Maximum Entropy IRL (continue)

Recap on the setting for inverse RL

Finite horizon MDP $\mathscr{M} = \{S, A, H, c, P, \mu, \pi^\star\}$

We have a dataset $\mathscr{D} = (s_i^\star, a_i^\star)_{i=1}^M \sim d_\mu^{\pi^\star}$

Key Assumption on cost:

$c(s, a) = \langle \theta^\star, \phi(s, a) \rangle$, **linear w.r.t feature** $\phi(s, a)$

Plan for Today:

1. MaxEnt IRL alg

2. Case study of AlphaGo

Maximum Entropy Inverse RL formulation

Matching expert's feature with an entropy regularization

$$\arg\min_{\pi} \left\| \mathbb{E}_{s,a\sim d^{\pi}}\phi(s,a) - \mathbb{E}_{s,a\sim d^{\pi^\star}}\phi(s,a) \right\|_2 - \lambda \mathbb{E}_{s\sim d^{\pi}} \text{entropy}(\pi(\cdot \mid s))$$

π's expected feature $\quad$ $\pi^\star$'s expected feature $\quad$ Encourage diversity in π

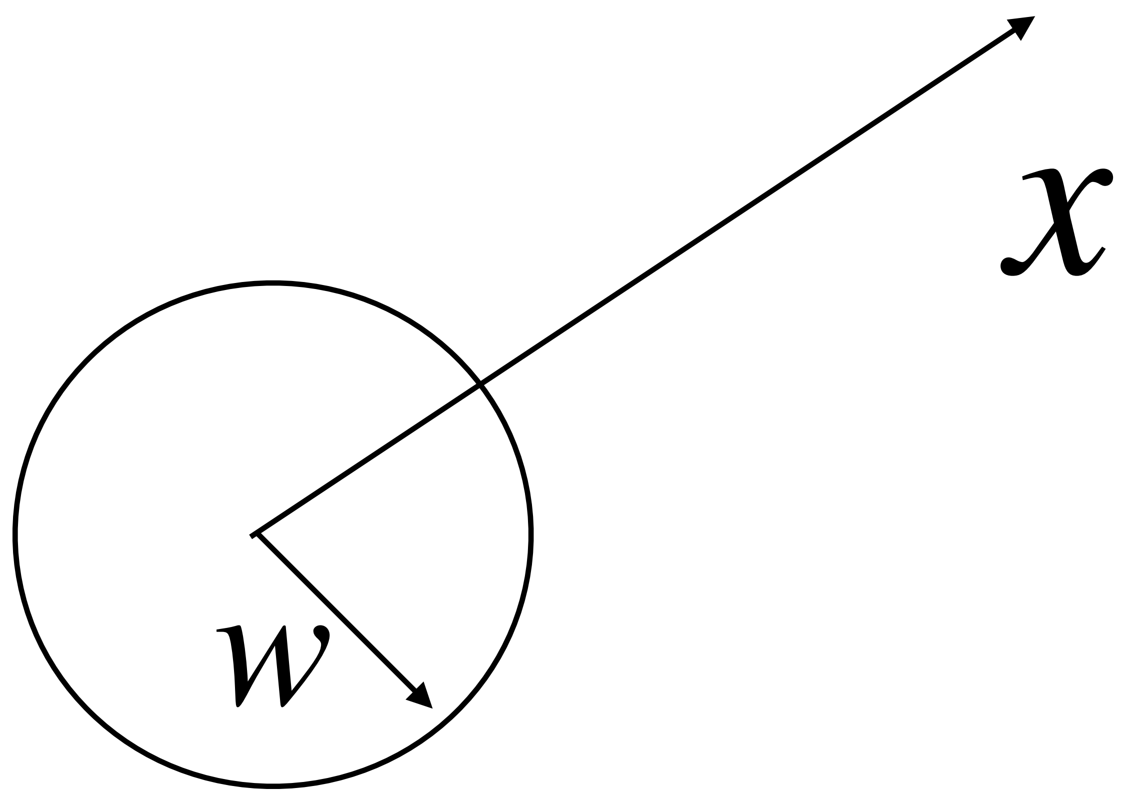
Q: why matching experts feature is enough? (Reward the linear reward assumption..)

