AlphaGo

Shusen Wang

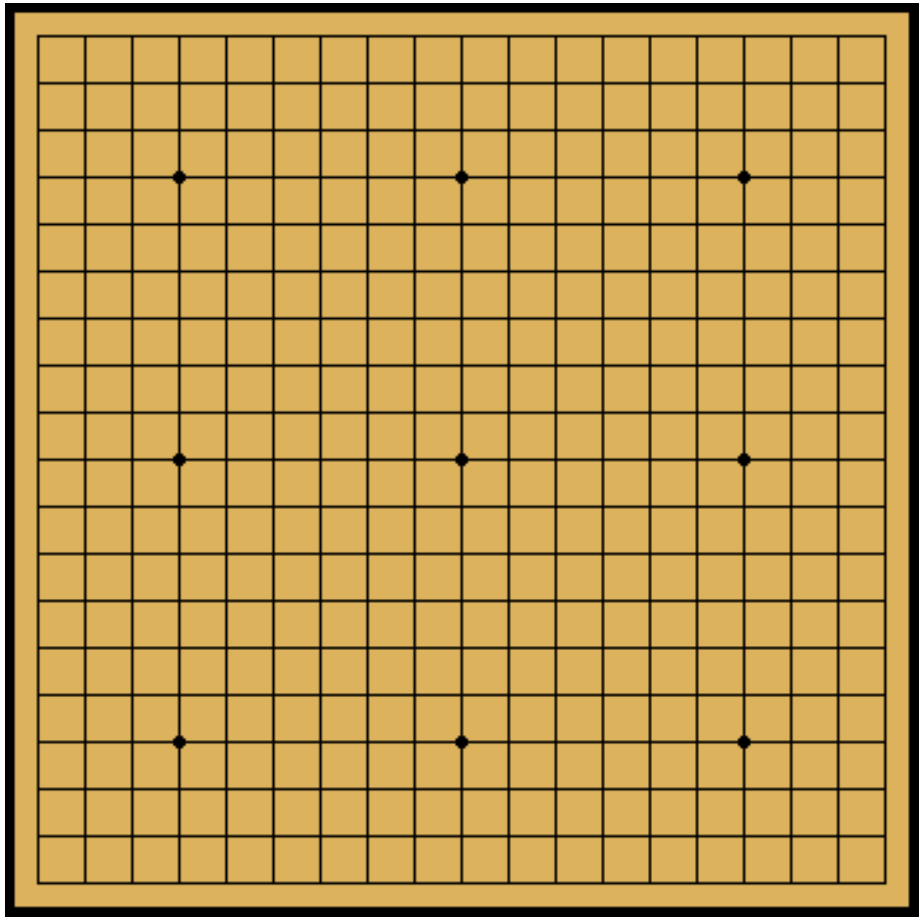
Disclaimer

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

Reference

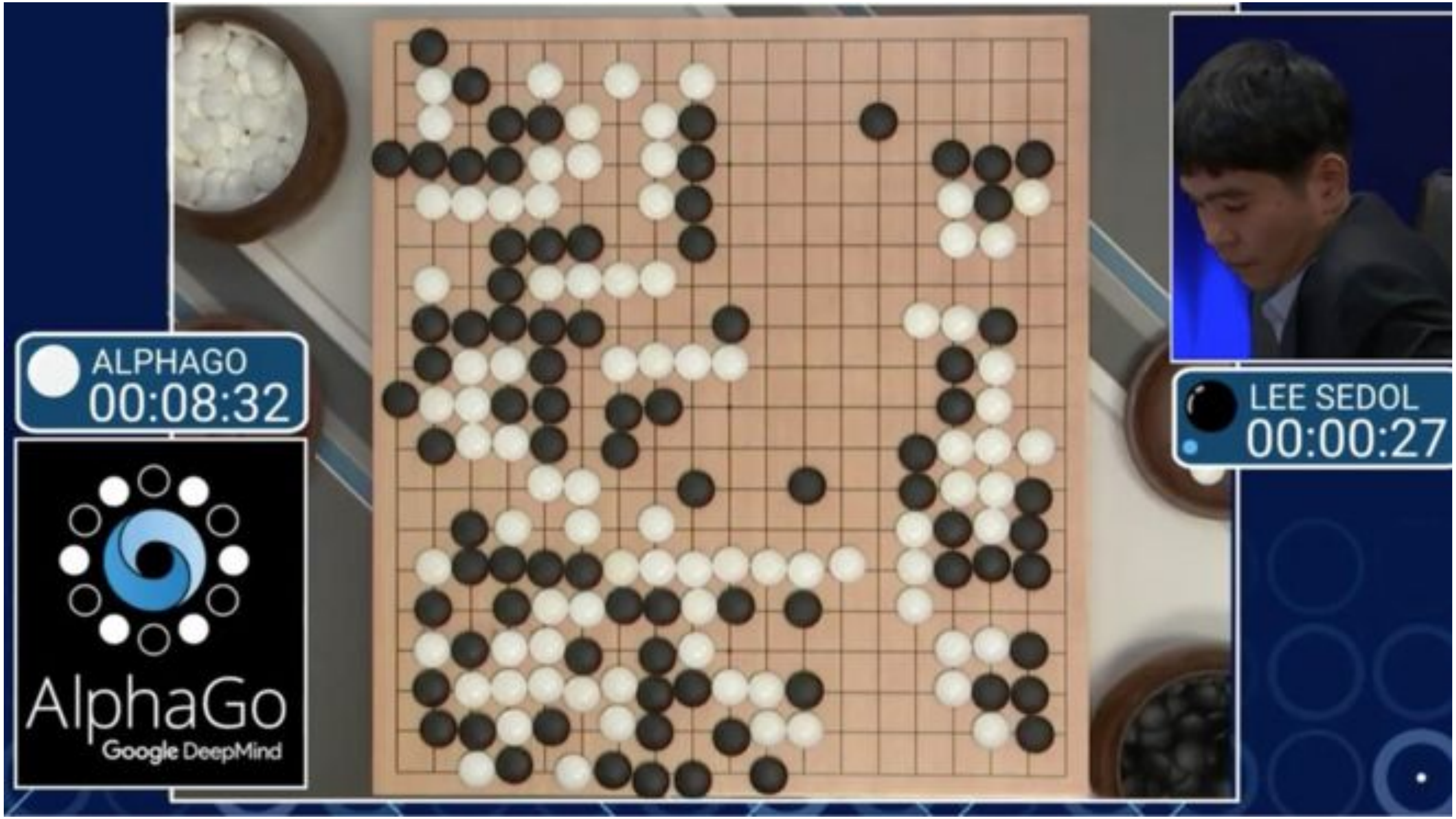
1. Silver and others: Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.
2. Silver and others: Mastering the game of Go without human knowledge. *Nature*, 2017.

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 $19 \times 19 \times 3$ 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 10^{170}.

AlphaGo



High-Level Ideas

Training and Execution

Training in 3 steps:

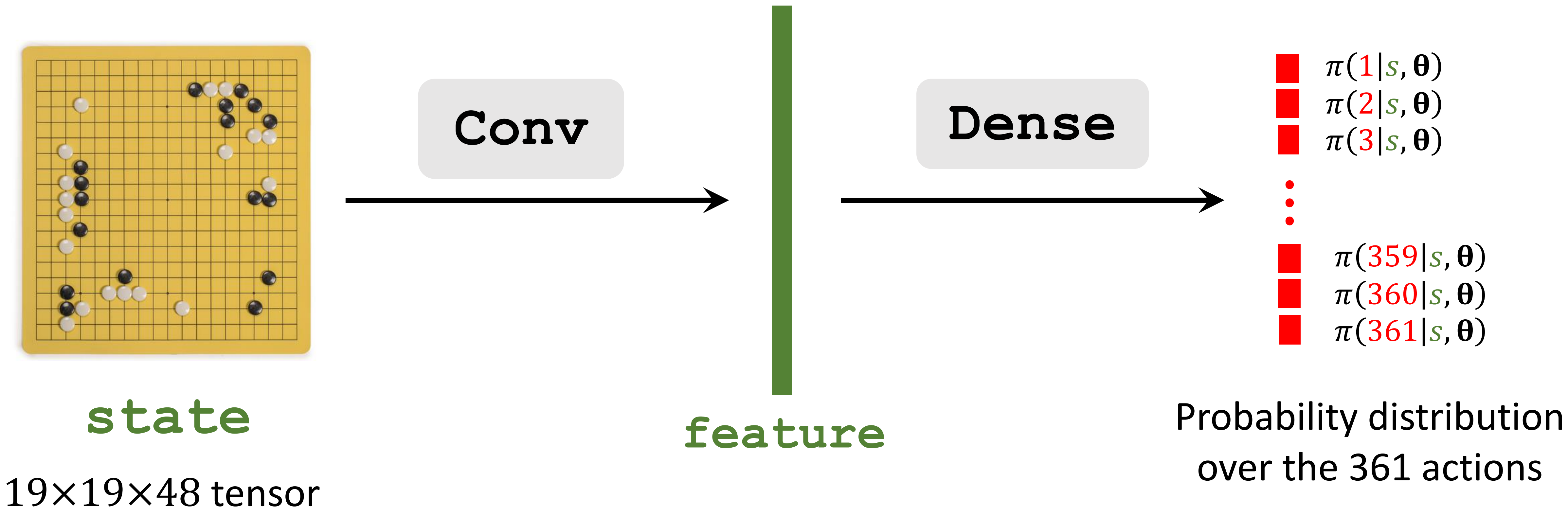
1. Initialize policy network using behavior cloning. (Supervised learning from human experience.)
2. Train the policy network using policy gradient. (Two policy networks play against each other.)
3. After training the policy network, use it to train a value network.

