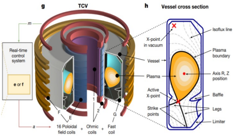
Introduction to Reinforcement Learning

Ke Wei
Fudan University

Success of RL

(Nature, 2016)

(Nature, 2019)

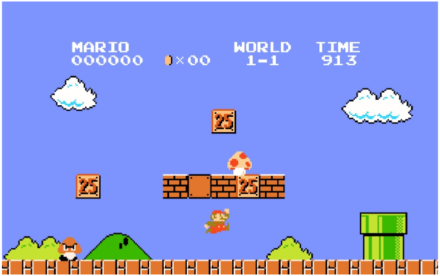


(Nature, 2022)

(ChatGPT, 2023)

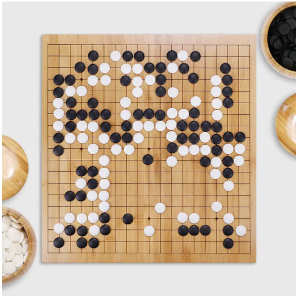
- RL has also been used to solve computationally difficult problems such as traveling salesman problem and plays an important role in “AI for Science”.

Illustration: Super Mario



Super Mario makes a decision, then receives a reward and transfers to the next state;
Goal: high long term cumulative reward by making right decisions.

Challenges in RL

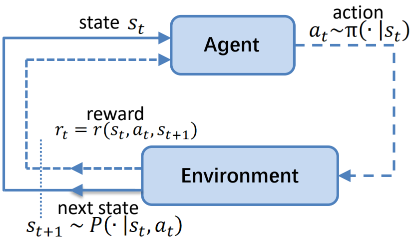


RL is a sequential decision problem and is essentially about efficient search (dynamic programming, control, game theory).

- High dimension (large state/action spaces)
- Highly nonconvex (distribution optimization, parameterization)
- Computational efficiency vs Reliability
- Plenty of scenarios $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$
- $\cdots\cdots$

MDP and Basic Setup

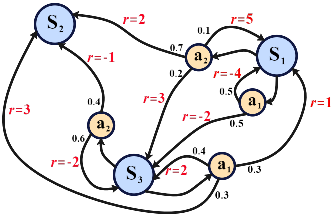
Markov Decision Process (MDP)



Markov chain augmented with decision and reward: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$

- $\mathcal{S}$: state space (状态空间)
- $P(\cdot|s, a)$: state transition model (状态转移模型)
- $\gamma \in [0, 1]$: discount factor (折扣因子)
- $\mathcal{A}$: action space (动作空间)
- $r(s, a, s')$: immediate reward (即时奖励)
- $\pi(\cdot|s) : \mathcal{A} \to \Delta$ (策略)

Illustrative Example



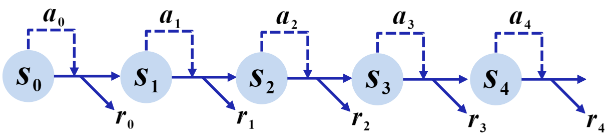
- three states: $\mathcal{S} = \{s_1, s_1, s_3\}$
- two actions: $\mathcal{A} = \{a_1, a_2\}$

Each edge is associated with a transition probability and a reward.

For instance, we can observe that:

$$P(s_3|s_3, a_2) = 0.6,\ \ P(s_2|s_3, a_2) = 0.4,$$
$$r(s_3, a_2, s_3) = -2,\ \ r(s_3, a_2, s_2) = -1.$$

State Value and Action Value



- Trajectory (轨迹):

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, s_3, \cdots), \quad r_t = r(s_t, a_t, s_{t+1}).$$

- Given s_0, the probability of trajectory τ is given by

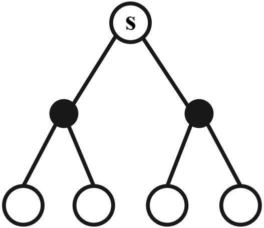
$$P_{s_0}^{\pi}(\tau) = \prod_{t=0}^{\infty} \pi(a_t|s_t)P(s_{t+1}|s_t, a_t).$$

- Infinite horizon discounted return (折扣回报):

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots = \sum_{t=0}^{\infty} \gamma^t r_t.$$

Here we consider infinite horizon discounted return which enable us to focus on the stationary policy. In finite horizon problems, it may be beneficial to select a different action depending on the remaining time steps which has the form $\pi(s) = (\pi_0(s), \pi_1(s), \cdots)$.

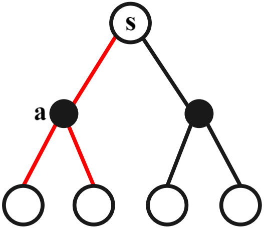
State Value and Action Value



- State value (状态价值函数):

$$V^{\pi}(s) = \mathbb{E}_{\tau \sim P_{s_0}^{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right], \quad \forall s \in \mathcal{S}.$$

State Value and Action Value



- Action value (Q-value, 动作价值函数):

$$Q^{\pi}(s,a)=\mathbb{E}_{\tau\sim P^{\pi}_{s_0,a_0}}\left[\sum_{t=0}^{\infty}\gamma^t r_t|s_0=s,a_0=a\right],\quad \forall(s,a)\in\mathcal{S}\times\mathcal{A},$$

where the probability for $\tau=(s_0,a_0,r_0,s_1,a_1,r_1,s_2,a_2,r_2,s_3,\cdots)$ is given by

$$P^{\pi}_{s_0,a_0}(\tau)=P(s_1|s_0,a_0)\prod_{t=1}^{\infty}\pi_t(a_t|s_t)P(s_{t+1}|s_t,a_t).$$

State Value and Action Value

- State and action values can be used to quantify goodness/badness of policies and actions.
- The relation between the state value and the action value is given by

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} \left[Q^{\pi}(s, a) \right],$$
$$Q^{\pi}(s, a) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s, a, s') + \gamma V^{\pi}(s') \right].$$

- Computing the expectation seems not easy. However, the MDP structure enables us to compute the values by finding the solutions to linear systems (i.e., Bellman equations).

Bellman Equation for State Value

Theorem
Given policy π, state value satisfies the following ***Bellman equation****:*

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma V^\pi(s')\right].$$

Alternatively, if for any $V \in \mathbb{R}^{|\mathcal{S}|}$, define the ***Bellman operator****:*

$$[\mathcal{T}^\pi V](s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma V(s')\right],$$

Bellman equation can be rewritten as

$$V^\pi = \mathcal{T}^\pi V^\pi.$$

That is, V^π is a fixed point of $\mathcal{T}^\pi$, and can be computed via fixed point iteration.

- $\mathcal{T}^\pi$ looks one step ahead using policy π.

Matrix Form for Bellman Operator

Lemma
The Bellman operator can be expressed as the following matrix form:

$$\mathcal{T}^\pi V = r^\pi + \gamma P^\pi V,$$

where

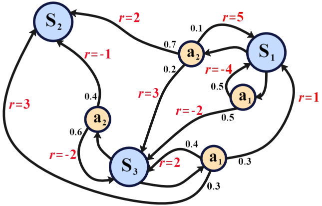
$$r^\pi = \begin{bmatrix} r^\pi(s_1) \\ \vdots \\ r^\pi(s_n) \end{bmatrix}, \quad P^\pi = \begin{bmatrix} p^\pi_{s_1 s_1} & \cdots & p^\pi_{s_1 s_n} \\ \vdots & \ddots & \vdots \\ p^\pi_{s_n s_1} & \cdots & p^\pi_{s_n s_n} \end{bmatrix},$$

and the entries of r_π and P^π are

$$r^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s')\right] \quad \textit{and} \quad p^\pi_{ss'} = \sum_a \pi(a|s)P(s'|s,a).$$

- $p^\pi_{ss'}$ is the transition probability from s to s' under policy π.

