Lecture 5: Value-based RL

Last time we saw:

Actor-critic methods.

$$\theta_{t+1} = \theta_t + \eta \, \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{i=0}^{T} \nabla_\theta \log \pi_\theta (a_i | s_i) \cdot \underbrace{A^\pi(s_i, a_i)}_{\text{advantage fn.}} \right]$$

$a \sim \pi_\theta(a|s)$

The advantage is defined as:

$$A^\pi(s_i, a_i) = \underbrace{r(s_i, a_i) + \gamma V^\pi(s_{i+1})}_{\text{this is the Q-value of } (s_i, a_i)} - V^\pi(s_i)$$

$$\Rightarrow \quad A^\pi(s_i, a_i) = \underbrace{Q^\pi(s_i, a_i)}_{\text{how good action } a_i \text{ is}} - \underbrace{V^\pi(s_i)}_{\text{avg. value of an action}}$$

Back to the RL algorithm skeleton

1. Collect data.
2. Estimate return (advantage) by evaluating the policy
3. Improve the policy

how do we do this?

A^π using rollouts from π

high variance

$R(s_i, a_i) - V_\phi(s_i)$

Value-based methods

"off-policy"

2. $\rightarrow$ estimate A^π as perfectly as possible
3. $\rightarrow$ update policy to argmax

$\mathcal{A} = \{a_1 \ldots a_N\}$

$$\boxed{\pi^*(s) \leftarrow \max_a A^\pi(s,a)}$$

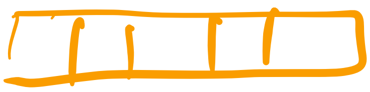
Skeleton of a value-based RL algo

1. Collect data in an env. using current policy π
   (+ some exploration) "noise"

2. Train $A^{\pi}(s,a) \Rightarrow$?? $A^{\pi}_{\phi}(s,a)$

3. update π as $\pi' := \mathbb{1}\{a = \arg\max_a A^{\pi}(s,a)\}$

$a_1 \; . \; . \; . \; a_N$



$A^{\pi}_{\phi}(s,a_1) \; . \; . \; . \; a_N$

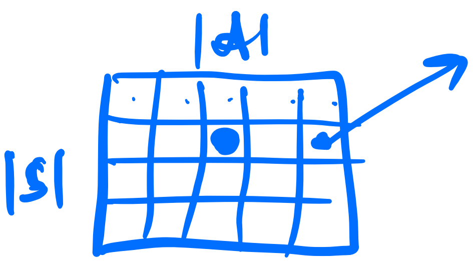
Theoretical algorithm:

"Dynamic programming"

$$\boxed{Q(s,a)} = r(s,a) + \gamma \, \mathbb{E}_{s' \sim p(s'|s,a)} [V(s')]$$

How can we implement this in a concrete algorithm?

$Q(s,a)$



$|S| = \text{finite}$, $|A| = \text{finite}$

Value iteration

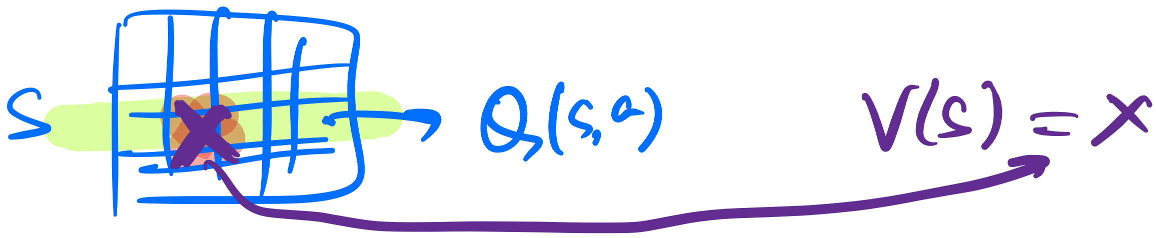
$V(s)$ $Q(s,a)$ $V(s)$

while converged:

① $Q(s,a) \leftarrow r(s,a) + \gamma E_{s'}[V(s')]$

② $V(s) \leftarrow \max_a Q(s,a)$

A what does this mean?



Fitted Q-iteration

Do this using neural networks!

$$Q(s,a) \leftarrow r(s,a) + \gamma E_{s'}[V(s')]$$

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

Train Q_θ to satisfy:

$$Q_\theta(s,a) \leftarrow r(s,a) + \gamma E_{s'}[\max_{a'} Q_\theta(s',a')]$$

θ_1 θ_0

How do I write this down as a loss fn.??