This isn't an RL problem (e.g., not maximizing some reward), seems hard to optimize π…

Maximum Entropy Inverse RL formulation

Re-write the ℓ_2 norm as an optimization problem..

$$\|x\|_2 = \max_{w:\|w\|_2 \leq 1} w^\top x$$



Maximum Entropy Inverse RL formulation

$$\arg\min_{\pi} \left\| \mathbb{E}_{s,a\sim d^{\pi}}\phi(s,a) - \mathbb{E}_{s,a\sim d^{\pi^\star}}\phi(s,a) \right\|_2 - \lambda \mathbb{E}_{s\sim d^{\pi}}\text{entropy}(\pi(\cdot\,|\,s))$$

Using this new form for $\ell_2$norm

$$\arg\min_{\pi} \max_{w:\|w\|_2\le 1} \mathbb{E}_{s,a\sim d^{\pi}} w^\top\phi(s,a) - \mathbb{E}_{s,a\sim d^{\pi^\star}} w^\top\phi(s,a) - \lambda \mathbb{E}_{s\sim d^{\pi}}\text{entropy}(\pi(\cdot\,|\,s))$$

1. We can swap the order and write this as $\max_{w:\|w\|_2\le 1} \min_{\pi} \ldots$ (proof out of the scope) …

2. Given w, optimize π is like an RL with cost $w^\top\phi$ and entropy reg…

Maximum Entropy Inverse RL Algorithm framework

$$\max_{w:\|w\|_2\le 1} \min_{\pi} \mathbb{E}_{s,a\sim d^{\pi}} w^{\top}\phi(s,a) - \mathbb{E}_{s,a\sim d^{\pi^\star}} w^{\top}\phi(s,a) - \lambda \mathbb{E}_{s\sim d^{\pi}} \text{entropy}(\pi(\cdot\,|\,s))$$

Initialize $w^0 \in \mathbb{R}^d$

For $t = 0 \to T-1$

$$\pi^t = \arg\min_{\pi} \mathbb{E}_{s,a\sim d^{\pi}_{\mu}}\left[(w^t)^{\top}\phi(x,a) - \lambda \text{Ent}(\pi(\cdot\,|\,s))\right]$$

An RL problem w/ cost $c(s,a) := (w^t)^{\top}\phi(s,a)$ and entropy reg (e.g., in practice, run PPO w/ entroy regularization)

(# compute the best policy given the current cost)

$$w^{t+1} = w^t + \eta\left(\mathbb{E}_{s,a\sim d^{\pi^t}_{\mu}}\phi(s,a) - \mathbb{E}_{s,a\sim d^{\pi^\star}_{\mu}}\phi(s,a)\right)$$

(# gradient update on cost vector w)

Return w^T, π^T

(# Learned cost function $\phi^{\top}(w^T)$, and its optimal policy)

Plan for Today:

1. MaxEnt IRL alg

2. Case study of AlphaGo


ALPHAGO
00:08:32
LEE SEDOL
00:00:27
BBC NEWS

Setting: Two player Markov Games:

$$\mathcal{M} = \{S, A, f, r, H, s_0\}$$

We have two players π_1 and π_2, they take turn to play:

$$s_0, \quad a_0 \sim \pi_1(s_0), s_1 = f(s_0, a_0), \quad a_1 \sim \pi_2(s_1), s_2 = f(s_1, a_1), \ldots, s_H$$

Sparse reward at the termination state: $r(s_H) = 1$ if wins, -1 otherwise

Min-max formulation:

$$\max_{\pi_1} \min_{\pi_2} \mathbb{E}\left[r(s_H) \mid \pi_1, \pi_2\right]$$

Setting: Two player Markov Games:

It's a zero-sum game, i.e., they cannot both win or both lose…

Player 2 tries to minimize the expected win rate of player 1, which is equivalent to maximizes its own win rate

Setting: Two player Markov Games:

Min-max formulation:

$$\max_{\pi_1} \min_{\pi_2} \mathbb{E}\left[r(s_H) \,|\, \pi_1, \pi_2 \right]$$

Go has known and deterministic dynamic, i.e., $s' = f(s, a)$ is known and simple, in theory we can do **Dynamic Programming** to solve the max-min formulation..