Illustrative Example



Consider policy $\pi(a|s) = 0.5$ for all s, a and let $\gamma = 0.9$:

$$P^{\pi} = \begin{bmatrix} 0.3 & 0.35 & 0.35 \\ 0 & 1 & 0 \\ 0.15 & 0.35 & 0.5 \end{bmatrix},$$

$$r^{\pi} = [-0.25, 0, 0.2]^T,$$

$$V^{\pi} = [-0.21, 0, 0.31]^T.$$

We can also verify the correctness of V^{π}. Taking the state s_0 as an example, it is not hard to show that

$$\begin{aligned} V^{\pi}(s_3) &= \sum_a \pi(a|s_3) \sum_{s'} p(s'|s_3, a) \left(r(s_3, a, s') + \gamma V^{\pi}(s') \right) \\ &= 0.5\left(-1.6 + 0.9 \times 0.6 \times 0.31\right) + 0.5\left(2 + 0.9(0.4 \times 0.31 - 0.3 \times 0.21)\right) \\ &= 0.31. \end{aligned}$$

Bellman Equation for Action Value

Theorem

*Given policy π, action value satisfies the following **Bellman equation**:*

$$Q^\pi(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma \mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q^\pi(s',a')\right]\right].$$

*Alternatively, if for any $Q \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$, define the **Bellman operator**:*

$$[\mathcal{F}^\pi Q](s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma \mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q(s',a')\right]\right],$$

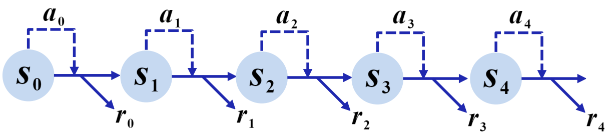
Bellman equation can be rewritten as

$$Q^\pi = \mathcal{F}^\pi Q^\pi.$$

That is, Q^π is a fixed point of $\mathcal{F}^\pi$.

- $\mathcal{F}^\pi$ also admits a matrix form and it is also a contraction with infinity norm.

Goal of RL



Trajectory: $\tau = (s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots)$

Recall definitions of state value at s and action value at (s, a):

$$V^{\pi}(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) | s_0 = s, \pi\right],$$

$$Q^{\pi}(s, a) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right].$$

Goal of RL is to a find a policy that maximizes weighted state values:

$$\max_{\pi} V^{\pi}(\mu), \quad \text{where } V^{\pi}(\mu) := \mathbb{E}_{s\sim\mu}[V^{\pi}(s)].$$

Basic RL Methods

- Valued-based methods: Not directly optimize policy but seek optimal state or action values based on fixed point iteration or dynamic programming:

$$\begin{cases} \text{Value Iteration} \\ \text{Policy Iteration} \end{cases} \xrightarrow{\textit{sampling}} \begin{cases} \text{MC Learning} \\ \text{SARSA} \\ \text{Q-Learning} \end{cases} \xrightarrow[\textit{approximation}]{\textit{function}} \text{Deep Q-Learning}$$

Basic RL Methods

- Policy optimization: Directly optimize policy via parameterization $\pi_\theta(\cdot|s)$:

$$V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right],$$

where $P_\mu^{\pi_\theta}(\tau) = \mu(s_0) \prod_{t=0}^{\infty} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$. Then maximize $V^{\pi_\theta}(\mu)$ is finite dimensional optimization problem about θ. Value exists in expression of policy gradient and policy optimization+value update= Actor-Critic.

Course: Algorithmic and Theoretical Foundations of RL (https://makwei.github.io/rlIndex.html), also includes materials about online planning.

Value-Based Methods

Optimal State and Action Values

$$V^* (s) := \sup_\pi V^\pi (s) , \quad Q^* (s, a) := \sup_\pi Q^\pi (s, a) .$$

- The optimal state and action values satisfy (Bellman optimality equations)

$$Q^*(s, a) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s, a, s') + \gamma V^* (s') \right] \text{ and } V^*(s) = \max_a Q^*(s, a).$$

- Given optimal values, an optimal policy can be retrieved as

$$\pi^*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_a \underbrace{\mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s, a, s') + \gamma V^*(s') \right]}_{Q^*(s,a)}, \\ 0 & \text{otherwise.} \end{cases}$$

Value-based methods learn optimal values, then retrieval optimal policies.

Bellman Optimality Equation for Optimal State Value

Theorem

The optimal state value satisfies the following ***Bellman optimality equation****:*

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s,a,s') + \gamma V^*\left(s'\right) \right].$$

Alternatively, if for any $V \in \mathbb{R}^{|\mathcal{S}|}$, define the ***Bellman optimality operator****:*

$$[\mathcal{T} V](s) = \max_a \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s,a,s') + \gamma V\left(s'\right) \right],$$

Bellman optimality equation can be rewritten as

$$V^* = \mathcal{T} V^*.$$

That is, V^* is a fixed point of $\mathcal{T}$.

- $\mathcal{T}$ is a contraction with infinity norm.

Bellman Optimality Equation for Optimal Action Value

Theorem

The optimal action value satisfies the following ***Bellman optimality equation****:*

$$Q^*(s,a) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s,a,s') + \gamma \max_{a' \in \mathcal{A}} Q^*(s',a') \right].$$

Alternatively, if for any $Q \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$, define the ***Bellman optimality operator****:*

$$[\mathcal{F}Q](s,a) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s,a,s') + \gamma \max_{a' \in \mathcal{A}} Q(s',a') \right],$$

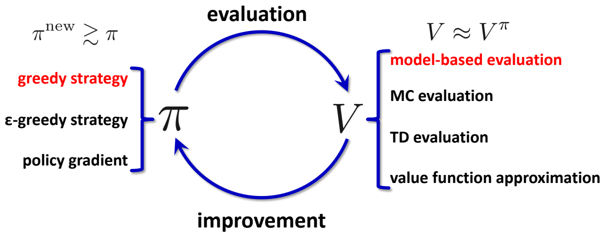
Bellman optimality equation can be rewritten as

$$Q^* = \mathcal{F}Q^*.$$

That is, Q^* is a fixed point of $\mathcal{F}$.

- $\mathcal{F}$ is a contraction with infinity norm. This is the foundation of Q-learning.

An Overall Framework



Overall, different RL algorithms can be viewed as implementing the idea of alternative update of value and policy in different ways. We first present the idea in the model based setting.

Value Iteration

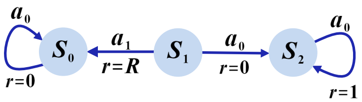
Value Iteration (VI): Solve Bellman optimality equation by fixed point iteration,

$$V^{k+1}(s) = \max_a \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s,a,s') + \gamma V^k(s') \right].$$

- To retrieve a policy after value iteration:

$$\pi_{k+1}(a|s) = \begin{cases} 1 & \arg\max_a \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s,a,s') + \gamma V^k(s') \right], \\ 0 & \text{otherwise}. \end{cases}$$

Illustrative Example



- three states: $\mathcal{S} = \{s_0, s_1, s_2\}$
- two actions: $\mathcal{A} = \{a_0, a_1\}$

Each edge is associated with a deterministic transition and a reward.

Suppose we start from $V^0 = 0$. Then

$$V^k(s_0) = r(s_0, a_0, s_0) + \gamma V^{k-1}(s_0) = \gamma V^{k-1}(s_0) = \gamma^k V^0(s_0) = 0,$$

$$V^k(s_2) = r(s_2, a_0, s_2) + \gamma V^{k-1}(s_2) = 1 + \gamma V^{k-1}(s_2) = \frac{1-\gamma^k}{1-\gamma} + \gamma^k V^0(s_2) = \frac{1-\gamma^k}{1-\gamma},$$

$$\begin{aligned} V^k(s_1) &= \max\left\{ r(s_1, a_0, s_2) + \gamma V^{k-1}(s_2), r(s_1, a_1, s_0) + \gamma V^{k-1}(s_0) \right\} \\ &= \max\left\{ \frac{\gamma}{1-\gamma}\left(1-\gamma^{k-1}\right), R \right\}. \end{aligned}$$

Thus (assuming $R < \frac{\gamma}{1-\gamma}$),

$$V^*(s_0) = 0, \quad V^*(s_1) = \frac{\gamma}{1-\gamma}, \quad V^*(s_2) = \frac{1}{1-\gamma}.$$

Asynchronous Value Iteration

State values in VI are updated synchronously. An alternative is **asynchronous value iteration**: Rather than sweeping through all states to create a new value vector, only updates one state (an entry of vector) at a time.

