Q-Learning

Recap: Bellman Optimality

Bellman Optimality

$$Q^\star(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \max_{a'} Q^\star(s',a'), \forall s, a$$

VI: An iterative approach for estimating $Q^\star$

$$Q \Leftarrow \mathcal{T} Q$$

$$(\mathcal{T}Q)(s,a) = r(s,a) + \gamma \cdot \mathbb{E}_{s' \sim P(\cdot|s,a)} \max_{a'} Q(s',a')$$

Recap: Bellman Optimality

Bellman Optimality

$$Q^\star(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \max_{a'} Q^\star(s',a'), \forall s, a$$

VI: An iterative approach for estimating $Q^\star$

$$Q \Leftarrow \mathcal{T} Q \qquad P$$

1. Need to know the transition
2. Only works for discrete small MDPs

Recap: Bellman Optimality

Q: if there is some $Q(s, a)$, such that the following holds:

$$Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \max_{a'} Q(s',a'), \forall s, a$$

is this $Q = Q^\star$?

$\|Q(s,a) - (r(s,a) + \gamma \mathbb{E}_{s'} \max_{a'} Q(s',a'))\| = 0$

$Q^* = TQ^*$

$Q = TQ$

$\cdot\, Q^* \xrightarrow{T} Q^* \xrightarrow{T} Q^* \rightarrow \ldots$

$\cdot\, Q \xrightarrow{T} Q \xrightarrow{T} Q \rightarrow \ldots$

Today

V^{π}

Given MDP $\mathscr{M} = (S, A, r, P, \gamma)$,

how to estimate $Q^\star(s, a), \forall s$ WITHOUT knowing P
i.e., how to learn $Q^\star$ (thus, $\pi^\star$) from experience

$Q \rightarrow Q^*$

$\pi \leftarrow \arg\max_a Q(s,a)$

Motivation

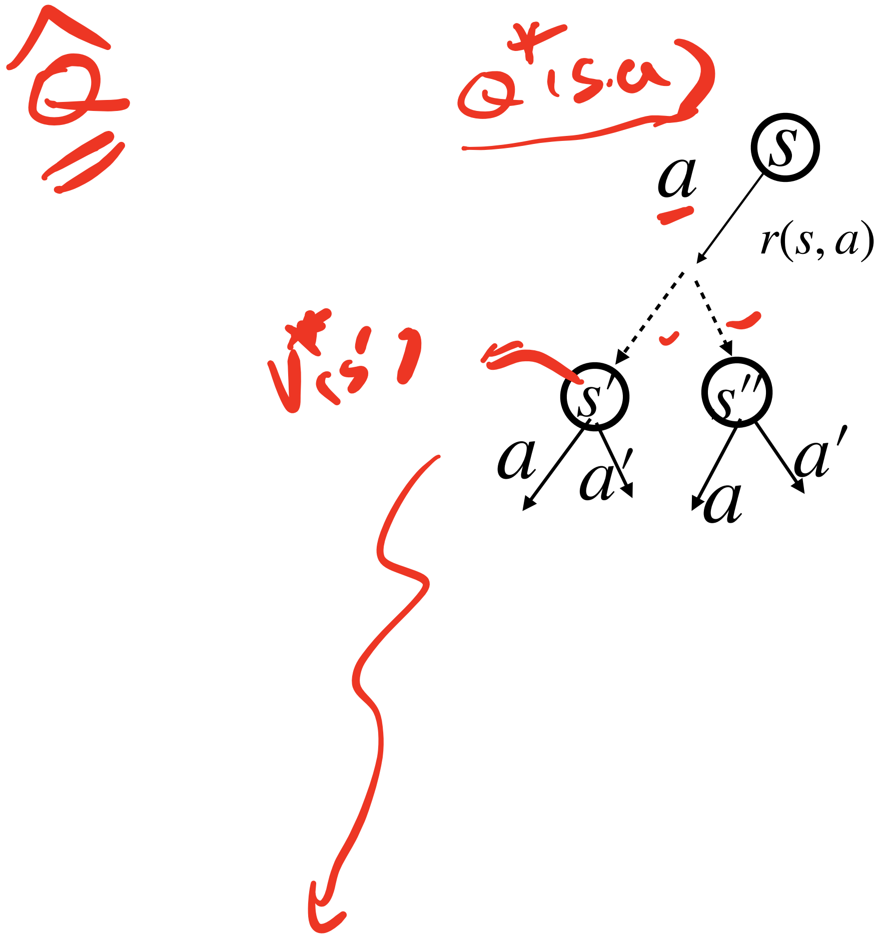
Computing a near-optimal policy to achieve the long-term goals w/o knowing or explicitly modeling the world

