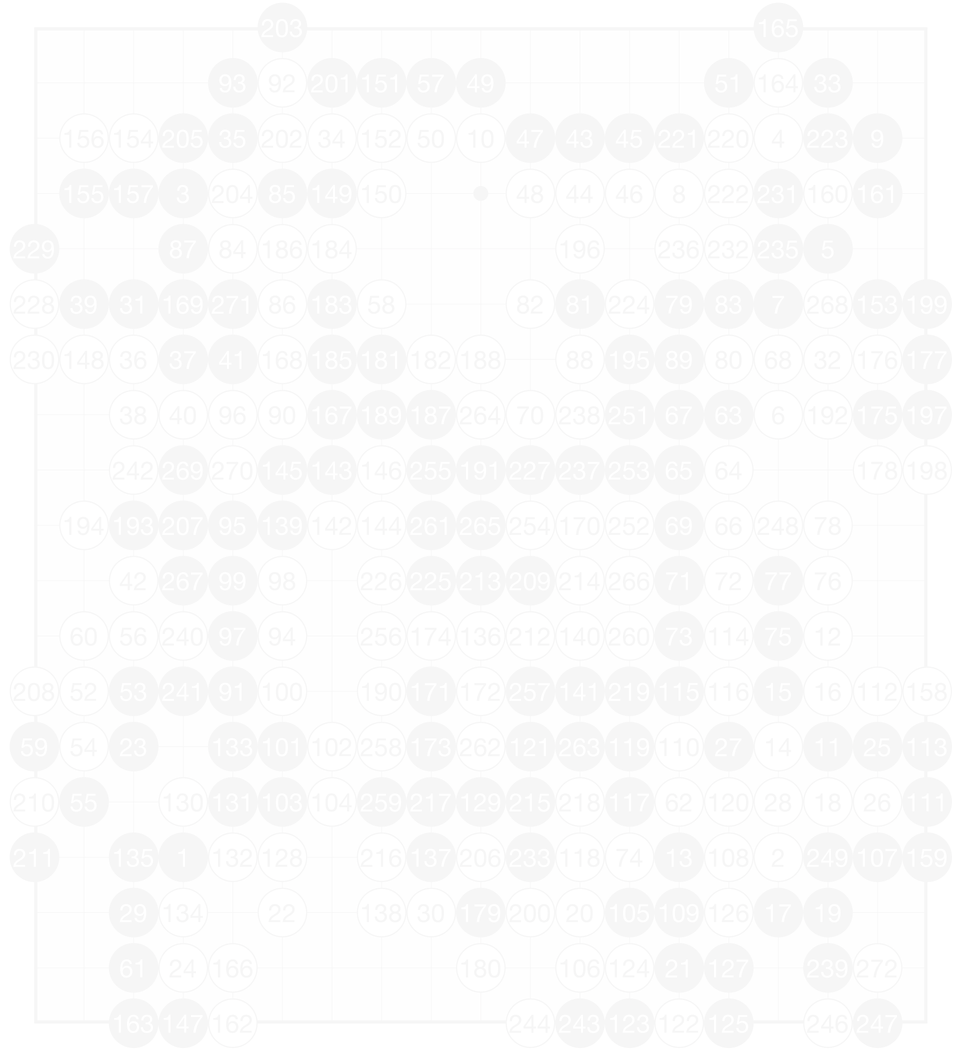
🟥 $\pi(360|s,\boldsymbol{\theta})$

🟥 $\pi(361|s,\boldsymbol{\theta})$

Probability distribution over the 361 actions

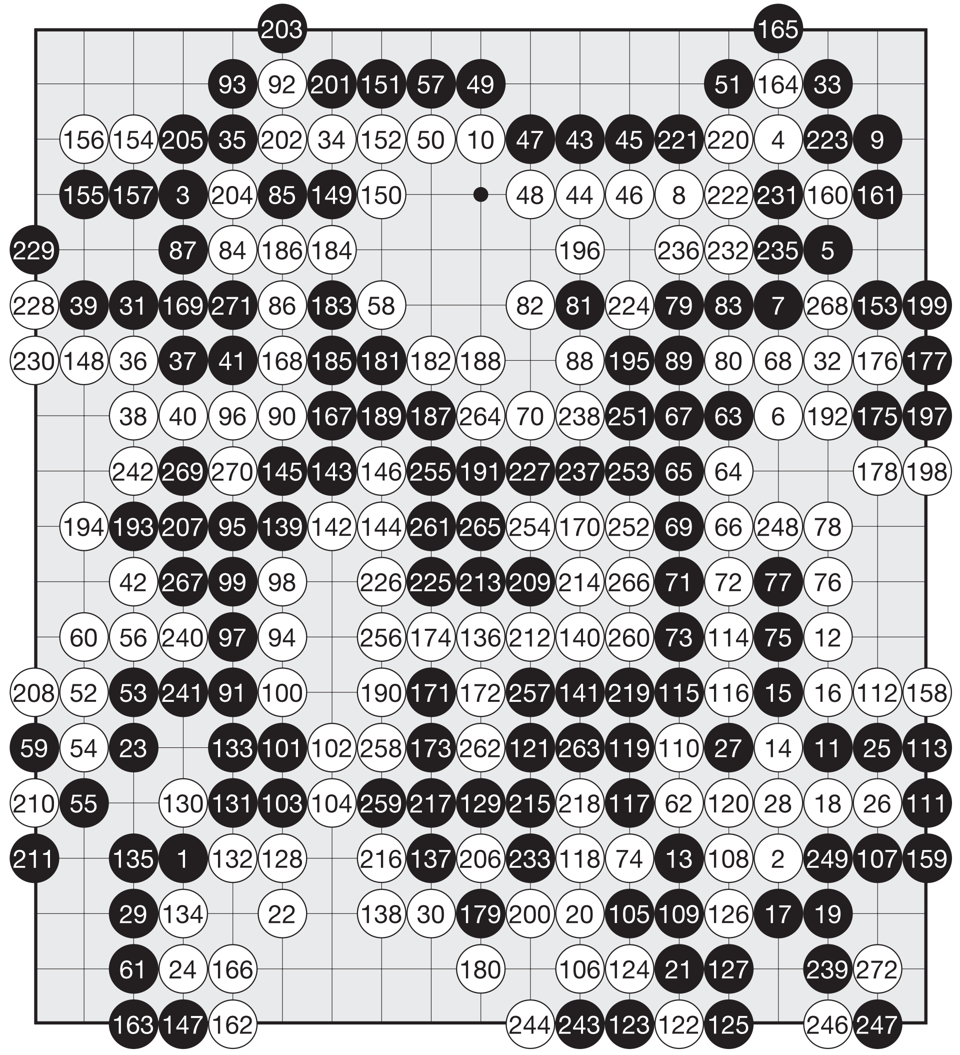
Initialize Policy Network by Behavior Cloning

Learning from human's record



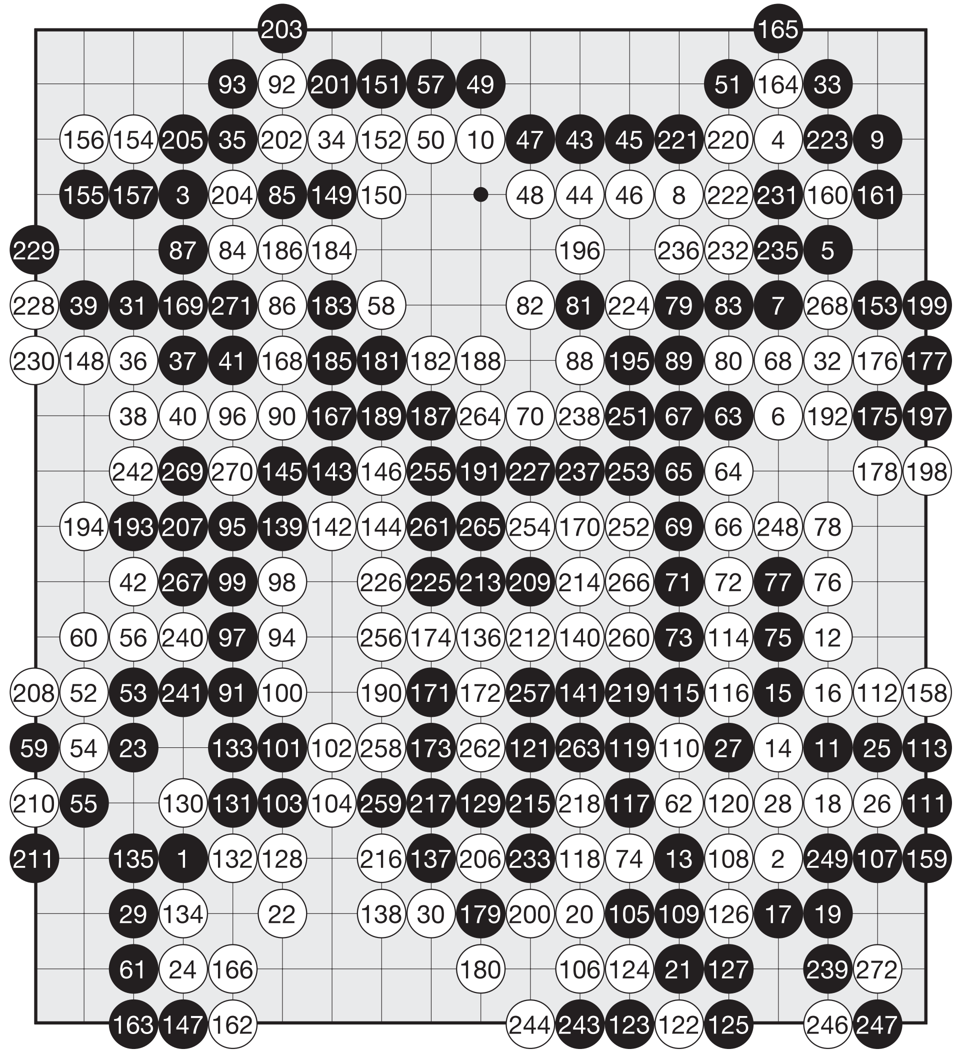
- Initially, the network's parameters are random.
  - If two policy networks play against each other, they would do random actions.
  - It would take very long before they make reasonable actions.