Gauss-Seidel Value Iteration:

$$\text{for } s = 1, 2, 3, \dots$$
$$\quad V(s) \leftarrow \max_{a} \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s, a, s') + \gamma V(s') \right]$$

Policy Iteration

$$\pi_0 \xrightarrow{\mathsf{E}} V^{\pi_0} \xrightarrow{\mathsf{I}} \pi_1 \xrightarrow{\mathsf{E}} V^{\pi_1} \xrightarrow{\mathsf{I}} \pi_2 \xrightarrow{\mathsf{E}} \cdots \xrightarrow{\mathsf{I}} \pi^*$$

There are two ingredients in **Policy Iteration (PI)**.

Policy Evaluation:

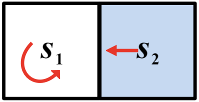
$$V^{\pi_k} = r^{\pi_k} + \gamma P^{\pi_k} V^{\pi_k}.$$

Policy Improvement:

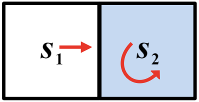
$$\pi_{k+1}(a|s) = \begin{cases} 1 & a = \arg\max_a \underbrace{\mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s,a,s') + \gamma V^{\pi_k}(s') \right]}_{Q^{\pi_k}(s,a)}, \\ 0 & \text{otherwise.} \end{cases}$$

Illustrative Example

Consider the example in following figure, where each state is associated with three possible actions: a_l, a_0, a_r (move leftwards, stay unchanged, and move rightwards). The reward is r_{s_1} = -1 and r_{s_2} = 1. The discount rate is γ = 0.9.



(a)



(b)

Assume the initial policy π_0 is given in (a). This policy satisfies $\pi_0(a_0|s_1) = 1$ and $\pi_0(a_l|s_2) = 1$. This policy is not good because it does not move toward s_2. We next apply policy iteration problem.

Illustrative Example

- Policy Evaluation

$$\begin{cases} V^{\pi_0}(s_1) = -1 + \gamma V^{\pi_0}(s_1) \\ V^{\pi_0}(s_2) = -1 + \gamma V^{\pi_0}(s_1) \end{cases} \Rightarrow \begin{cases} V^{\pi_0}(s_1) = -10 \\ V^{\pi_0}(s_2) = -10 \end{cases}$$

- Policy Improvement

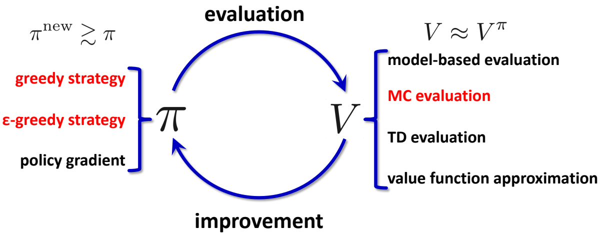
$Q^{\pi_0}(s,a)$	a_ℓ	a_0	a_r
s_1	$-$	-10	-8
s_2	-10	-8	$-$

Since π_1 choose the action that maximize $Q^{\pi_0}(s,a)$, one has (see (b)):

$$\pi_1(a_r|s_1) = 1, \quad \pi_1(a_0|s_2) = 1.$$

It is evident that this is an optimal policy.

Monte Carlo (MC) Learning



Policy Iteration: greedy policy is improved via

$$\pi_{k+1}(s) = \arg\max_{a} \underbrace{\mathbb{E}_{s' \sim P(\cdot|s,a)}[r(s,a,s') + \gamma V^{\pi_k}(s')]}_{Q^{\pi_k}(s,a)},$$

where $Q^{\pi_k}(s,a)$ is evaluated via Bellman equation based on the model.

— What if system information (P and r) is not available?

— Replace model by data (model free).

— How to collect data? How to use data?

— · · · · · ·

MC Policy Evaluation

Basic idea. Given π, estimate $V^\pi(s)$ and $Q^\pi(s,a)$ from sampled trajectories

$$\tau_i = \{(s_0^i, a_0^i, r_0^i, s_1^i, a_1^i, r_1^i, \cdots)\}_{i=1}^n \sim \pi.$$

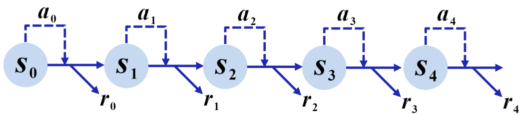
- MC evaluation of $V^\pi(s)$: $s_0^i = s$,

$$V^\pi(s) \approx \frac{1}{n}\sum_{i=1}^n \left(\sum_{t=0}^\infty \gamma^t r_t^i\right).$$

- MC evaluation of $Q^\pi(s,a)$: $s_0^i = s$, $a_0^i = a$,

$$Q^\pi(s,a) \approx \frac{1}{n}\sum_{i=1}^n \left(\sum_{t=0}^\infty \gamma^t r_t^i\right).$$

Use Trajectory More Efficiently



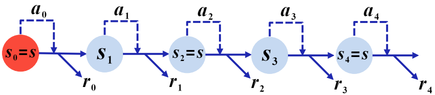
Trajectory $(s_0, a_0, r_0, s_1, a_1, r_1, \cdots) \sim \pi$ starting from s contains sub-trajectories $(s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \cdots)$ that starts from other states (e.g. $s_t = s'$). Thus, return from the sub-trajectory

$$G_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}$$

can be used to build an estimator of $V^\pi(s')$. Namely, one trajectory can be used to estimate different $V^\pi(s)$.

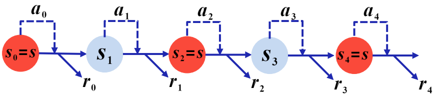
There is no essential difference in the MC evaluations of state value and action value in methodology.

First-Visit and Every Visit



First Visit

- Only sub-trajectory that starts from the first visit of s is used in the estimation of $V^\pi(s)$; One trajectory is only used once in the evaluation of $V^\pi(s)$.



Every Visit

- All sub-trajectories that start from of s is used in the estimation of $V^\pi(s)$; One trajectory might be used many times in the evaluation of $V^\pi(s)$.

Incremental Update

Given a new single estimation G of state value or action value,

- state value update:

$$N(s_t) \leftarrow N(s_t) + 1, \quad V(s_t) \leftarrow V(s_t) + \frac{1}{N(s_t)}(G - V(s_t));$$

- action value update:

$$N(s_t, a_t) \leftarrow N(s_t, a_t) + 1, \quad Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{1}{N(s_t, a_t)}(G - Q(s_t, a_t)).$$

MC Learning with ϵ-Greedy Policy

Algorithm 1: MC Learning with ϵ-Greedy Exploration

Initialization: $N(s, a) = 0, Q(s, a) = 0, \forall s, a$, π_0

for $k = 0, 1, 2, \dots$ **do**

Initialize s_0 and sample an episode following π_k:

$$(s_0, a_0, r_0, s_1, a_1, r_1, \cdots, s_{T-1}, a_{T-1}, r_{T-1}, s_T) \sim \pi_k$$

$G \leftarrow 0$

for $t = T-1, T-2, \dots, 0$ **do**

$G \leftarrow \gamma G + r_t$

if (s_t, a_t) *does not appear in* $(s_0, a_0, s_1, a_1, \dots, s_{t-1}, a_{t-1})$ **then**

$N(s_t, a_t) \leftarrow N(s_t, a_t) + 1$

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{1}{N(s_t, a_t)}(G - Q(s_t, a_t))$

Update policy of visited state via ϵ_k-greedy:

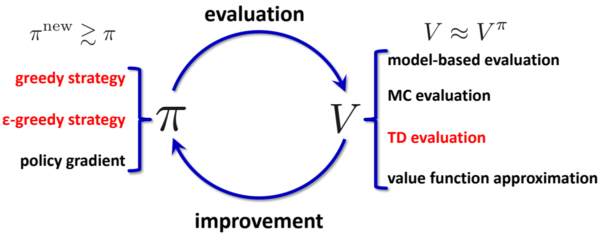
$$\pi_{k+1}(a|s_t) = \begin{cases} 1 - \epsilon_k + \frac{\epsilon_k}{|\mathcal{A}|} & \text{if } a = \arg\max_{a'} Q(s_t, a') \\ \frac{\epsilon_k}{|\mathcal{A}|} & \text{otherwise} \end{cases}$$

end

end

end

Temporal-Difference (TD) Learning



- Model-based evaluation: Solve Bellman equation accurately based on model;
- MC evaluation: Value estimation via sample mean;
- TD evaluation: Solve Bellman equation in a stochastic and online manner.

