

Value Function Methods

CS 294-112: Deep Reinforcement Learning

Sergey Levine

Class Notes

1. Homework 2 is due in one week
2. Remember to start forming final project groups and writing your proposal!
 - Proposal due 9/26, in two weeks
 - Submission instructions coming soon

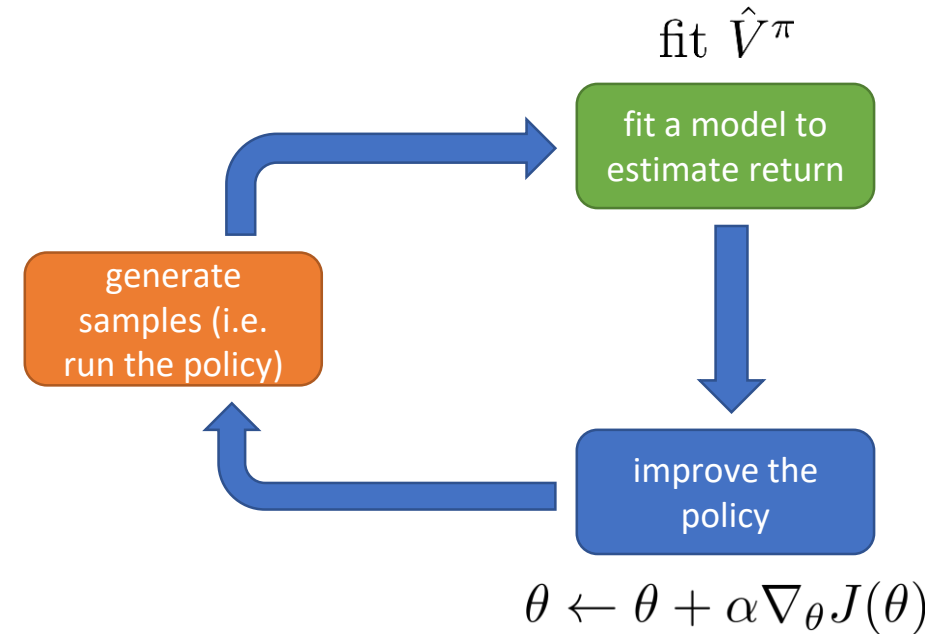
Today's Lecture

1. What if we just use a critic, without an actor?
 2. Extracting a policy from a value function
 3. The Q-learning algorithm
 4. Extensions: continuous actions, improvements
- Goals:
 - Understand how value functions give rise to policies
 - Understand the Q-learning algorithm
 - Understand practical considerations for Q-learning

Recap: actor-critic

batch actor-critic algorithm:

1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
2. fit $\hat{V}_\phi^\pi(\mathbf{s})$ to sampled reward sums
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



Can we omit policy gradient completely?

$A^\pi(\mathbf{s}_t, \mathbf{a}_t)$: how much better is \mathbf{a}_t than the average action according to π

$\arg \max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t)$: best action from \mathbf{s}_t , if we then follow π

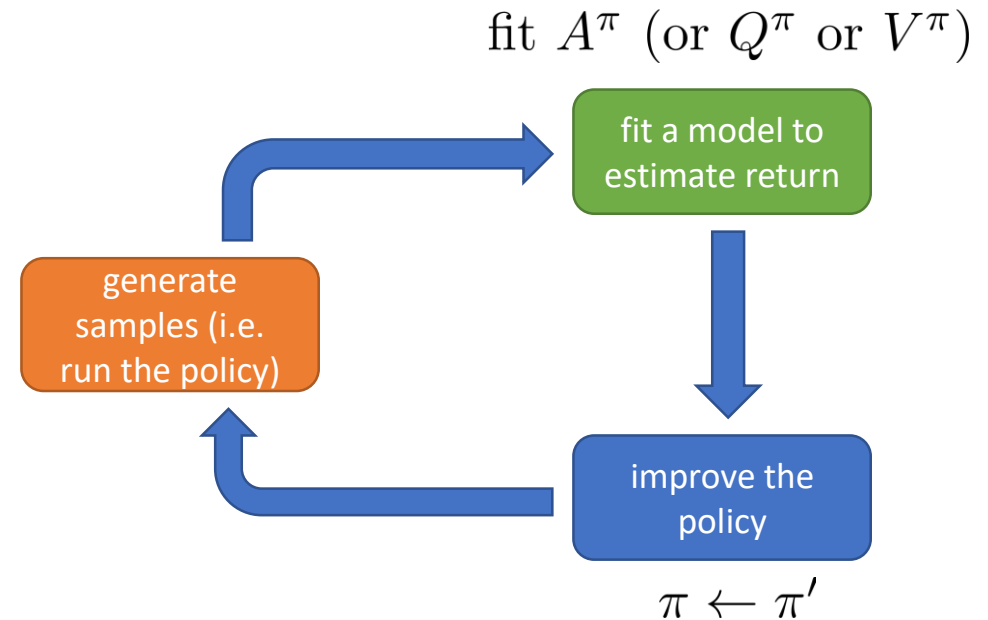
at *least* as good as any $\mathbf{a}_t \sim \pi(\mathbf{a}_t | \mathbf{s}_t)$