Outline:

1. Q Learning

2. Revisit TD: Off-policy TD Learning

Q Learning

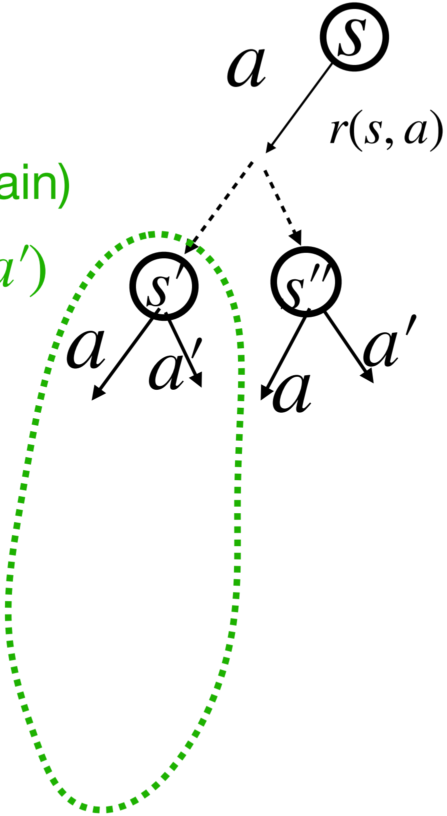


Q Learning

(Bootstrapping again)

$\max_{a'} \hat{Q}(s', a')$

$\approx V^*(s')$



Q Learning

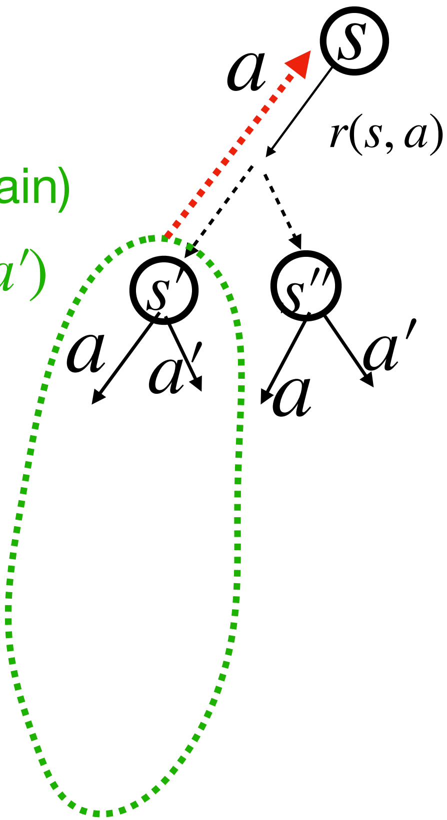
$$Q^\star(s,a) \approx \underbrace{\underline{r(s,a)} + \gamma \underline{\max_{a'} \hat{Q}(s',a')}}_{\text{target}}$$

