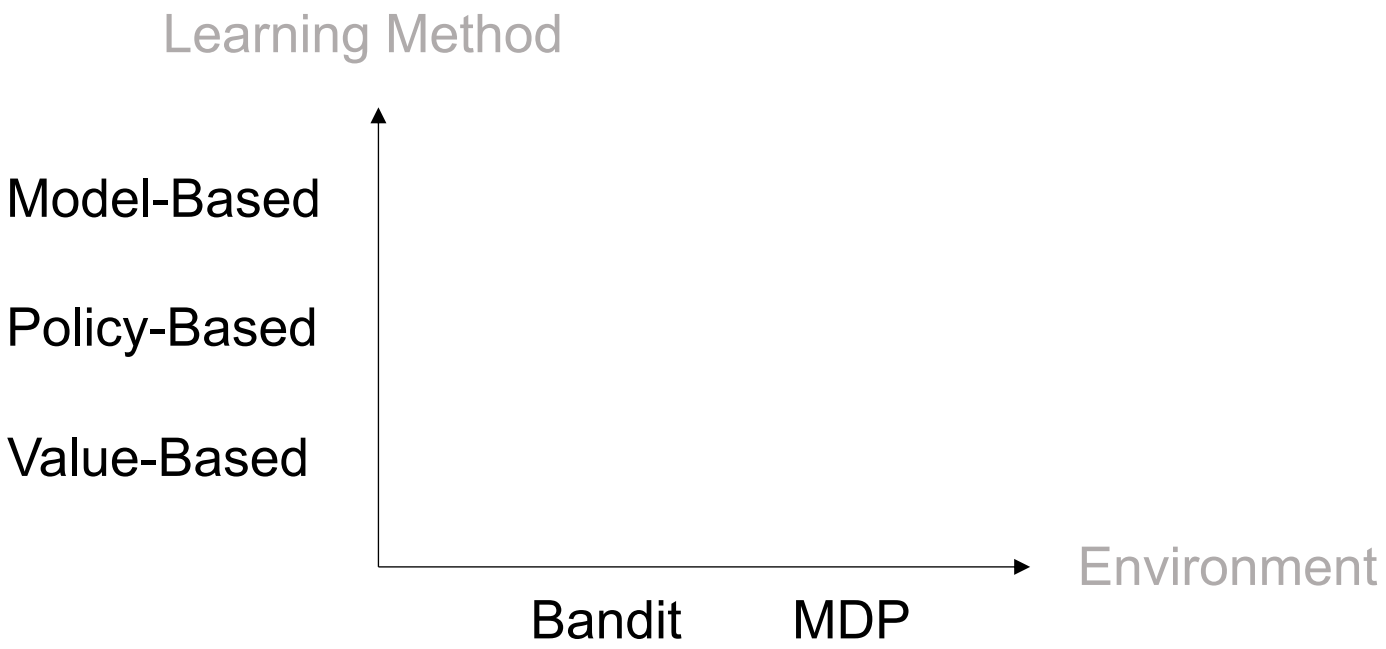
Reinforcement Learning: Selective Review

Chen-Yu Wei

Topics in This Course



Core Algorithms

Exploration in MDPs

Inference-Time Algorithms (MCTS)

Imitation Learning

Special Topics

RL vs SL

Learning sequential decision making

→ Learning sequential decision making **with bandit and delayed feedback**

SL: "what to do in each step" (full-information, immediate)

RL: "how you're doing overall" (bandit, delayed)

When Is IL (SL) Insufficient?

- The truly best policy is unknown / expert is imperfect
  - Atari game, Go
  - Faster matrix multiplication

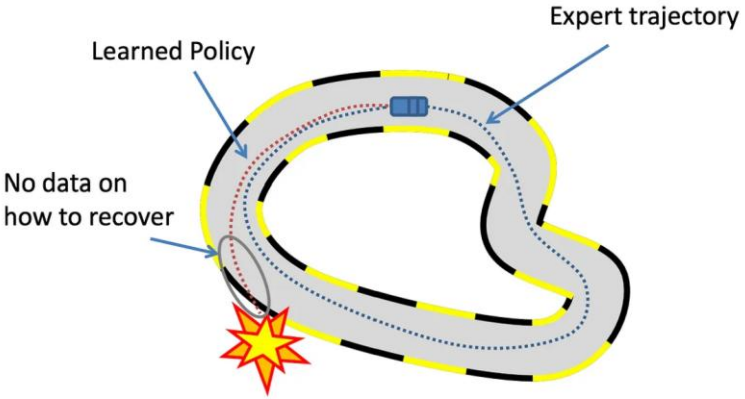
  ⇒ RL can **search** for better solutions