regardless of what $\pi(\mathbf{a}_t | \mathbf{s}_t)$ is!

forget policies, let's just do this!

$$\pi'(\mathbf{a}_t | \mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg \max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 & \text{otherwise} \end{cases}$$

as good as π
(probably better)



Policy iteration

High level idea:

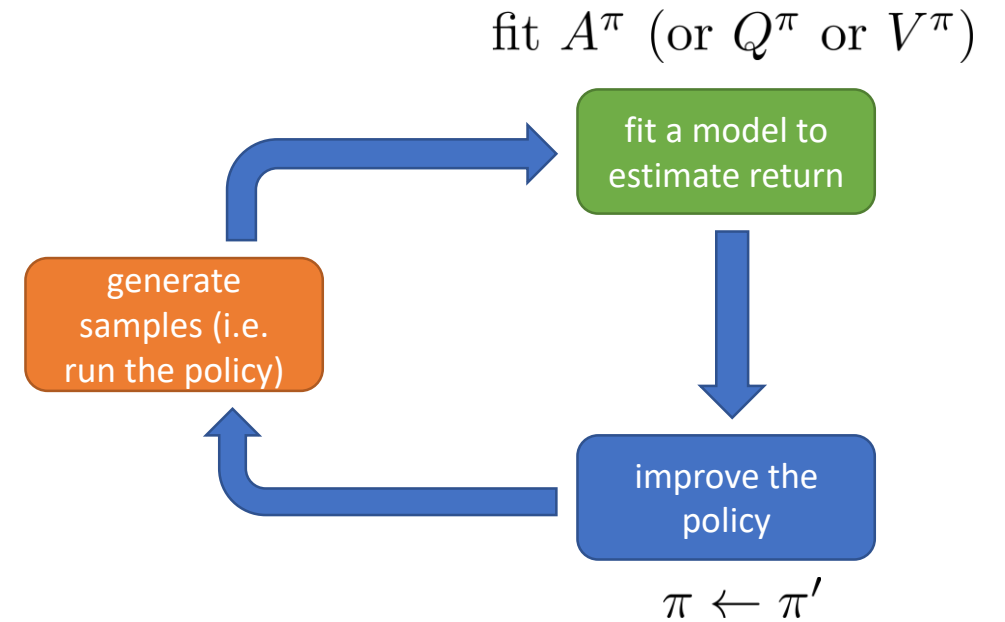
policy iteration algorithm:

- ↻ 1. evaluate $A^\pi(\mathbf{s}, \mathbf{a})$ ← how to do this?
2. set $\pi \leftarrow \pi'$

$$\pi'(\mathbf{a}_t | \mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg \max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 & \text{otherwise} \end{cases}$$

as before: $A^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma E[V^\pi(\mathbf{s}')] - V^\pi(\mathbf{s})$

let's evaluate $V^\pi(\mathbf{s})!$



Dynamic programming

Let's assume we know $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$, and \mathbf{s} and \mathbf{a} are both discrete (and small)

0.2	0.3	0.4	0.3
0.3	0.3	0.5	0.3
0.4	0.4	0.6	0.4
0.5	0.5	0.7	0.5

16 states, 4 actions per state

can store full $V^\pi(\mathbf{s})$ in a table!

