Lecture 12: Typical Deep RL Algorithms

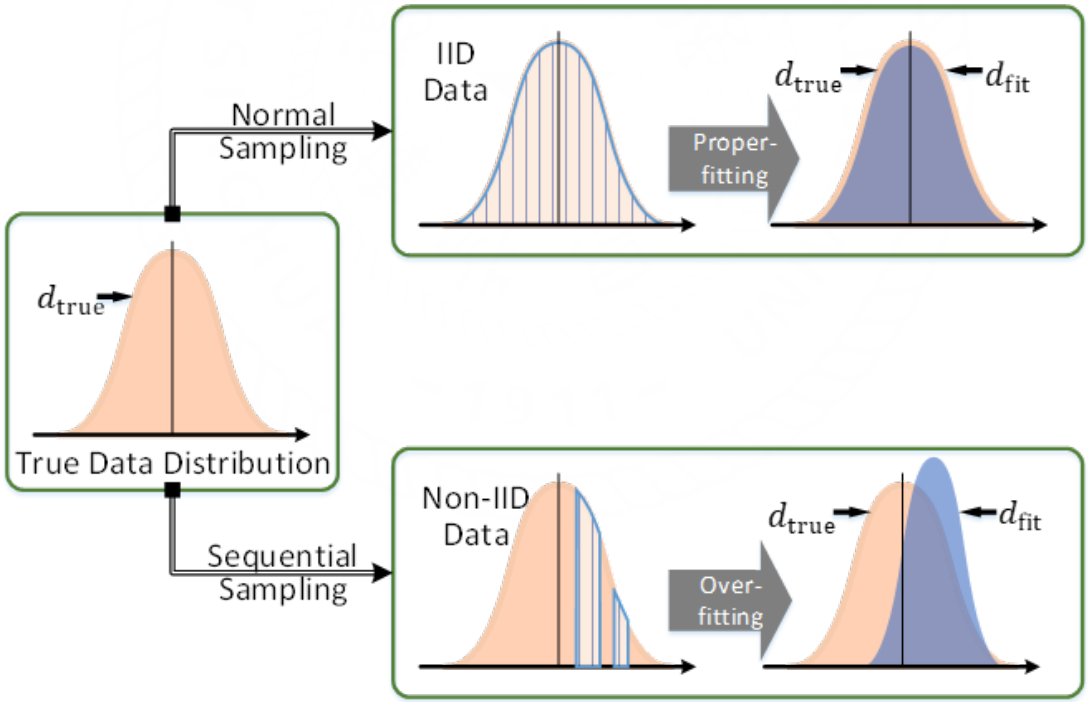
Dr. Wen Fuxi

Challenges

- Non-iid data breaks the guarantee of NN convergence
- Mutable update targets lead to diverged policy/value
- Inhomogeneous overestimation negatively affects policy
- Low efficiency in sampling the whole state-action space

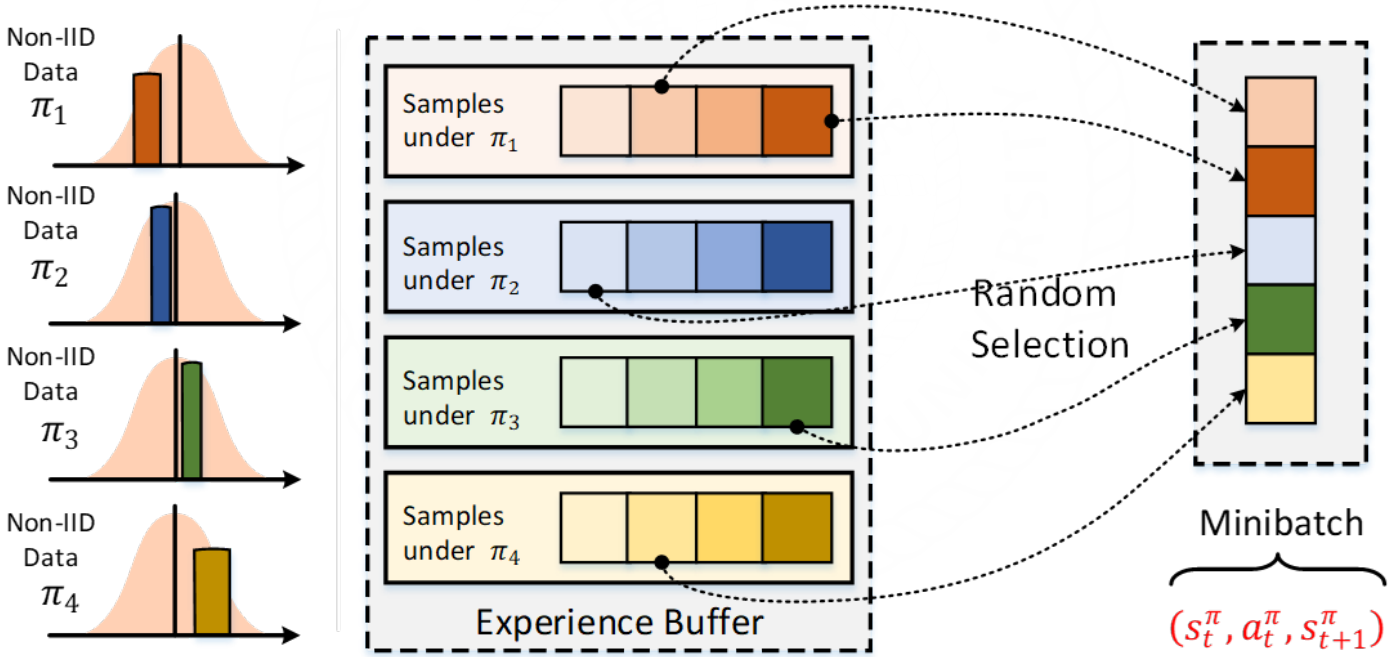
Challenge A: Non-iid sequential data

- In deep learning, samples need to be independent and identically distributed (iid)
- Samples are explored sequentially in reinforcement learning
- Overfitting: generalization ability is negatively affected
- Training distribution changes with diverse sequential data