- RL signal may more faithfully reflect our real objective
  - RL from Human Feedback

  ⇒ RL can provide **alignment** to the real objective

- The expert data has limited coverage
  - Autonomous driving

  ⇒ RL can explore edge cases and **robustify** solutions



Core Challenges in RL

- Bandit feedback
  - Need exploration
- Delayed and aggregated feedback
  - Need credit assignment
- Unseen input / untried decisions (also a challenge in SL)
  - Need generalization

Other Challenges

- Reward design
- Simulation-to-reality gap
- Safety, robustness under attacks, … (similar challenges are also in SL)

Exploration

Ideas of Exploration

- Sample actions with randomization: **ϵ-greedy, Boltzmann exploration** (action space)
- Encourage sampling recently unchosen actions: **Baseline** (action space)
- Encourage sampling historically underchosen actions or states: **Exploration bonus, Thompson sampling** (state and/or action space)

Exploration Bonus

Modify the reward as

$$\tilde{R}(s,a) = R(s,a) + \text{bonus}(s) \qquad \text{or} \qquad \tilde{R}(s,a) = R(s,a) + \text{bonus}(s,a)$$

where $\text{bonus}(s)$ or $\text{bonus}(s,a)$ quantifies how **uncertain the learner is** about s or (s,a).

After this modifications, perform standard RL algorithm over $\tilde{R}$.

Exploration Bonus (HW5)

- Tabular Bonus

  $g(s,a)$: the group (s,a) belongs to

  $b(s,a) = c \cdot \frac{1}{\sqrt{N(g(s,a))}}$

- Random Network Distillation

  $f^\star(s,a)$: random target function, $f_\theta(s,a)$: prediction network

  $b(s,a) = c\|f_\theta(s,a) - f^\star(s,a)\|^2$

Bootstrapped DQN

Randomly initialize K instances of DQN $\theta_1, \dots, \theta_K$
(each θ_i has their own target network $\bar{\theta}_i$ and replay buffer $\mathcal{B}_i$).

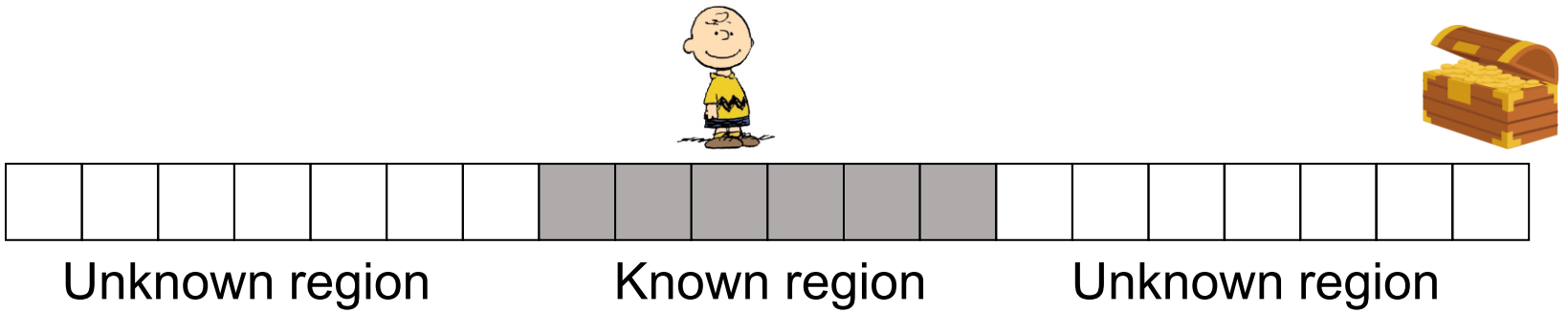
For each episode:

- Randomly sample $i \sim \text{Unif}\{1,2,\dots,K\}$
- Execute $\pi(s) = \max_a Q_{\theta_i}(s,a)$ in the whole episode.
- Randomly place the obtained (s,a,r,s') in **all/randomly selected** replay buffers.
- Update all DQN parameters.