Learning from human's record



- Initially, the network's parameters are random.
- Human' sequences of actions have been recorded. (KGS dataset has 160K games' records.)

Learning from human's record



- Initially, the network's parameters are random.
- Human' sequences of actions have been recorded. (KGS dataset has 160K games' records.)
- Behavior cloning: Let the policy network imitate human players.监督学习训练策略网络
- After behavior cloning, the policy network beats advanced amateur.

Behavior Cloning

Behavior cloning is not reinforcement learning!

- Reinforcement learning: Supervision is from rewards given by the environment.
- Imitation learning: Supervision is from experts' actions.
  - Agent does not see rewards.
  - Agent simply imitates experts' actions.

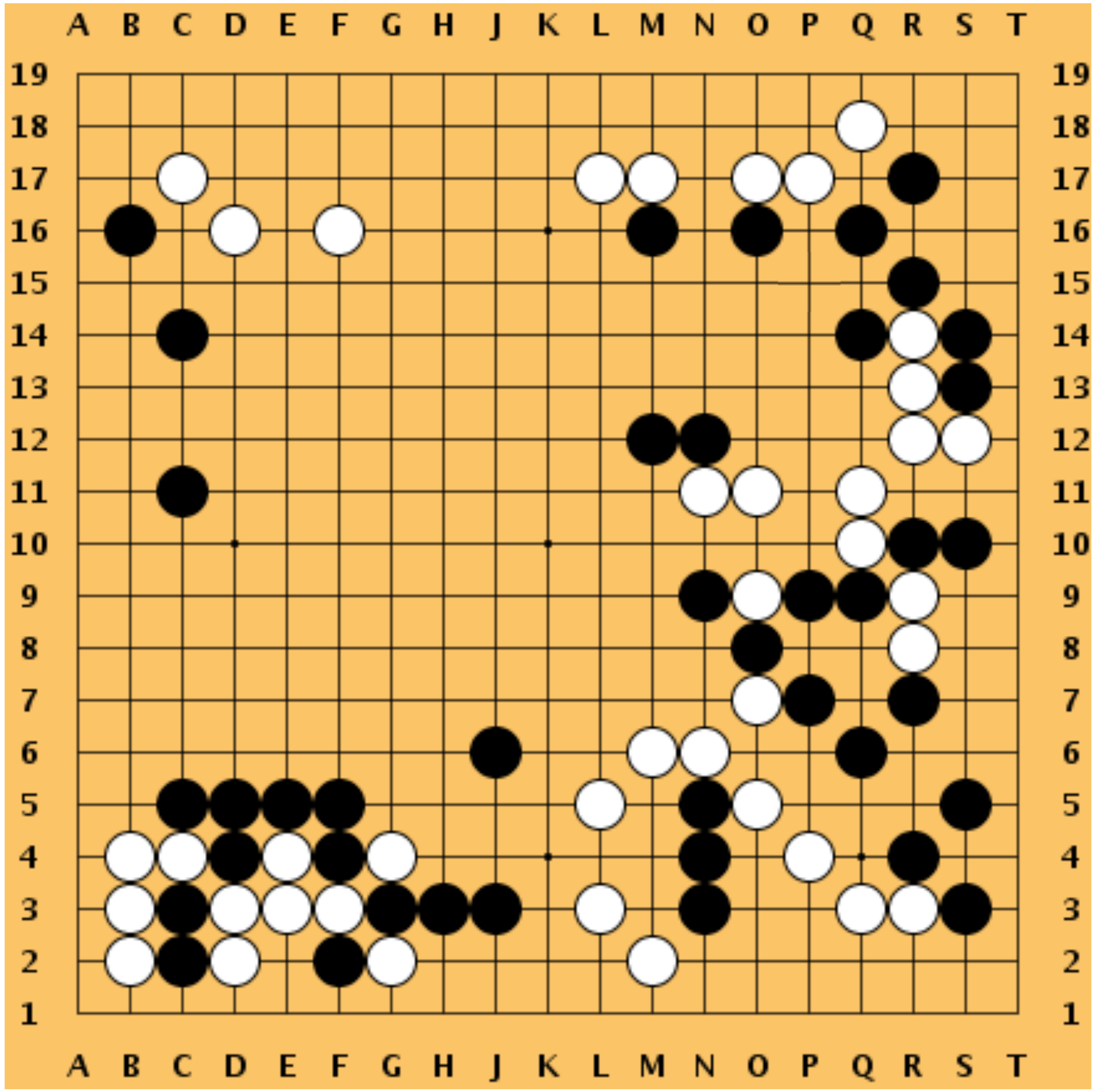
Behavior Cloning

Behavior cloning is not reinforcement learning!

- Reinforcement learning: Supervision is from rewards given by the environment.
- Imitation learning: Supervision is from experts' actions.

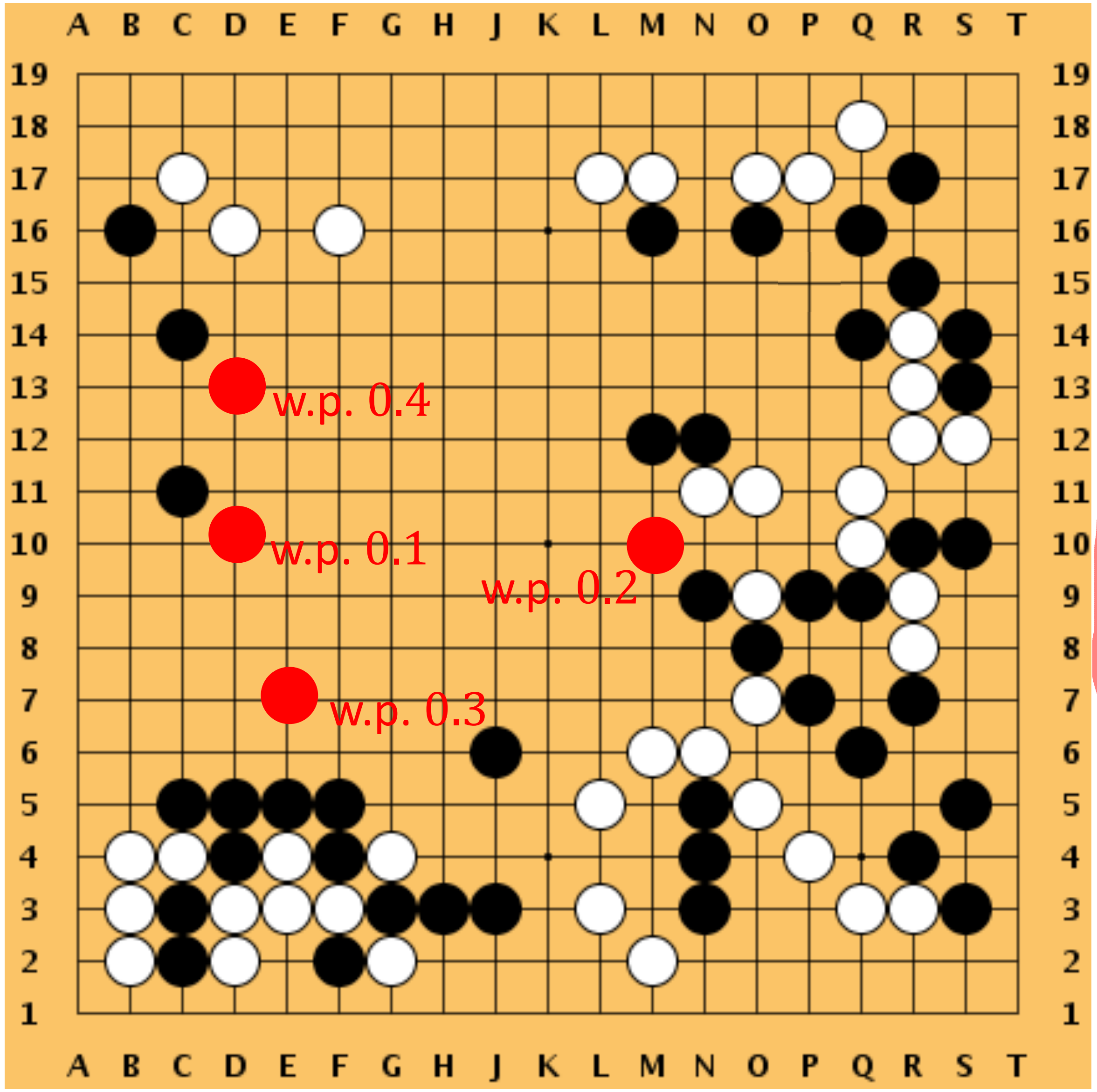
- Behavior cloning is one of the imitation learning methods.
- Behavior cloning is simply classification or regression.