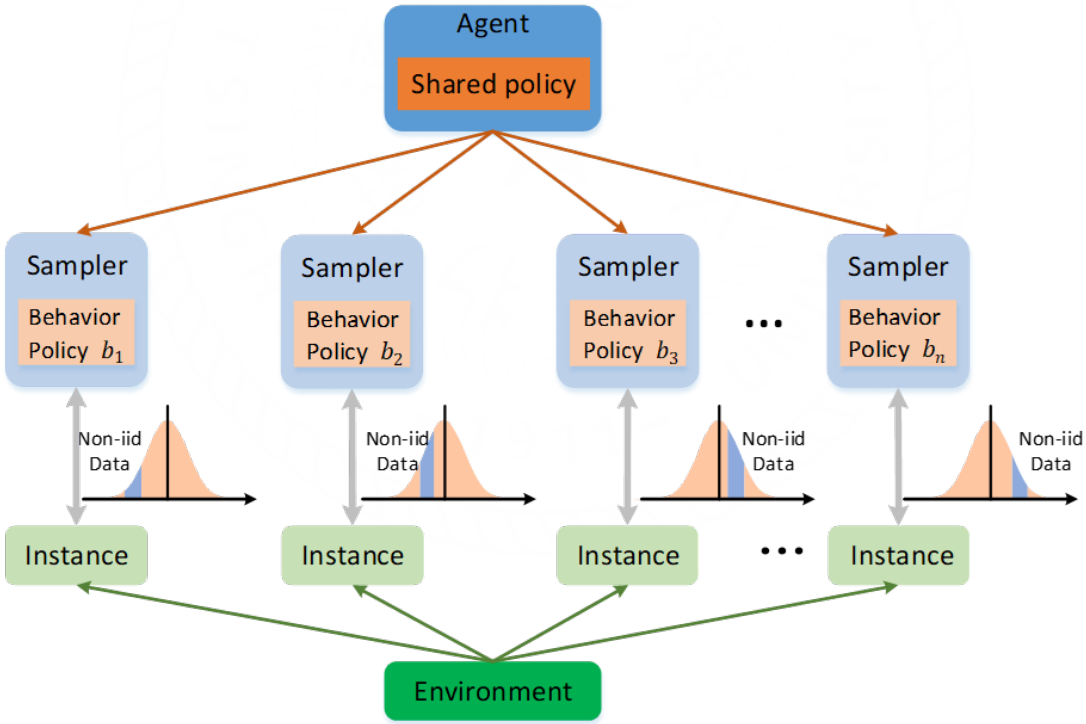
Trick 1: Experience Replay (ExR)

- Only suitable for off-policy RL
- Store samples in a replay buffer; reuse randomly sampled mini-batch to update
- Average training distribution over previous experiences



Trick 2: Parallel Exploration (PEx)

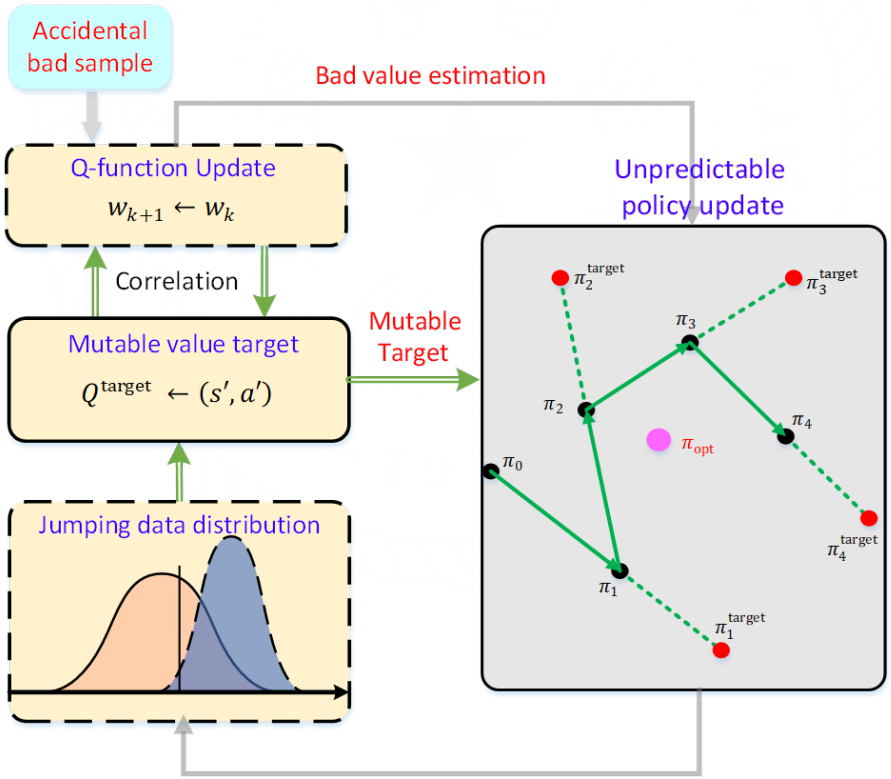
- Suitable for both on-policy RL & off-policy RL
- Collect data from different parts of the environment to alleviate the sample correlation and average the training distribution
- Off-policy can maximize the diversity of training data