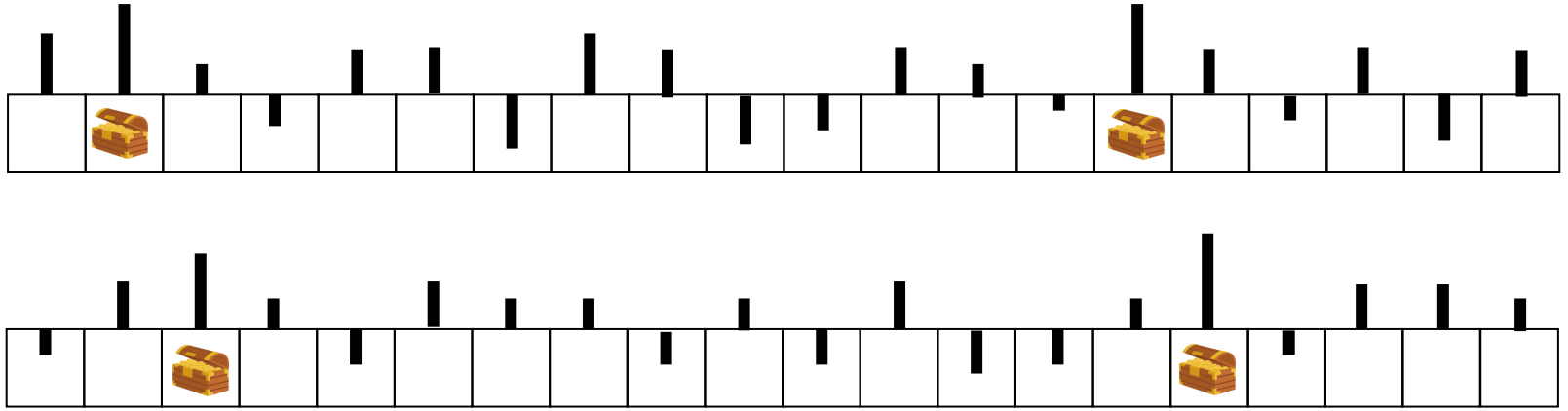
Something to Think About

What makes Exploration Bonus / Bootstrapped DQN able to explore state space?

-- what makes the known region expand?



Bootstrapped DQN



$\max_a Q(s,a)$ of randomly initialized Q networks

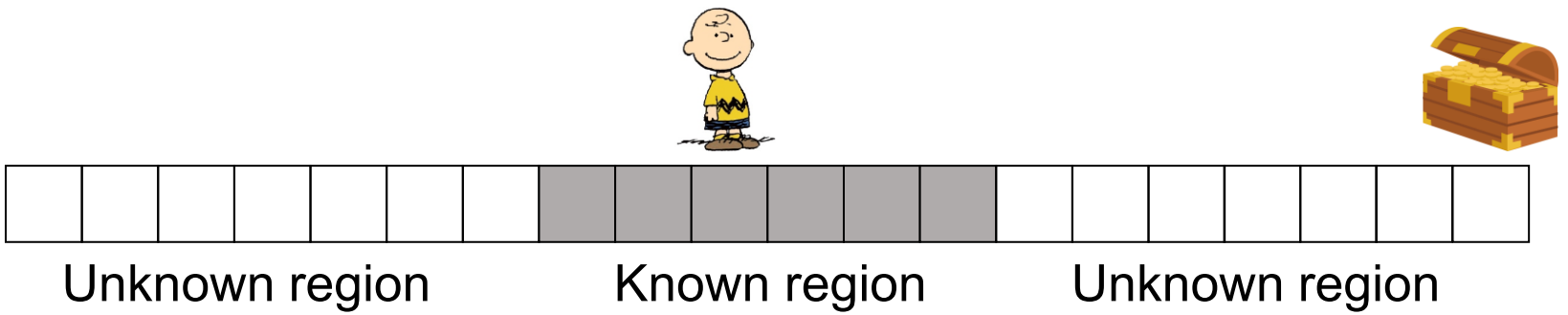
These networks essentially put some random fake reward in the grid

If a network puts a fake reward close to the boundary, it guides expanding the boundary

Something to Think About

What makes Exploration Bonus / Bootstrapped DQN able to explore state space?

-- what makes the known region expand?



For a given algorithm, can you tell whether it can explore this grid efficiently?