Behavior Cloning



- Observe this state s_t.

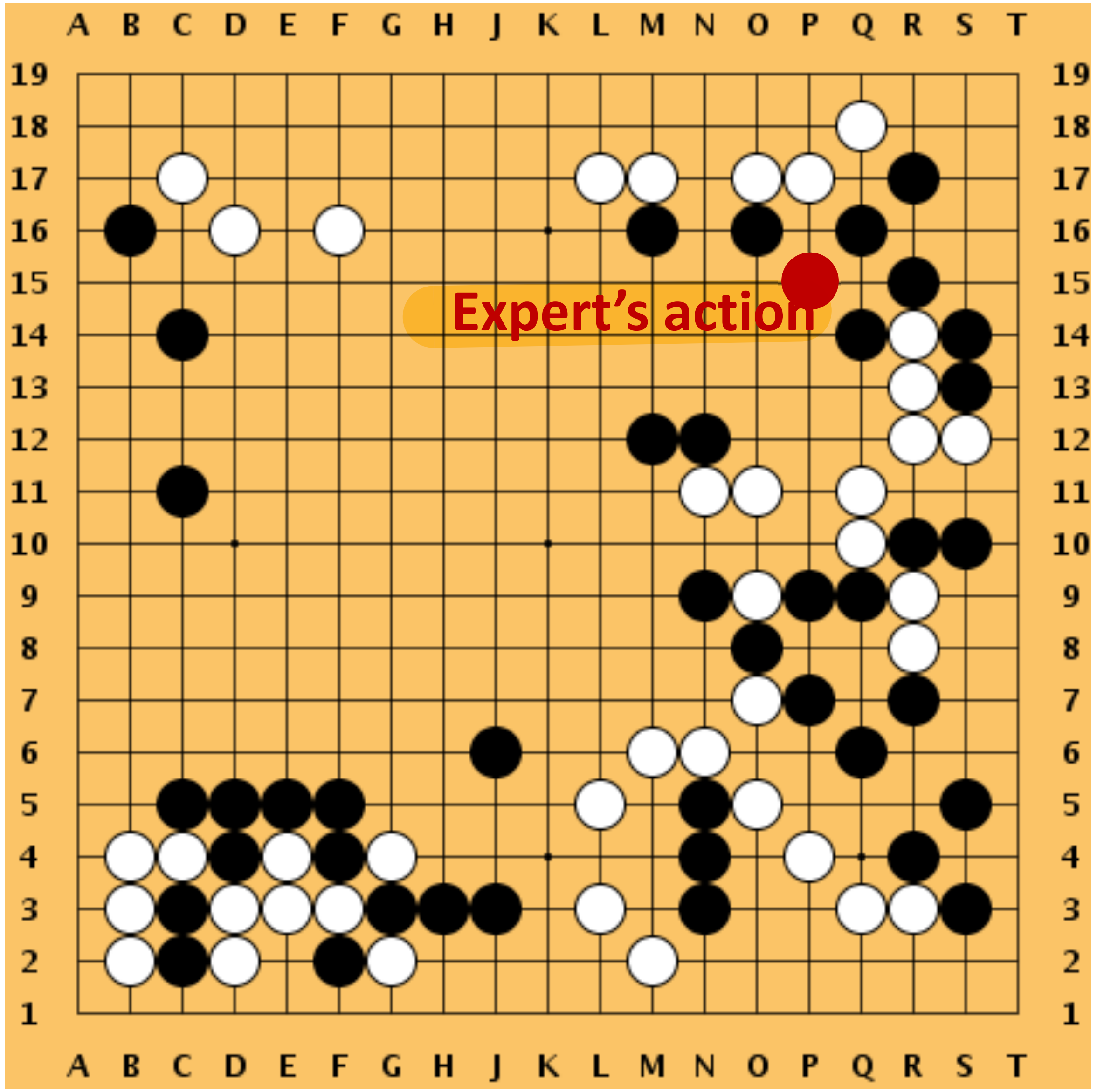
Behavior Cloning



- Observe this state s_t.
- Policy network makes a prediction:

$$\underbrace{\mathbf{p}_t}_{\text{361个动作的概率}} = [\pi(1|\underset{\uparrow \text{input}}{s_t}, \boldsymbol{\theta}), \cdots, \pi(361|s_t, \boldsymbol{\theta})] \in (0,1)^{361}.$$

Behavior Cloning



- Observe this state s_t.
- Policy network makes a prediction:

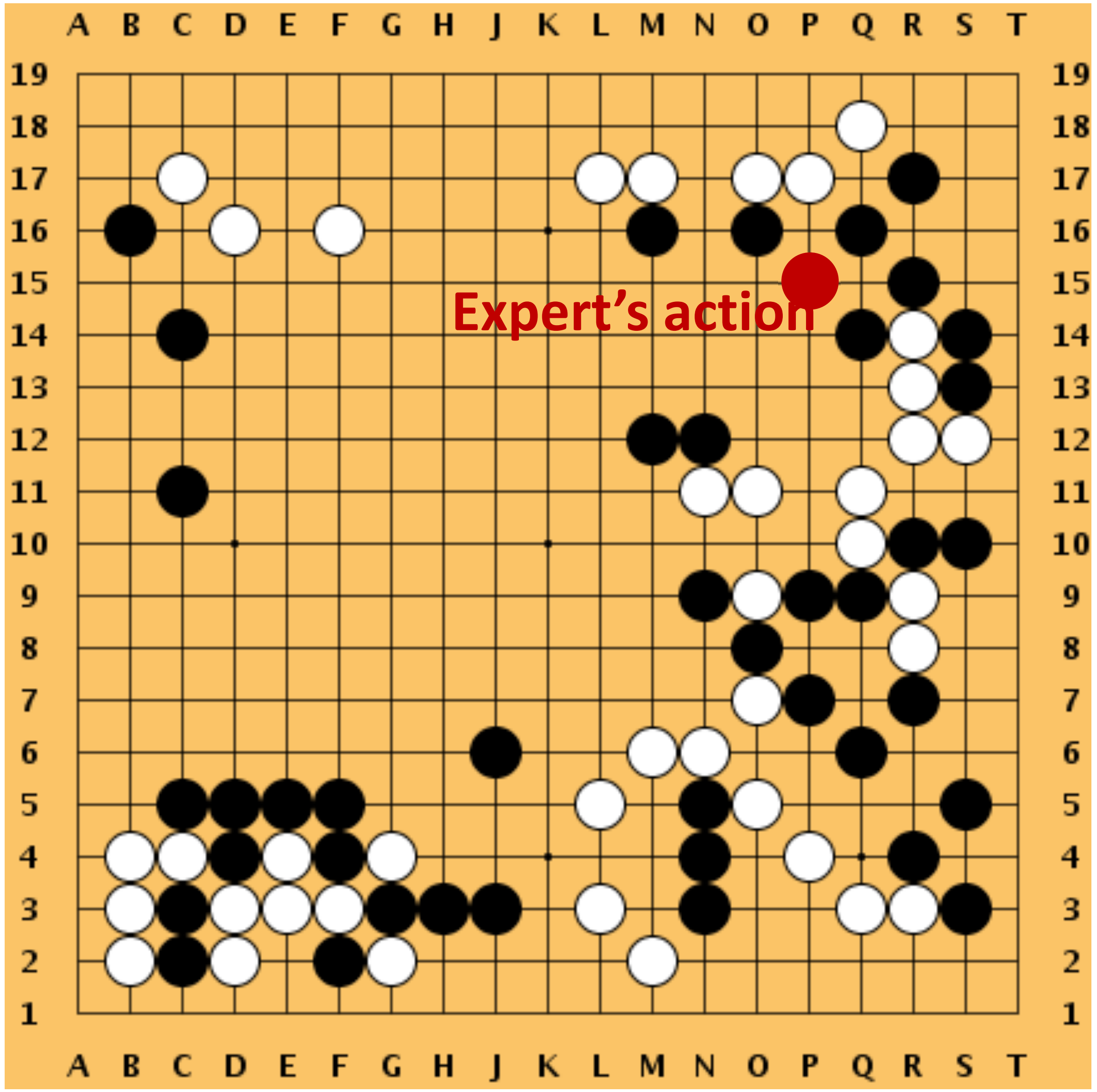
$$\mathbf{p}_t = [\pi(1|s_t,\boldsymbol{\theta}), \cdots, \pi(361|s_t,\boldsymbol{\theta})] \in (0,1)^{361}.$$

人类真实的落棋点

- The expert's action is $a_t^\star = 281$.
- Let $\mathbf{y}_t \in \{0,1\}^{361}$ be the one-hot encode of $a_t^\star = 281$.