Execution (actually play Go games):

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

Initialize Policy Network by Behavior Cloning

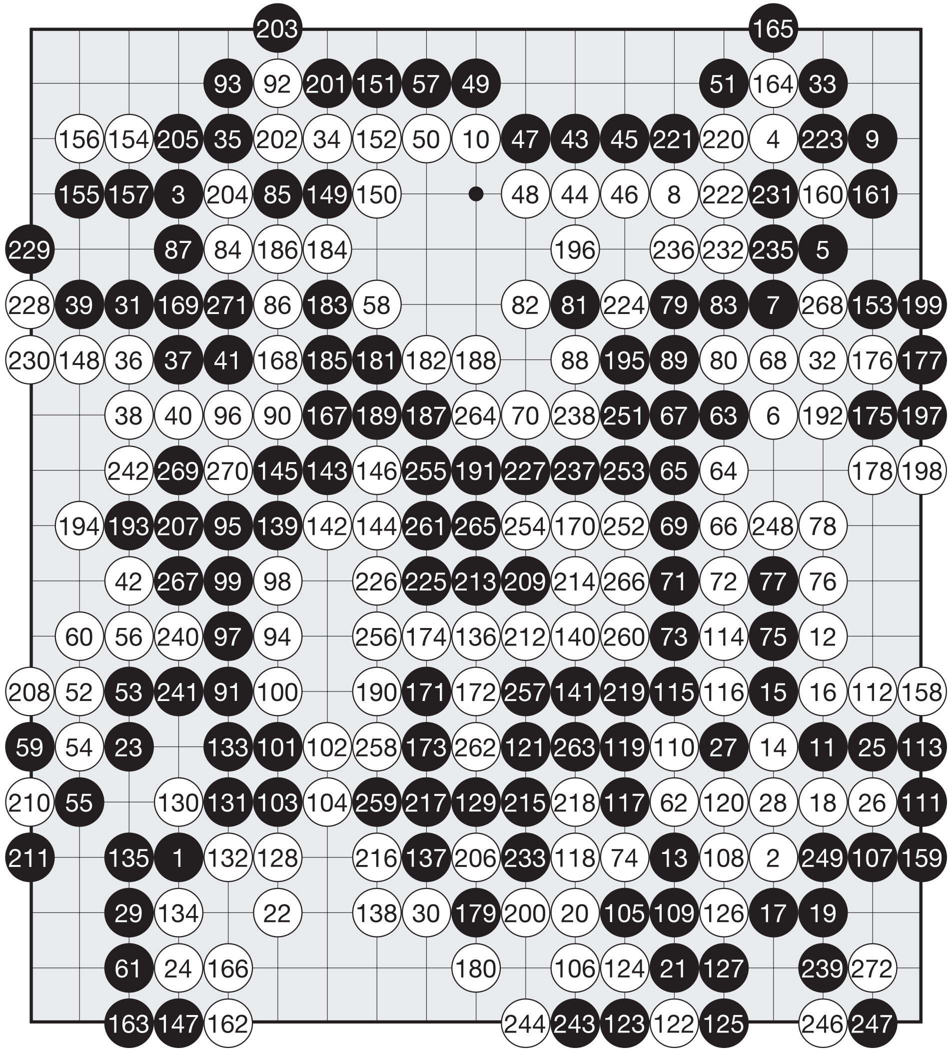
Policy Network



Human players’ 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.

Human players' record



- Initially, the network's parameters are random.
- Human' sequences of actions have been recorded. (KGS dataset has 30M games' records.)
- Behavior cloning: Let the policy network imitate human players.
- The policy network improves very quickly.
- 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.

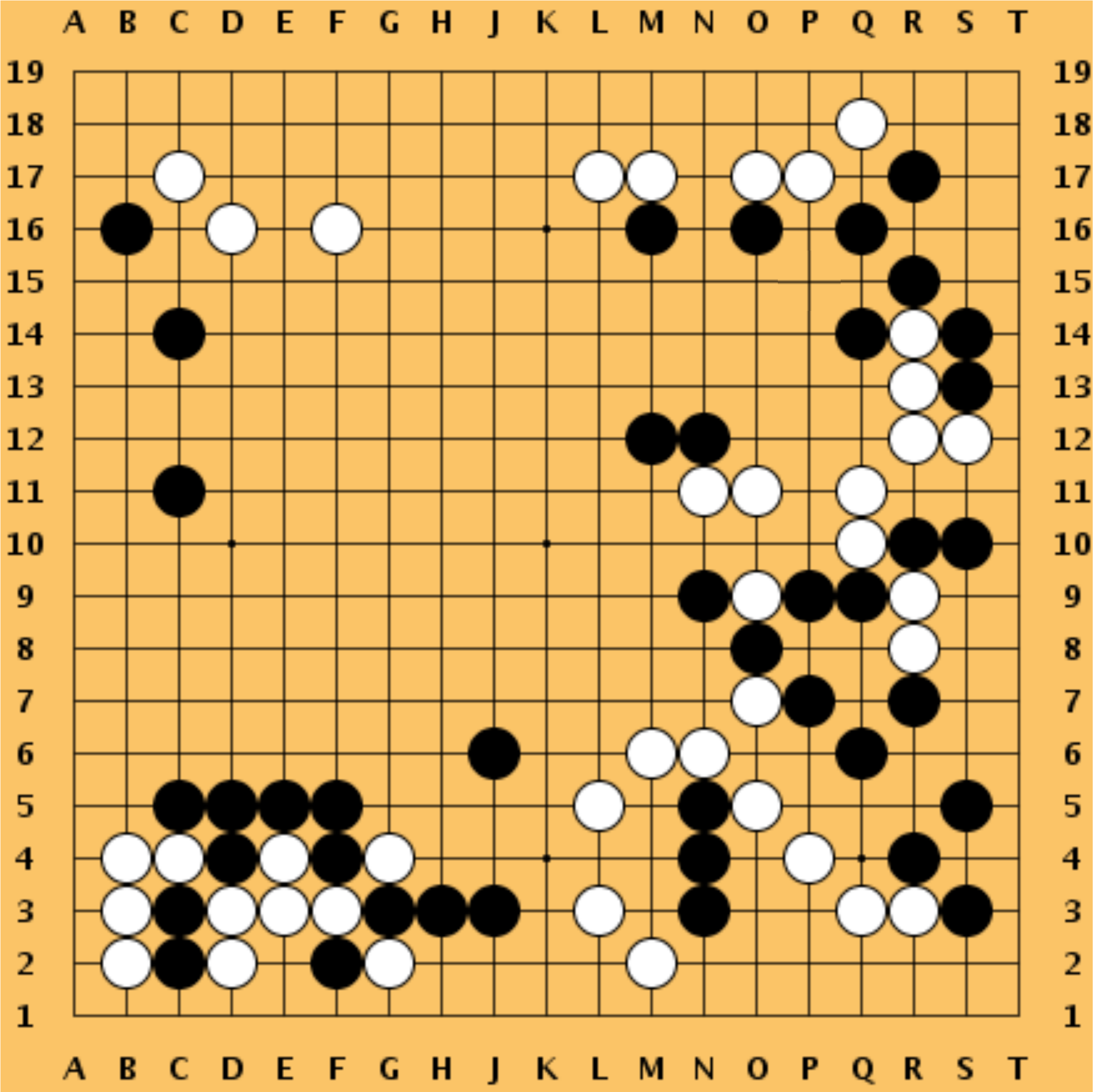
Behavior Cloning

Behavior cloning is not reinforcement learning!