-- might be in the exam

Ensemble Methods

Ensemble Methods

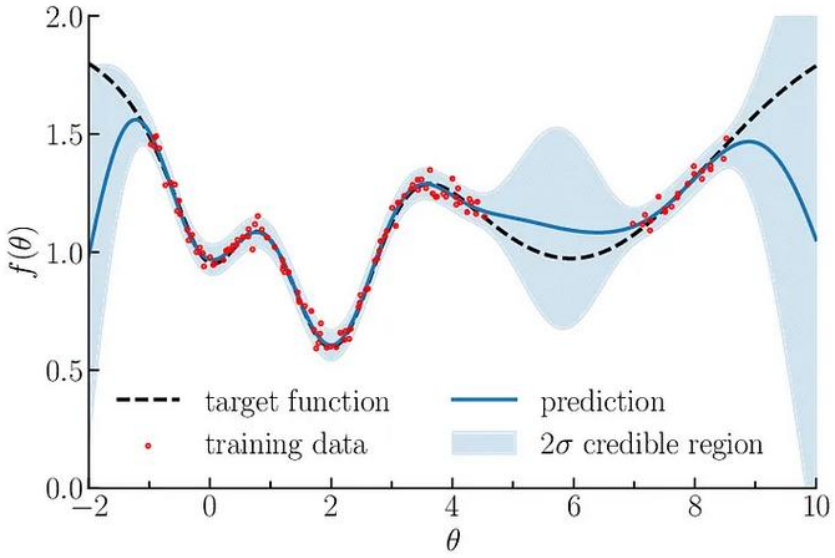
In the course, we have discussed several methods that involve training multiple networks simultaneously, and use them to jointly make decisions

- Bootstrapped DQN (Page 21 exploration)
- TD3 (two target networks) (Page 17 continuous action)
- Mitigating model error in model-based RL (Page 24 model-based)
- Estimating expert's action uncertainty (Page 17 imitation)

Ensemble Methods

- While applying to different scenarios, these ensemble methods are all related to the **uncertainty of the learner.**
  - The more the networks agree with each other, the less uncertainty.
- However, in different scenarios, the purposes are different
  - Being **optimistic** (i.e., having positive bias) when there is uncertainty
  - Trying to be **not optimistic** (i.e., avoiding positive bias) when there is uncertainty

E.g. using TD3 idea in Bootstrapped DQN might hurt the exploration capability of Bootstrapped DQN.



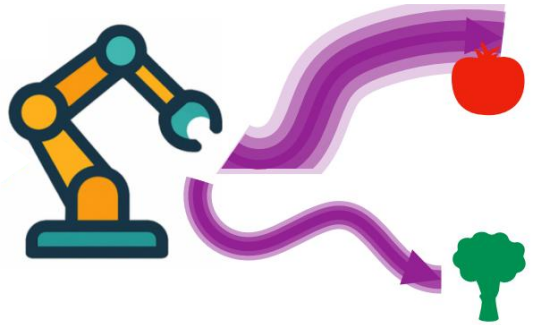
Optimistic or Not?

Application dependent: Does the learner want to keep exploring or not?

For complex problems, exploration is usually performed **only in a controlled region:** the learner should not always try to visit new states if the state space is very large.

In this case, encouragement to exploration (optimism) should be limited. After exploration, the agent should then focus on stably improving its policy in the known region (avoid optimism).

The advanced problems in the exam may test your ability to choose and apply appropriate RL tools depending on the scenario.