把281转化为361维向量,
除了281维,其余全为0

Behavior Cloning



- Observe this state s_t.
- Policy network makes a prediction:

$$\mathbf{p}_t = [\pi(1|s_t,\boldsymbol{\theta}), \cdots, \pi(361|s_t,\boldsymbol{\theta})] \in (0,1)^{361}.$$

- The expert's action is $a_t^\star = 281$.
- Let $\mathbf{y}_t \in \{0,1\}^{361}$ be the one-hot encode of $a_t^\star = 281$.
- $\text{Loss} = \text{CrossEntropy}(\mathbf{y}_t, \mathbf{p}_t)$.
- Use gradient descent to update policy network.

Behavior其实就是多分类, 361个类别,每个类别的概

After behavior cloning…

- Suppose the current sate s_t has appeared in training data.
- The policy network imitates expert's action a_t. (Which is a good action!)

After behavior cloning…

- Suppose the current sate s_t has appeared in training data.
- The policy network imitates expert’s action a_t. (Which is a good action!)

Question: Why bother doing RL after behavior cloning?

- What if the current state s_t has not appeared in training data?
- Then the policy network’ action a_t can be bad.

若当前St出现过,那么at的结果肯定非常好,因为有迹可循

After behavior cloning…

- Suppose the current sate s_t has appeared in training data.
- The policy network imitates expert’s action a_t. (Which is a good action!)

Question: Why bother doing RL after behavior cloning?

- What if the current state s_t has not appeared in training data?
- Then the policy network’ action a_t can be bad.
- Number of possible states is too big.
- There is a big chance that s_t has not appeared in training data.

Behavior cloning + RL beats behavior cloning with 80% chance.

Train Policy Network Using Policy Gradient

Reinforcement learning of policy network

- Player’s parameters are the latest parameters of the policy network.
- Opponent’s parameters are randomly selected from previous iterations.

博弈

Player
(Agent)
policy network with latest param

V.S.

Opponent
(Environment)
policy network with old param

Reinforcement learning of policy network

Reinforcement learning is guided by rewards.

- Suppose a game ends at step T.
- Rewards:

游戏没结束时,奖励就是0

  - $r_1 = r_2 = r_3 = \cdots = r_{T-1} = 0$. (When the game has not ended.)
  - $r_T = +1$ (winner).
  - $r_T = -1$ (loser).

Reinforcement learning of policy network

Reinforcement learning is guided by rewards.

- Suppose a game ends at step T.
- Rewards:
  - $r_1 = r_2 = r_3 = \cdots = r_{T-1} = 0$. (When the game has not ended.)
  - $r_T = +1$ (winner).
  - $r_T = -1$ (loser).
- Recall that return is defined by $u_t = \sum_{i=t}^{T} r_i$. (No discount here.)
- Winner's returns: $u_1 = u_2 = \cdots = u_T = +1$.
- Loser's returns: $u_1 = u_2 = \cdots = u_T = -1$.

无法判断哪一步棋下的好,只能
用最终结果来给予回报

Policy Gradient

Policy gradient: Derivative of state-value function $V(s;\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$.

- Recall that $\frac{\partial \log \pi(a_t|s_t,\boldsymbol{\theta})}{\partial \, \boldsymbol{\theta}} \cdot Q_\pi(s_t, a_t)$ is approximate policy gradient.

Policy Gradient

Policy gradient: Derivative of state-value function $V(s; \boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$.

- Recall that $\frac{\partial \log \pi(a_t|s_t,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot \boxed{Q_\pi(s_t, a_t)}$ is approximate policy gradient.
- By definition, the action value is $Q_\pi(s_t, a_t) = \mathbb{E}\left[U_t | s_t, a_t\right]$.
- Thus, we can replace $Q_\pi(s_t, a_t)$ by the observed return u_t.
- Approximate policy gradient: $\frac{\partial \log \pi(a_t|s_t,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot \boxed{u_t}$.

Train policy network using policy gradient

Repeat the followings:

- Two policy networks play a game to the end. (Player v.s. Opponent.)
- Get a trajectory: $s_1, a_1, s_2, a_2, \cdots, s_T, a_T$. 轨迹
- After the game ends, update the player's policy network.
  - The player's returns: $u_1 = u_2 = \cdots = u_T$. (Either $+1$ or -1.)
  - Sum of approximate policy gradients: $\mathbf{g}_\theta = \sum_{t=1}^{T} \frac{\partial \log \pi(a_t | s_t, \boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot u_t$.
  - Update policy network: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \cdot \mathbf{g}_\theta$.

Play Go using the policy network

- The policy network π has been learned. ✓
- Observing the current state s_t, randomly sample action

$$\underbrace{a_t}_{\text{out}} \sim \pi(\cdot \mid \underbrace{s_t}_{\text{input}}, \boldsymbol{\theta}).$$

从out中随机抽样得到一个位置

Play Go using the policy network

- The policy network π has been learned.
- Observing the current state s_t, randomly sample action

$$a_t \sim \pi(\cdot \mid s_t, \boldsymbol{\theta}).$$

- The learned policy network π is strong, but not strong enough.
- A small mistake may change the game result.

Train the Value Network

State-Value Function

Definition: State-value function.

- $V_\pi(s) = \mathbb{E}[U_t | S_t = s]$, where $U_t = +1$ (if win) and -1 (if lose) .
- The expectation is taken with respect to
  - The future actions $A_t, A_{t+1}, \cdots, A_T$.
  - The future states $S_{t+1}, S_{t+2}, \cdots, S_T$

State-Value Function

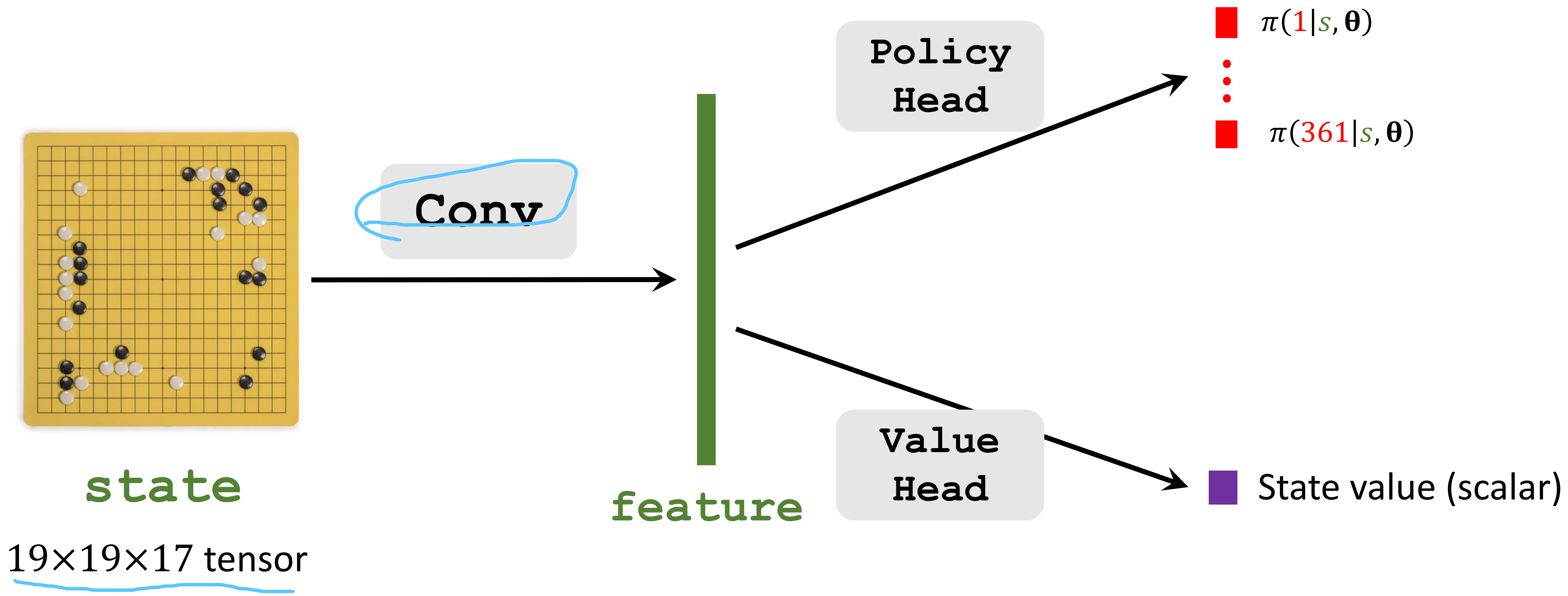
Definition: State-value function.