TD Policy Evaluation of Action Values

Recall that the Bellman equation for Q-values is

$$\begin{aligned} Q^\pi(s,a) &= [\mathcal{F}^\pi Q^\pi](s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma \mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q^\pi(s',a')\right]\right] \\ &= \mathbb{E}_{s'\sim P(\cdot|s,a)}\mathbb{E}_{a'\sim\pi(\cdot|s')}\left[r(s,a,s') + \gamma Q^\pi(s',a')\right], \quad (s,a) \in \mathcal{S}\times\mathcal{A}. \end{aligned}$$

The Bellman iteration for computing Q-values is given by

$$\begin{aligned} Q^{t+1}(s,a) &= \mathbb{E}_{s'\sim P(\cdot|s,a)}\mathbb{E}_{a'\sim\pi(\cdot|s')}\left[r(s,a,s') + \gamma Q^t(s',a')\right] \\ &= Q^t(s,a) + \alpha_t(s,a)\left(\mathbb{E}_{s'\sim P(\cdot|s,a)}\mathbb{E}_{a'\sim\pi(\cdot|s')}\left[r(s,a,s') + \gamma Q^t(s',a')\right] - Q^t(s,a)\right). \end{aligned}$$

Given a random sample (s,a,r,s',a'), the RM algorithm is

$$Q^{t+1}(s,a) = Q^t(s,a) + \alpha_t(s,a)\left(r(s,a,s') + \gamma Q^t(s',a') - Q^t(s,a)\right).$$

TD evaluation of actions values implements this in an online manner.

SARSA: On Policy TD Learning

Algorithm 2: SARSA

Initialization: $Q^0(s,a) = 0$, s_0, π_0, $a_0 \sim \pi_0(\cdot|s_0)$

for $t = 0, 1, 2, \dots$ **do**

Sample a tuple $(s_t, a_t, r_t, s_{t+1}, a_{t+1}) \sim \pi_t$ from (s_t, a_t)

$Q^{t+1}(s_t, a_t) = Q^t(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma Q^t(s_{t+1}, a_{t+1}) - Q^t(s_t, a_t)\right)$

Update policy of visited state via ϵ_t-greedy:

$$\pi_{t+1}(a|s_t) = \begin{cases} 1 - \epsilon_t + \frac{\epsilon_t}{|\mathcal{A}|} & \text{if } a = \arg\max_{a'} Q^{t+1}(s_t, a'), \\ \frac{\epsilon_t}{|\mathcal{A}|} & \text{otherwise.} \end{cases}$$

end

- SARSA is the abbreviation of “state-action-reward-state-action”, and it is an on policy algorithm which updates the policy after every time step.

Q-Learning: Off-Policy TD-Learning

Recall that the optimal state-action values Q^* is the fixed point of the Bellman optimality operator $\mathcal{F}$ where

$$[\mathcal{F}Q](s,a) = \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma \cdot \max_{a' \in \mathcal{A}} Q(s',a')\right], \quad (s,a) \in \mathcal{S} \times \mathcal{A}.$$

It can be shown that $\mathcal{F}$ is a contraction with factor γ. Assuming the model (probability transition model) is known we can find Q^* via Q-value iteration:

$$\begin{aligned} Q^{t+1}(s,a) &= [\mathcal{F}Q^t](s,a) \\ &= Q^t(s,a) + \alpha_t(s,a)([\mathcal{F}Q^t](s,a) - Q^t(s,a)), \quad (s,a) \in \mathcal{S} \times \mathcal{A}. \end{aligned}$$

Q-learning is a model free and online implementation of Q-value iteration: Sample a tuple (s,a,r,s') via a behavior policy, noting that

$$r + \gamma \cdot \max_{a' \in \mathcal{A}} Q^t(s',a')$$

is an unbiased estimator of $\mathcal{F}Q^t(s,a)$, we can update action-value at (s,a) by

$$Q^{t+1}(s,a) = Q^t(s,a) + \alpha_t(s,a)\left(r + \gamma \cdot \max_{a' \in \mathcal{A}} Q^t(s',a') - Q^t(s,a)\right).$$

Q-Learning: Off-Policy TD-Learning

Algorithm 3: Q-Learning

Initialization: $Q^0(s, a) = 0$, s_0

for $t = 0, 1, 2, \dots$ **do**

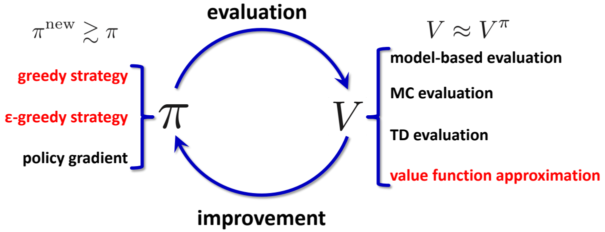
Sample a tuple $(s_t, a_t, r_t, s_{t+1}) \sim b_t$ from s_t, where b_t is a behavior policy

Update Q-value at visited state-action pair (s_t, a_t):

$$Q^{t+1}(s_t, a_t) = Q^t(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \cdot \max_{a' \in \mathcal{A}} Q^t\left(s_{t+1}, a'\right) - Q^t(s_t, a_t)\right)$$

end

Value Function Approximation (VFA)



Approximately represent state/action values with functions

$$V^\pi(s) \approx V(s;\omega) \quad \text{or} \quad Q^\pi(s,a) \approx Q(s,a;\omega)$$

- Learn parameter ω instead of state/action value directly
- Generalize from seen states/actions to unseen states/actions

Policy Evaluation of Actions Values with VFA

With an oracle for $Q^\pi(s, a)$, we can form the following optimization problem

$$\min_\omega J(\omega) = \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\|Q(s, a; \omega) - Q^\pi(s, a)\|_2^2 \right].$$

The SGD for this problem is given by

$$\omega_{t+1} = \omega_t + \alpha_t \cdot (Q^\pi(s, a) - Q(s, a; \omega_t))\nabla_\omega Q(s, a; \omega_t).$$

Sample a tuple (s, a, r, s', a'). We can estimate $Q^\pi(s, a)$ by $r + \gamma \cdot Q(s', a'; \omega_t)$, yielding the update

$$\omega_{t+1} = \omega_t + \alpha_t \cdot (r + \gamma \cdot Q(s', a'; \omega_t) - Q(s, a; \omega_t))\nabla_\omega Q(s, a; \omega_t).$$

Linear VFA of Action Values

In linear VFA for action values, we have

$$Q(s, a; \omega) = \phi(s, a)^T \omega, \quad \text{where } \omega \in \mathbb{R}^n \text{ and } \begin{bmatrix} \phi_1(s, a) \\ \phi_2(s, a) \\ \vdots \\ \phi_n(s, a) \end{bmatrix} \in \mathbb{R}^n.$$

It is clear that $\nabla_\omega Q(s, a; \omega) = \phi(s, a)$.