Challenge B: Easy Divergence

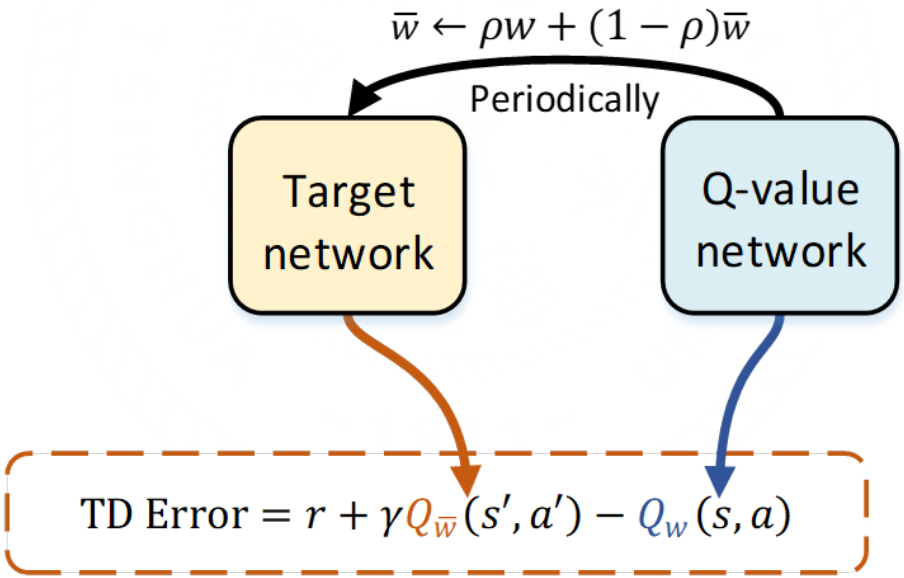
Oscillating target values and unstable policy updates lead to oscillation of the learning process and non-convergence of the policy.

- Bad value estimation provides wrong direction of policy update
- Improper policy improvement deteriorates the critic quality



Trick 3: Separated Target Network (STN)

- The target value is obtained from a separate target network rather than from the online value network
- The target is fixed during most updates of the online value network
- Reduce the tight correlation between Q-value and its target



Tricks 4-6

Trick 4: Delayed policy updates (DPU)

- Learn the policy for a more precise value network
- The policy update waits until the value approximation error becomes small enough

Trick 5: Constrained policy updates (CPU)

- Stabilize policy update by constraining the policy change

$$\|\pi_{k+1} - \pi_k\| \leq \delta_\pi$$

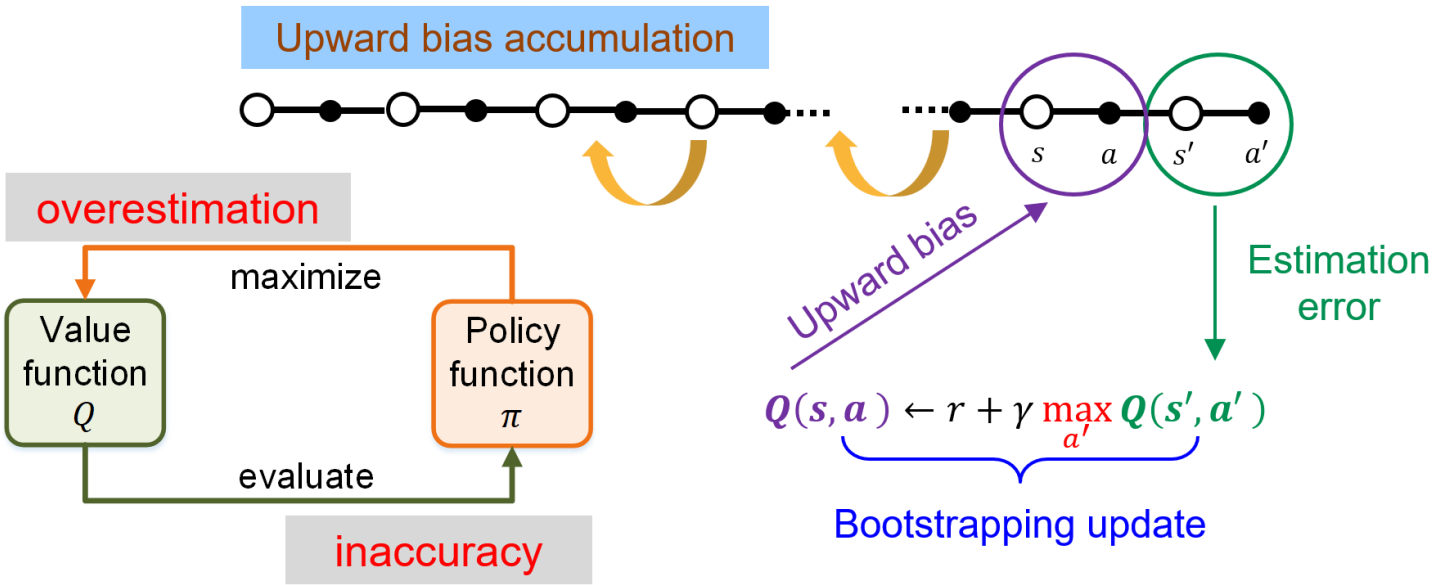
Trick 6: Clipped actor criterion (CAC)

- Stabilize policy update by constraining actor criterion change

$$0 \leq J_{\text{Actor}}\ (\pi_{k+1}) - J_{\text{Actor}}\ (\pi_k) \leq \delta_J$$