- $V_\pi(s) = \mathbb{E}[U_t | S_t = s]$, where $U_t = +1$ (if win) and -1 (if lose) .
- The expectation is taken with respect to
  - The future actions $A_t, A_{t+1}, \cdots, A_T$.
  - The future states $S_{t+1}, S_{t+2}, \cdots, S_T$

Approximate state-value function using a value network.

- Use a neural network $v(s; \mathbf{w})$ to approximate $V_\pi(s)$.
- It evaluate how good the current situation is.

Policy Value Networks (AlphaGo Zero)


Conv
Policy Head
Value Head
$\pi(1|s,\boldsymbol{\theta})$
$\pi(361|s,\boldsymbol{\theta})$
State value (scalar)
state
$19\times19\times17$ tensor
feature

Train the value network

After finishing training the policy network, train the value network.

Train the value network

After finishing training the policy network, train the value network.

Repeat the followings:

1. Play a game to the end.
   - If win, let $u_1 = u_2 = \cdots = u_T = +1$.
   - If lose, let $u_1 = u_2 = \cdots = u_T = -1$.
2. Loss: $L = \sum_{t=1}^{T} \frac{1}{2} \left[v(s_t; \mathbf{w}) - u_t \right]^2$. 反映v是否准确
3. Update: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \frac{\partial L}{\partial \mathbf{w}}$.

Monte Carlo Tree Search

How do human play Go?

Players must look ahead two or more steps.

- Suppose I choose action a_t.
- What will be my opponent's action? (His action leads to state s_{t+1}.)
- What will I be my action a_{t+1} upon observing s_{t+1}?
- What will be my opponent's action? (His action leads to state s_{t+2}.)
- What will I be my action a_{t+2} upon observing s_{t+3}?

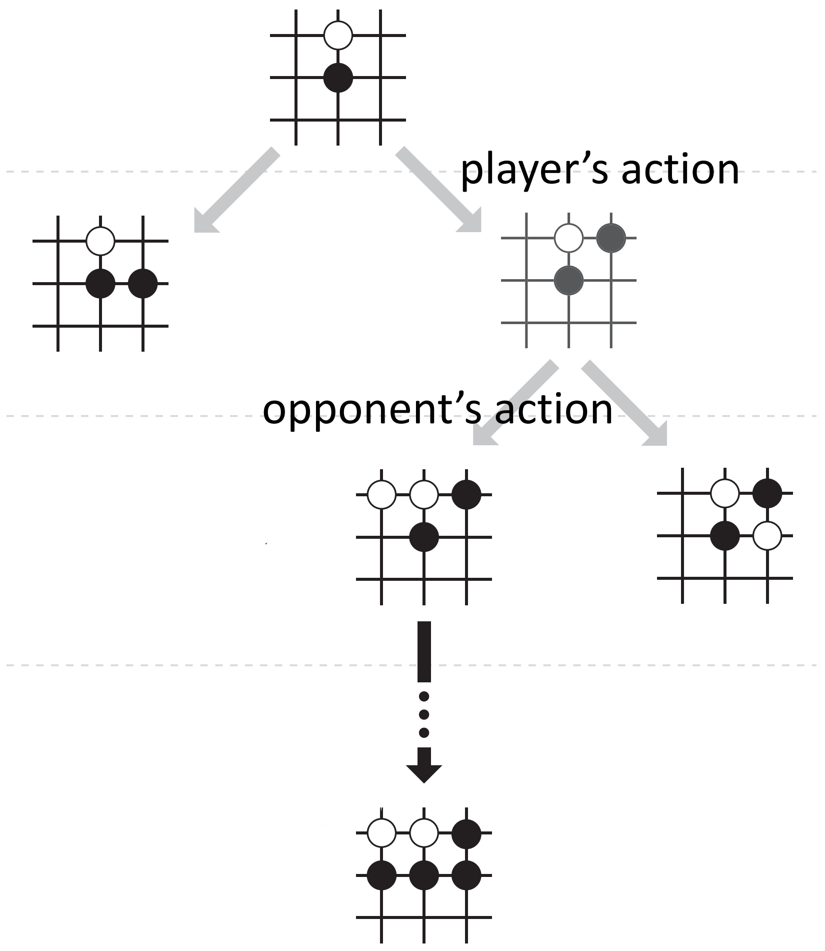
$\vdots$

- If you can exhaustively foresee all the possibilities, you will win.

- Strange: I went forward in time... to view alternate futures. To see all the possible outcomes of the coming conflict.
- Quill: How many did you see?
- Strange: Fourteen million six hundred and five.

- Stark: How many did we win?
- Strange: … One.

Select actions by look-ahead search



Main idea

策略函数可以选出好的动作

1. • Randomly select an action a.
2. • Look ahead and see whether a leads to win or lose. **自我博弈**
3. • Repeat this procedure many times.
4. • Choose the action a that has the highest score.

看哪个动作的分数最高

Monte Carlo Tree Search (MCTS)

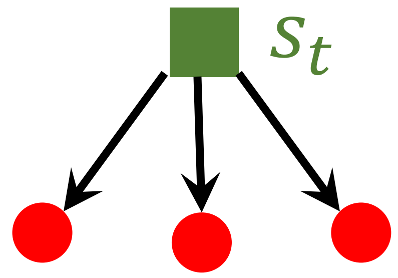
Every simulation of Monte Carlo Tree Search (MCTS) has 4 steps:

1. **Selection:** The player makes an action a. (Imaginary action; not actual move.) **按分数选一个动作**

2. **Expansion:** The opponent makes an action; the state updates. (Also imaginary action; made by the policy network.)

3. **Evaluation:** Evaluate the state-value and get score v. Play the game to the end to receive reward r. Assign score $\frac{v+r}{2}$ to action a.

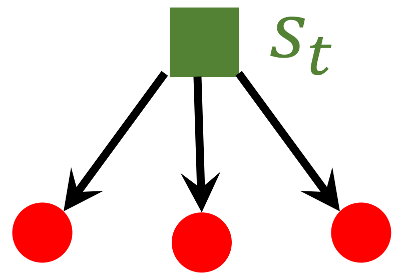
4. **Backup:** Use the score $\frac{v+r}{2}$ to update action-values.



valid actions

Step 1: Selection

Question: Observing s_t, which action shall we explore?



valid actions

Step 1: Selection

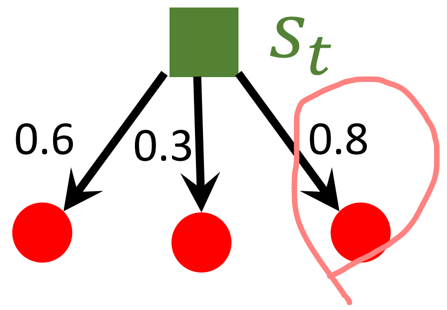
Question: Observing s_t, which action shall we explore?

优先探索好的动作

First, for all the valid actions a, calculate the score:

$$\text{score}(a) = Q(a) + \eta \cdot \frac{\pi(a \mid s_t; \boldsymbol{\theta})}{1 + N(a)}.$$

- $Q(a)$: Action-value computed by MCTS. (To be defined.)
- $\pi(a \mid s_t; \boldsymbol{\theta})$: The learned policy network.
- $N(a)$: Given s_t, how many times we have selected a so far.



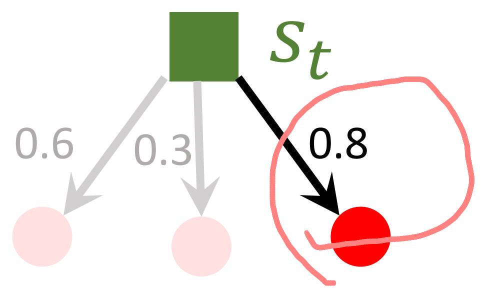
Step 1: Selection

Question: Observing s_t, which action shall we explore?

First, for all the valid actions a, calculate the score:

$$\text{score}(a) = Q(a) + \eta \cdot \frac{\pi(a \mid s_t; \boldsymbol{\theta})}{1 + N(a)}.$$

- $Q(a)$: Action-value computed by MCTS. (To be defined.)
- $\pi(a \mid s_t; \boldsymbol{\theta})$: The learned policy network.
- $N(a)$: Given s_t, how many times we have selected a so far.



Step 1: Selection

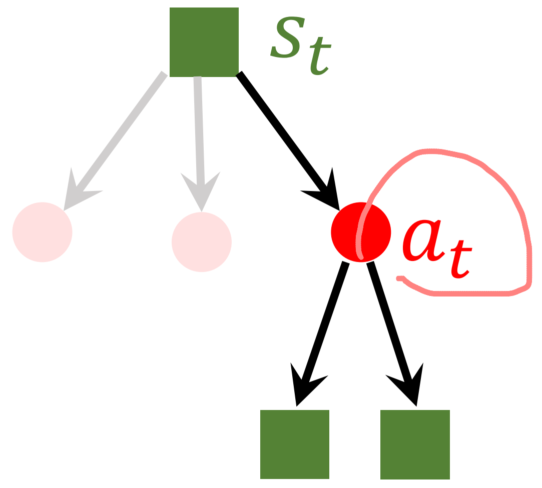
Question: Observing s_t, which action shall we explore?