SARSA with Linear VFA

Algorithm 4: SARSA with Linear VFA

Initialization: $\phi_{s,a}$, s_0, π_0, $a_0 \sim \pi_0(s_0)$

for $t = 0, 1, 2, \ldots$ **do**

Sample a tuple $(s_t, a_t, r_t, s_{t+1}, a_{t+1}) \sim \pi_t$ from (s_t, a_t)

$\omega_{t+1} = \omega_t + \alpha_t \left(r_t + \gamma \cdot \phi(s_{t+1}, a_{t+1})^T \omega_t - \phi(s_t, a_t)^T \omega_t \right) \phi(s_t, a_t)$

Update policy of visited state via ϵ_t-greedy:

$$\pi_{t+1}(a|s_t) = \begin{cases} 1 - \epsilon_t + \frac{\epsilon_t}{|\mathcal{A}|} & \text{if } a = \arg\max_{a'} \phi(s_t, a')^T \omega_{t+1}, \\ \frac{\epsilon_t}{|\mathcal{A}|} & \text{otherwise.} \end{cases}$$

end

Q-Learning with Linear VFA

In Q-learning $Q(s, a; \omega)$ is used to approximate $Q^*(s, a)$. Having a transition $(s_t, a_t, r_t, s_{t+1}) \sim b_t$, we can construct $r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \omega_t)$ as a better estimation of $Q^*(s_t, a_t)$ than $Q(s_t, a_t; \omega_t)$ since one-step lookahead reward r_t is accurate (or approximate error is discounted by γ), and update ω_t via

$$\omega_{t+1} = \omega_t + \alpha_t \left(r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \omega_t) - Q(s_t, a_t; \omega_t) \right) \nabla_\omega Q(s_t, a_t; \omega_t)$$

to reduce $\mathcal{L}(\omega) = \frac{1}{2} \left(r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \omega_t) - Q(s_t, a_t; \omega) \right)^2$.

Algorithm 5: Q-Learning with linear VFA

Initialization: $\phi(s, a)$, s_0

for $t = 0, 1, 2, \ldots$ **do**

Sample a tuple $(s_t, a_t, r_t, s_{t+1}) \sim b_t$ from s_t where b_t is a behavior policy

Update parameter

$$\omega_{t+1} = \omega_t + \alpha_t \left(r_t + \gamma \cdot \max_a \phi(s_{t+1}, a)^T \omega_t - \phi(s_t, a_t)^T \omega_t \right) \phi(s_t, a_t)$$

end

Q-Learning with VFA as Approximate Q-Value Iteration

Recall that the Q-value iteration has the following form:

$$Q^{t+1} = \mathcal{F}Q^t, \quad \text{where } [\mathcal{F}Q](s,a) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s,a,s') + \gamma \cdot \max_{a' \in \mathcal{A}} Q(s',a') \right].$$

With Q^t being replaced by $Q(:;\omega_t)$, there may not be a function $Q(:;\omega_{t+1})$ such that $Q(:;\omega_{t+1}) = \mathcal{F}Q(:;\omega_t)$ holds exactly. We can solve for $Q(:;\omega_{t+1})$ via

$$\begin{aligned}
\omega_{t+1} &= \arg\min_{\omega} \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[(Q(s,a;\omega) - [\mathcal{F}Q](s,a;\omega_t))^2 \right] \\
&= \arg\min_{\omega} \mathbb{E}_{(s,a)\sim\mathcal{D}, s' \sim P(\cdot|s,a)} \left[\left(Q(s,a;\omega) - \left(r(s,a,s') + \gamma \cdot \max_{a' \in \mathcal{A}} Q(s',a';\omega_t) \right) \right)^2 \right].
\end{aligned}$$

Solving it via one step SGD yields Q-learning with VFA.

Batch Method

Let $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^n$ be a batch of experience data. At time t, we can form an sample version of $\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[(Q(s,a;\omega) - \mathcal{F}Q(s,a;\omega_t))^2\right]$ and update ω by finding a solution to the empirical risk minimization (or regression) problem

$$\omega_{t+1} = \arg\min_{\omega} \sum_{i=1}^{n} \big(Q(s_i, a_i; \omega) - \big(r_i + \gamma \cdot \max_{a' \in \mathcal{A}} Q(s_i', a'; \omega_t)\big)\big)^2.$$

Solving this problem by batch SGD yields an instance of Fitted Q-Iteration.

Fitted Q-Iteration (FQI): Offline Approximate Q-Value Iteration

Algorithm 6: FQI

Initialization: Dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^n$, initial VFA parameter ω

for $t = 0, 1, 2, \ldots$ *until some stopping criterion is met* **do**

  Copy parameter: $\tilde{\omega} \leftarrow \omega$

  for $k = 0, 1, 2, \ldots$ *until some stopping criterion is met* **do**

    Sample a mini-batch $\mathcal{B}$ of $\mathcal{D}$

    $\omega \leftarrow \omega + \alpha \sum_{(s_i, a_i, r_i, s_i') \in \mathcal{B}} \left(r_i + \gamma \cdot \max_{a'} Q(s_i', a'; \tilde{\omega}) - Q(s_i, a_i; \omega)\right) \nabla_\omega Q(s_i, a_i; \omega)$

  end

end

Deep Q-Learning

Deep Q-learning is a variant of FQI which uses deep neural network for VFA and adopts incremental learning by maintaining a buffer and experience replay.

Algorithm 7: DQN

Initialization: Replay buffer $\mathcal{D}$ to capacity N, Q network $Q(s, a; \omega)$ with ω, target Q network $q(s, a; \tilde{\omega})$ with $\tilde{\omega} = \omega$, SGD iteration number C, $k = 0$, and s_0

for $t = 0, 1, 2, \ldots$ *until some stopping criterion* **do**

  $k \leftarrow k + 1$

  Sample a tuple $(s_t, a_t, r_t, s_{t+1}) \sim b_t$ from s_t and add it to buffer $\mathcal{D}$

  sample a mini-batch $\mathcal{B}$ of $\mathcal{D}$

  $\omega \leftarrow \omega + \alpha \sum_{(s_i, a_i, r_i, s_i') \in \mathcal{B}} (r_i + \gamma \cdot \max_{a'} Q(s_i', a'; \tilde{\omega}) - Q(s_i, a_i; \omega)) \nabla_\omega Q(s_i, a_i; \omega)$

  if $k == C$ **then**

    $\tilde{\omega} \leftarrow \omega$

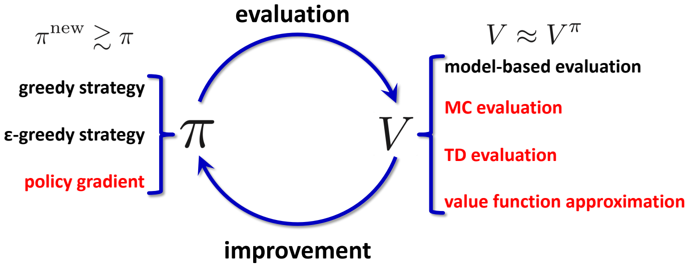
    $k \leftarrow 0$

  end

end

Policy Optimization

Value-Based RL vs Policy-Based RL



- Value-based RL: Learn optimal values and policy is implicitly inferred;
- Policy-based RL: Parametrize policy and conduct search in policy space.

Policy-Based RL

Consider a policy parameterization (which is essentially about how to represent a distribution) such that :

$$\pi_\theta(\cdot|s) \text{ defines a probability distribution on } \mathcal{A}.$$

Note that once θ is given, policy is determined.

Goal: Search for best θ subject to certain performance measure.

Typical advantages of policy-based methods include:

- Better convergence properties
- Effective in high dimensional or continuous action spaces
- Can learn stochastic policies

Policy Optimization

Consider average state value with initial distribution μ as performance measure:

$$V^{\pi_\theta}(\mu) = \mathbb{E}_{s_0 \sim \mu}\left[V^{\pi_\theta}(s_0)\right] = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\left[r(\tau)\right],$$

where given $\tau = (s_t, a_t, r_t)_{t=0}^{\infty}$,

$$P_\mu^{\pi_\theta}(\tau) = \mu(s_0)\prod_{t=0}^{\infty}\pi_\theta(a_t|s_t)P(s_{t+1}|s_t, a_t) \quad \text{and} \quad r(\tau) = \sum_{t=0}^{\infty}\gamma^t r_t.$$

It is natural to formulate RL as

$$\theta^* = \arg\max_\theta V^{\pi_\theta}(\mu).$$

For simplicity, we only discuss the case where sate and action spaces are discrete.