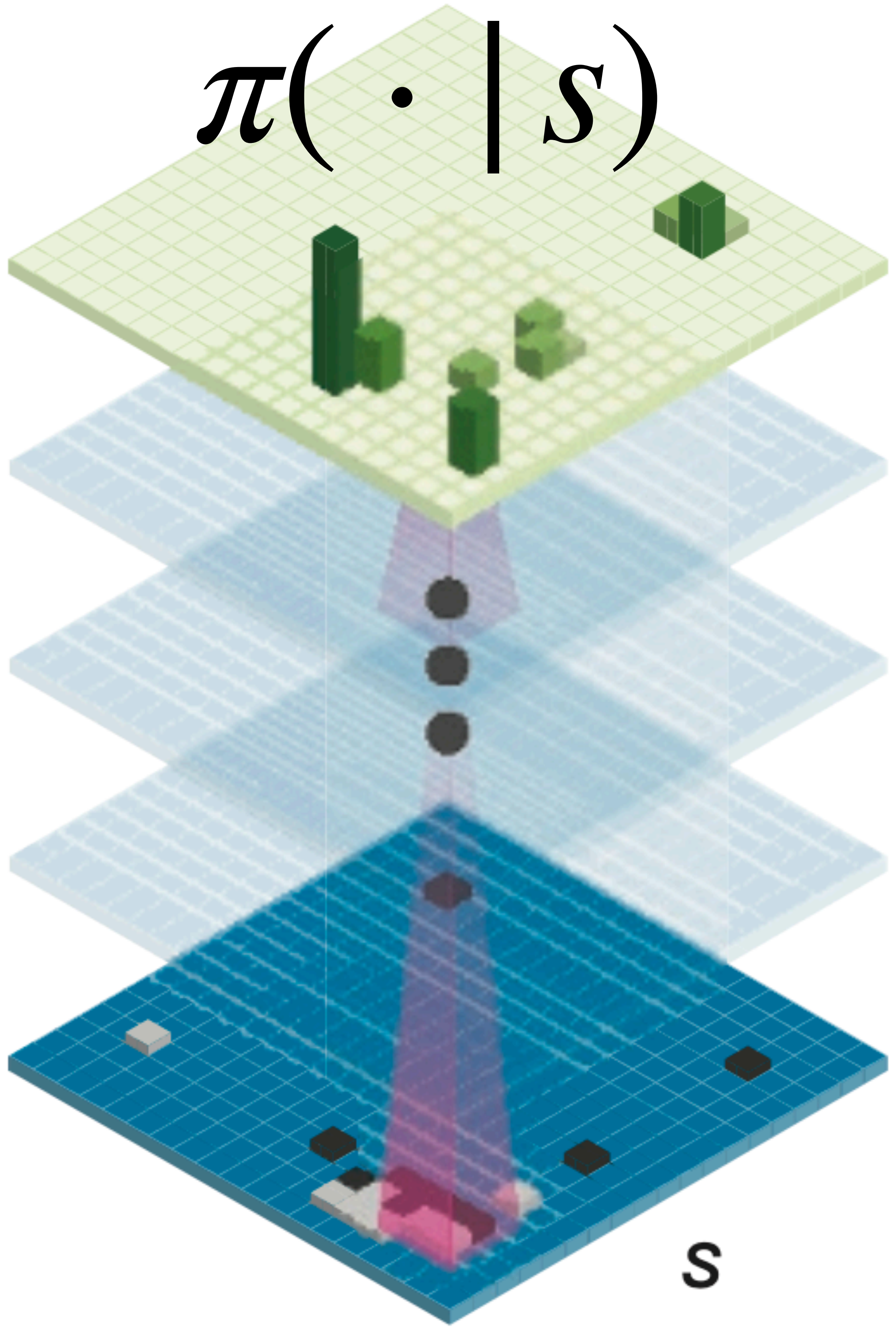
But…

For Go, state space is huge…

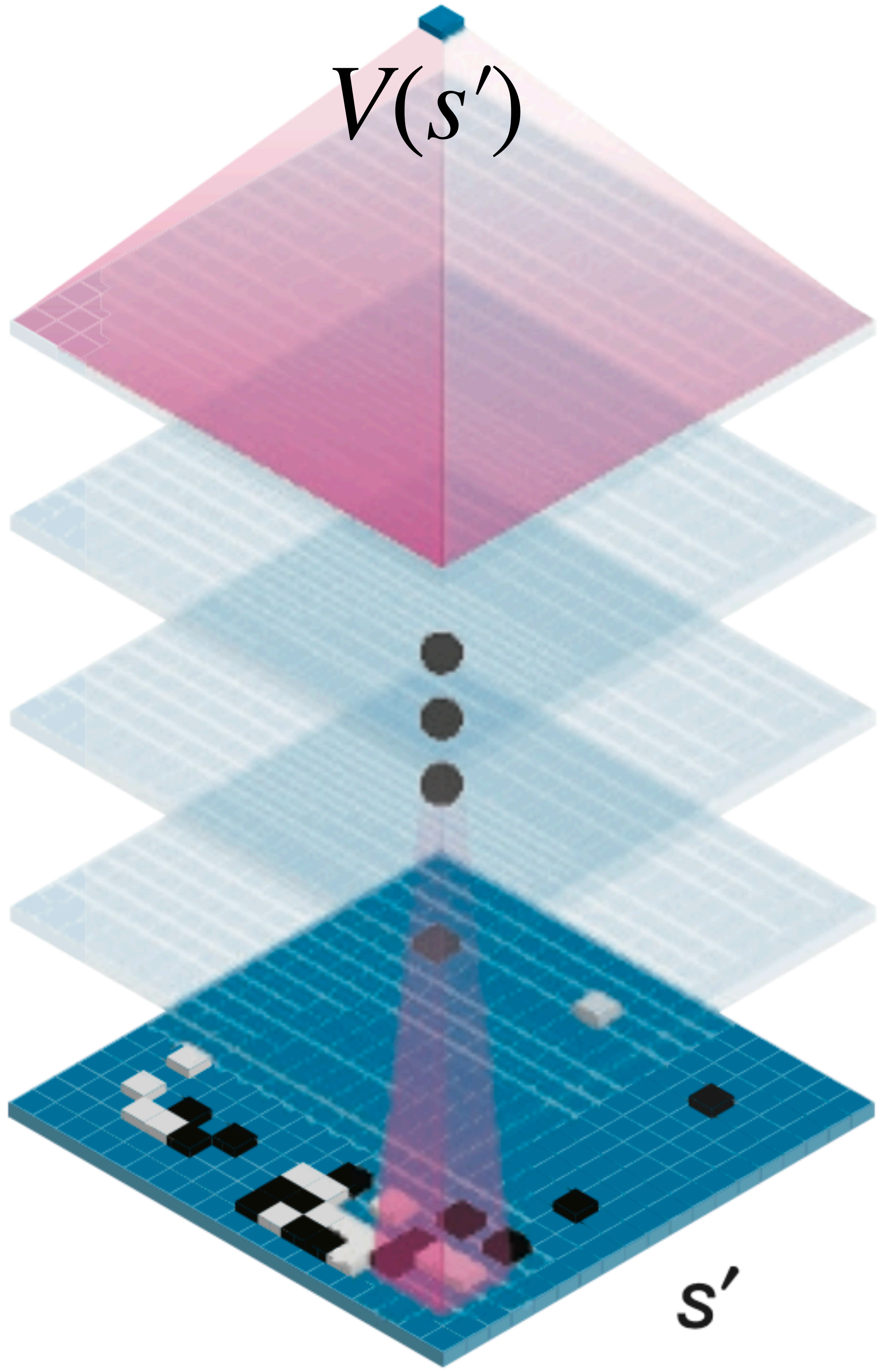
Thus, we cannot enumerate, we must **generalize via function approximation..**