First, for all the valid actions a, calculate the score:

$$\text{score}(a) = Q(a) + \eta \cdot \frac{\pi(a \mid s_t; \boldsymbol{\theta})}{1+N(a)}.$$

- $Q(a)$: Action-value computed by MCTS. (To be defined.)
- $\pi(a \mid s_t; \boldsymbol{\theta})$: The learned policy network.
- $N(a)$: Given s_t, how many times we have selected a so far.

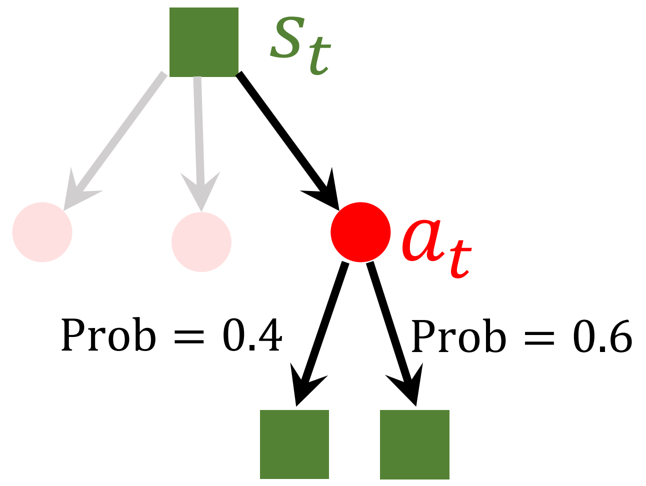
Second, the action with the biggest score(a) is selected.



Step 2: Expansion

Question: What will be the opponent's action?

- Given a_t, the opponent's action a'_t will lead to the new state s_{t+1}.



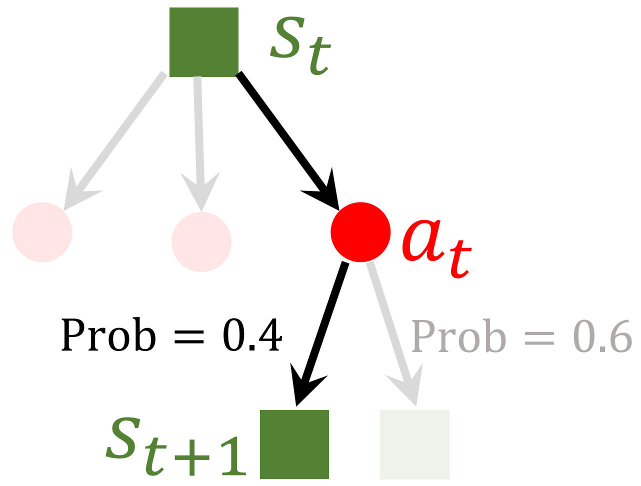
Step 2: Expansion

Question: What will be the opponent's action?

- Given a_t, the opponent’s action a_t' will lead to the new state s_{t+1}.
- The opponent’s action is randomly sampled from

$$a_t' \sim \pi(\cdot \mid s_t'; \boldsymbol{\theta}).$$

Here, s_t' is the state observed by the opponent.



Step 2: Expansion

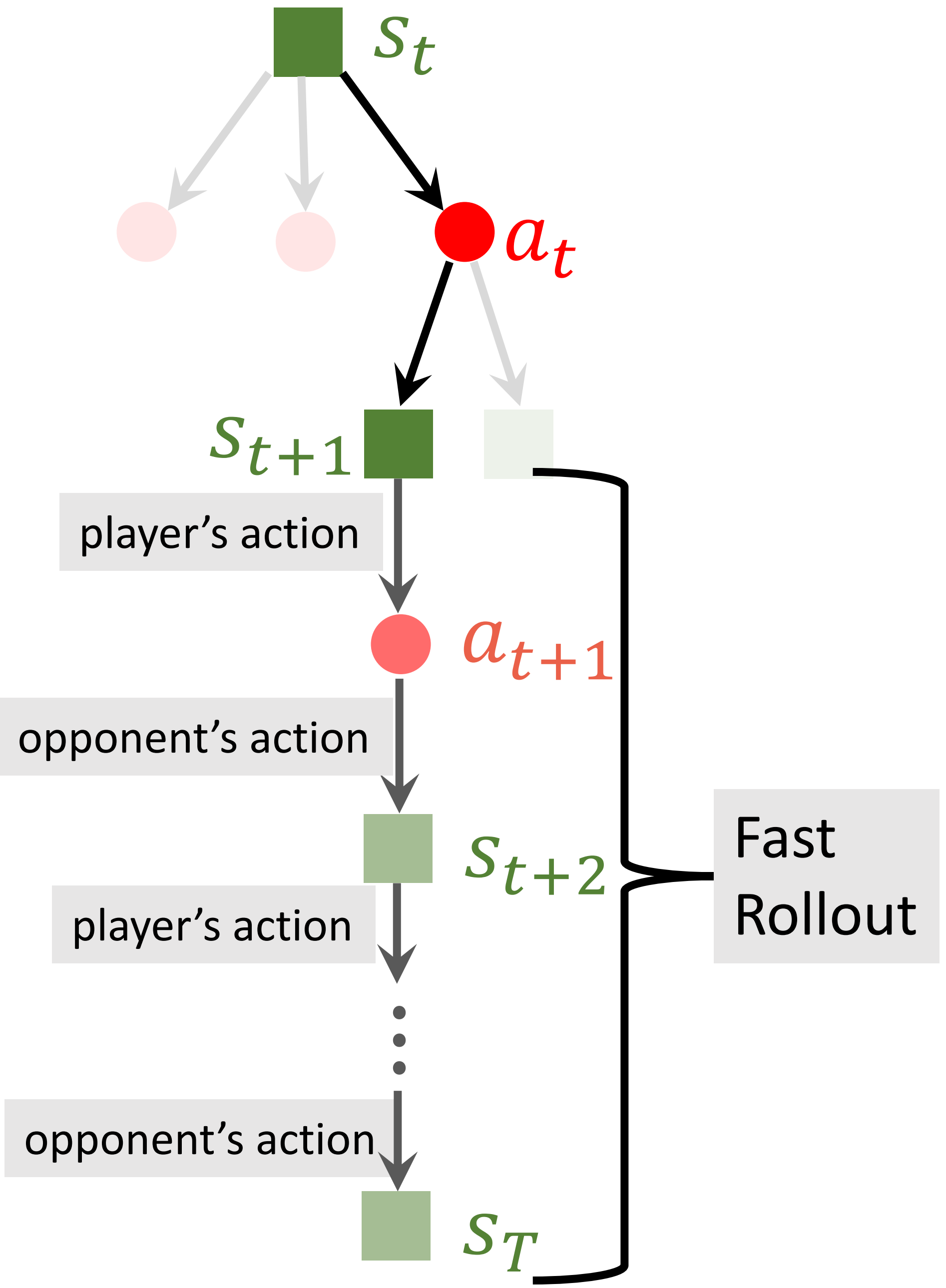
Question: What will be the opponent's action?

- Given a_t, the opponent's action a_t' will lead to the new state s_{t+1}.
- The opponent's action is randomly sampled from

$$a_t' \sim \pi(\cdot \mid s_t'; \boldsymbol{\theta}).$$

Here, s_t' is the state observed by the opponent.