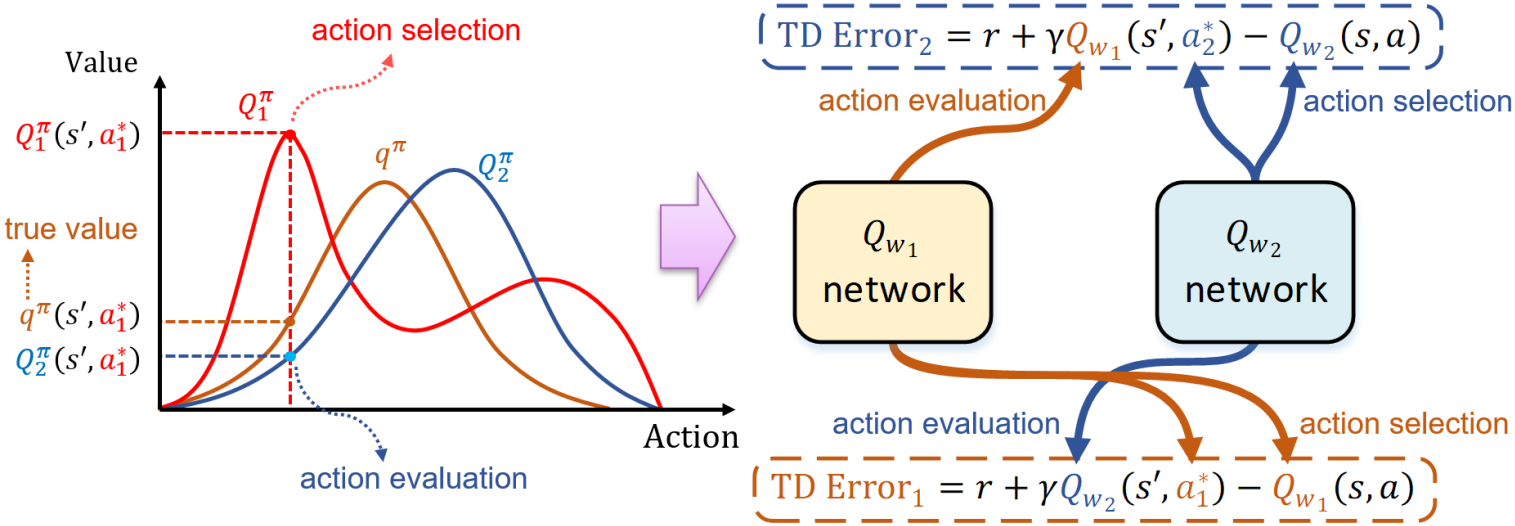
Challenge C: Overestimation

- When the maximizer acts on a value function, any estimation error will cause an upward bias
- Upward bias will accumulate through value network update, i.e., bootstrapping update



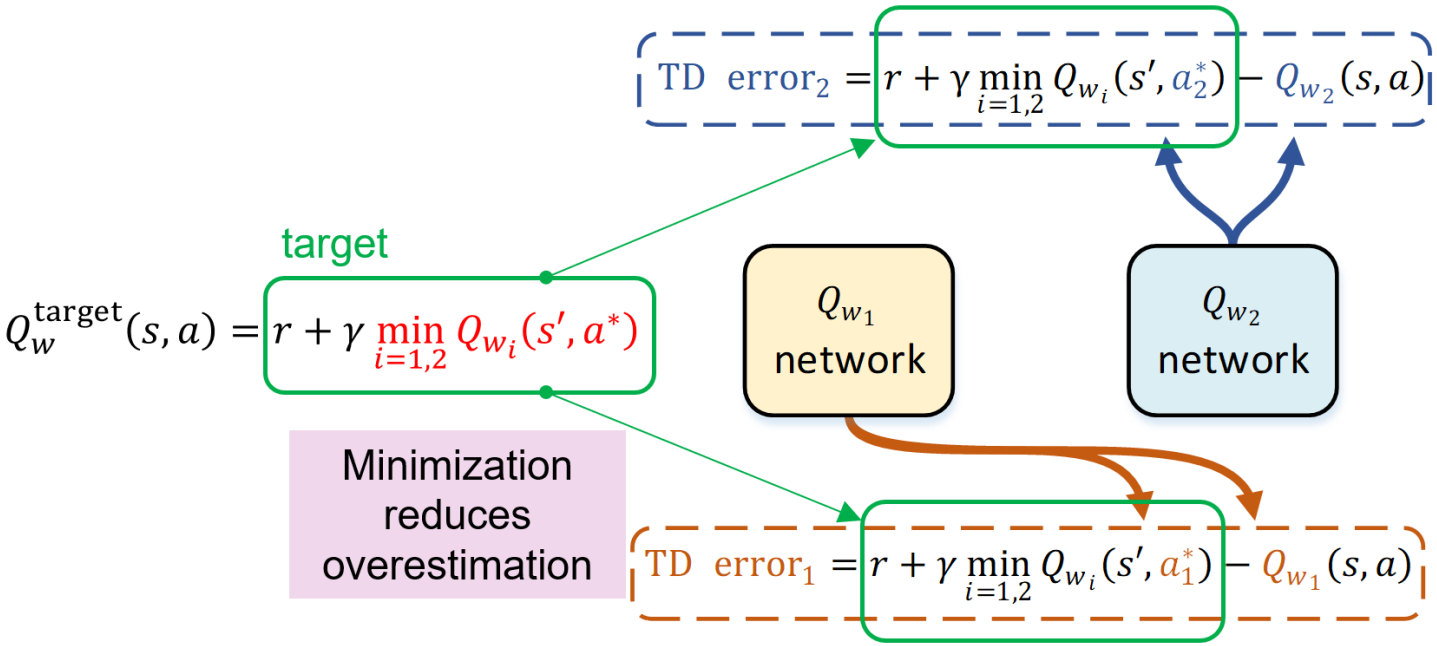
Trick 7: Double Q-functions (DQF)

- Decouple maximizer into action selection and action evaluation
- Use two independent value functions: action is selected by one value function, and then evaluated by the other value function