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

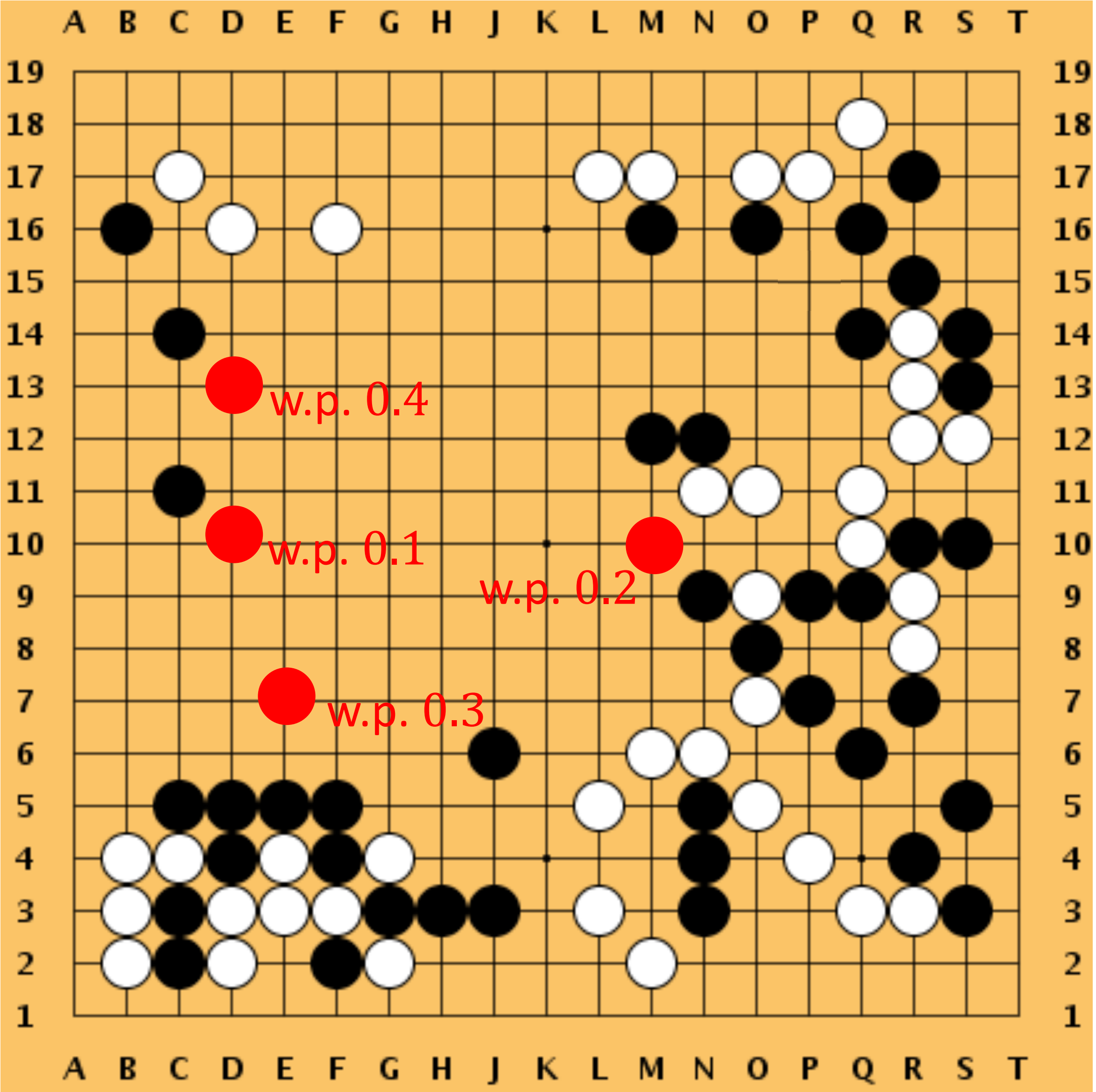
- Behavior cloning is one of the imitation learning methods.
- Behavior cloning is simply classification or regression.

Behavior Cloning



- Observe this state s_t.