$\approx \underline{V^*(s')}$

$= \max_{a'} Q^*(s',a')$

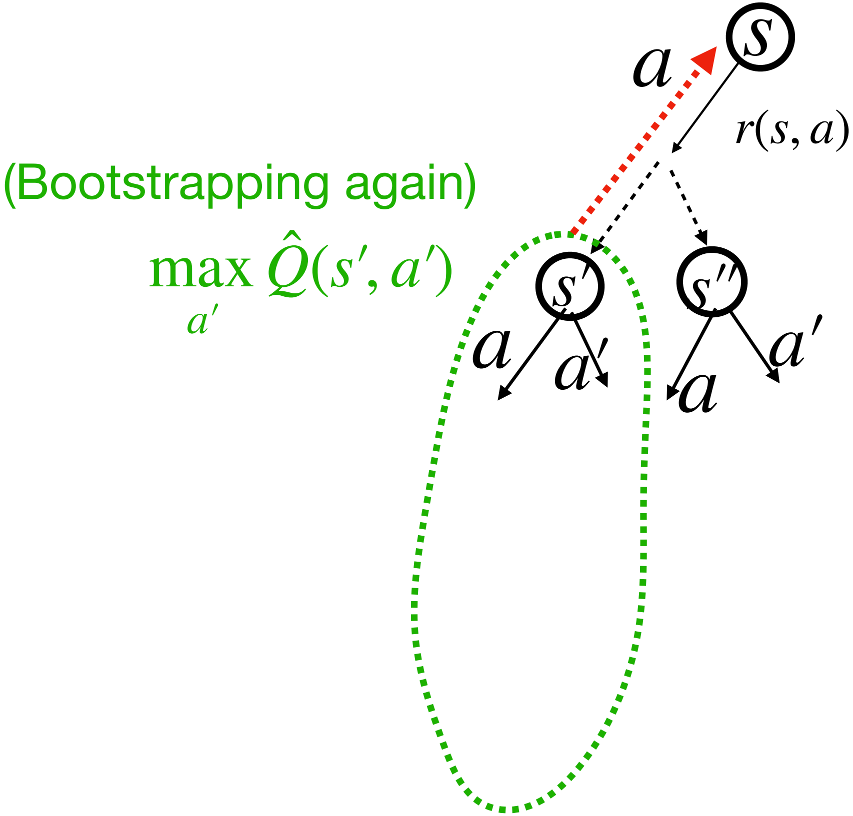
(Bootstrapping again)

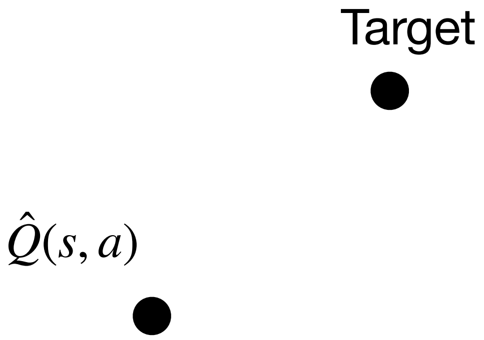
$\max_{a'} \hat{Q}(s',a')$



Q Learning

$$Q^\star(s,a) \approx \underbrace{r(s,a) + \gamma \max_{a'} \hat{Q}(s',a')}_{\text{target}}$$