Performance Difference Lemma

Given a policy π, the advantage function is defined as

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

which measures how well a single action is compared with average state value.

Lemma (Performance Difference Lemma)

For any two policies π_1, π_2, one has

$$V^{\pi_1}(\mu) - V^{\pi_2}(\mu) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\mu}^{\pi_1}}\left[\mathbb{E}_{a\sim\pi_1(\cdot|s)}\left[A^{\pi_2}(s,a)\right]\right].$$

Policy Gradient Theorem

Theorem (Policy Gradient Theorem)

Recalling the definition of visitation measure, we have

$$\begin{aligned}\nabla_\theta V^{\pi_\theta}(\mu) &= \mathbb{E}_{\tau\sim P_\mu^{\pi_\theta}}\left[\sum_{t=0}^{\infty}\gamma^t Q^{\pi_\theta}(s_t,a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\right] \\ &= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^{\pi_\theta}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)\right].\end{aligned}$$

- Policy gradient theorem expresses policy gradient as a weighted average of $\nabla_\theta \log \pi_\theta(a|s)$ over all state-action pairs. Note that $\nabla_\theta \log \pi_\theta(a|s)$ is direction that $\pi_\theta(a|s)$ increases (i.e., probability of selecting a at s increases).

Policy Gradient in Terms of Advantage Function

Theorem (Policy Gradient in Terms of Advantage Function)

We have

$$\nabla_\theta V^{\pi_\theta}(\mu) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^{\pi_\theta}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[A^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)\right],$$

provided $\sum_a \pi_\theta(a|s) = 1$ *for any* θ.

Proof. The result follows from the fact

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a|s)\right] = \nabla_\theta\Big(\sum_a \pi_\theta(a|s)\Big) = 0.$$

Policy Gradient Ascent

$$\begin{aligned}\theta &\leftarrow \theta + \alpha \cdot \mathbb{E}_{s,a}\left[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)\right] \\ &= \theta + \alpha \cdot \mathbb{E}_{s,a}\left[\frac{Q^{\pi_\theta}(s,a)}{\pi_\theta(a|s)}\nabla_\theta \pi_\theta(a|s)\right]\end{aligned}$$

- Large $Q^{\pi_\theta}(s,a)$ means that weight in front of the direction $\nabla_\theta \pi_\theta(a|s)$ is large. Thus, the method attempts to exploit actions with large action values.
- Small $\pi_\theta(a|s)$ means that weight in front of the direction $\nabla_\theta \pi_\theta(a|s)$ is large. This reflects that the method attempts to explore actions with low probability.
- Policy gradient method also fits into the framework of policy evaluation and policy improvement, where policy evaluation affects direction to improve the policy and policy improvement is achieved by updating policy parameter. Thus, analysis of policy gradient methods often boils down to analysis of improvement ability in policy domain.

Example: Softmax Parameterization

Lemma

The policy gradient under softmax parameterization is given by

$$\nabla_{\theta_s} V^{\pi_\theta}(\mu) = \frac{d_\mu^{\pi_\theta}(s)}{1-\gamma} \pi_\theta(\cdot|s) A^{\pi_\theta}(s, \cdot).$$

- Softmax PG: in the parameter space,

$$\theta_{s,a}^{+} = \theta_{s,a} + \eta \frac{d_\mu^{\pi_\theta}(s)}{1-\gamma} \pi_\theta(a|s) A_\tau^{\pi_\theta}(s, a).$$

  In the policy space,

$$\pi_{s,a}^{+} \propto \pi_{s,a} \exp\left(\eta \frac{d_\mu^{\pi}(s)}{1-\gamma} \pi_\theta(a|s) A_\tau^{\pi_\theta}(s, a) \right).$$

Trust Region Policy Optimization (TRPO)

Overall Idea

Given a policy π_{θ_t}, by performance difference lemma, we can rewrite $V^{\pi_\theta}(\mu)$ as

$$V^{\pi_\theta}(\mu) = V^{\pi_{\theta_t}}(\mu) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^{\pi_\theta}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[A^{\pi_{\theta_t}}(s,a)\right].$$

Since we do not have access to $d_\mu^{\pi_\theta}$, instead maximize the approximation:

$$\max_\theta \; V_t(\theta) = V^{\pi_{\theta_t}}(\mu) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^{\pi_{\theta_t}}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[A^{\pi_{\theta_t}}(s,a)\right].$$

Trust Region Policy Optimization (TRPO)

Two Facts

- Assume $\sum_a \pi_\theta(a|s) = 1$ for any θ. It is easy to see that $V^{\pi_\theta}(\mu)$ and $V_t(\theta)$ match at θ_t up to first derivative.
- It can be shown that

$$V^{\pi_\theta}(\mu) \geq V_t(\theta) - \frac{2\gamma\varepsilon_t}{(1-\gamma)^2} \max_s \mathrm{KL}(\pi_{\theta_t}(\cdot|s) \| \pi_\theta(\cdot|s)),$$

where $\varepsilon_t = \max_{s,a} |A^{\pi_{\theta_t}}(s,a)|$.

See "Trust region policy optimization" by Schulman et al. 2017 for derivation of second fact.

Trust Region Policy Optimization (TRPO)

The second fact suggests that we may seek a new estimator by maximizing $V_t(\theta)$ in a small neighborhood of θ_t:

$$\max_{\theta} \; V_t(\theta) \quad \text{subject to} \quad \max_{s} \mathrm{KL}(\pi_{\theta_t}(\cdot|s) \| \pi_\theta(\cdot|s)) \leq \delta.$$

Moreover, replace constraint by the average version and instead solve

$$\max_{\theta} \; V_t(\theta) \quad \text{subject to} \quad \mathbb{E}_{s \sim d_\mu^{\pi_{\theta_t}}} \left[\mathrm{KL}(\pi_{\theta_t}(\cdot|s) \| \pi_\theta(\cdot|s))\right] \leq \delta.$$

Trust Region Policy Optimization (TRPO)

After linear approximation to $V_t(\theta)$ and quadratic approximation to KL at θ_t,

$$V_t(\theta) \approx (\nabla_\theta V^{\pi_{\theta_t}}(\mu))^T(\theta - \theta_t),\ \mathbb{E}_{s \sim d_\mu^{\pi_{\theta_t}}}\left[\mathrm{KL}(\pi_{\theta_t}(\cdot|s)\|\pi_\theta(\cdot|s))\right] \approx \frac{1}{2}(\theta - \theta_t)^T F(\theta_t)(\theta - \theta_t),$$

we arrive at the same problem as that for NPG,

$$\max_\theta (\nabla_\theta V^{\pi_{\theta_t}}(\mu))^T(\theta - \theta_t) \quad \text{subject to} \quad \frac{1}{2}(\theta - \theta_t)^T F(\theta_t)(\theta - \theta_t) \leq \delta.$$

- TRPO is overall natural policy gradient (NPG) with adaptive line search.

Proximal Policy Optimization (PPO)

Recall from last section that

$$
\begin{aligned}
V_t(\theta) &\propto \mathbb{E}_{s\sim d_\mu^{\pi_{\theta_t}}} \mathbb{E}_{a\sim\pi_\theta(\cdot|s)} \left[A^{\pi_{\theta_t}}(s,a)\right] \\
&= \mathbb{E}_{s\sim d_\mu^{\pi_{\theta_t}}} \mathbb{E}_{a\sim\pi_{\theta_t}(\cdot|s)} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_t}(a|s)} A^{\pi_{\theta_t}}(s,a)\right],
\end{aligned}
$$

serves as a surrogate function of true target in small region around θ_t.