\mathcal{T} is $16 \times 16 \times 4$ tensor

bootstrapped update: $V^\pi(\mathbf{s}) \leftarrow E_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} [r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} [V^\pi(\mathbf{s}')]]$

↑
just use the current estimate here

$\pi'(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg \max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 & \text{otherwise} \end{cases} \longrightarrow$ deterministic policy $\pi(\mathbf{s}) = \mathbf{a}$

simplified: $V^\pi(\mathbf{s}) \leftarrow r(\mathbf{s}, \pi(\mathbf{s})) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \pi(\mathbf{s}))} [V^\pi(\mathbf{s}')]]$

Policy iteration with dynamic programming

policy iteration:

- 1. evaluate $V^\pi(\mathbf{s})$
- 2. set $\pi \leftarrow \pi'$

$$\pi'(\mathbf{a}_t | \mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg \max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 & \text{otherwise} \end{cases}$$

policy evaluation:

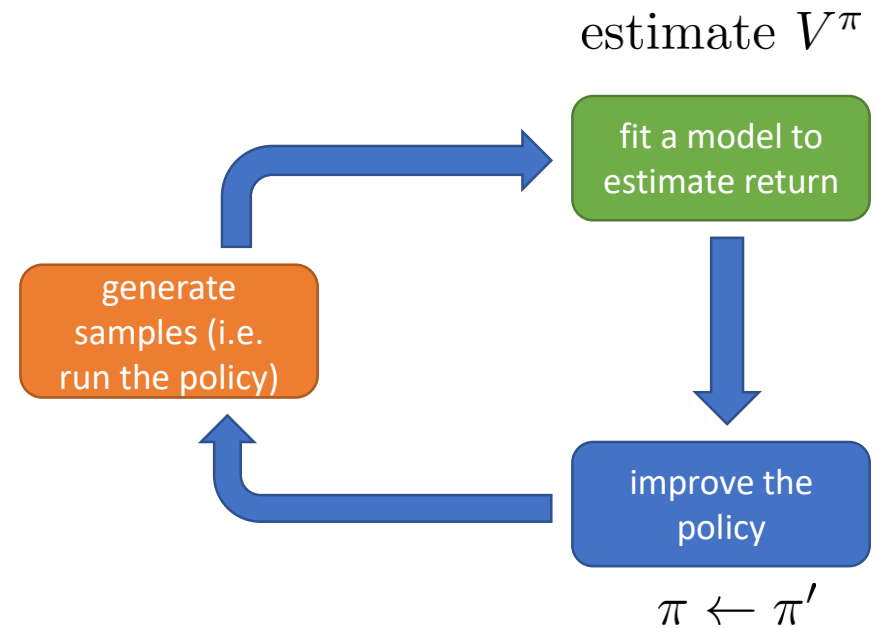
$$V^\pi(\mathbf{s}) \leftarrow r(\mathbf{s}, \pi(\mathbf{s})) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}' | \mathbf{s}, \pi(\mathbf{s}))} [V^\pi(\mathbf{s}')]]$$

0.2	0.3	0.4	0.3
0.3	0.3	0.5	0.3
0.4	0.4	0.6	0.4
0.5	0.5	0.7	0.5

16 states, 4 actions per state

can store full $V^\pi(\mathbf{s})$ in a table!

\mathcal{T} is $16 \times 16 \times 4$ tensor



Even simpler dynamic programming

$$\pi'(\mathbf{a}_t | \mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg \max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 & \text{otherwise} \end{cases}$$

$$A^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma E[V^\pi(\mathbf{s}')] - V^\pi(\mathbf{s})$$

$$\arg \max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t) = \arg \max_{\mathbf{a}_t} Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$$

$$Q^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma E[V^\pi(\mathbf{s}')] \quad (\text{a bit simpler})$$

skip the policy and compute values directly!

value iteration algorithm:

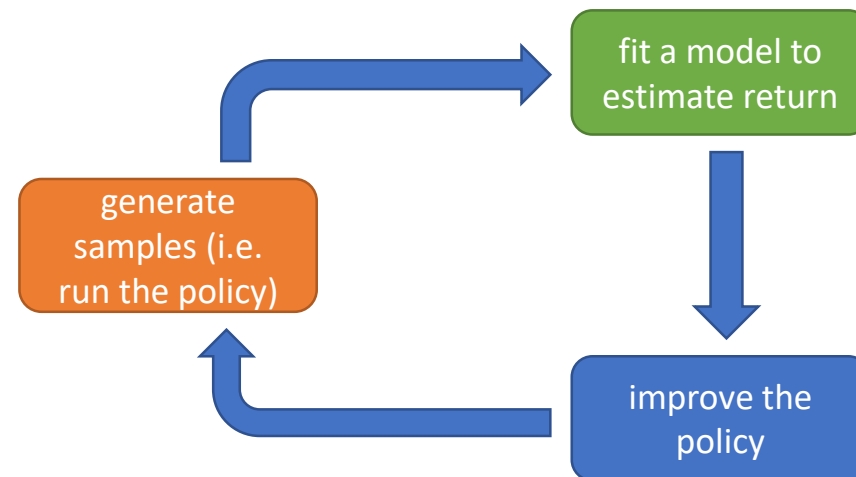
1. set $Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E[V(\mathbf{s}')]$
2. set $V(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$

		a		
s	Q(s, a)		Q(s, a) Q(s, a)	
	Q(s, a)	Q(s, a)		Q(s, a)
		Q(s, a)	Q(s, a)	Q(s, a)
	Q(s, a)	Q(s, a)	Q(s, a)	
	Q(s, a)		Q(s, a)	Q(s, a)
	Q(s, a)	Q(s, a)		Q(s, a)

$\arg \max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a}) \rightarrow \text{policy}$

approximates the new value!

$$Q^\pi(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}' | \mathbf{s}, \mathbf{a})} [V^\pi(\mathbf{s}')]]$$



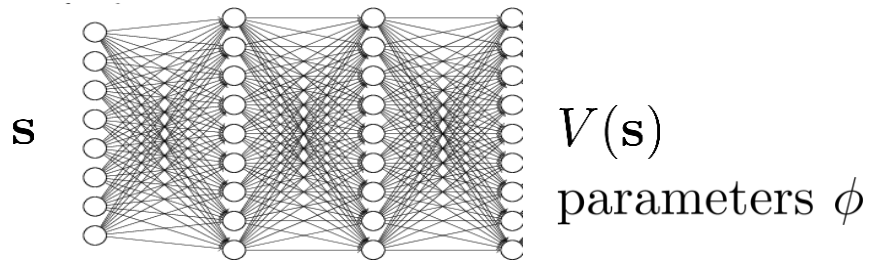
$$V^\pi(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})$$

Fitted value iteration

how do we represent $V(\mathbf{s})$?

big table, one entry for each discrete \mathbf{s}

neural net function $V : \mathcal{S} \rightarrow \mathbb{R}$



$$\mathbf{s} = 0 : V(\mathbf{s}) = 0.2$$

$$\mathbf{s} = 1 : V(\mathbf{s}) = 0.3$$

$$\mathbf{s} = 2 : V(\mathbf{s}) = 0.5$$



curse of dimensionality

$$|\mathcal{S}| = (255^3)^{200 \times 200}$$

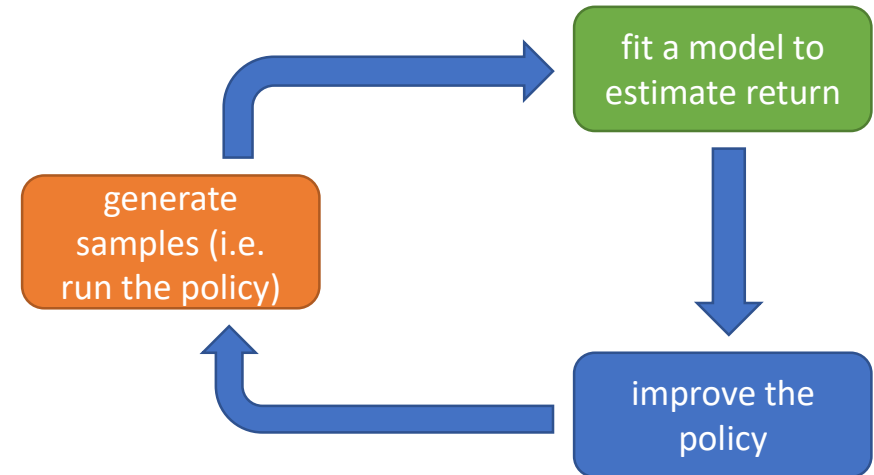
(more than atoms in the universe)