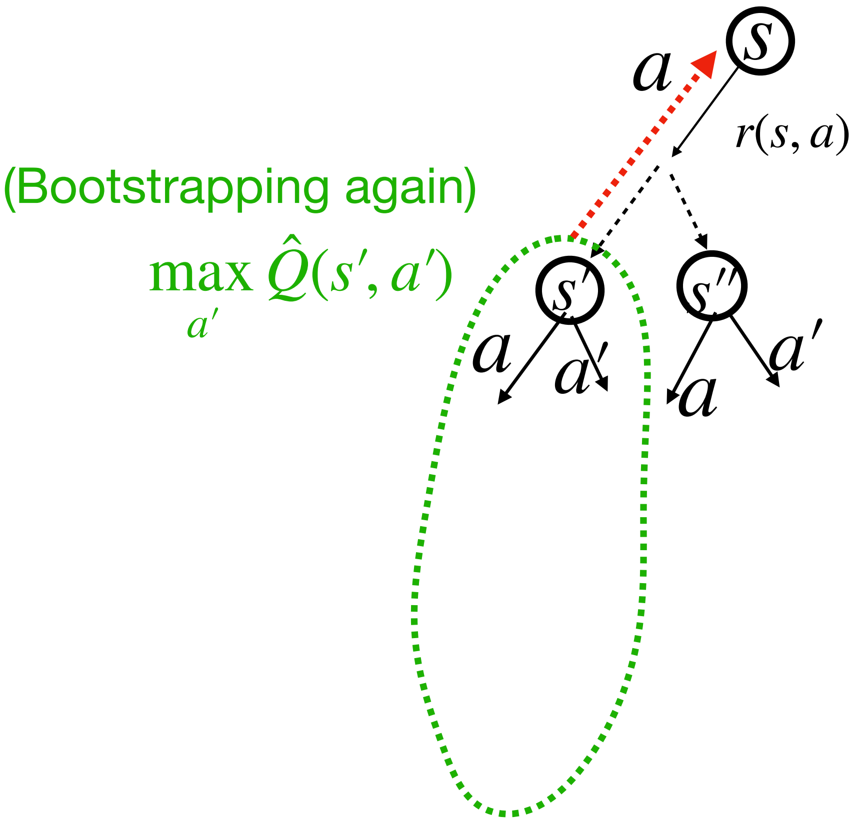


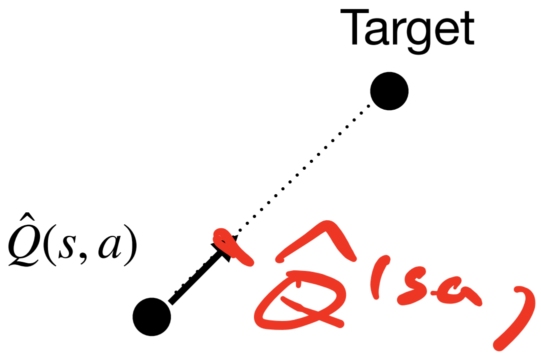


Q Learning

$$Q^\star(s,a) \approx \underbrace{r(s,a) + \gamma \max_{a'} \hat{Q}(s',a')}_{\text{target}}$$

