

MODEL

FREE

CONTROL

ACTION-VALUE FUNCTION

If we don't know $f(\cdot)$, knowing $V^*(\cdot)$ isn't enough to compute π^* .
Instead of learning $V(\cdot)$, RL methods learn $Q(\cdot)$.

Cost starting from state x , applying input u , and then following policy π

$$Q^\pi(x, u) = J_\pi(x, u) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = x, u_0 = u, u_{k>0} \sim \pi \right]$$

Can be decomposed as:

$$Q^\pi(x, u) = l(x, u) + \gamma Q^\pi(f(x, u), \pi(f(x, u)))$$

This is the **Bellman equation** for the Q function.

CONTROL LEARNING METHODS

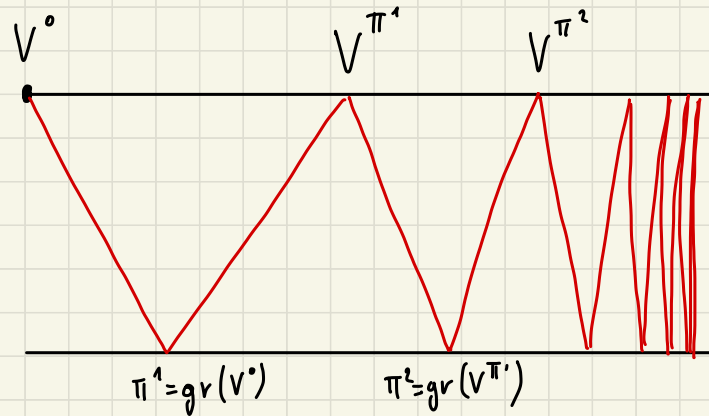
		LEARNED QUANTITIES	
		$V(x) / Q(x, u)$	$V(x) / Q(x, u) + \pi(x)$
Dynamics	Known	Value Iteration	Policy Iteration
	Unknown	Direct methods	Actor-Critic Methods

• Why learning Q instead of V ?

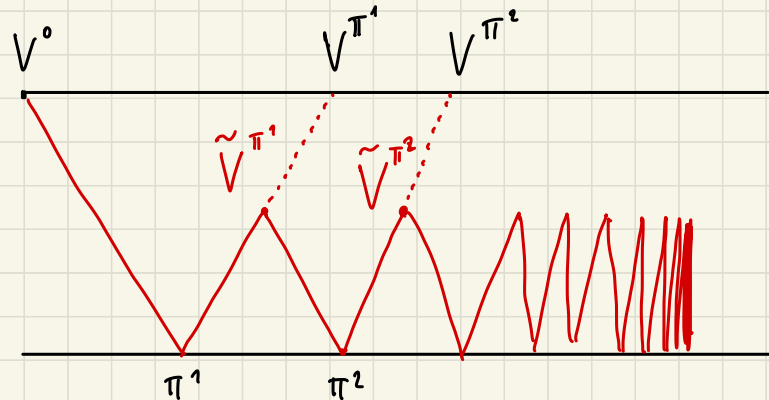
$$\pi(x) = \underset{u}{\operatorname{argmin}} l(x, u) + \gamma V(x') = \underset{u}{\operatorname{argmin}} Q(x, u)$$

ACTOR-CRITIC METHODS: GENERALIZED POLICY ITERATION

- Extension of Policy Iteration to unknown MDP
 - PI alternates "policy evaluation" and "policy improvement"
- Unknown $f(\cdot)$ $\Rightarrow$ use MC or TD to evaluate policy
- Exact policy evaluation can take infinitely many samples
- Solution: improve policy based on approximate value function



PI: exact policy evaluation



GPI: approximate policy evaluation

- The value function is called "critic" because it evaluates the policy, which is called "actor", in order to improve it
- The policy used to generate transitions is typically not the same that is evaluated and improved
 - The "behavior policy" mixes small amount of exploration into "target policy"
- GPl can generate policies worse than previous ones

SARSA: CRITIC

- Use TD(0) to learn Q from **on-policy** samples
- After each transition $(\underset{s}{x}, \underset{a}{u}, \underset{r}{l}, \underset{s'}{x'}, \underset{a'}{u'})$ update Q as:

$$\text{TD ERROR} \rightarrow \delta(Q) := l(x, u) + \gamma Q(x', u') - Q(x, u)$$

$$Q_{k+1}(x, u) := Q_k(x, u) + \alpha_k \delta(Q_k)$$

- If π was fixed $\Rightarrow$ SARSA = TD(0)
- As TD, SARSA can diverge in **off-policy** situations
- How to improve policy?

HOW TO CHOOSE CONTROL?

- What about being greedy w.r.t. Q ? $u(x) = \arg\min_w Q_u(x, w)$

EXAMPLE

- You can choose one of two doors in front of you

- Left door $\Rightarrow l = 0 \Rightarrow V(\text{left}) = 0$
- Right door $\Rightarrow l = -1 \Rightarrow V(\text{right}) = -1$
- Right door $\Rightarrow l = -2 \Rightarrow V(\text{right}) = -2$

Are you sure the right door is the best one?