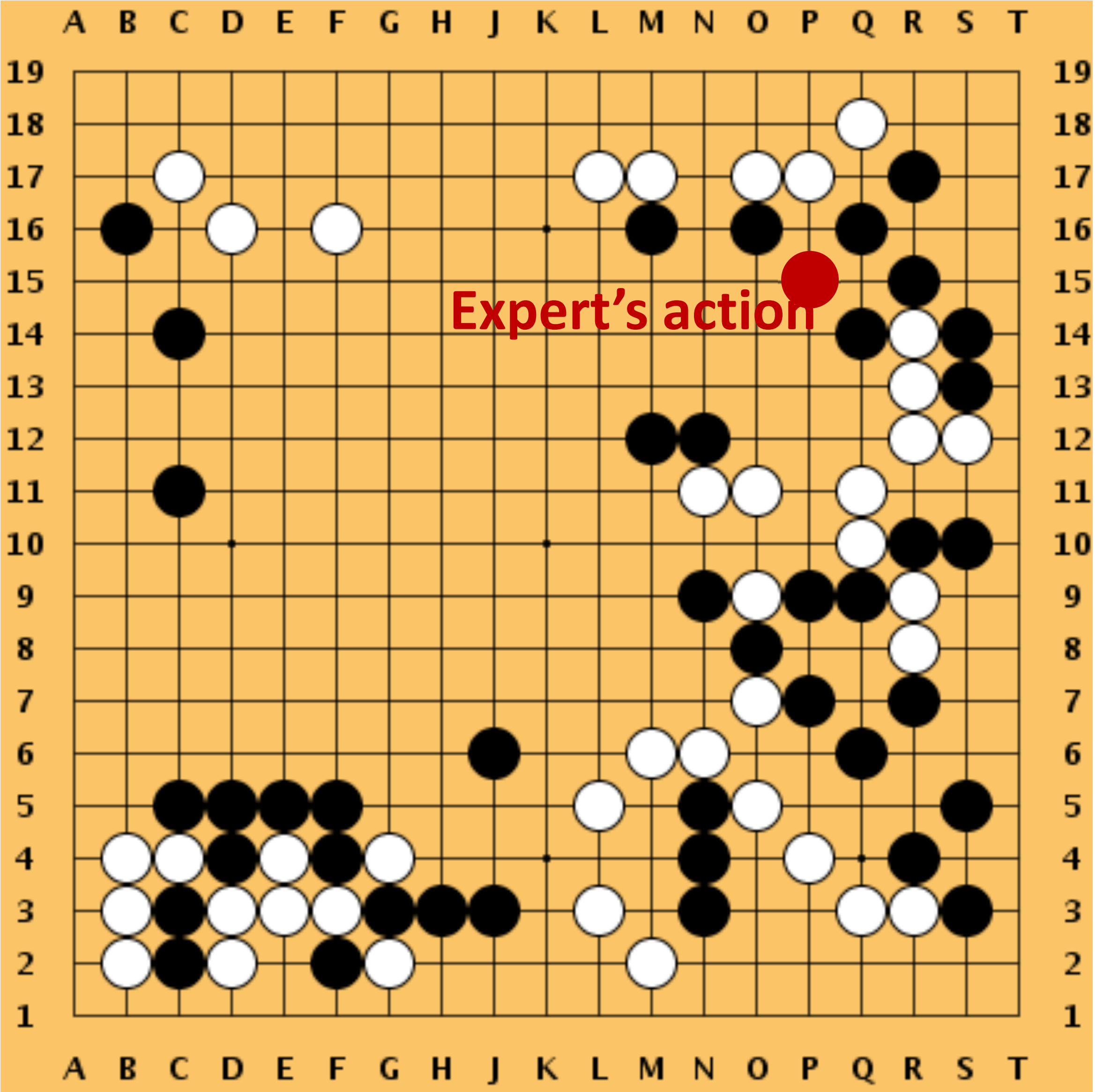
Behavior Cloning



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

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

Behavior Cloning

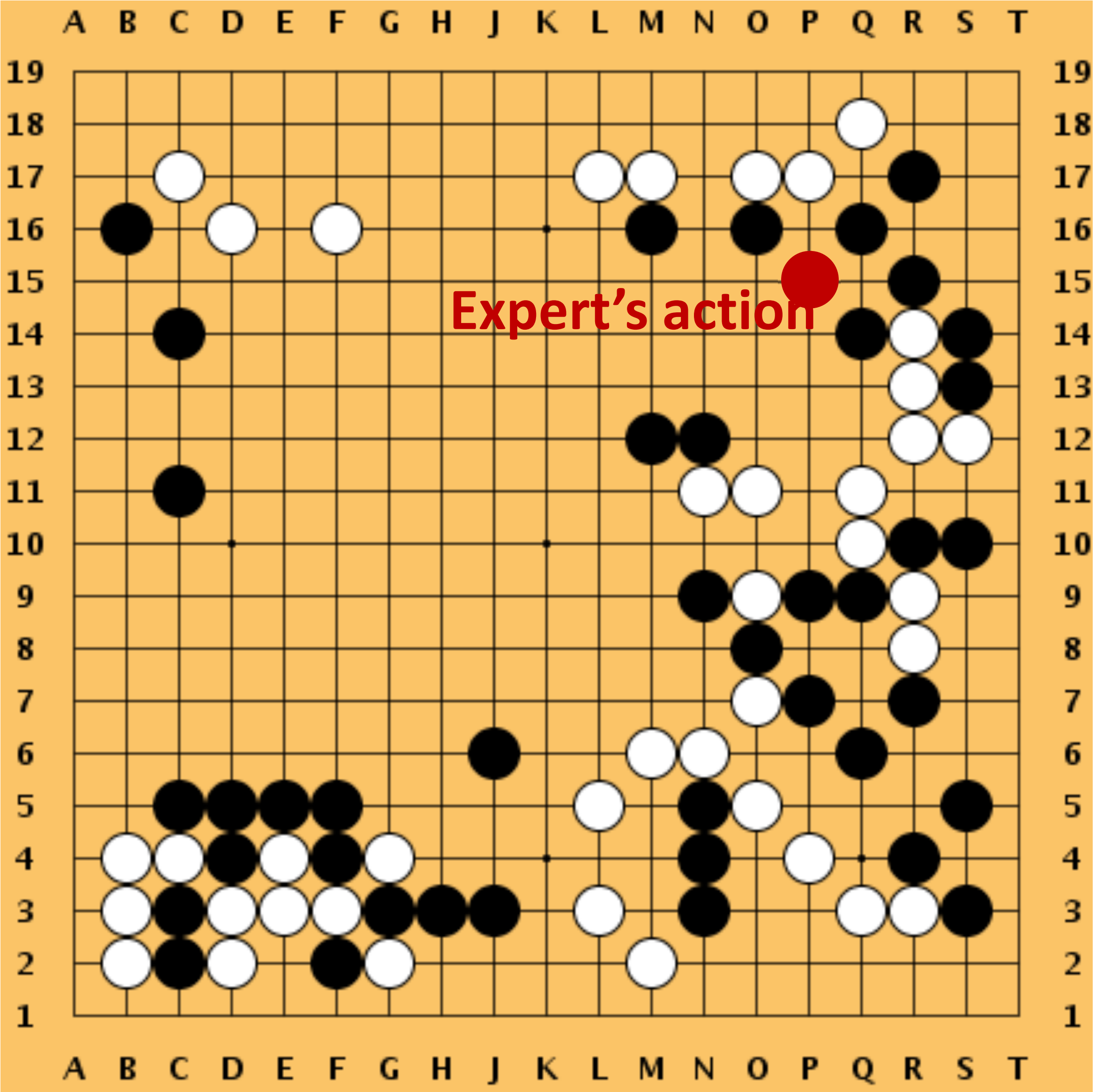


- Observe this state s_t.
- Policy network makes a prediction:

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

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

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.

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?

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

Player
(policy network with latest param)

V.S.

Opponent
(policy network with old param)

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).

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 $R_t = \sum_{i=1}^{t} r_i$. (No discount here.)
- Winner's returns: $R_1 = R_2 = \cdots = R_T = +1$.
- Loser's returns: $R_1 = R_2 = \cdots = R_T = -1$.

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 a stochastic policy gradient.
  - It is unbiased estimate of $\frac{\partial \, V(s; \boldsymbol{\theta})}{\partial \, \boldsymbol{\theta}}$.

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 a stochastic policy gradient.
- By definition, the action value is $Q_\pi(s_t, a_t) = \mathbb{E}\,[R_t | s_t, a_t, \pi]$.
- Thus, we can replace $Q_\pi(s_t, a_t)$ by R_t.
- Stochastic policy gradient: $\frac{\partial \log \pi(a_t|s_t,\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot R_t$.

Update policy network using policy gradient

Stochastic policy gradient: $\frac{\partial \log \pi(a_t|s_t,\mathbf{\theta})}{\partial \ \mathbf{\theta}} \cdot R_t.$

Update policy network using policy gradient

Stochastic policy gradient: $\frac{\partial \log \pi(a_t | s_t, \boldsymbol{\theta})}{\partial \, \boldsymbol{\theta}} \cdot R_t$.

Repeat the followings:

- Two policy networks play a game to the end. (Player v.s. Opponent.)
- Get a trajectory: $s_1, a_1, s_2, a_2, \cdots, s_T, a_T$.
- After the game ends, update the player's policy network.
  - The player's returns: $R_1 = R_2 = \cdots = R_T$. (Either $+1$ or -1.)
  - Sum of stochastic policy gradients: $\mathbf{g}_\theta = \sum_{t=1}^{T} \frac{\partial \log \pi(a_t | s_t, \boldsymbol{\theta})}{\partial \, \boldsymbol{\theta}} \cdot R_t$.
  - Update policy network: $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \cdot \mathbf{g}_\theta$.

Can we play game using the policy network?

- We have learned the policy network $\pi(a|s, \boldsymbol{\theta})$.
- Observing the current state s_t, randomly sample action

$$a_t \sim \pi(\cdot\,|s, \boldsymbol{\theta}).$$

Can we play game using the policy network?

- We have learned the policy network $\pi(a|s,\boldsymbol{\theta})$.
- Observing the current state s_t, randomly sample action

$$a_t \sim \pi(\cdot\,|s,\boldsymbol{\theta}).$$

The learned policy network is strong enough. But it is not the best…

- The policy network alone is not good enough.
- A small mistake may change the game result.

Train the Value Network

State-Value Function

Definition: State-value function.

- $V_\pi(s_t) = \mathbb{E}[R_t | s_t]$, where $R_t = +1$ (if win) and -1 (if lose) .
- The expectation is taken with respect to
  - The actions $a_t, a_{t+1}, \cdots, a_{T-1}$ sampled from the policy function $\pi(\cdot|s; \boldsymbol{\theta})$.
  - The new states $s_{t+1}, s_{t+2}, \cdots, s_T$ sampled from the state transition $p(\cdot|s, a)$.
- $V_\pi(s_t)$ depends on the policy function $\pi(a|s; \boldsymbol{\theta})$.

State-Value Function

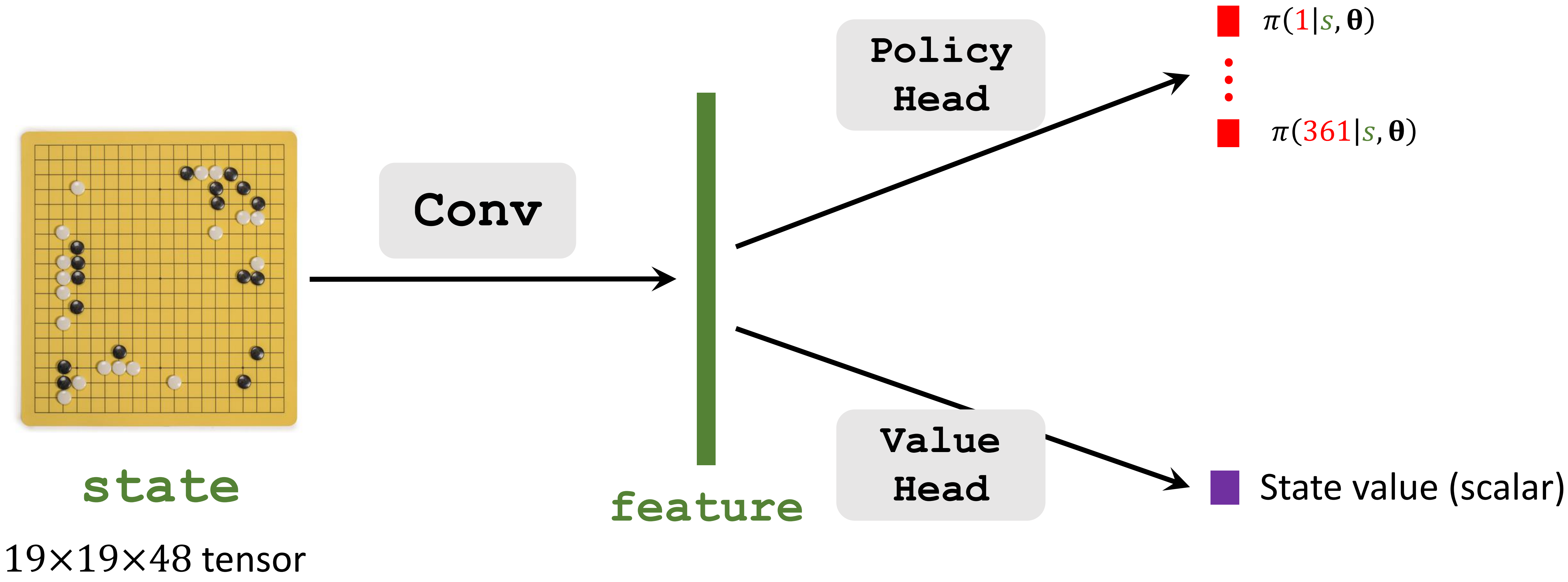
Definition: State-value function.