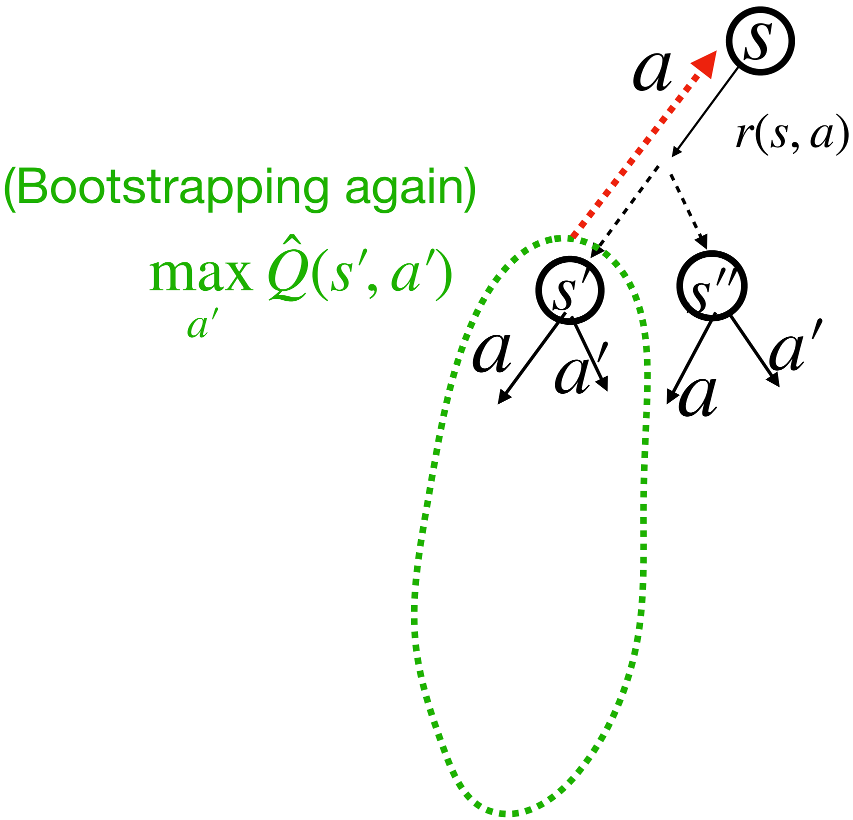
$\neq Q^\star(s,a')$

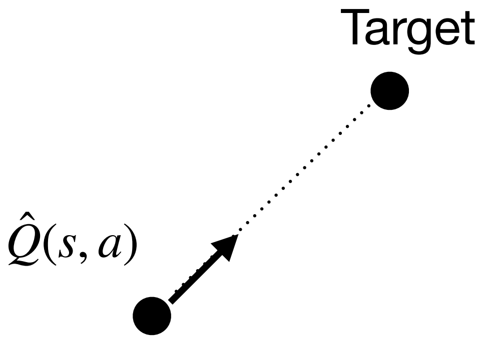




Q Learning

$$Q^\star(s,a) \approx \underbrace{r(s,a) + \gamma \max_{a'} \hat{Q}(s',a')}_{\text{target}}$$





Q-learning update: move to the target with a small step

V^π Q^π

Q Learning

Q^* $\arg\max_a Q^*(s,a)$

Given a **one-step transition** (s, a, r, s') where $r = r(s, a)$, $s' \sim P(\cdot \mid s, a)$:

$a \sim \pi(\cdot \mid s)$

Q-learning updates the guess at (s, a) as follows:

$$\hat{Q}(s,a) \Leftarrow \hat{Q}(s,a) + \eta \left(r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right)$$

Q-Target
(Constructed via Bootstrapping!)

Q Learning

How to collect data?

<u>Choice one</u>: trust current estimator $\hat{Q}$, always use $\arg\max_a \hat{Q}(s, a)$

Q Learning

How to collect data?

<u>Choice one</u>: trust current estimator $\hat{Q}$, always use $\arg\max_a \hat{Q}(s, a)$

Issue: cannot explore (i.e., need to try something that hasn't been tried)

Q Learning

How to collect data?

<u>Choice one</u>: trust current estimator $\hat{Q}$, always use $\arg\max_a \hat{Q}(s, a)$

Issue: cannot explore (i.e., need to try something that hasn't been tried)

<u>Choice two (quite effective in practice)</u>: ϵ**-greedy**

$\epsilon \in (0,1)$

$\hat{Q}^\star(s,a) = w^\top \binom{s}{a}$

$\hat{Q}(s, a_1) = 100$

$\hat{Q}(s, a_2) = 0.1$

Q Learning

How to collect data?

<u>**Choice one**</u>: trust current estimator $\hat{Q}$, always use $\arg\max_a \hat{Q}(s,a)$

Issue: cannot explore (i.e., need to try something that hasn't been tried)

<u>**Choice two (quite effective in practice)**</u>: **ϵ-greedy**

W/ prob ϵ, select action uniform randomly ← Explore

W/ prob $1-\epsilon$, select greedy action $\arg\max_a \hat{Q}(s,a)$ ← Exploitation

Choice three: $\pi(a|s) = \exp(\hat{Q}(s,a)) \Big/ \sum_{a'} \exp(\hat{Q}(s,a'))$

Q ~~TD~~ Learning

Initialize $\hat{Q}(s, a) = 0, \forall s, a$. Set initial state $s \in \mathcal{S}$

While True:

TD Learning

Initialize $\hat{Q}(s,a) = 0, \forall s, a$. Set initial state $s \in \mathcal{S}$

While True:

(s, a, r, s')

Take action a based on ϵ-greedy of $\hat{Q}$, get reward r and next state $s' \sim P(\cdot \mid s, a)$

TD Learning

Initialize $\hat{Q}(s, a) = 0, \forall s, a$. Set initial state $s \in \mathcal{S}$

While True:

Take action a based on ϵ-greedy of $\hat{Q}$, get reward r and next state $s' \sim P(\cdot \mid s, a)$