SARSA: CONTROL

ϵ -GREEDY STRATEGY:

- Choose random control with probability ϵ
- Choose greedy control with probability $1-\epsilon$
- Simple way to ensure exploration
- ϵ can change over time (typically decreasing to 0)

GREEDY PLUS NOISE

$$\pi(x) = \arg \min_u Q(x, u)$$

$$u = \pi(x) + \mathcal{N}(0, \Sigma)$$

SARSA: CONVERGENCE

- Use ϵ -greedy as behavior policy to ensure exploration
- SARSA converges to $Q^*(x, a)$ if:
 - ① Greedy in the Limit with Infinite Exploration (GLIE) policies:
 - All state-action pairs visited infinitely many times
 - Policy converges to greedy policy (e.g. $\epsilon_n = \frac{1}{n}$)
 - ② Robbins-Monro sequence of step size:

$$\sum_{n=1}^{\infty} \alpha_n = \infty$$

$$\sum_{n=1}^{\infty} \alpha_n^2 < \infty$$

What if we want to learn off-policy?

ON-POLICY vs OFF-POLICY LEARNING

- On-policy (e.g. SARSA)
 - learn policy π from experience sampled from π
 - behavior policy $\approx$ target policy (e.g. ϵ -greedy, with ϵ small)
 - limited exploration $\Rightarrow$ low sample efficiency
- Off-policy
 - learn policy π from experience sampled from μ
 - behavior policy μ can be arbitrarily different from target policy π
 - higher sample efficiency

DIRECT METHODS: Q-LEARNING

- Iteratively update an estimate $Q_n(x, u)$ of $Q^*(x, u)$
- After observing transition (x, u, l, x') update with:

$$\delta(Q) := l + \gamma \min_{u' \in U} Q(x', u') - Q(x, u)$$

$$Q_{n+1}(x, u) := Q_n(x, u) + \alpha_n \delta(Q_n)$$

- Same as SARSA, but uses $\min_{u'} Q(x', u')$ instead of $Q(x', u')$
- Error $\delta(Q)$ is independent of behavior policy, which is the one that generates $u' \Rightarrow$ OFF-POLICY
- Target policy is: $\pi(x) = \arg \min_u Q(x, u)$

Q-LEARNING: CONVERGENCE

need exploration!
↓

- At stochastic equilibrium, $V(x, u)$ visited infinitely often

$$E[\delta(Q)] = 0 = TQ - Q \Rightarrow Q = Q^* \Leftarrow \text{Unique fixed point of } T$$

- Q_n converges when appropriate learning rates are used (Robbins-Monro conditions) and sufficient exploration is ensured

SARSA

INPUTS: Q, ϵ

For $k = 0 \dots K$

$x \leftarrow \text{env.reset}()$

$u \leftarrow \text{EpsGreedy}(Q, x, \epsilon)$

For $t = 0 \dots N$

$x', l \leftarrow \text{env.step}(u)$

$u' \leftarrow \text{EpsGreedy}(Q, x', \epsilon)$

$Q^{\text{target}} \leftarrow l + \gamma Q[x', u']$

$Q[x, u] += \alpha (Q^{\text{target}} - Q[x, u])$

$x, u \leftarrow x', u'$

$\epsilon \leftarrow \text{updateEpsilon}(\epsilon)$

Q-LEARNING

INPUTS: Q , behavior policy π_B

For $k = 0 \dots K$

$x \leftarrow \text{env.reset}()$

For $t = 0 \dots N$

$u \leftarrow \pi_B(x)$

$x', l \leftarrow \text{env.step}(u)$

$Q^{\text{target}} \leftarrow l + \gamma \min_w (Q[x', w])$

$Q[x, u] += \alpha (Q^{\text{target}} - Q[x, u])$

$x \leftarrow x'$

SARSA

VS

Q-LEARNING

- On-policy
- Learn Q of policy you actually execute, including exploration
- Know the policy will some times explore and plans accordingly
- Behavior policy = Target policy
 $\Rightarrow$ needs to ensure both exploration and exploitation

- Off-policy
- Learn Q of optimal greedy policy, while executing another exploratory policy
- Pretend the policy will be greedy in the future
- Behavior policy just needs to ensure exploration

SUMMARY

D P

Policy Evaluation

$$V(x) = l + \gamma V(x')$$

Policy Iteration

$$V(x) = l + \gamma V(x')$$

$$\pi(x) = \underset{u}{\operatorname{argmin}} l + \gamma V(x'(u))$$

Value Iteration

$$V(x) = \min_u l + \gamma V(x'(u))$$

T D

TD-Learning

$$V(x) \stackrel{d}{\leftarrow} l + \gamma V(x')$$

SARSA

$$Q(x, u) \stackrel{d}{\leftarrow} l + \gamma Q(x', u')$$

$$\pi(x) = \begin{cases} \underset{u}{\operatorname{argmin}} Q(x, u) & \text{with prob. } 1-\epsilon \\ \text{random} & \text{with prob. } \epsilon \end{cases}$$

Q-Learning

$$Q(x, u) \stackrel{d}{\leftarrow} l + \gamma \min_{u'} Q(x', u')$$