$$Q^\pi(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} [V^\pi(\mathbf{s}')]]$$

$$\mathcal{L}(\phi) = \frac{1}{2} \left\| V_\phi(\mathbf{s}) - \max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a}) \right\|^2$$

fitted value iteration algorithm:

1. set $\mathbf{y}_i \leftarrow \max_{\mathbf{a}_i} (r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)])$
2. set $\phi \leftarrow \arg \min_{\phi} \frac{1}{2} \sum_i \|V_\phi(\mathbf{s}_i) - \mathbf{y}_i\|^2$



$$V^\pi(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})$$

What if we don't know the transition dynamics?

fitted value iteration algorithm:

1. set $\mathbf{y}_i \leftarrow \max_{\mathbf{a}_i} (r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)])$
 2. set $\phi \leftarrow \arg \min_{\phi} \frac{1}{2} \sum_i \|V_\phi(\mathbf{s}_i) - \mathbf{y}_i\|^2$
- need to know outcomes for different actions!

Back to policy iteration...

policy iteration:

1. evaluate $Q^\pi(\mathbf{s}, \mathbf{a})$
2. set $\pi \leftarrow \pi'$

$$\pi'(\mathbf{a}_t | \mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg \max_{\mathbf{a}_t} Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 & \text{otherwise} \end{cases}$$

policy evaluation:

1. ~~$V^\pi(\mathbf{s}) \leftarrow r(\mathbf{s}, \pi(\mathbf{s})) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}' | \mathbf{s}, \pi(\mathbf{s}))} [V^\pi(\mathbf{s}')]$~~
2. $Q^\pi(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}' | \mathbf{s}, \mathbf{a})} [Q^\pi(\mathbf{s}', \pi(\mathbf{s}'))]$

can fit this using samples

Can we do the “max” trick again?

policy iteration:



1. evaluate $V^\pi(\mathbf{s})$
2. set $\pi \leftarrow \pi'$



fitted value iteration algorithm:



1. set $\mathbf{y}_i \leftarrow \max_{\mathbf{a}_i} (r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)])$
2. set $\phi \leftarrow \arg \min_\phi \frac{1}{2} \sum_i \|V_\phi(\mathbf{s}_i) - \mathbf{y}_i\|^2$

forget policy, compute value directly

can we do this with Q-values **also**, without knowing the transitions?

fitted Q iteration algorithm:



1. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)]$ ← approximate $E[V(\mathbf{s}'_i)] \approx \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
2. set $\phi \leftarrow \arg \min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

doesn't require simulation of actions!

+ works even for off-policy samples (unlike actor-critic)

+ only one network, no high-variance policy gradient

- no convergence guarantees for non-linear function approximation (more on this later)

Fitted Q-iteration

full fitted Q-iteration algorithm:

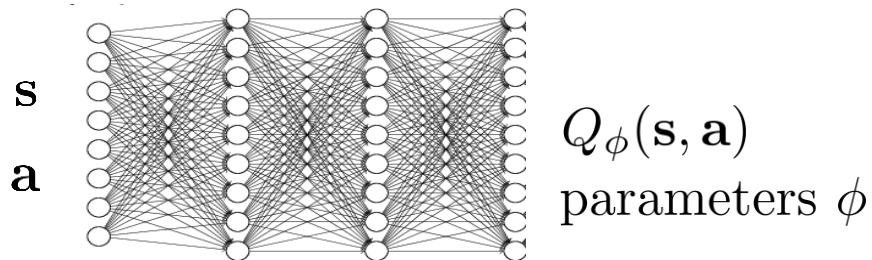
1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy
2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. set $\phi \leftarrow \arg \min_{\phi} \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

parameters

dataset size N , collection policy

iterations K

gradient steps S



Why is this algorithm off-policy?