Form Q-target $r + \gamma \max_{a'} \hat{Q}(s', a')$ ← Bootstrapping

TD Learning

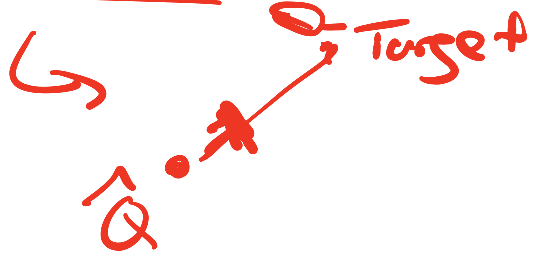
Initialize $\hat{Q}(s,a) = 0, \forall s, a$. Set initial state $s \in \mathcal{S}$

While True:

Take action a based on ϵ-greedy of $\hat{Q}$, get reward r and next state $s' \sim P(\cdot \mid s, a)$

Form Q-target $r + \gamma \max_{a'} \hat{Q}(s', a')$

Update for s, a: $\hat{Q}(s,a) \Leftarrow \hat{Q}(s,a) + \eta \left(r + \gamma \max_{a'} \hat{Q}(s', a') - \hat{Q}(s,a) \right)$



TD Learning

Initialize $\hat{Q}(s, a) = 0, \forall s, a$. Set initial state $s \in \mathcal{S}$

While True:

Take action a based on ϵ-greedy of $\hat{Q}$, get reward r and next state $s' \sim P(\cdot \mid s, a)$

Form Q-target $r + \gamma \max_{a'} \hat{Q}(s', a')$

Update for s, a: $\hat{Q}(s, a) \Leftarrow \hat{Q}(s, a) + \eta \left(r + \gamma \max_{a'} \hat{Q}(s', a') - \hat{Q}(s, a) \right)$

Set $s \Leftarrow s'$

Interpret Q-learning as “SGD” on Bellman error

Q-learning is not the usual SGD, i.e., it is **not** running SGD to minimize a fixed loss function

Q-learning may be interpreted as running SGD on an **evolving** loss function (Bellman error)

Interpret Q-learning as “SGD” on Bellman error

Q-learning is not the usual SGD, i.e., it is **not** running SGD to minimize a fixed loss function

Q-learning may be interpreted as running SGD on an **evolving** loss function (Bellman error)

$$\ell_{be}(\hat{Q}(s,a)) := \left(\hat{Q}(s,a) - y\right)^2, \text{ where } y = r(s,a) + \gamma \underline{\mathbb{E}_{s' \sim P(\cdot|s,a)}} \max_{a'} \hat{Q}(s',a')$$

$\Delta = 0$

$\hat{Q} = T\hat{Q}$

Bootstrapping

Interpret Q-learning as “SGD” on Bellman error

Q-learning is not the usual SGD, i.e., it is **not** running SGD to minimize a fixed loss function

Q-learning may be interpreted as running SGD on an **evolving** loss function (Bellman error)

$$\ell_{be}(\hat{Q}(s,a)) := \left(\hat{Q}(s,a) - y\right)^2, \text{ where } y = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \max_{a'} \hat{Q}(s', a')$$

This keeps changing as we learning

Interpret Q-learning as "SGD" on Bellman error

Q-learning is not the usual SGD, i.e., it is **not** running SGD to minimize a fixed loss function

Q-learning may be interpreted as running SGD on an **evolving** loss function (Bellman error)

$$\ell_{be}(\hat{Q}(s,a)) := \left(\hat{Q}(s,a) - y\right)^2, \text{ where } y = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \max_{a'} \hat{Q}(s',a')$$

$$\nabla \ell_{be}(x)\,|_{x=\hat{Q}(s,a)} := 2\left(\hat{Q}(s,a) - y\right)$$

$s' \sim P(\cdot|sa)$

This keeps changing as we learning

Interpret Q-learning as “SGD” on Bellman error

Q-learning is not the usual SGD, i.e., it is **not** running SGD to minimize a fixed loss function