- $V_\pi(s_t) = \mathbb{E}[R_t | s_t]$, where $R_t = +1$ (if win) and -1 (if lose) .
- The expectation is taken with respect to
  - The actions $a_t, a_{t+1}, \cdots, a_{T-1}$ sampled from the policy function $\pi(\cdot | s; \boldsymbol{\theta})$.
  - The new states $s_{t+1}, s_{t+2}, \cdots, s_T$ sampled from the state transition $p(\cdot | s, a)$.
- $V_\pi(s_t)$ depends on the policy function $\pi(a|s; \boldsymbol{\theta})$.

Approximate state-value function using a value network.

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

Policy Network and 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 $R_1 = R_2 = \cdots = R_T = +1$.
    - If lose, let $R_1 = R_2 = \cdots = R_T = -1$.
2. Loss: $L(\mathbf{w}) = \sum_{t=1}^{T} \frac{1}{2} [v(s_t; \mathbf{w}) - R_t]^2$.
3. Update: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \frac{\partial\, L(\mathbf{w})}{\partial\, \mathbf{w}}$.

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}?

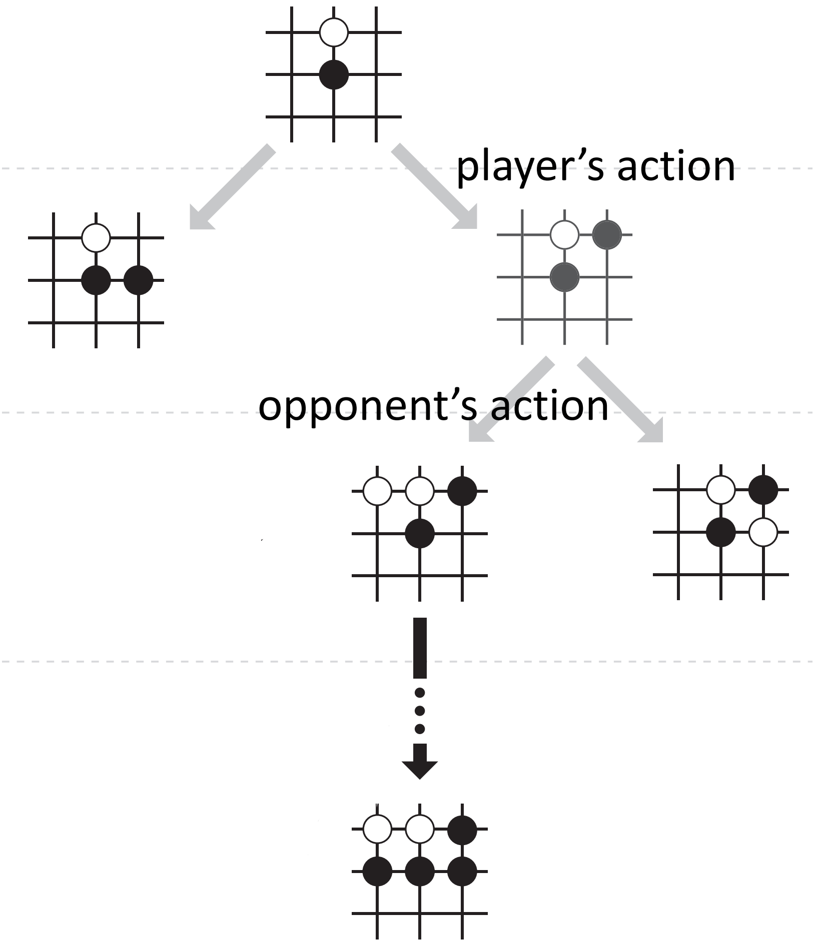
$$\vdots$$

- 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.

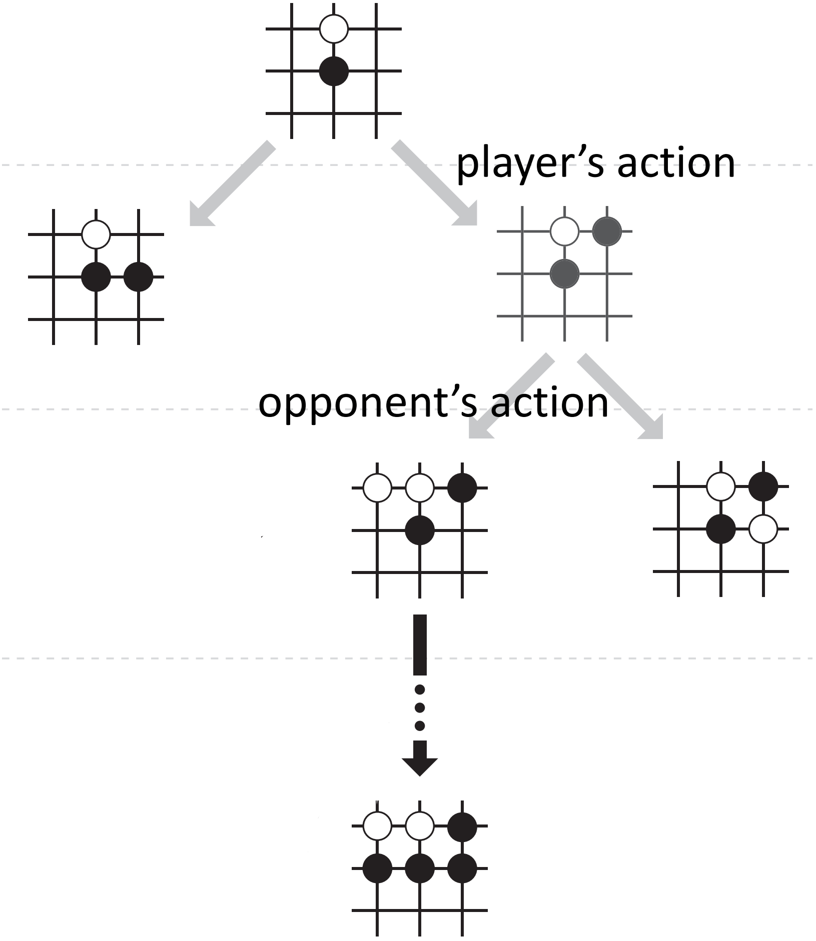
- Stark: How many did we win?
- Strange: … One.

Select actions by look-ahead search



- For all the possible actions a, look ahead and see whether a leads to win or lose.
- Repeat this procedure many times.
- Choose the action a that is most likely to win.