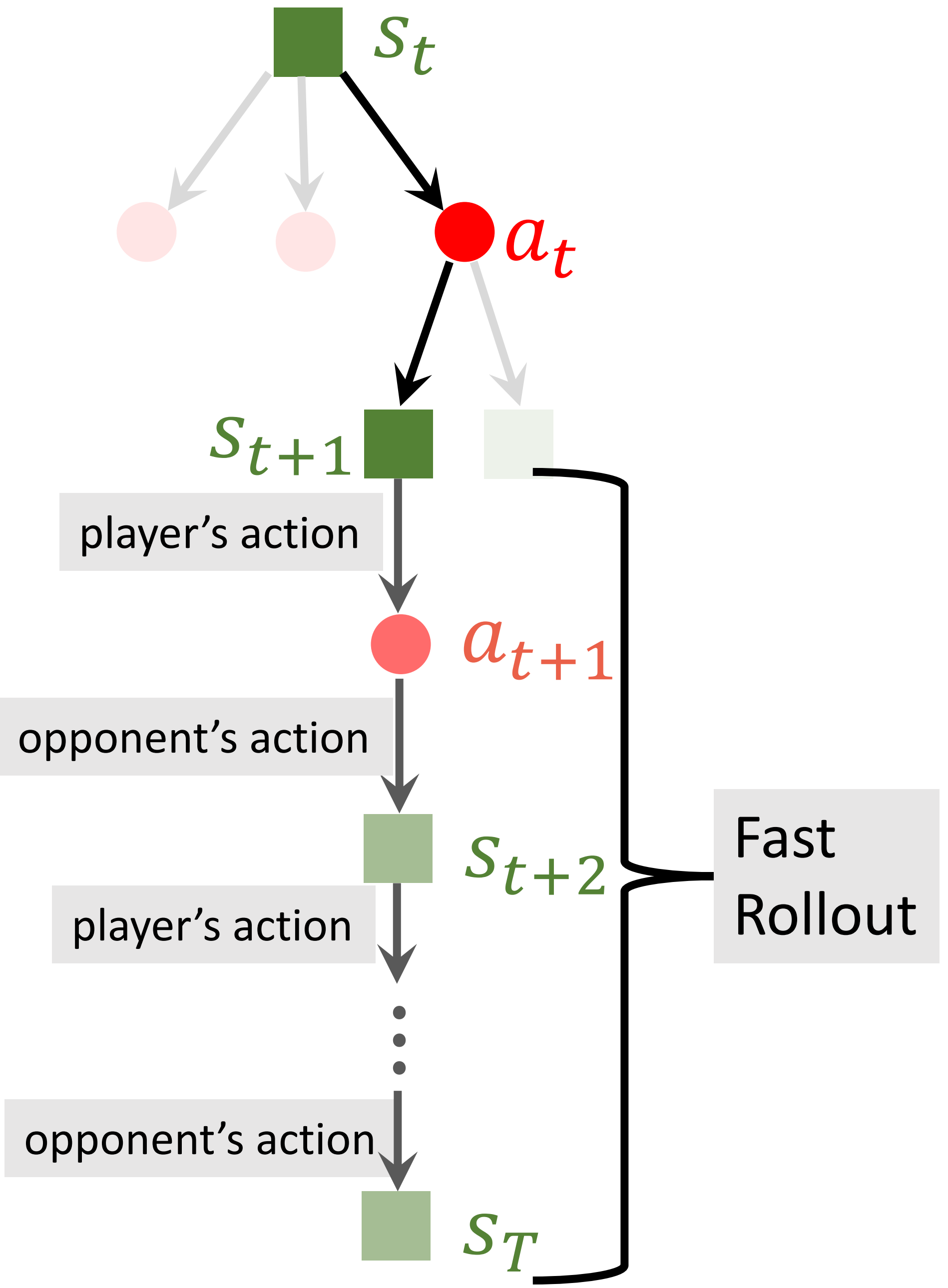
- The state-transition probability $p(s_{t+1}|s_t, a_t)$ is unknown.
- Use the policy function π as the state-transition function p.



Step 3: Evaluation

Run a rollout to the end of the game (step T).

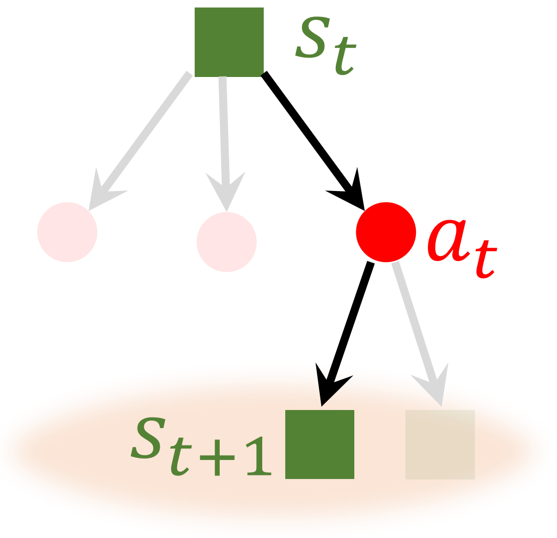
- Player's action: $a_k \sim \pi(\cdot \mid s_k; \boldsymbol{\theta})$.
- Opponent's action: $a'_k \sim \pi(\cdot \mid s'_k; \boldsymbol{\theta})$.



Step 3: Evaluation

Run a rollout to the end of the game (step T).

- Player's action: $a_k \sim \pi(\cdot \mid s_k; \boldsymbol{\theta})$.
- Opponent's action: $a'_k \sim \pi(\cdot \mid s'_k; \boldsymbol{\theta})$.
- Receive reward r_T at the end.
  - Win: $r_T = +1$.
  - Lose: $r_T = -1$.



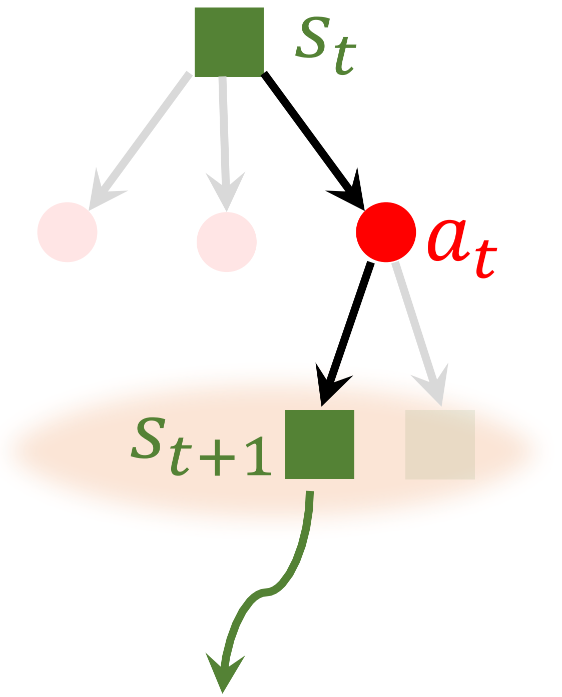
Step 3: Evaluation

Run a rollout to the end of the game (step T).

- Player's action: $a_k \sim \pi(\cdot \mid s_k; \boldsymbol{\theta})$.
- Opponent's action: $a'_k \sim \pi(\cdot \mid s'_k; \boldsymbol{\theta})$.
- Receive reward r_T at the end.
  - Win: $r_T = +1$.
  - Lose: $r_T = -1$.

Evaluate the state s_{t+1}.

- $v(s_{t+1}; \mathbf{w})$: output of the value network.



Record $V(s_{t+1})$.

$$V(s_{t+1}) = \frac{1}{2} v(s_{t+1}; \mathbf{w}) + \frac{1}{2} r_T .$$

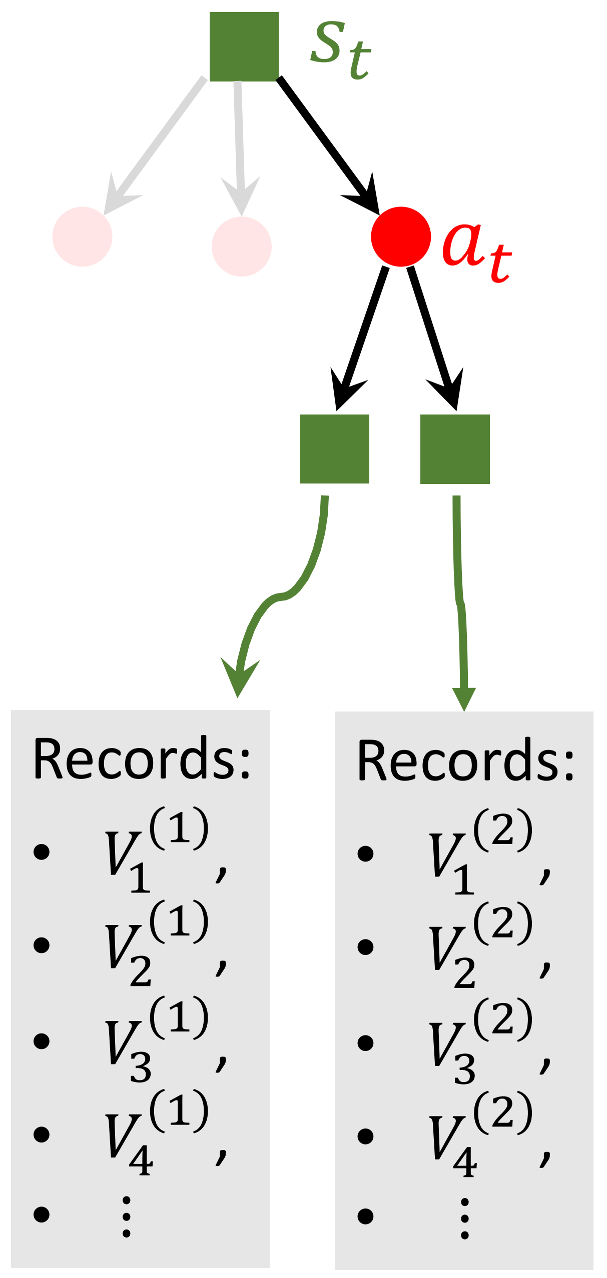
Step 3: Evaluation

Run a rollout to the end of the game (step T).

- Player's action: $a_k \sim \pi(\cdot \mid s_k; \boldsymbol{\theta})$.
- Opponent's action: $a'_k \sim \pi(\cdot \mid s'_k; \boldsymbol{\theta})$.
- Receive reward r_T at the end.
  - Win: $r_T = +1$.
  - Lose: $r_T = -1$.

Evaluate the state s_{t+1}.