Q-learning may be interpreted as running SGD on an **evolving** loss function (Bellman error)

$$\ell_{be}(\hat{Q}(s,a)) := \left(\hat{Q}(s,a) - y\right)^2, \text{ where } y = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \max_{a'} \hat{Q}(s',a')$$

This keeps changing as we learning

$$\nabla \ell_{be}(x)\,|_{x=\hat{Q}(s,a)} := 2\left(\hat{Q}(s,a) - y\right)$$

$$\widetilde{\nabla} \ell_{be}(x)\,|_{x=\hat{Q}(s,a)} := 2\left(\hat{Q}(s,a) - \left(r + \gamma \max_{a'} \hat{Q}(s',a')\right)\right)$$

Interpret Q-learning as “SGD” on Bellman error

Q-learning is not the usual SGD, i.e., it is **not** running SGD to minimize a fixed loss function

Q-learning may be interpreted as running SGD on an **evolving** loss function (Bellman error)

$$\ell_{be}(\hat{Q}(s,a)) := \left(\hat{Q}(s,a) - y\right)^2, \text{ where } y = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \max_{a'} \hat{Q}(s', a')$$

Unbiased estimate of y

This keeps changing as we learning

$$\nabla \ell_{be}(x)\,|_{x=\hat{Q}(s,a)} := 2\left(\hat{Q}(s,a) - y\right)$$

$$\widetilde{\nabla} \ell_{be}(x)\,|_{x=\hat{Q}(s,a)} := 2\left(\hat{Q}(s,a) - \left(r + \gamma \max_{a'} \hat{Q}(s', a')\right)\right)$$

Q-Target

$$\hat{Q}(s,a) - \eta \cdot \widetilde{\nabla} \ell_{BE}(x)\,\Big|_{x=\hat{Q}(s,a)}$$

Q Learning Theory

[Informal] Assume the ϵ-greedy strategy has non-trivial probability of visiting every state-action pair. Setting learning rate η properly, we will have:

$\frac{1}{\sqrt{n}}$ $\frac{1}{n}$

$$\hat{Q}(s, a) \to Q^{\star}(s, a), \forall s, a$$

when # of interactions approaches to ∞

(concrete convergence rates are known as well)

Demo: Q-learning on CartPole

Note: Cartpole's state is continuous, so we will need Q-learning w/ function approximation, e.g., neural network (we will get there very soon)

1. Does Q learning eventually learn a good policy

2. How does the ϵ affect the learning

Outline:

1. Q Learning

2. Revisit TD: Off-policy TD Learning

TD Learning

Given (s, a, r, s'), where $a \sim \pi(\cdot \mid s), s' \sim P(\cdot \mid s, a)$, TD updates:

TD-Error

$$\hat{V}^{\pi}(s) \Leftarrow \hat{V}^{\pi}(s) + \eta \left(\underbrace{r + \gamma \hat{V}^{\pi}(s')}_{\text{TD-Target}} - \hat{V}^{\pi}(s') \right)$$

On-policy: data is generated from the policy π itself

Off-policy: data is generated from policy π_b where $\pi_b \neq \pi$

TD Learning

Given (s, a, r, s'), where $a \sim \pi(\cdot \mid s), s' \sim P(\cdot \mid s, a)$, TD updates:

$$\hat{V}^{\pi}(s) \Leftarrow \hat{V}^{\pi}(s) + \eta \left(r + \gamma \hat{V}^{\pi}(s') - \hat{V}^{\pi}(s') \right)$$

On-policy: data is generated from the policy π itself

Off-policy: data is generated from policy π_b where $\pi_b \neq \pi$

Q: is Q-learning off-policy or on-policy?

Q-learning

$\arg\max_a Q^*$

Motivation for off-policy evaluation

Counterfactual: what would happen if I did something different?