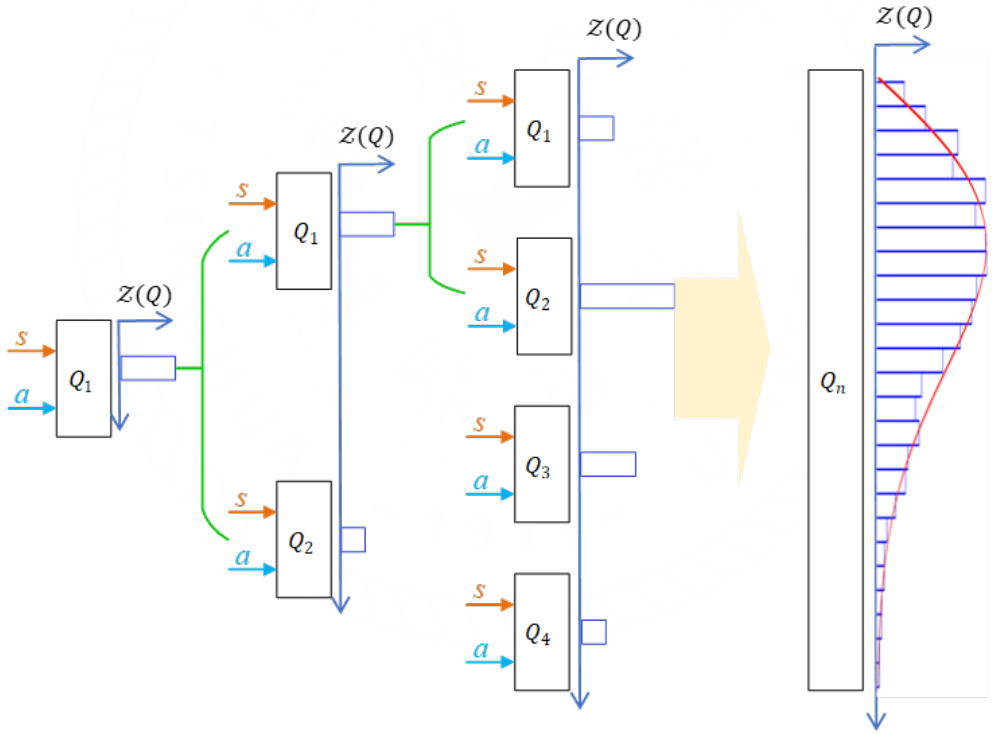
Trick 8: Bounded Double Q-functions (BDQ)

- Any two value functions are not exactly independent since each creates the target for the other value function
- Obtain the target from the minimum of two value functions



Trick 9: Distributional Return Function (DRF)

- More independent action-value functions can better alleviate overestimation
- Learn an infinite number of action-value functions to maximize the estimation accuracy

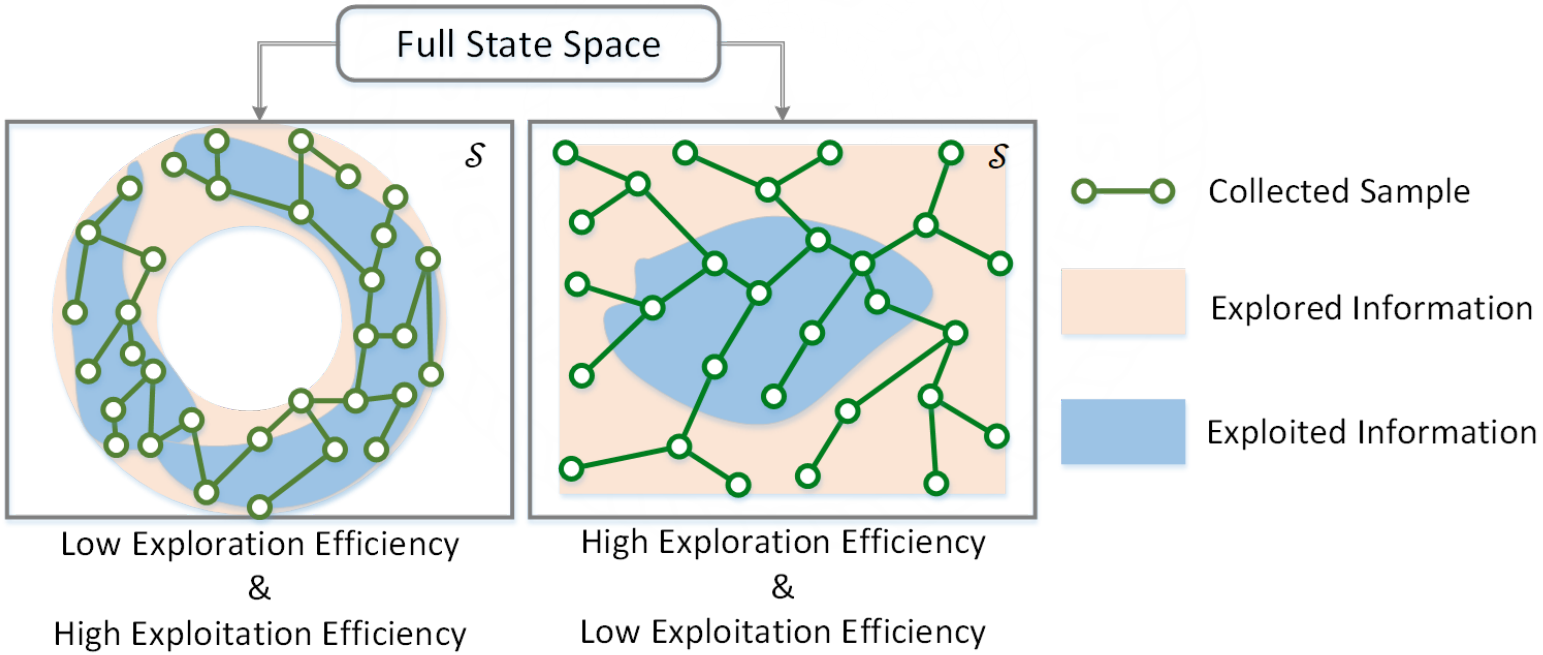


State-action return $Z(s, a) \sim Z_w(\cdot \mid s, a)$ Action-value return distribution

Challenge D: Sample Inefficiency

- High-dimensional and continuous state-action space in DRL
- Sample efficiency: how many samples are required when reaching a specific policy performance

  Exploration efficiency & Exploitation efficiency



Trick 10 and 11

Trick 10: Entropy Regularization (EnR)