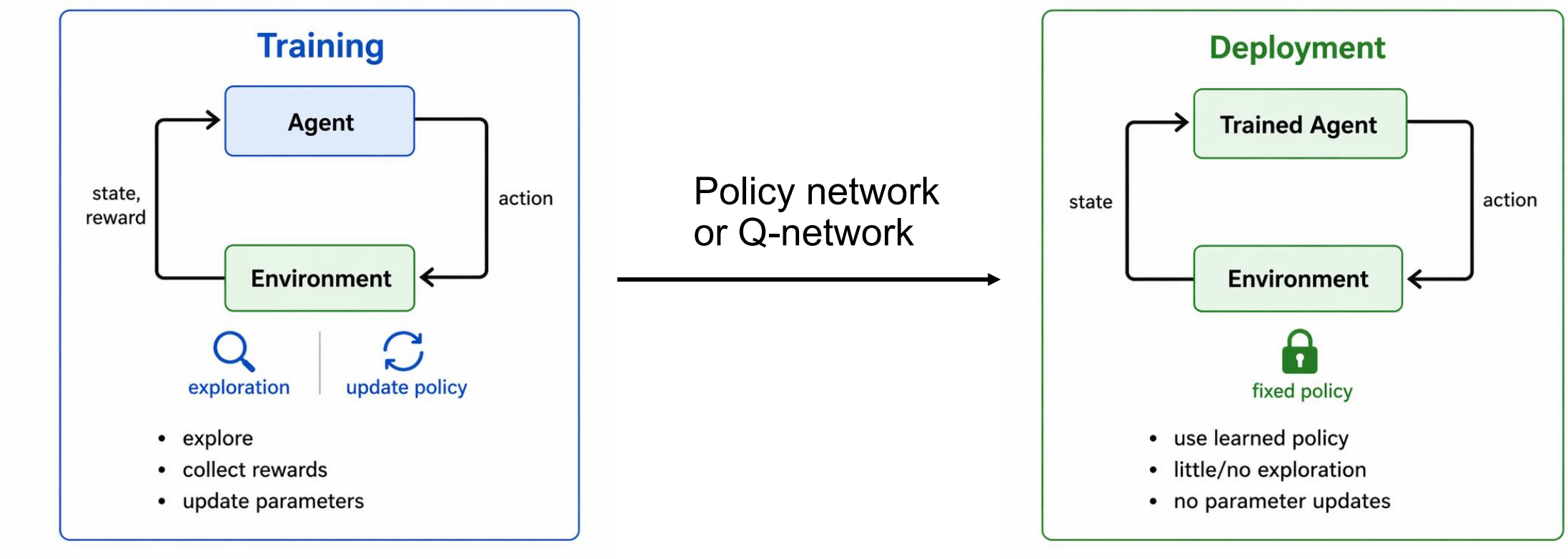


Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS)

MCTS is a somewhat more special algorithm we cover in the class: it is a **deployment-time** algorithm.

Monte Carlo Tree Search (MCTS)



Limitation of the policy network / Q-network output from the training phase:

- It only handles well on states seen in training time
- The single network has the burden to perform well on all possible states

Monte Carlo Tree Search (MCTS)

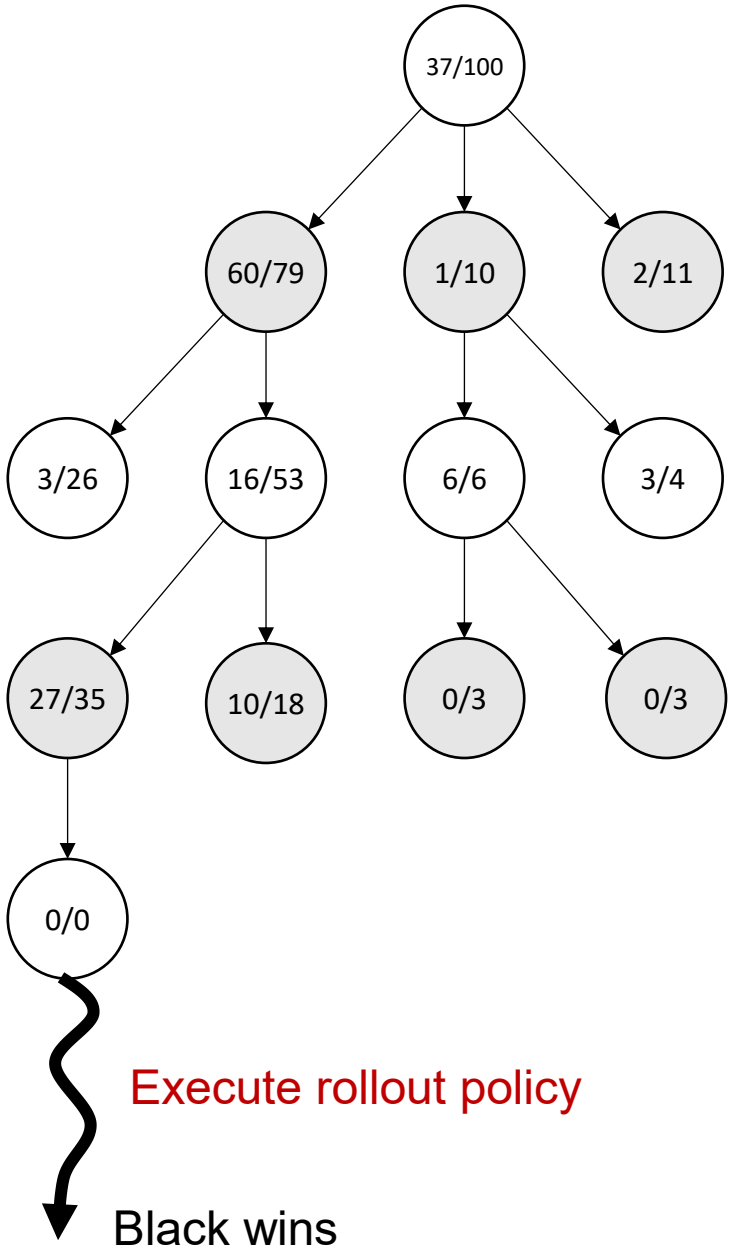
MCTS is a way to enhance the policy / Q-network **in the deployment time** that better handles unseen states.