Setting: Function Approximation

1. Policy Network $\approx \pi^\star$



2. Value Network $\approx V^\star(s')$



1. Warm start our policy net via Imitation Learning

1. Randomly sampled an expert dataset containing
30m (s, a) pairs from KGS Go Server…

2. Form imitation learning loss function, e.g., Negative Log-likelihood

$$\min_{\pi} \sum_{s,a} -\ln \pi(a \mid s)$$

3. Optimize via Stochastic Gradient Descent:

$$\theta_{t+1} = \theta_t - \eta \sum_{(s,a)\in B} \nabla_\theta \left(-\ln \pi_{\theta_t}(a \mid s) \right) / |B|$$

Behavior Cloning!

How well can it predict expert moves on a hold out test dataset?

It achieves 57% accuracy on expert test dataset

How well does this BC policy perform?

Test it against the open-source Go program: Pachi (ranked 2 amateur dan on KGS)

Win rate: 11%

2. Further Improving Policy via PG on Self-playing

1. We warm start our PG procedure using the BC policy…

2. We then iterate as follows:

$$\pi_{\theta_0} = \pi_{BC}$$

For $t = 0 \to T - 1$ (# fictitious play to avoid catastrophic forgetting..)

Randomly select a previous policy $\pi_{\theta_\tau}, \quad \tau < t$

Play π_{θ_t} against π_{θ_τ}, get a trajectory $(s_0, a_0, s_1, a_1', s_2, a_2, s_3, a_3' \ldots s_H)$

PG update: $\theta_{t+1} = \theta_t + \eta \sum_{h:a_h \sim \pi_{\theta_t}} \nabla_\theta \ln \pi_{\theta_t}(a_h \mid s_h) r(s_H)$

How does the performance improved after PG optimization?

Test it against the open-source Go program: Pachi (ranked 2 amateur dan on KGS)

RL policy has win rate 85%

Comment: this is where we are for LLM training:
pre-training + SFT (e..g., BC on internet web data), followed by RLHF
with REINFORCE, PPO, DPO, REBEL, etc

But to beat human champions on Go, this is clearly not enough yet…

3. Final stage of training: Learn a value function $\hat{V}(s) \approx V^\star$

Denote the PG policy as $\hat{\pi}$, we will approximate $V^{\hat{\pi}}$ instead:

$$V^{\hat{\pi}}(s) = \mathbb{E}\left[r(s_H) \mid s_0 = s, \hat{\pi}, \hat{\pi}\right]$$

i.e., the value of the game when both players play $\hat{\pi}$, starting at s

We use simple least square regression here:

$$\min_{\beta} \sum_{s,z} (V_\beta(s) - z)^2$$

Where s is a **random state in one game play**, and z is the outcome of the play..

(We only keep one sample per game play, i.e., we are really sampling $s \sim d^{\hat{\pi}}$ i.i.d)