- Policy entropy: A measure of policy randomness
- Augment overall RL objective function with policy entropy

$$J(\theta) = \mathbb{E}_{\pi_\theta} \left\{ v^{\pi_\theta}(s) + \alpha \mathcal{H}\left(\pi_\theta(\cdot \mid s)\right) \right\}$$

Trick 11: Soft Value Function (SVF)

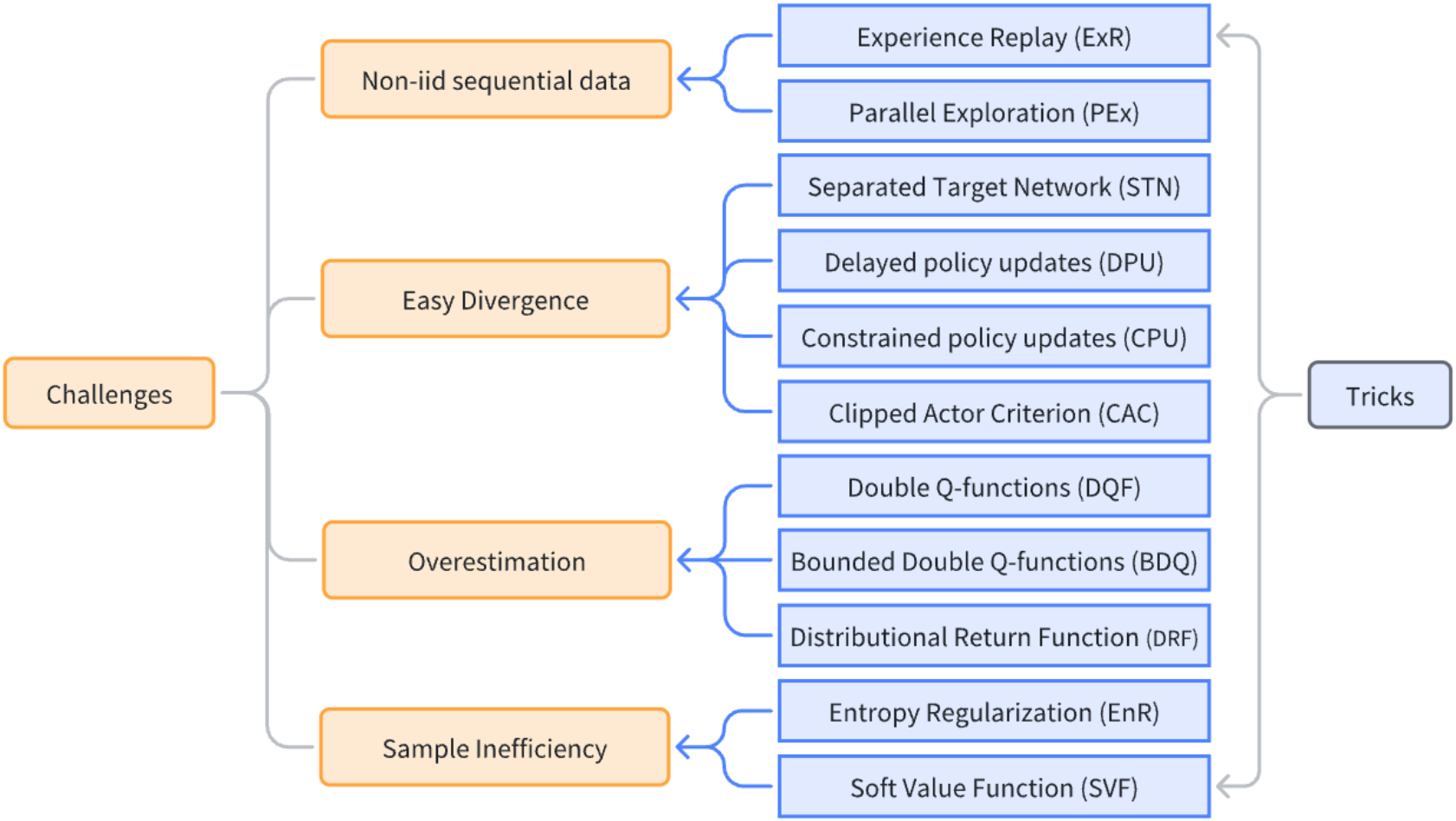
- Augment each reward signal with policy entropy

$$r_{\text{aug}}\left(s, a, s'\right) = r\left(s, a, s'\right) + \alpha \mathcal{H}(\pi(\cdot \mid s))$$

- Maximum entropy RL framework

$$v^{\pi}(s) = \mathbb{E}_{\pi} \left\{ \sum_{i=0}^{\infty} \gamma^i \left(r_{t+i} + \alpha \mathcal{H}\left(\pi\left(\cdot \mid s_{t+i}\right)\right) \right) \mid s_t = s \right\}$$

Challenges and Tricks of Deep RL



Deep RL Algorithms

⋆ Formally proposed for the first time | • Inherited tricks from previous DRLs

Algorithm	ExR	PEx	STN	DPU	CPU	CAC	DQF	BDQ	DRF	EnR	SVF	π
DQN	★		★									off
Dueling DQN	•		•									off
DDQN	•		•				★					off
TRPO					★							on
PPO						★						on
A3C		★								★		on
DDPG	•		•				•					off
TD3	•		•	•			•	★				off
SAC	•		•				•	•			★	off
DSAC	•	•	•	•			•		★		•	off

Notations:

Experience Replay (ExR) Parallel Exploration (PEx) Separated Target Network (STN)
Delayed policy updates (DPU) Constrained policy updates (CPU) Clipped actor criterion (CAC)
Double Q-functions (DQF) Bounded Double Q-functions (BDQ) Distributional Return Function (DRF)
Entropy Regularization (EnR) Soft Value Function (SVF)

Reference:

Deep Q-Network (DQN)