π_b π_{new}

Off-policy TD Learning

$a \sim \pi_b(\cdot|s)$

Setting: data is generated by π_b, but we want to estimate V^π for some $\pi \neq \pi_b$

Key trick: importance weighting

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

BE $-$ π

$a \sim \pi_b$

$$= \mathbb{E}_{a \sim \pi_b(\cdot|s)} \frac{\pi(a|s)}{\pi_b(a|s)} (\sim \ldots)$$

$$= \sum_a \pi(a|s)(\cdots\cdots) = \sum_a \pi_b(a|s) \cdot \frac{\pi(a|s)}{\pi_b(a|s)} (\cdots\cdots)$$

Off-policy TD Learning

Setting: data is generated by π_b, but we want to estimate V^π for some $\pi \neq \pi_b$

Key trick: importance weighting

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

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

Off-policy TD Learning

Setting: data is generated by π_b, but we want to estimate V^π for some $\pi \neq \pi_b$

Key trick: importance weighting

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

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

Importance weight

Off-policy TD Learning

Setting: data is generated by π_b, but we want to estimate V^π for some $\pi \neq \pi_b$

Key trick: importance weighting

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

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

Now action is sampled from π_b

Importance weight

$a \sim \pi_b(\cdot|s)$

Off-policy TD Learning

Given (s, a, r, s'), where $a \sim \pi_b(\cdot \mid s)$, $s' \sim P(\cdot \mid s, a)$,

Off-policy TD updates as follows:

$$\hat{V}^{\pi}(s) \Leftarrow \hat{V}^{\pi}(s) + \eta \frac{\pi(a \mid s)}{\pi_b(a \mid s)} \left(r + \gamma \hat{V}^{\pi}(s') - V^{\pi}(s) \right)$$

Off-policy TD Learning

Given (s, a, r, s'), where $a \sim \pi_b(\cdot \mid s)$, $s' \sim P(\cdot \mid s, a)$,

Off-policy TD updates as follows:

$$\hat{V}^\pi(s) \Leftarrow \hat{V}^\pi(s) + \eta \frac{\pi(a \mid s)}{\pi_b(a \mid s)} \left(r + \gamma \hat{V}^\pi(s') - V^\pi(s) \right)$$

Case 1: $\pi(a \mid s)$ is large but $\pi_b(a \mid s)$ is small

$\frac{\pi(a|s)}{\pi_b(a|s)} \nearrow$

Off-policy TD Learning

Given (s, a, r, s'), where $a \sim \pi_b(\cdot \mid s), s' \sim P(\cdot \mid s, a)$,

Off-policy TD updates as follows:

$$\hat{V}^\pi(s) \Leftarrow \hat{V}^\pi(s) + \eta \frac{\pi(a \mid s)}{\pi_b(a \mid s)} \left(r + \gamma \hat{V}^\pi(s') - V^\pi(s) \right)$$

Case 1: $\pi(a \mid s)$ is large but $\pi_b(a \mid s)$ is small

Case 2: $\pi(a \mid s)$ is small but $\pi_b(a \mid s)$ is large

$\frac{\pi}{\pi_b} \simeq 0$

Off-policy TD Learning is SGD on TD loss

Given (s, a, r, s'), where $a \sim \pi_b(\cdot \mid s), s' \sim P(\cdot \mid s, a)$, Off-policy TD updates:

$$\hat{V}^\pi(s) \Leftarrow \hat{V}^\pi(s) + \eta \frac{\pi(a \mid s)}{\pi_b(a \mid s)} \left(r + \gamma \hat{V}^\pi(s') - V^\pi(s) \right)$$

Check if it is doing one-step SGD on the TD loss:

$$\ell_{td}(\hat{V}^\pi(s)) = \left(\hat{V}^\pi(s) - y \right)^2 \text{ where } y = \mathbb{E}_{a \sim \pi(\cdot|s)} \left(r + \gamma \mathbb{E}_{s' \sim P(s,a)} \hat{V}^\pi(s') \right)$$

The off-policy TD update is one-step SGD on ℓ_{td} (more in HW2)

Summary

Q-Learning: online algorithm that learns $Q^\star$ (bootstrapping)

Exploration & Exploitation tradeoff: ϵ-greedy is an effective heuristic

Off-policy policy evaluation: importance weighting
(also known as inverse probability weighting in causal inference)