Select actions by look-ahead search

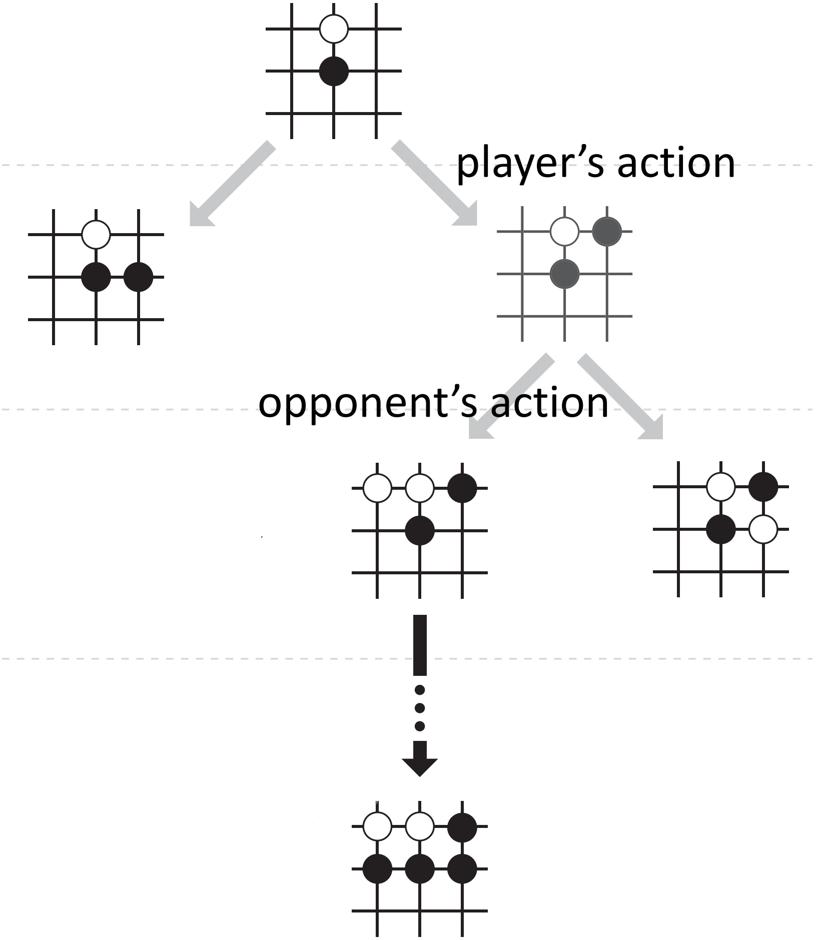


- For all the possible actions a, look ahead and see whether a leads to win or lose.
- Repeat this procedure many times.
- Choose the action a that is most likely to win.

Challenge: Exhaustive search is not feasible!

- Number of sequences of actions is 10^{170}.
- (Earth has only 10^{50} atoms.)
- (Dr. Strange saw only 10^{7} possible outcomes.)

Select actions by look-ahead search



Challenge: Exhaustive search is not feasible!

Fortunately, search does not have to be exhaustive.

- Policy function and value function have been learned.
- Search the actions that worth searching.

Monte Carlo Tree Search (MCTS)

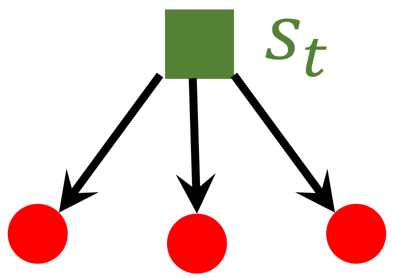
- Observe the current state s_t, look ahead till the end of the game.
- Do many simulations; see which action has the biggest winning chance.

Monte Carlo Tree Search (MCTS)

- Observe the current state s_t, look ahead till the end of the game.
- Do many simulations; see which action has the biggest winning chance.

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

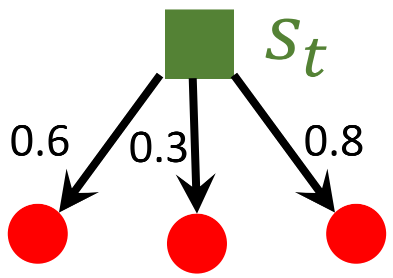
1. **Selection:** The player makes an action. (Imaginary action; not actual move.)
2. **Expansion:** The opponent makes an action and the state updates. (Also imaginary action; made by the policy network.)
3. **Evaluation:** Evaluate the state-value; then play the game to the end (known as ``roll-out”); finally, receive a reward.
4. **Backup:** Use the state-value and reward to update action-values.



valid actions

Step 1: Selection

Question: Observing s_t, which action shall we explore?



Step 1: Selection

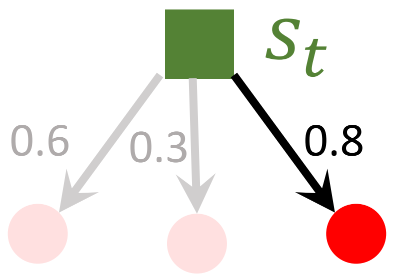
Question: Observing s_t, which action shall we explore?

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.

- $\text{score}(a)$ is an upper bound of $Q(a)$.
- If a has been visited many times (big $N(a)$), then the confidence is high, and $\text{score}(a)$ is close to $Q(a)$.



Step 1: Selection

Question: Observing s_t, which action shall we explore?

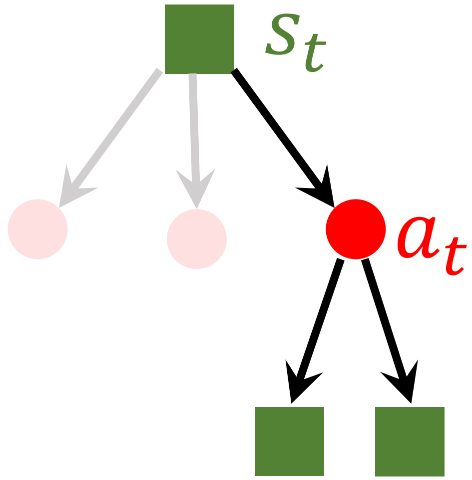
First, for all the valid actions a, calculate the score:

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

- $Q(a)$: 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, $a_t \leftarrow$ the action a that has the biggest $\text{score}(a)$.

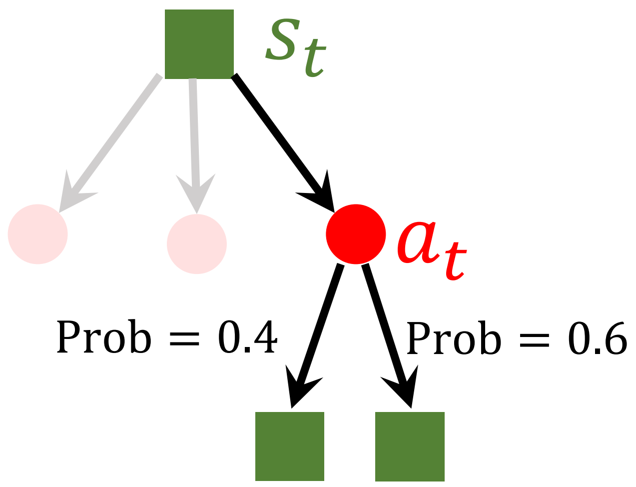
Imaginary action; not actually performed.



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

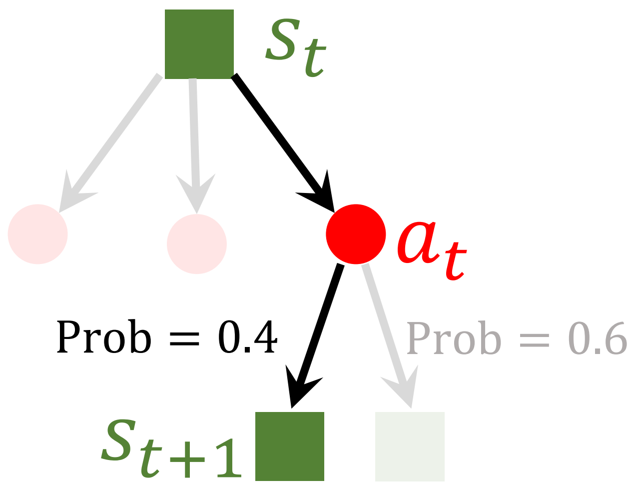
Question: What will be the opponent's action?

- 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.