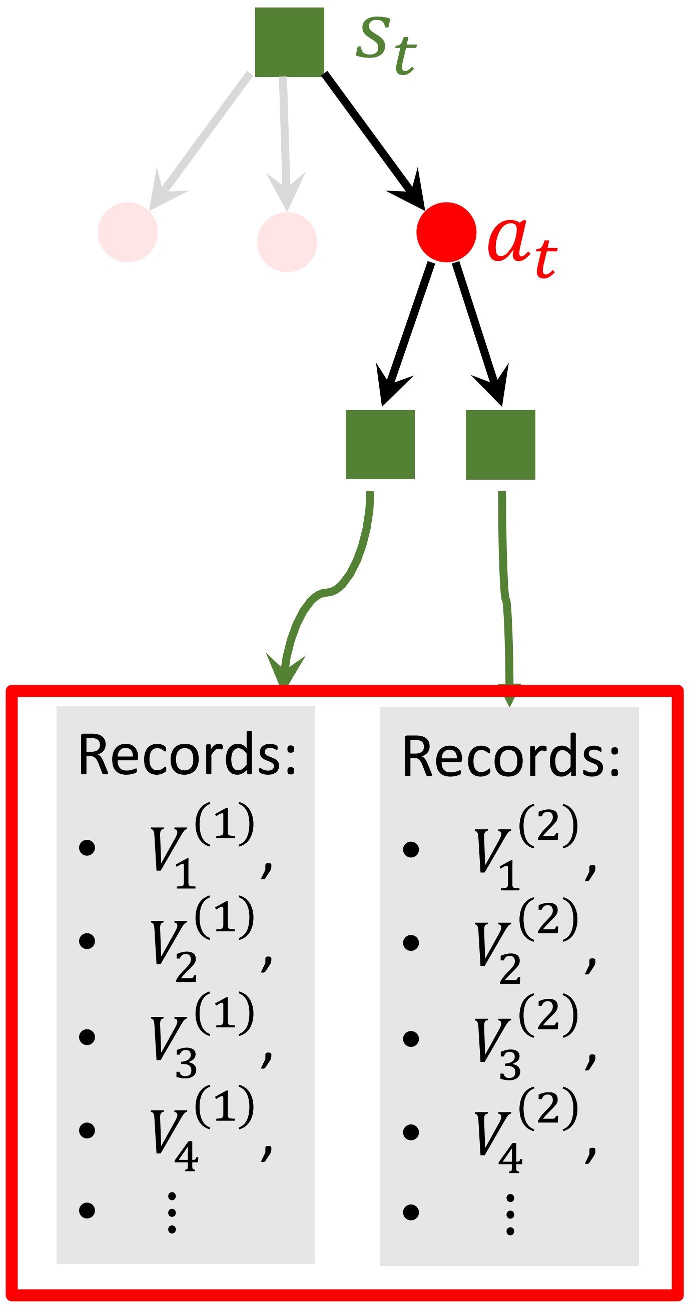
- $v(s_{t+1}; \mathbf{w})$: output of the value network.



Step 4: Backup

- MCTS repeats such a simulation many times.
- Each child of a_t has multiple recorded $V(s_{t+1})$.

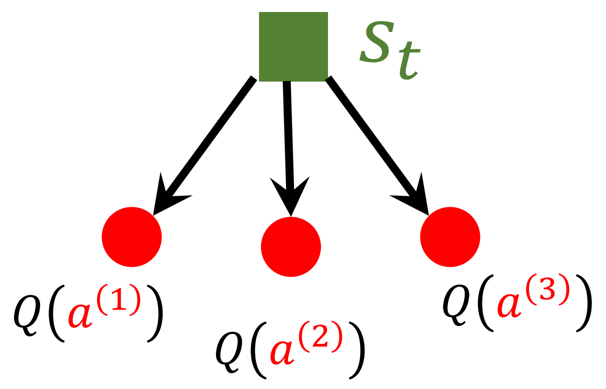
Step 4: Backup



- MCTS repeats such a simulation many times.
- Each child of a_t has multiple recorded $V(s_{t+1})$.
- Update action-value:

$$Q(a_t) = \text{mean}(\text{the recorded } V's).$$

- The Q values will be used in Step 1 (selection).

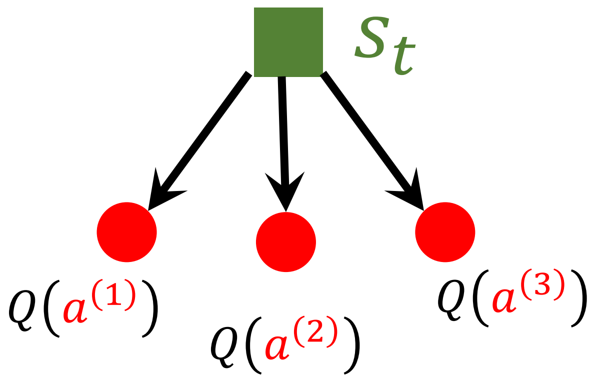


Step 4: Backup

- MCTS repeats such a simulation many times.
- Each child of a_t has multiple recorded $V(s_{t+1})$.
- Update action-value:

$$Q(a_t) = \text{mean}(\text{the recorded } V's).$$

- The Q values will be used in Step 1 (selection).

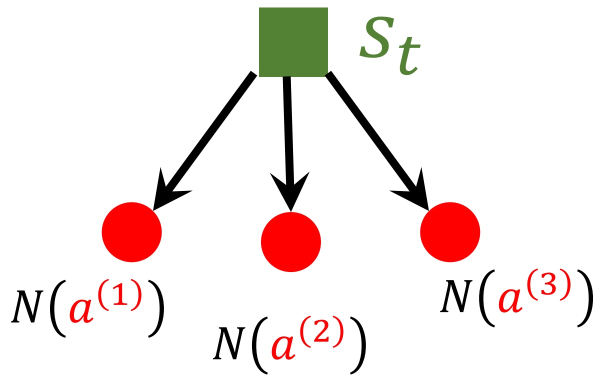


Revisit Step 1 Selection

First, for all the valid actions a, calculate the score:

$$\text{score}(a) = Q(a) + \eta \cdot \frac{\pi(a \mid s_t; \boldsymbol{\theta})}{1 + N(a)}.$$

Second, the action with the biggest $\text{score}(a)$ is selected.



Decision Making after MCTS

- $N(a)$: How many times a has been selected so far.
- After MCTS, the player makes **actual** decision:

$$a_t = \underset{a}{\operatorname{argmax}}\, N(a)$$

MCTS: Summary

- MCTS has 4 steps: selection, expansion, evaluation, and backup.
- To perform one action, AlphaGo repeats the 4 steps for many times to calculate $Q(a)$ and $N(a)$ (for every action a.)
- AlphaGo executes the action a with the highest $N(a)$ value.
- To perform the next action, AlphaGo do MCTS all over again. (Initialize $Q(a)$ and $N(a)$ to zeros.)

Summary

Training and Execution

Training in 3 steps:

1. Train a policy network using behavior cloning.
2. Train the policy network using policy gradient algorithm.
3. Train a value network.

Training and Execution

Training in 3 steps:

1. Train a policy network using behavior cloning.
2. Train the policy network using policy gradient algorithm.
3. Train a value network.

Execution (actually play Go games):

- Select the "best" action by Monte Carlo Tree Search.
- To perform one action, AlphaGo repeats simulations many times.

AlphaGo Zero

AlphaGo Zero v.s. AlphaGo

- AlphaGo Zero is stronger than AlphaGo. (100-0 against AlphaGo.)
- Differences:
    - AlphaGo Zero does not use human experience. (No behavior cloning.)
    - MCTS is used to train the policy network.

Is behavior cloning useless?

- AlphaGo Zero does not use human experience. (No behavior cloning.)
- For the Go game, human experience is harmful.

- **In general, is behavior cloning useless?**
- What if a surgery robot (randomly initialized) is learned purely by performing surgery? (Human experience is not used.)

Training of policy network

AlphaGo Zero uses MCTS in training. (AlphaGo does not.)

1. Observe state s_t.
2. Predictions made by policy network:
$$\mathbf{p} = [\pi(a = 1 | s_t; \boldsymbol{\theta}), \cdots, \pi(a = 361 | s_t; \boldsymbol{\theta})] \in \mathbb{R}^{361}.$$

Training of policy network

AlphaGo Zero uses MCTS in training. (AlphaGo does not.)

1. Observe state s_t.
2. Predictions made by policy network:
$$\mathbf{p} = [\pi(a=1|s_t;\boldsymbol{\theta}), \cdots, \pi(a=361|s_t;\boldsymbol{\theta})] \in \mathbb{R}^{361}.$$
3. Predictions made by MCTS:

归一化,变成概率值
$$\mathbf{n} = \text{normalize}[N(a=1), N(a=2), \cdots, N(a=361)] \in \mathbb{R}^{361}.$$
4. Loss: $L = \text{CrossEntropy}(\mathbf{n}, \mathbf{p})$.
5. Use $\frac{\partial\, L}{\partial\, \boldsymbol{\theta}}$ to update $\boldsymbol{\theta}$.

Reference

- AlphaGo:
  - Silver and others: Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

- AlphaGo Zero:
  - Silver and others: Mastering the game of Go without human knowledge. *Nature*, 2017.