Suitable for continuous state, discrete action

2 Q-networks, only one needs Backpropagation (BP)

- Trick 1: experience replay
- Trick 3: separated target network

$$J(w) = \mathbb{E}_{s,a,s' \sim \mathcal{D}_{\text{Replay}}} \left\{ \left(r + \gamma \max_{a'} Q_{\bar{w}}\left(s', a'\right) - Q_w(s, a) \right)^2 \right\}$$

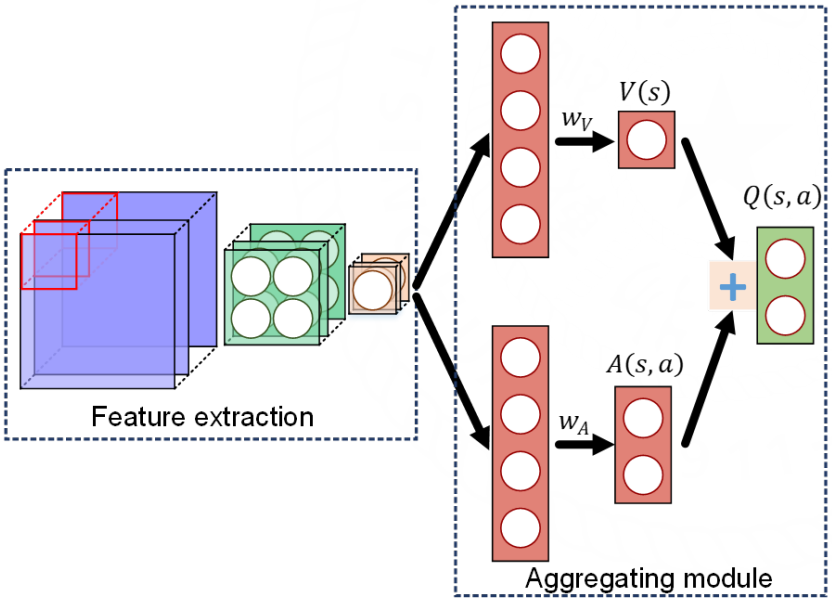
Q-network weight Replay buffer Target network weight

* Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature, 2015

Dueling DQN

2 Q-networks (only one needs BP)

- Trick 1: experience replay
- Trick 3: separated target network
- Dueling Q-network: Split Q-value into V-value and A-value



$$Q^{\pi}(s,a) = V^{\pi}(s) + A^{\pi}(s,a)$$

* Wang Z, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning. ICML 2016, New York, USA.

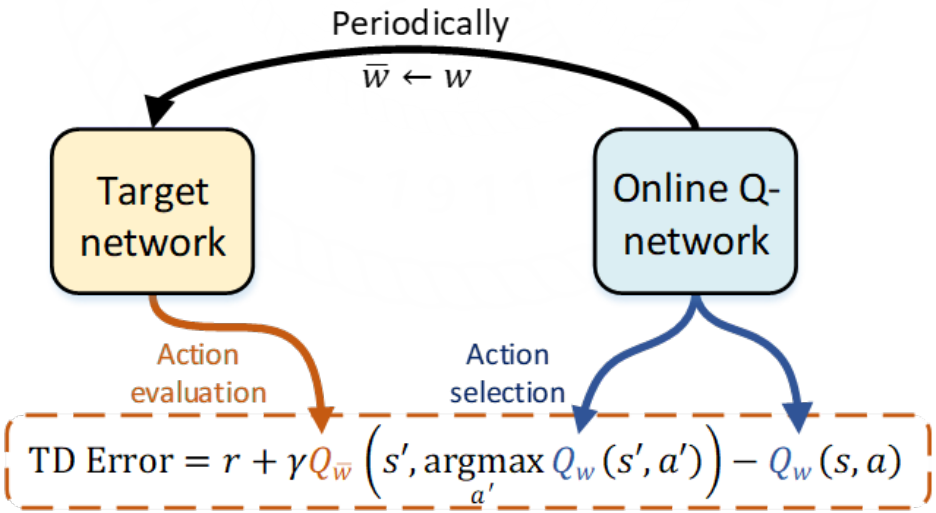
Double DQN (DDQN)

2 Q-networks (only 1 needs BP)

- Trick 1: experience replay
- Trick 3: separated target network
- Trick 7: double Q-functions

The target network in DQN provides a natural candidate for action evaluation

$$J(w) = \mathbb{E}_{s,a,s' \sim \mathcal{D}_{\text{Replay}}} \left\{ \left(r + \gamma Q_{\bar{w}} \left(s', a^* \left(s' \right) \right) - Q_w(s,a) \right)^2 \right\}$$



Trust Region Policy Optimization (TRPO)

1 V-network, 1 stochastic policy network (both need BP)

Minorize-maximization (MM) optimization

- Trick 5: constrained policy updates

Convert the penalty term into a trust region constraint

$$\theta \leftarrow \arg\max_{\theta} \mathbb{E}_{s\sim d_{\pi_{\text{old}}},a\sim\pi_{\text{old}}} \left\{ \frac{\pi_\theta(a \mid s)}{\pi_{\text{old}}(a \mid s)} A^{\pi_{\text{old}}}(s,a) \right\}$$

$$\text{s.t. } D_{\text{KL}}(\pi_{\text{old}}, \pi_\theta) \le \delta_\pi$$

Use average KL divergence to replace max KL divergence

$$\bar{D}_{\text{KL}} = \mathbb{E}_{s\sim d_{\pi_{\text{old}}}} \{D_{\text{KL}}(\pi_{\text{old}}, \pi_\theta)\} \le \delta_\pi$$

Computationally expensive!

$$(\theta - \theta_{\text{old}})^{\text{T}} H (\theta - \theta_{\text{old}}) \le \delta_\pi$$

$$H = \mathbb{E}\left\{ \frac{\partial^2 \bar{D}_{\text{KL}}}{\partial \theta^2} \right\}$$

* Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization. ICML 2015, Lille, France.