Monte Carlo Tree Search (MCTS)

Tree policy

$$\frac{W(n)}{N(n)} + C \times \sqrt{\frac{\log N(\text{parent}(n))}{N(n)}}$$

Rollout policy: The policy network / Q-network from the training time, or heuristic policies.



Comparing Different Types of Algorithms

Value Iteration and Policy Iteration

Repeat until Q, V becomes stable:

$$Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V(s') \quad \text{for all } (s,a)$$

$$V(s) = \max_a Q(s,a) \quad \text{for all } s$$

$$\pi^\star(s) = \operatorname*{argmax}_a Q(s,a)$$

For $i = 1,\ 2, \ldots$

$$\forall s, \qquad \pi_i(s) \leftarrow \operatorname*{argmax}_a Q^{\pi_{i-1}}(s,a)$$

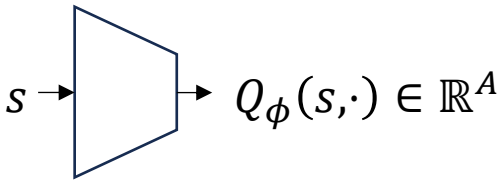
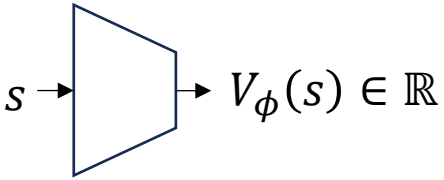
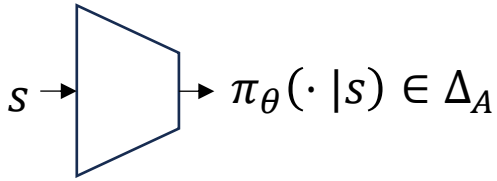
Generalized Policy Iteration

$Q \leftarrow \mathcal{T}^{\pi} Q$ means $Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s',a'} P(s'|s,a)\,\pi(a'|s')Q(s',a')$ for all s, a

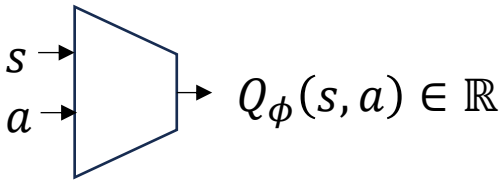
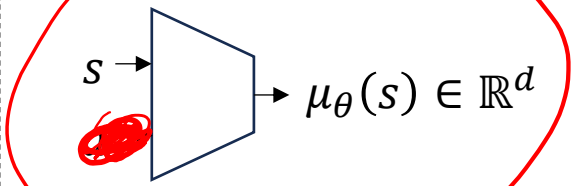
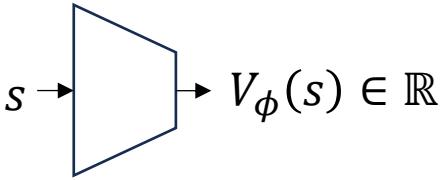
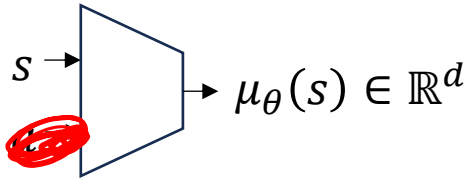
$N = \infty \Rightarrow$ Policy Iteration