Final stage of training: Learn a value function $V(s) \approx V^\star$

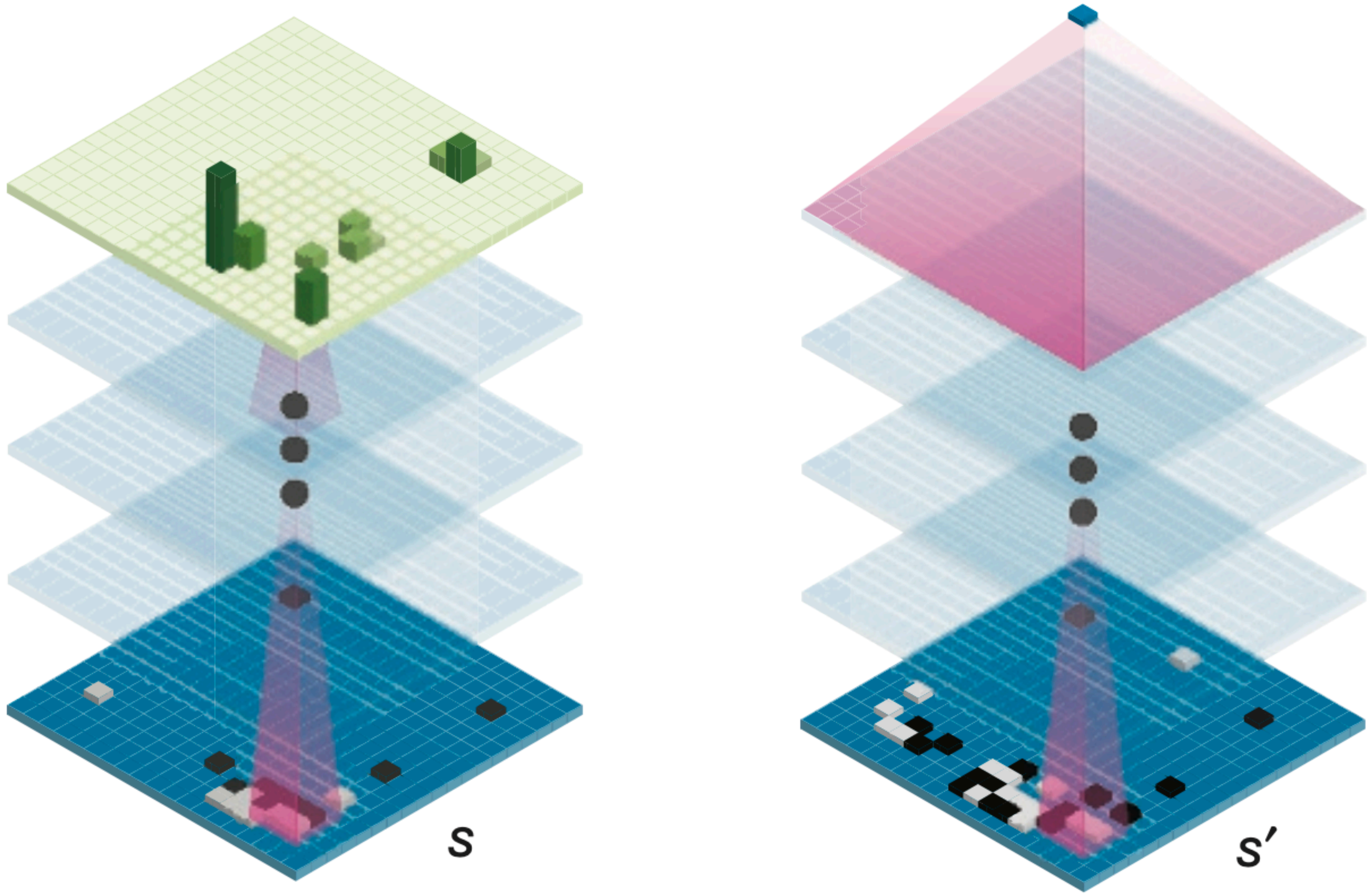
Self-play 30m games, and get 30m (s, z) pairs

Optimize least square via SGD again:

$$\beta_{t+1} = \beta_t - \eta \sum_{(s,z) \in B} (V_\beta(s) - z) \nabla_\beta V_\beta(s)$$