Draw inspiration from TD learning: using Q_θ

gradient descent

$$\theta_1 \leftarrow \min_\theta \mathbb{E}_{\substack{s,a \sim \text{somewhere} \\ s' \sim p(\cdot|s,a)}} \left[\left(Q_\theta(s,a) - \text{RHS} \right)^2 \right]$$

anything

target value

$Q_\theta(s,a)$

$$\text{RHS} :\Rightarrow \quad y(s,a) = r(s,a) + \gamma \max_{a'} Q(s',a')$$

$$(s, a, r, s') \qquad \sum_{(s,a,r,s')} \left(Q(s,a) - y \right)^2$$

This algorithm can use data from any policy!!

$$\mathbb{E}_{\substack{s,a \sim \text{somewhere} \\ s'}} [\text{ squared error }]$$

Important: This algorithm is not doing gradient descent!

$$\boxed{\text{FQI} \neq \min_\theta \mathcal{L}(\theta)}$$

References:

Martin Riedmiller. "Neural FQI" paper, 2009.

Fu*, Kumar* et. al. "Diagnosing Bottlenecks in Deep Q-learning Algos" ICML 2019.

Deep Q-Networks (DQN)

FQI w/ gradients.

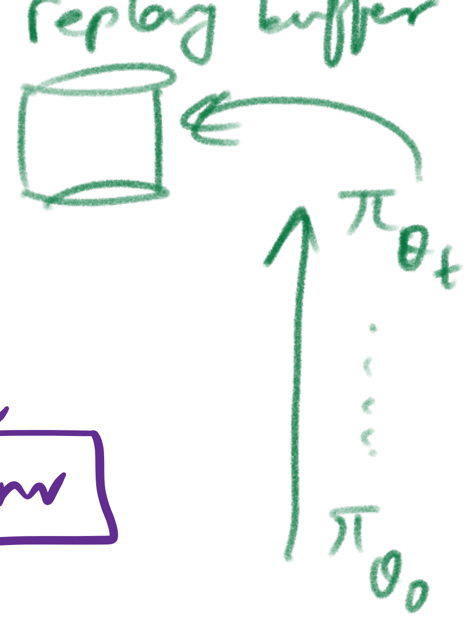
$$\text{TD-Error}(\theta) = \mathbb{E}_{s,a,s' \sim \mathcal{D}}\left[\left(Q_\theta(s,a) - y(s,a)\right)^2\right]$$

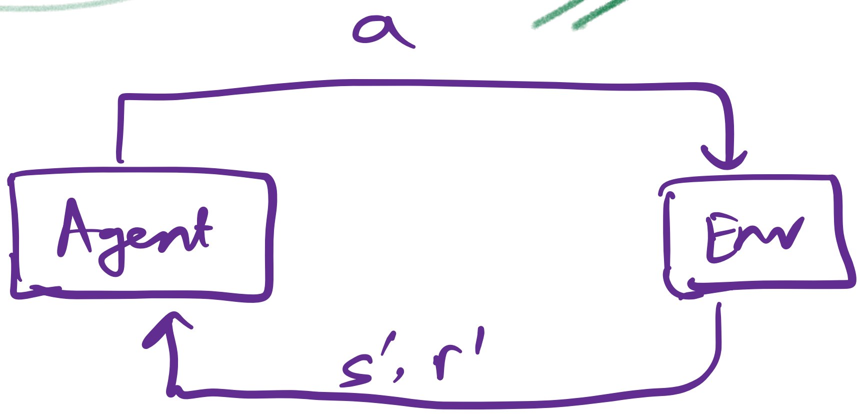
gradient descent

$$\theta_{t+1} \leftarrow \theta_t - \alpha \nabla_\theta \, \text{TD-Error}(\theta) \Big|_{\theta=\theta_t}$$

Problem with doing this on on-policy data??

1 Replay buffer:





store (s,a,r,s') in a dataset of states!

$a \sim \pi_{\theta_0} \,|\, \pi_{\theta_1} \,|\, \cdots \,\pi_{\theta_t}$

1) collect data using π_t
2) update Q_{θ_t} using $(s,a,r,s') \in \mathcal{D}$

2 Target networks:

↳ do one gradient step on (s,a,r,s')

↳ update target network.

Soft targets:

$$\min_{\theta_t} \left(Q_{\theta_t}(s,a) - y_{\theta_{t-1}}(s,a)\right)^2$$

few gradient updates

& calculate targets

& refresh targets

Hard targets: $\underline{Q_\theta}$ "target": $Q_{\bar{\theta}}$

1) K gradient updates on TD-error using targets from $\underline{Q_{\bar{\theta}}}$

2) update: $\bar{\theta} \leftarrow \theta_t$

$K=1 \rightarrow$ instabilities / correlations b/w target & Q-function

Challenges w/ DQNs:

① Best way to sample from replay buffers

② Noise & stochasticity

"the overestimation problem"