PPO keeps new policy close to old one through clipped objective.

PPO with Clipped Objective

Let $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_t}(a|s)}$. Then $r(\theta_t) = 1$. The clipped objective function is given by

$$V_t^{\text{clip}}(\theta) = \mathbb{E}_{s\sim d_\mu^{\pi_{\theta_t}}} \mathbb{E}_{a\sim\pi_{\theta_t}(\cdot|s)} \left[\min\left(r(\theta)A^{\pi_{\theta_t}}(s,a), \mathrm{clip}\left(r(\theta), 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_t}}(s,a)\right)\right],$$

where

$$\mathrm{clip}\left(r(\theta), 1-\epsilon, 1+\epsilon\right) = \begin{cases} 1+\epsilon, & r(\theta) > 1+\epsilon, \\ r(\theta), & r(\theta) \in [1-\epsilon, 1+\epsilon], \\ 1-\epsilon, & r(\theta) < 1-\epsilon. \end{cases}$$

- The $\min$ operation ensure $V_t^{\text{clip}}(\theta)$ provides a lower bound. Since a maximal point will be computed subsequently, $\min$ will not cancel the effect of clip.
- PPO policy update (in expectation): $\theta_{t+1} = \arg\max_\theta V_t^{\text{clip}}(\theta)$.
- In flat region, gradient of $V_t^{\text{clip}}(\theta)$ is zero, thus won't move far from θ_t is using policy gradient type method to solve the sub-problem.

See "Proximal policy optimization algorithms" by Schulman et al. 2017 for details.

MC Evaluation of Policy Gradient

The expectation in policy gradient expression requires MC evaluation.

- Sample N episodes:

$$\tau^{(i)} = (s_0^{(i)}, a_0^{(i)}, r_0^{(i)}, \cdots, s_{T-1}^{(i)}, a_{T-1}^{(i)}, r_{T-1}^{(i)}, s_T^{(i)}) \sim \pi_\theta;$$

- Use return $G_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$ as an unbiased estimate of $Q^{\pi_\theta}(s_t, a_t)$:

$$\nabla_\theta V^{\pi_\theta}(\mu) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \gamma^t G_t^{(i)} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}).$$

As illustration, we present policy gradient ascent with MC evaluation next.

REINFORCE

Algorithm 8: REINFORCE

Initialization: $\pi_\theta(a|s)$ and θ_0.

for $k = 0, 1, 2, \ldots$ **do**

Sample episodes $\mathcal{D}_k = \{\tau^{(i)}\}$:

$$\tau^{(i)} = (s_0^{(i)}, a_0^{(i)}, r_0^{(i)}, \cdots, s_{T-1}^{(i)}, a_{T-1}^{(i)}, r_{T-1}^{(i)}, s_T^{(i)}) \sim \pi_{\theta_k}$$

Policy gradient calculation:

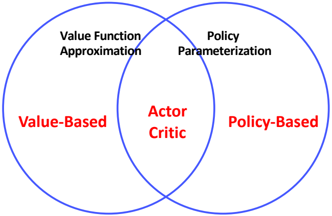
$$g_k = \frac{1}{|\mathcal{D}_k|} \sum_{i=1}^{|\mathcal{D}_k|} \sum_{t=0}^{T-1} \gamma^t G_t^{(i)} \nabla_\theta \log \pi_{\theta_k}(a_t^{(i)}|s_t^{(i)})$$

Policy parameter update:

$$\theta_{k+1} = \theta_k + \alpha_k g_k$$

end

Actor-Critic Methods



- Value-based: Learn value function
- Policy-based: Learn policy function
- Actor-critic: Learn value and policy functions

Actor-Critic Methods

Motivation. MC policy gradient evaluation is sample inefficient and has high variance. Similar to VFA in value-based RL, we can approximate values that appears in policy gradient and update VFA parameters in learning process.

- Actor: Learn parameterized policy π_θ via policy gradient;
- Critic: Learn value function $V(:;\omega)$ or $Q(:;\omega)$ in $\nabla V^{\pi_\theta}(\mu)$ via policy evaluation.

Recall TD evaluation for state value and action value parameter as follows:

$$
\begin{aligned}
\text{(State value)} \quad & \delta_t = r_t + \gamma \cdot V(s_{t+1};\omega) - V(s_t;\omega) \\
& \omega \leftarrow \omega + \alpha_t \, \delta_t \, \nabla_\omega V(s_t;\omega) \\
\text{(Action value)} \quad & \delta_t = r_t + \gamma \cdot Q(s_{t+1}, a_{t+1};\omega) - Q(s_t, a_t;\omega) \\
& \omega \leftarrow \omega + \alpha_t \, \delta_t \, \nabla_\omega Q(s_t, a_t;\omega)
\end{aligned}
$$

Action-Value Actor-Critic

Algorithm 9: Action-Value Actor-Critic

Initialization: policy parameters θ_0, action value function parameter ω_0.

for $t = 0, 1, \cdots$ **do**

Sample a tuple $(s_t, a_t, r_t, s_{t+1}, a_{t+1}) \sim \pi_\theta$

Calculate $\delta_t \leftarrow r_t + \gamma \cdot Q(s_{t+1}, a_{t+1}; \omega) - Q(s_t, a_t; \omega)$

Critic update: $\omega \leftarrow \omega + \alpha_t \, \delta_t \, \nabla_\omega Q(s_t, a_t; \omega)$

Actor update: $\theta \leftarrow \theta + \beta_t \, Q(s_t, a_t; \omega) \nabla_\theta \log \pi_\theta \, (a_t | s_t)$

end

There are other versions of actor-critic, for example, the parameters are only updated at the end of an episode by using all the episode data simultaneously.

Advantage Actor-Critic Method (A2C)

In A2C, advantage function expression for policy gradient is used and value function approximation is applied to state values:

$$Q(s_t, a_t) \approx r_t + \gamma V(s_{t+1}; \omega), \quad A(s_t, a_t) \approx \underbrace{r_t + \gamma V(s_{t+1}; \omega) - V(s_t; \omega)}_{\delta_t}$$

Algorithm 10: Advantage Actor-Critic (A2C)

Initialization: policy parameters θ_0, state value function parameter ω_0.

for $t = 0, 1, \cdots$ **do**

  Sample a tuple $(s_t, a_t, r_t, s_{t+1}) \sim \pi_\theta$

  Calculate $\delta_t \leftarrow r_t + \gamma V(s_{t+1}; \omega) - V(s_t; \omega)$

  Critic update: $\omega \leftarrow \omega + \alpha_t \, \delta_t \, \nabla_\omega V(s_t; \omega)$

  Actor update: $\theta \leftarrow \theta + \beta_t \, \delta_t \, \nabla_\theta \log \pi_\theta \, (a_t | s_t)$

end

Entropy Regularization

Entropy Regularized State Value

Given a policy π, the average entropy regularized state value is given by

$$
\begin{aligned}
V_\tau^\pi(\mu) &= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^\pi}\left\{\mathbb{E}_{a\sim\pi(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s')\right] + \tau H(\pi(\cdot|s))\right\} \\
&= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^\pi}\mathbb{E}_{a\sim\pi(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') - \tau\log\pi(a|s)\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\left(r(s_t,a_t,s_{t+1} - \tau\log\pi(a_t|s_t))\right)\ \middle|\ s_0\sim\mu,\pi\right],
\end{aligned}
$$

where $H(p) = \sum_a p_a \log p_a$ is the entropy of a probability distribution.

- Entropy regularized state value at s, denoted $V_\tau^\pi(s)$, can be similarly defined.
- In addition to the perspective based on entropy regularization for more exploration, it can also be interpreted as encouraging exploration via revising the reward (the third equation).

In this section, we will use τ to denote the regularization parameter, which should be distinguished from the trajectory.