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



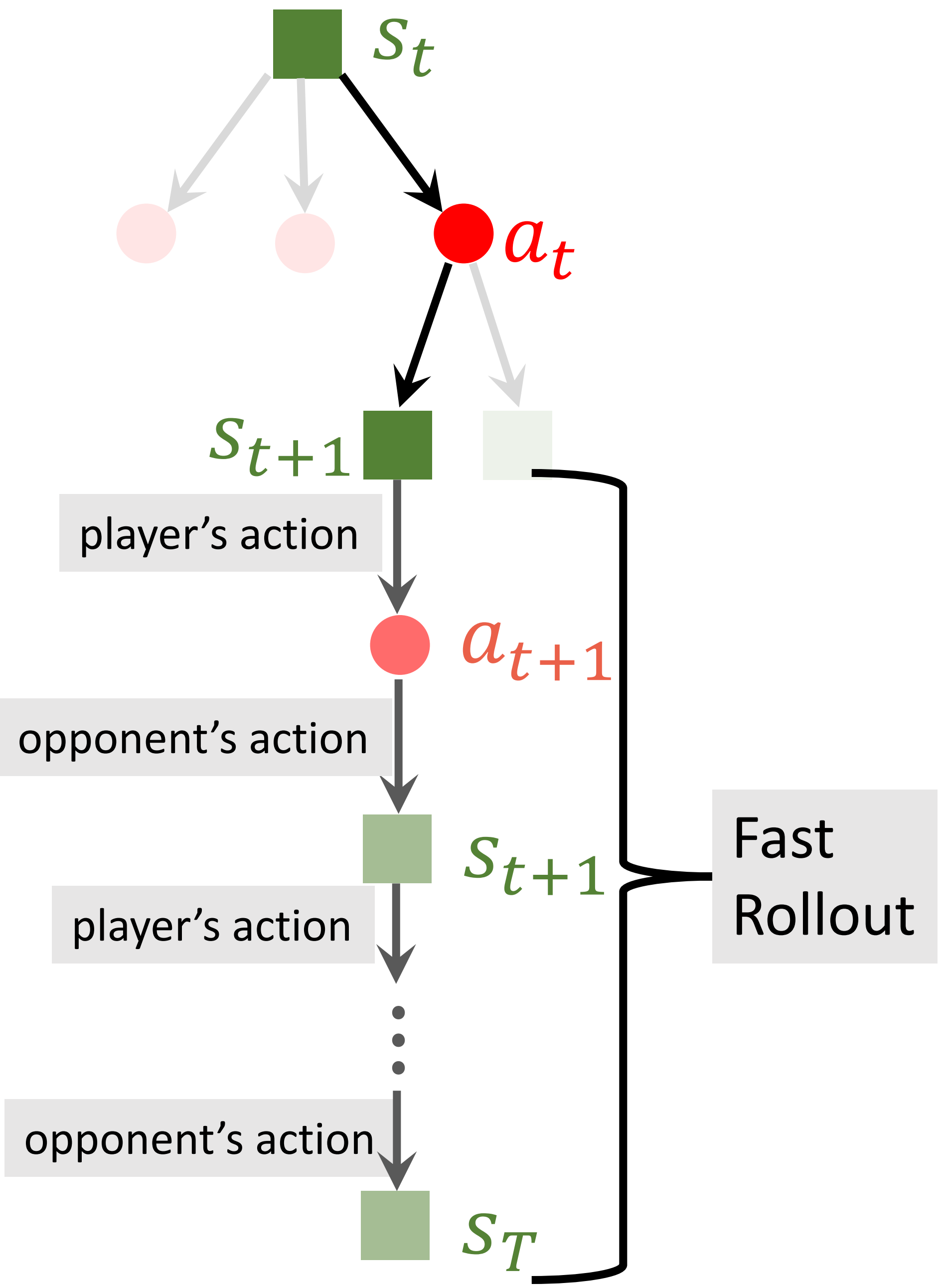
Step 2: Expansion

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

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

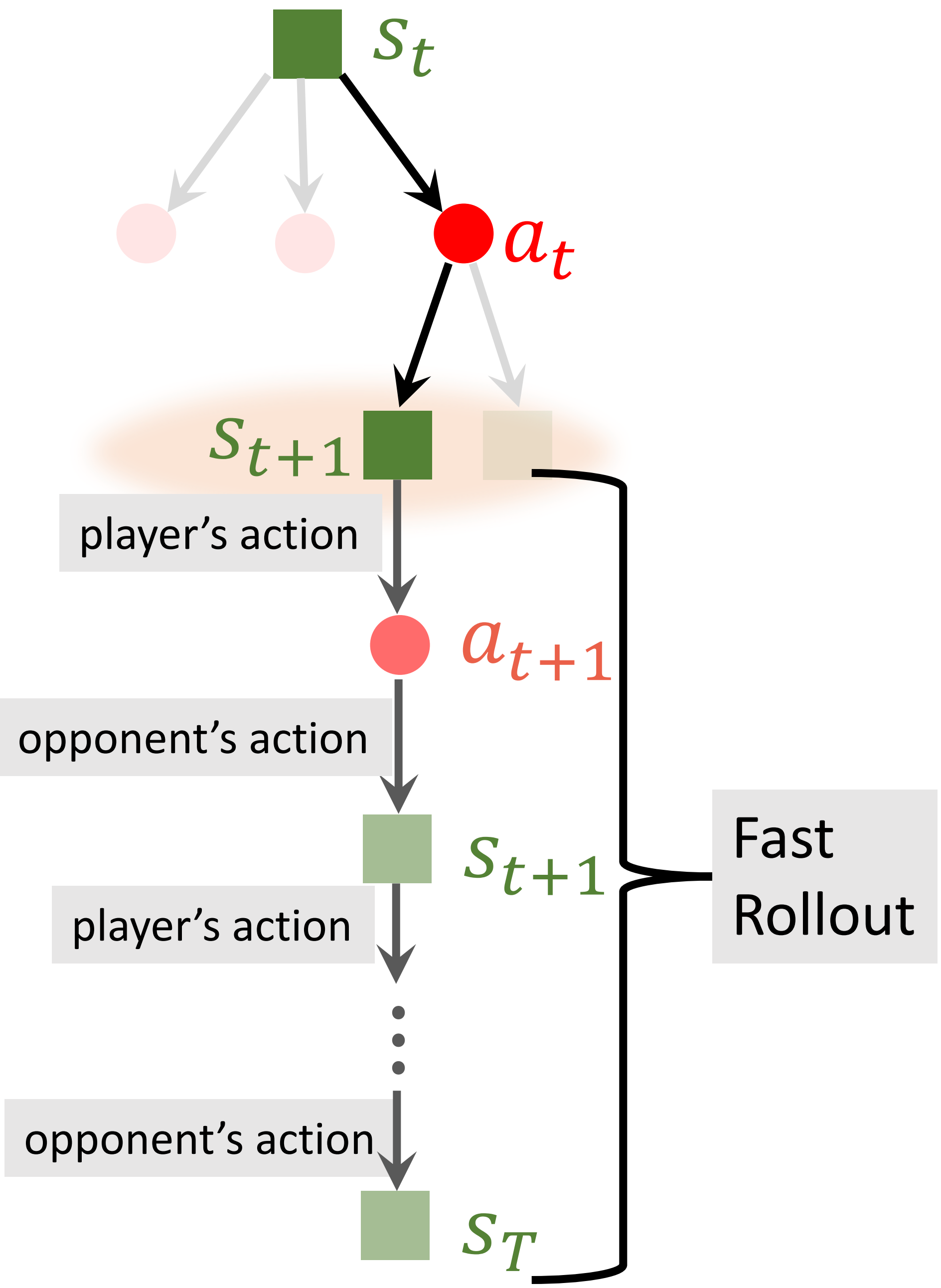
Here, s_t' is the state observed by the opponent.



Step 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: $A_k \sim \pi(\cdot \mid s'_k; \boldsymbol{\theta})$.



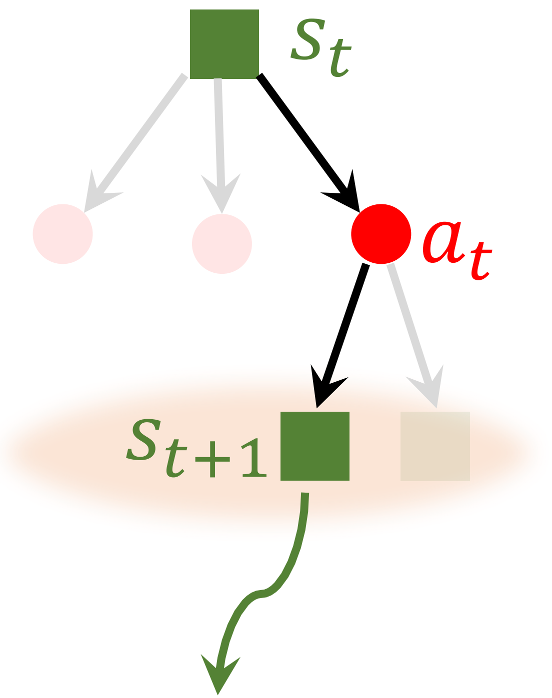
Step 3: Evaluation

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: $A_k \sim \pi(\cdot \mid s'_k; \boldsymbol{\theta})$.

Evaluate the state s_{t+1}.