$N = 1 \Rightarrow$ Value Iteration

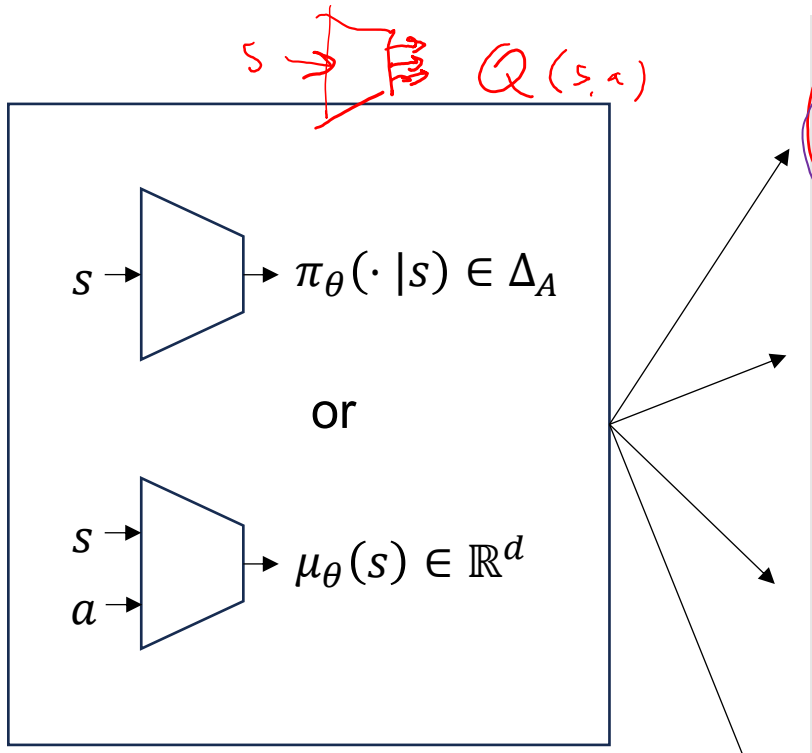
Comparison (Discrete Actions)

	Value $Q(s,a)$	Policy $\pi(a\|s)$
DQN		Induced by Q_ϕ
PPO	$\hat{Q}(s,a)$ constructed from 1. Pure real samples (MC estimator), or 2. Real samples + $V_\phi(s)$ (TD estimator)  (importance weighting needed)	

Comparison (Continuous Actions)

	Value $Q(s,a)$	Policy $\pi(a\|s)$
DDPG, TD3		
PPO	$\hat{Q}(s,a)$ constructed from 1. Pure real samples (MC estimator), or 2. Real samples + $V_\phi(s)$ (TD estimator)  (importance weighting needed)	

A Unified View



Policy (Actor)

Update policy to maximize

$\sum_a \pi_\theta(a|s_t) Q(s_t, a)$ or $Q(s_t, \mu_\theta(s_t))$

The $Q(s_t, a)$ used in each algorithm:

$$Q_\phi(s_t, a) - b(s_t)$$

DDPG, TD3, (DQN)

$$\frac{r_t + \gamma V_\phi(s_{t+1}) - b(s_t)}{\pi_{\theta_{\text{old}}}(a|s_t)} \mathbb{I}\{a_t = a\}$$

PPO with TD estimator

$$\frac{r_t + \gamma r_{t+1} + \gamma^2 V_\phi(s_{t+2}) - b(s_t)}{\pi_{\theta_{\text{old}}}(a|s_t)} \mathbb{I}\{a_t = a\}$$

$$\vdots$$

$$\frac{r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \text{end traj} - b(s_t)}{\pi_{\theta_{\text{old}}}(a|s_t)} \mathbb{I}\{a_t = a\}$$

PPO with MC estimator

Value (Critic)

Comparison: Importance Weight

Question: Why some algorithms use importance weight, some do not?

Recall that our goal is update actor (i.e., θ) to maximize

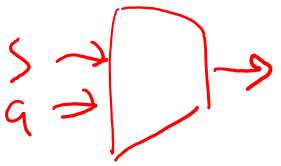
$$\sum_a \pi_\theta(a|s_t) Q(s_t, a) \qquad \text{or} \qquad Q(s_t, \mu_\theta(s_t))$$

Key: Does the critic tell us $Q(s_t, a)$ (i.e., long-term reward if choosing a) for some unchosen a?

$V(s)$

DDPG, TD3, DQN: Yes

PPO: No



Comparison: Replay Buffer

Question: Why some algorithms use replay buffer (i.e., reusing samples collected by previous policies), some do not?

Why replay buffer?

(s, a, r, s') contains the information of true $R(s, a)$, $P(s'|s, a)$

Even when the learner policy changes over time, $R(s, a)$, $P(s'|s, a)$ does not change.