Summary so far

We have learned a policy $\hat{\pi}$ (BC+PG) and $\hat{V} \approx V^{\hat{\pi}}$



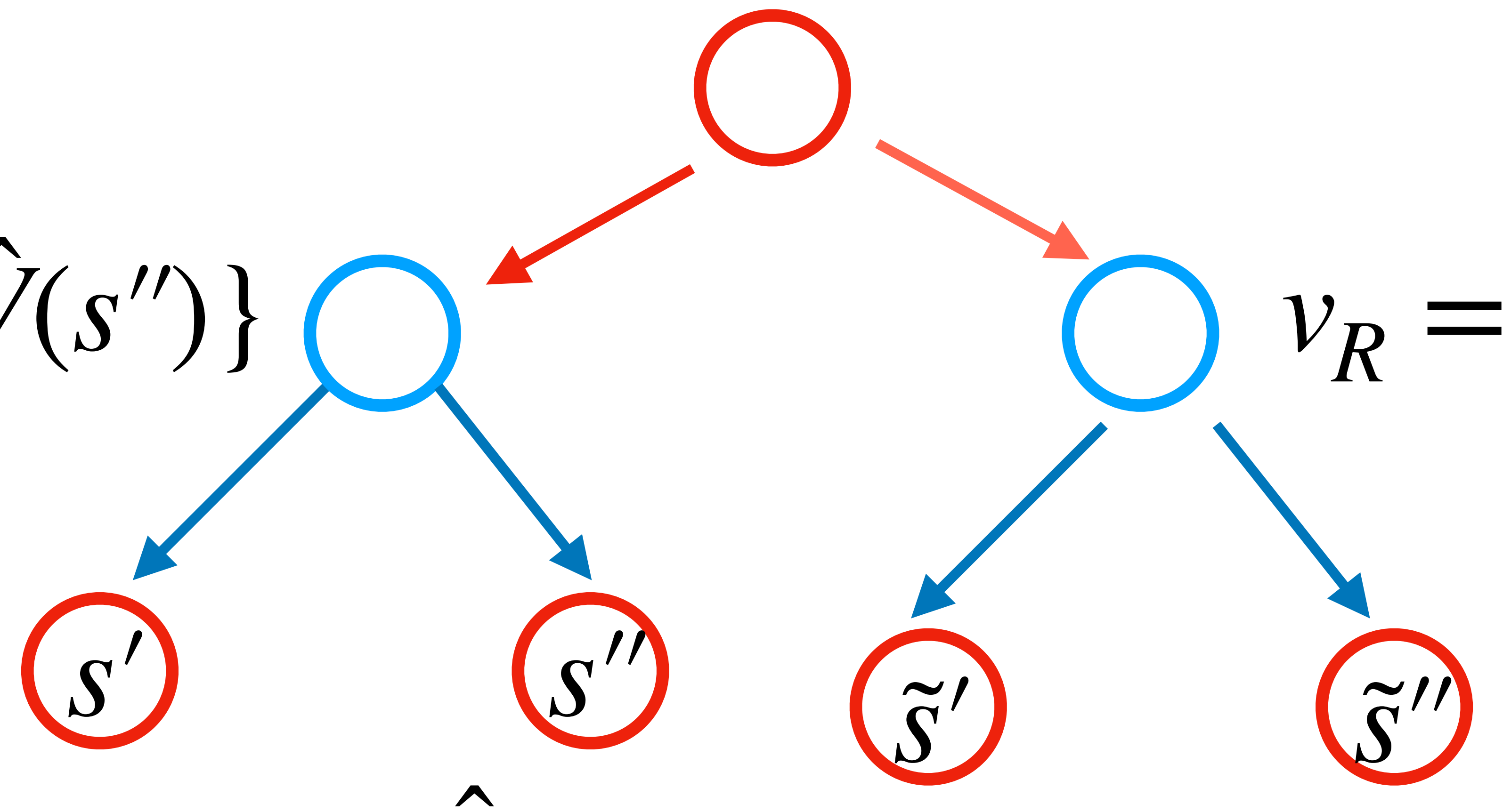
To make the program even more powerful, we combine them with a **Search Tree**

Combine with Tree Search (a naive version)

Imagine that we are at state s right now, let's simulate all possible moves into the future

$$v_{root} = \max\{v_L, v_R\}$$

$$v_L = \min\{\hat{V}(s'), \hat{V}(s'')\} \qquad v_R = \min\{\hat{V}(\tilde{s}'), \hat{V}(\tilde{s}'')\}$$



$\hat{V}(s')$ $\quad$ $\hat{V}(s'')$

$\hat{V}(s')$: win rate of red player starting at s'

AlphaGo uses Monte-Carlo Tree Search (MCTS)

Summary of the AlphaGo Program

1. Behavior cloning on 30m expert data samples

2. Classic Policy gradient on self-play games

3. Train a value network $\hat{V}$ to predict PG policy's outcome

4. Build search tree and use $\hat{V}$ to significantly reduce the search tree depth

Comment: might need step 4 in generative models if we really want them to dicover new things (dicover new drugs, prove open math problems, etc)