Bellman Equation and Operator

It is clear that $V_\tau^\pi(\mu)$ satisfies the following Bellman equation

$$V_\tau^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') - \tau\log(a|s) + \gamma V_\tau^\pi(s')\right].$$

Define the Bellman operator as follows

$$\mathcal{T}_\tau^\pi V(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') - \tau\log(a|s) + \gamma V(s')\right].$$

It is easy to see that $\mathcal{T}_\tau^\pi$ is of γ-contraction and V_τ^π is a fixed point of $\mathcal{T}_\tau^\pi$.

Entropy Regularized Action Value

The entropy regularized action value is defined as

$$Q_\tau^\pi(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma V_\tau^\pi(s')\right].$$

Note that we choose not to include $-\tau\log\pi(a|s)$ here. One immediately has

$$V_\tau^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q_\tau^\pi(s,a) - \tau\log\pi(a|s)\right].$$

- Action value is state value where initial policy is deterministic, thus entropy 0.
- It is convenient to give the maximum improvement policy (similar to PI policy). That is, the solution to

  $$\max_\pi \mathcal{T}_\tau^\pi V(s) = \max_\pi \mathbb{E}_{a\sim\pi(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') - \tau\log(a|s) + \gamma V(s')\right]$$

  is $\pi(\cdot|s) \propto \exp(Q^V(s,\cdot)/\tau)$, where $Q^V(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma V(s')\right]$. Entropy regularization moves the maxima to the interior so that it has an explicit solution in terms of softmax representation.

Performance Difference Lemma

Define the advantage function

$$A_\tau^\pi(s,a) = Q_\tau^\pi(s,a) - \tau \log \pi(a|s) - V_\tau^\pi(s).$$

It is evident that $\mathbb{E}_{a\sim\pi(\cdot|s)}\left[A_\tau^\pi(s,a)\right] = 0$.

Lemma

One has

$$\mathcal{T}_\tau^{\pi_1} V_\tau^{\pi_2}(s) - V_\tau^{\pi_2}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[A_\tau^\pi(s,a)\right] - \tau \mathrm{KL}(\pi_1(\cdot|s)\|\pi_2(\cdot|s)).$$

Lemma (Performance Difference Lemma)

There holds

$$V_\tau^{\pi_1}(\mu) - V_\tau^{\pi_2}(\mu) = \frac{1}{1-\gamma}\sum_s d_\mu^{\pi_1}(s)\left(\mathcal{T}_\tau^{\pi_1} V_\tau^{\pi_2}(s) - V_\tau^{\pi_2}(s)\right).$$

Optimality

Define the Bellman optimality operator $\mathcal{T}_\tau$ as follows:

$$\mathcal{T}_\tau V(s) = \max_\pi \mathbb{E}_{a\sim\pi(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') - \tau\log(a|s) + \gamma V(s')\right].$$

Then $\mathcal{T}_\tau$ is monotone and γ-contraction with respect to $\|\cdot\|_\infty$.

Theorem (Optimality)

Let V_τ^* be the solution to the Bellman optimality equation $\mathcal{T}_\tau V(s) = \mathcal{T}_\tau V(s)$. Then

$$V_\tau^*(s) = \max_\pi V_\tau^\pi(s).$$

Moreover, there exists an optimal policy π^* such that $V_\tau^{\pi^*} = V_\tau^*$.

Optimality

Proposition
Define $Q_\tau^*(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma V_\tau^*(s')\right]$*. It is evident that*

$$Q_\tau^*(s,a) = \max_\pi Q_\tau^\pi(s,a), \quad \forall s,\ a.$$

Moreover, one has $\pi^*(\cdot|s) \propto \exp\left(Q_\tau^*(s,\cdot)/\tau\right)$ *and*

$$V_\tau^*(s) = Q_\tau^*(s,a) - \tau\log\pi^*(a|s) \Leftrightarrow A_\tau^*(s,a) = 0, \quad \forall a.$$

- Recall that for the non-regularized case, one has $A^*(s,a) \leq 0,\ \forall a$. Moreover, $A_\tau^*(s,a) = 0,\ \forall a$ guarantees $\mathbb{E}_{a\sim\pi^*(\cdot|s)}\left[A_\tau^*(s,a)\right] = 0$ even $\pi^*(\cdot|s) > 0,\ \forall a$.

Lemma (Sub-Optimality Lemma)
There holds

$$V_\tau^*(\mu) - V_\tau^\pi(\mu) = \frac{\tau}{1-\gamma}\sum_s d_\mu^\pi(s)\mathrm{KL}(\pi(\cdot|s)\|\pi^*(\cdot|s)).$$

Reverse Direction

Theorem
If

$$V(s) = \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s,a,s') + \gamma V(s') \right] - \tau \log \pi(a|s), \quad \forall s, a,$$

then $V = V_\tau^*$ *and* $\pi = \pi_\tau^*$.

Proof. Taking expectation with respect to $\pi(\cdot|s)$ on both sides yields $V = V_\tau^\pi$. Thus, V is a value function. By Lemma 5 in Lecture 7, the condition also means

$$\pi(\cdot|s) = \arg\max_{\tilde{\pi}(\cdot|s)} \mathbb{E}_{a \sim \tilde{\pi}(\cdot|s)} \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[r(s,a,s') + \gamma V(s') \right] - \tau \log \tilde{\pi}(a|s),$$

which implies $\mathcal{T}_\tau V(s) = V(s)$.

- This result essentially states that if $A_\tau^\pi(s,a) = 0, \forall\, s, a$, then π is the optimal policy. It is parallel to the non-regularized case: if $A^\pi(s,a) \leq 0, \forall\, s, a$, then π is an optimal policy.

Remark

- The optimal policy is unique with entropy regularization.
- It is evident that as $\tau \to 0$, $\pi_\tau^*(a|s) \to 0$ for $a \notin \arg\max Q^*(s,a)$.
- Since one has

$$\max_a Q_\tau^*(s,a) \le \tau \log\left(\left\|\exp\left(Q_\tau^*\left(s,\cdot\right)/\tau\right)\right\|_1\right) \le \tau \log|\mathcal{A}| + \max_a Q_\tau^*(s,a),$$

it is easy to see that $V_\tau^*(s) \to \max_a Q^*(s,a) = V^*(s)$ as $\tau \to 0$.

Soft Policy Iteration

Soft Policy Iteration:

$$\pi_{k+1}(\cdot|s) = \arg\max_{\pi} \mathcal{T}_\tau^\pi V_\tau^{\pi_k} = \frac{\exp\left(Q_\tau^{\pi_k}(s,\cdot)/\tau\right)}{\|\exp\left(Q_\tau^{\pi_k}(s,\cdot)/\tau\right)\|_1}.$$

- γ-rate convergence, with local quadratic convergence.

"Elementary Analysis of Policy Gradient Methods" by Jiacai Liu, Wenye Li, and Ke Wei, 2024.

Policy Gradient Theorem

Theorem (Policy Gradient Theorem)
Assume $\forall \theta, \sum_a \pi_\theta(a|s) = 1$ *for simplicity. One has*

$$\nabla V_\tau^{\pi_\theta}(\mu) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_\mu^{\pi_\theta}} \mathbb{E}_{a \sim \pi_\theta(\cdot|s)} \left[A_\tau^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s) \right].$$

Example: Softmax Parameterization

Lemma

For softmax parameterization,

$$\nabla_{\theta_s} V_\tau^{\pi_\theta}(\mu) = \frac{d_\mu^{\pi_\theta}(s)}{1-\gamma} \pi_\theta(\cdot|s) A_\tau^{\pi_\theta}(s, \cdot).$$

- Entropy softmax PG: in the parameter space,

$$\theta_{s,a}^{+} = \theta_{s,a} + \eta \frac{d_\mu^{\pi_\theta}(s)}{1-\gamma} \pi_\theta(a|s) A_\tau^{\pi_\theta}(s, a).$$

  In the policy space,

$$\pi_{s,a}^{+} \propto \pi_{s,a} \exp\left(\eta \frac{d_\mu^{\pi}(s)}{1-\gamma} \pi_\theta(a|s) A_\tau^{\pi_\theta}(s, a)\right).$$

Questions?