Therefore, in principle, these old samples are always valid to use; we should try to exploit them whenever we can.

Replay Buffer for Q-type Critic (DDPG, TD3, DQN)

Using replay buffer to help training Q-type critic are straightforward (DDPG, TD3, DQN):

Ideal update: $$Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$$

NN update with replay buffer samples:

$$\phi \leftarrow \phi - \alpha \nabla_{\phi} \left(Q_{\phi}(s_i, a_i) - r_i - \gamma \max_{a'} Q_{\bar{\phi}}(s_i', a') \right)^2$$

(s_i, a_i, r_i, s_i')

$$\phi \leftarrow \phi - \alpha \nabla_{\phi} \left(Q_{\phi}(s_i, a_i) - r_i - \gamma Q_{\bar{\phi}}(s_i', \mu_{\theta}(s_i')) \right)^2$$

Replay Buffer for V-type Critic (PPO):

It is also possible to use replay buffer sample to create V-type critic (PPO):

$$Q(s_i, a) = \frac{r_i + \gamma V_\phi(s_i') - b(s_i)}{\pi_{\theta_{\text{old}}}(a|s_i)} \mathbb{I}\{a_i = a\}$$

$$\Rightarrow \sum_a \pi_\theta(a|s_i)\, Q(s_i, a) = \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{\text{old}}}(a_i|s_i)} \left(r_i + \gamma V_\phi(s_i') - b(s_i)\right)$$

$\pi_{\theta_{\text{old}}}(a_i|s_i)$ has to be the probability we use to collect the (s_i, a_i, r_i, s_i') data

Main limitation: the factor $\frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{\text{old}}}(a_i|s_i)}$ could potentially be very large if θ is already very different from $\pi_{\theta_{\text{old}}}$

Replay Buffer for V-type Critic (PPO):

Therefore, in PPO, we usually only reuse samples that are collected more recently when $\boldsymbol{\pi_{old}}$ **remains not very different from** $\boldsymbol{\pi_\theta}$

For $k = 1,\ 2, \ldots$

Collect N samples $(s_1, a_1, r_1, s_2, \ldots, s_N, a_N, r_N)$ using π_{θ_k}

Define for $i = 1, 2, \ldots, N$:

$$\hat{A}_{k,i} = \frac{\boldsymbol{G_i} - V_\phi(s_i)}{\pi_{\theta_k}(a|s_i)}$$

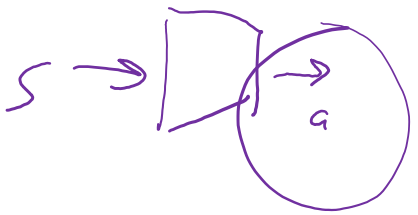
For $j = 1, 2, \ldots, M$:

Sample a minibatch $B \subset \{1,2, \ldots, N\}$

$$\theta \leftarrow \theta - \alpha \nabla_\theta \frac{1}{|B|} \sum_{i \in B} \left(\frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)} \hat{A}_{k,i} - \beta\, \mathrm{KL}\left(\pi_\theta(\cdot\,|s_i), \pi_{\theta_k}(\cdot\,|s_i)\right) \right)$$

Imitation Learning

Imitation Learning



$\pi(a|s) \propto \exp(Q(s,a))$
expert

We discussed direct imitation and inverse RL

- Direct imitation: Just train a policy network, with the label being the expert's action
- Inverse RL: Train a value network Q_ϕ based on expert demonstration, and then train a policy network π_θ based on Q_ϕ
- In Inverse RL we may also interleave the training of Q_ϕ and π_θ **(Generative Adversarial Imitation Learning)**

$\sum \pi_\theta(a|s) Q_\phi(s,a)$

Inverse RL vs. Actor-Critic Algorithms

One can view Inverse RL as a procedure similar to actor-critic algorithm. The difference is just in the critic

- Actor-critic RL: $Q(s,a)$ represents the value of the learner's own policy or the optimal policy in the real world

  Training goal: Make $Q_\phi(s,a)$ similar to $Q^{\pi_\theta}(s,a)$ or $Q^\star(s,a)$

$\pi(a|s) \propto \exp(Q(s,a))$

- Inverse RL: $Q(s,a)$ represents the **underlying value of the expert**

  Training goal: Make $Q_\phi(s,a)$ explain the expert's choices through e.g.

$$\pi^\star(a|s) \propto \exp\left(Q_\phi(s,a)\right)$$