- Record: $V(s_{t+1}) = \frac{1}{2} v(s_{t+1}; \mathbf{w}) + \frac{1}{2} r_T$.
- $v(s_{t+1}; \mathbf{w})$: is the value network.
- r_T: Reward received at the end of game.
  - Win: $r_T = +1$.
  - Lose: $r_T = -1$.



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

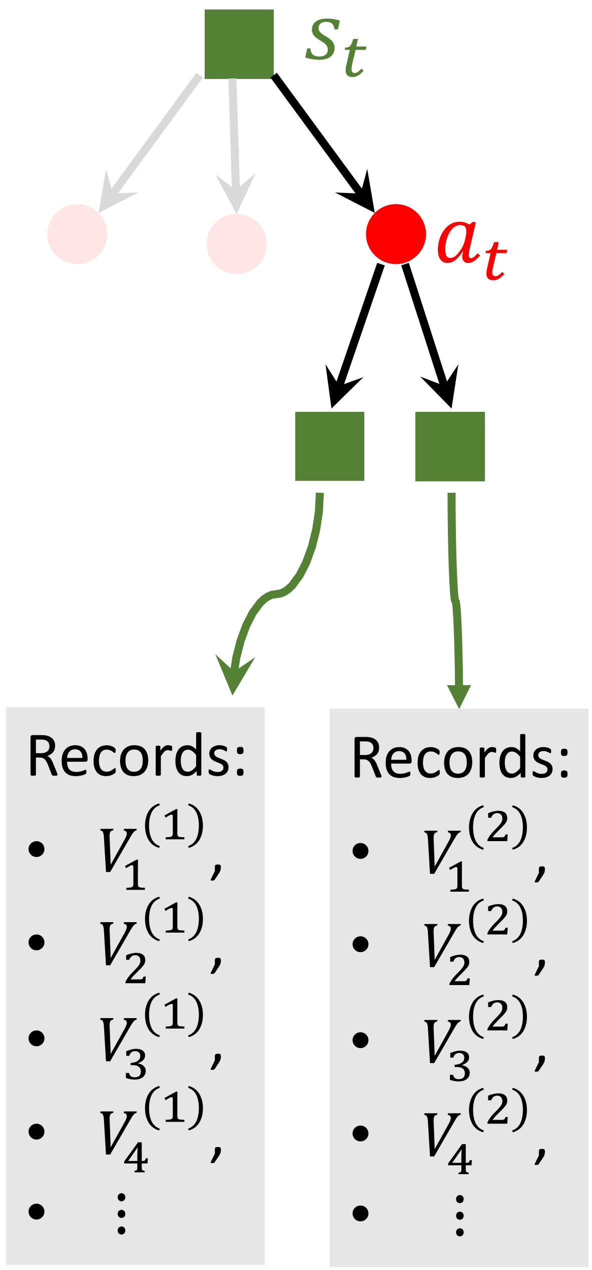
Step 3: Evaluation

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: $A_k \sim \pi(\cdot \mid s'_k; \boldsymbol{\theta})$.

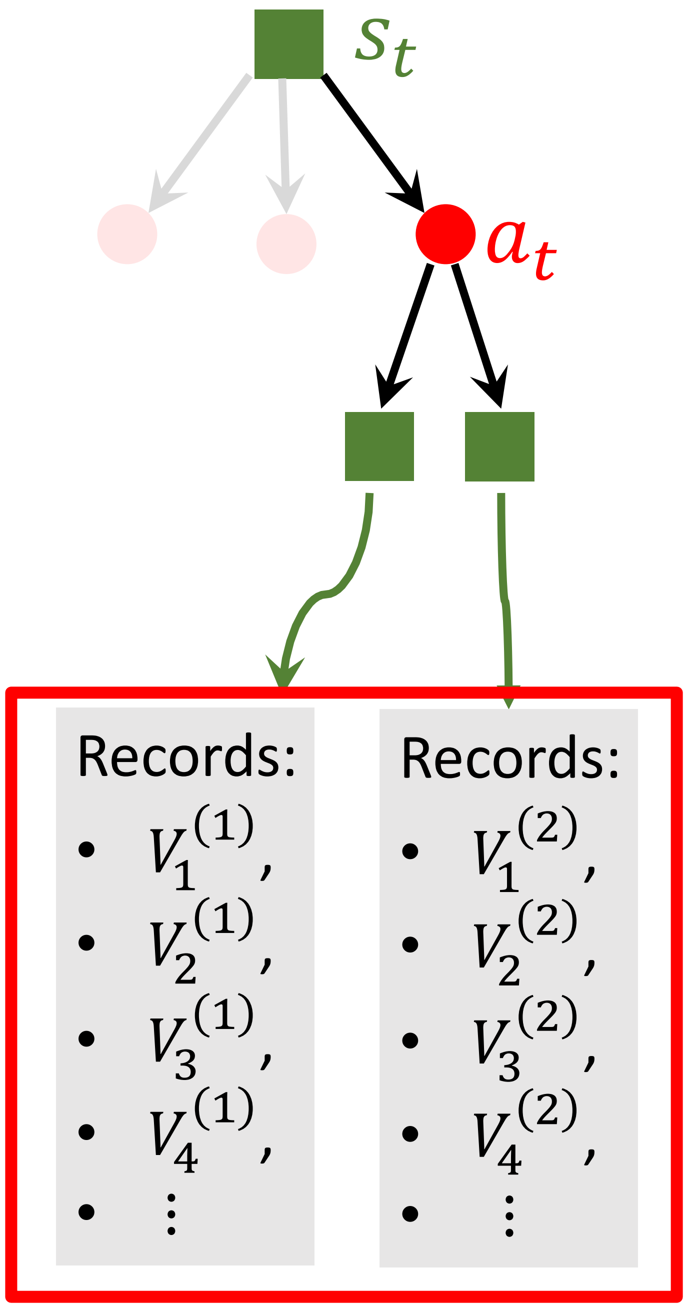
Evaluate the state s_{t+1}.

- Record: $V(s_{t+1}) = \frac{1}{2} v(s_{t+1}; \mathbf{w}) + \frac{1}{2} r_T$.
- $v(s_{t+1}; \mathbf{w})$: is the value network.
- r_T: Reward received at the end of game.
  - Win: $r_T = +1$.
  - Lose: $r_T = -1$.



Step 4: Backup

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

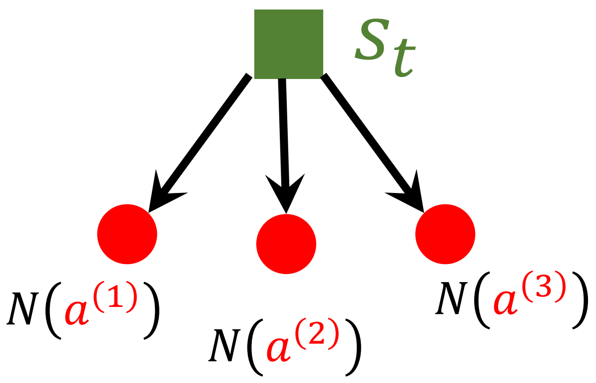


Step 4: Backup

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

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

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



Decision Making after MCTS

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

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

AlphaGo Zero

AlphaGo Zero v.s. AlphaGo

- AlphaGo Zero is stronger than AlphaGo. (100-0 against AlphaGo.)
- 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 AlphaGo, human experience is harmful.
- AlphaGo Zero is learned purely by playing against itself.

- Is behavior cloning useless?
- For Go game, self-playing does not have cost (except for energy.)
- What if a surgery robot (randomly initialized) is learned purely by performing surgery? (Human experience is not used.)
- What if a self-driving car (randomly initialized) is learned purely by self-driving? (Human's supervision is not used.)

Training of policy network

- AlphaGo Zero uses MCTS in training. (AlphaGo does not.)
- At state s_t.
- Predictions made by policy network:

$$\mathbf{p} = [\pi(a = 1|s_t; \boldsymbol{\theta}), \cdots, \pi(a = 361|s_t; \boldsymbol{\theta})] \in \mathbb{R}^{361}.$$

- Predictions made by MCTS:

$$\mathbf{n} = [N(a = 1), N(a = 2), \cdots, N(a = 361)] \in \mathbb{R}^{361}.$$

- Loss: $l_t(\boldsymbol{\theta}) = \text{CrossEntropy}(\mathbf{n}, \mathbf{p})$.
- Use $\frac{\partial\, l_t(\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}$ to update $\boldsymbol{\theta}$.
- In sum, supervision is from MCTS. (It is different from policy gradient.)

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.