Proximal Policy Optimization (PPO)

- 1 V-network, 1 stochastic policy network (both need BP)
- Pure first-order optimization that holds the monotonic improvement property
- Trick 6: Clipped Actor Criterion

$$\theta \leftarrow \arg\max_{\theta} \mathbb{E}_{s\sim d_{\pi_{\text{old}}},a\sim\pi_{\text{old}}} \left\{\min\left(\rho_{t:t} A^{\pi_{\text{old}}}(s,a), \rho_{\text{clip}}\ A^{\pi_{\text{old}}}(s,a)\right)\right\}$$

$$\rho_{\text{clip}} \overset{\text{def}}{=} \text{clip}\left(\rho_{t:t}, 1-\epsilon, 1+\epsilon\right)$$

Advantage function

$$A^{\pi}(s,a) = r + \gamma V^{\pi}\left(s'\right) - V^{\pi}(s)$$

- The minimum of clipped and unclipped criteria is a lower bound of original objective function (i.e., surrogate function)

* Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms. arXiv preprint, 2017.

Deep Deterministic Policy Gradient (DDPG)

Suitable for continuous state & action space

2 Q-networks (1 BP), 2 policy networks (1 BP)

- Trick 1: Experience Replay
- Trick 3: Separated Target Network
- Trick 7: Double Q-functions

Action evaluation and action selection are implemented by different networks

$$J_{\text{Critic}}(w) = \mathbb{E}_{s,a,s' \sim \mathcal{D}_{\text{Replay}}} \left\{ \left(r + \gamma Q_{\bar{w}} \left(s', \pi_{\bar{\theta}}(s') \right) - Q_w(s,a) \right)^2 \right\}$$

Learn a deterministic policy $\pi_\theta(s)$ which maximizes $Q_w(s,a)$

$$J_{\text{Actor}}(\theta) = \mathbb{E}_{s \sim \mathcal{D}_{\text{Replay}}} \left\{ Q_w \left(s, \pi_\theta(s) \right) \right\}$$

Twin Delayed DDPG (TD3)

4 Q-networks (2 BP), 2 policy networks (1 BP)

- Trick 1: experience replay
- Trick 3: separated target network
- Trick 7: double Q-functions
- Trick 8: bounded double Q-functions

$$Q_{\bar{w}}^{\text{target}}(s,a) = r + \gamma \min_{i=1,2} Q_{\bar{w}_i}\left(s', \pi_{\bar{\theta}}(s')\right)$$

$$J_{\text{Critic}}(w_1) = \mathbb{E}_{s,a,s' \sim \mathcal{D}_{\text{Replay}}}\left\{\left(Q_{\bar{w}}^{\text{target}}(s,a) - Q_{w_1}(s,a)\right)^2\right\}$$

$$J_{\text{Critic}}(w_2) = \mathbb{E}_{s,a,s' \sim \mathcal{D}_{\text{Replay}}}\left\{\left(Q_{\bar{w}}^{\text{target}}(s,a) - Q_{w_2}(s,a)\right)^2\right\}$$

- Trick 4: delayed policy updates. TD3 delays the policy update, i.e., the weights of policy network change less frequently

* Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods. ICML 2018, Stockholm, Sweden.

Soft Actor-Critic (SAC)

4 soft Q-networks (2 BP), 1 stochastic policy networks (1 BP)

- Trick 1: Experience Replay
- Trick 3: Separated Target Network
  - SAC concurrently learns a stochastic policy and two Q-networks. Both of two Q-networks have related target networks, while the policy has no target
- Trick 7: Double Q-functions
- Trick 8: Bounded Double Q-functions
- Trick 11: Soft Value Function
  - SAC uses maximum entropy RL framework, where each reward signal is augmented with a policy entropy term
  - SAC trains a stochastic policy to strike a balance between expected return and accumulated policy entropy

Soft Actor-Critic (SAC) – Soft Q-function

- Use common target to update two Q-networks (like TD3)

$$J_{\text{Critic}}\ (w_i) = \mathbb{E}_{s,a,s' \sim \mathcal{D}_{\text{replay}}} \left\{ \left(Q_{\bar{w}}^{\text{target}}\ (s,a) - Q_{w_i}(s,a) \right)^2 \right\}, \forall i = 1,2$$

- Common target is calculated through soft Q-function

$$Q_{\bar{w}}^{\text{target}}\ (s,a) = r + \gamma \mathbb{E}_{a' \sim \pi_\theta} \left\{ \min_{i=1,2} Q_{\bar{w}_i}\left(s', a'\right) - \alpha \log \pi_\theta \left(a' \mid s'\right) \right\}$$

- Either high return or high policy entropy would lead to high soft Q-function, thus the corresponding policy has the motivation to explore the state space with larger uncertainties

Soft Actor-Critic (SAC) – Stochastic policy

- Maximize soft V-function

$$J_{\text{Actor}}(\theta) = \mathbb{E}_{S\sim\mathcal{D}_{\text{replay}},a\sim\pi_\theta}\left\{Q^{\pi_\theta}(s,a) - \alpha \log \pi_\theta(a \mid s)\right\}$$

temperature parameter

- Reparameterization trick removes the pain point of the action distribution depending on policy weights and leads to a lower variance estimate
- SAC automatically update the temperature parameter, where the policy entropy is enforced to be no less than a lower bound

* Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. ICML 2018, Stockholm, Sweden.

Distributional SAC (DSAC)

2 distributional Q-networks (1 BP), 2 stochastic policy networks (1 BP)

DSAC integrates distributional return function into maximum entropy RL framework

- Trick 1: Experience Replay
- Trick 2: Parallel Exploration
- Trick 3: Separated Target Network
- Trick 4: Delayed Policy Updates
- Trick 7: Double Q-functions
- Trick 9: Distributional Return Function
  - Instead of learning two independent Q-networks, DSAC learns a single distributional return function for more accurate action-value estimation
- Trick 11: Soft Value Function

* Duan J, Guan Y, Li S E*, et al. Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. IEEE Trans Neural Networks and Learning Systems, 2021