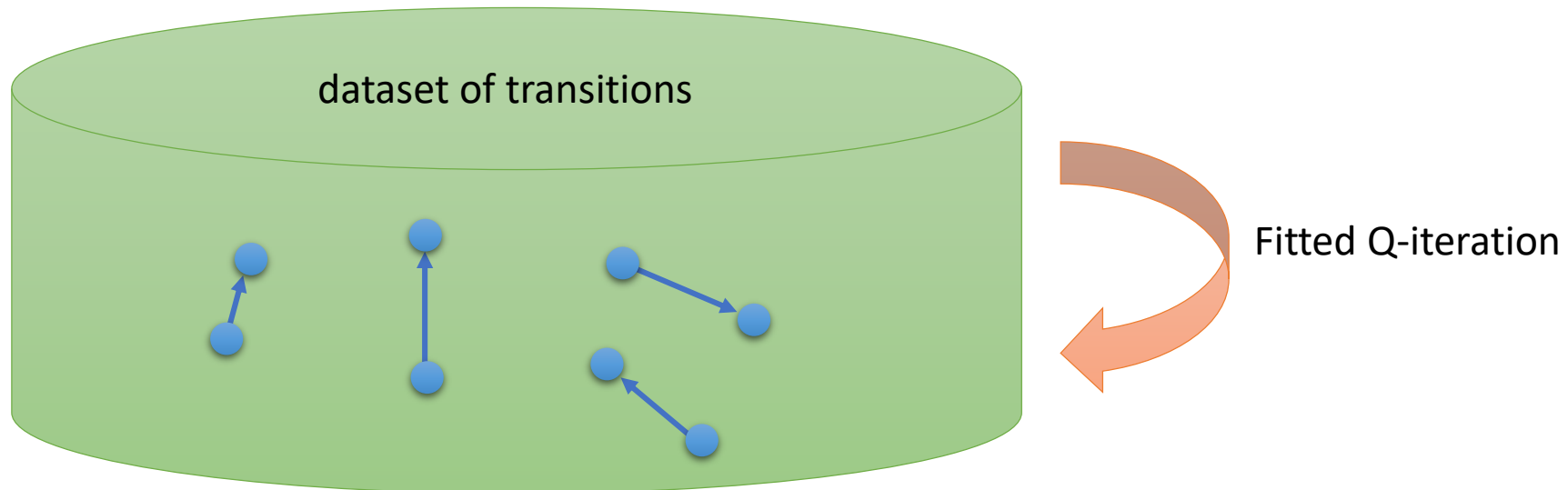
full fitted Q-iteration algorithm:

1. collect dataset $\{(s_i, \mathbf{a}_i, s'_i, r_i)\}$ using some policy
2. set $\mathbf{y}_i \leftarrow r(s_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(s'_i, \mathbf{a}'_i)$
3. set $\phi \leftarrow \arg \min_{\phi} \frac{1}{2} \sum_i \|Q_\phi(s_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

this approximates the value of π' at s'_i

$$\pi'(\mathbf{a}_t | s_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg \max_{\mathbf{a}_t} Q^\pi(s_t, \mathbf{a}_t) \\ 0 & \text{otherwise} \end{cases}$$

given \mathbf{s} and \mathbf{a} , transition is independent of π



What is fitted Q-iteration optimizing?

full fitted Q-iteration algorithm:

1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy
 2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$ ← this max improves the policy (tabular case)
 3. set $\phi \leftarrow \arg \min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$
- ↑
error \mathcal{E}

$$\mathcal{E} = \frac{1}{2} E_{(\mathbf{s}, \mathbf{a}) \sim \beta} \left[Q_\phi(\mathbf{s}, \mathbf{a}) - [r(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}', \mathbf{a}')] \right]^2$$

if $\mathcal{E} = 0$, then $Q_\phi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}', \mathbf{a}')$

this is an *optimal* Q-function, corresponding to optimal policy π' :

$$\pi'(\mathbf{a}_t | \mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg \max_{\mathbf{a}_t} Q_\phi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 & \text{otherwise} \end{cases} \quad \begin{array}{l} \text{maximizes reward} \\ \text{sometimes written } Q^* \text{ and } \pi^* \end{array}$$

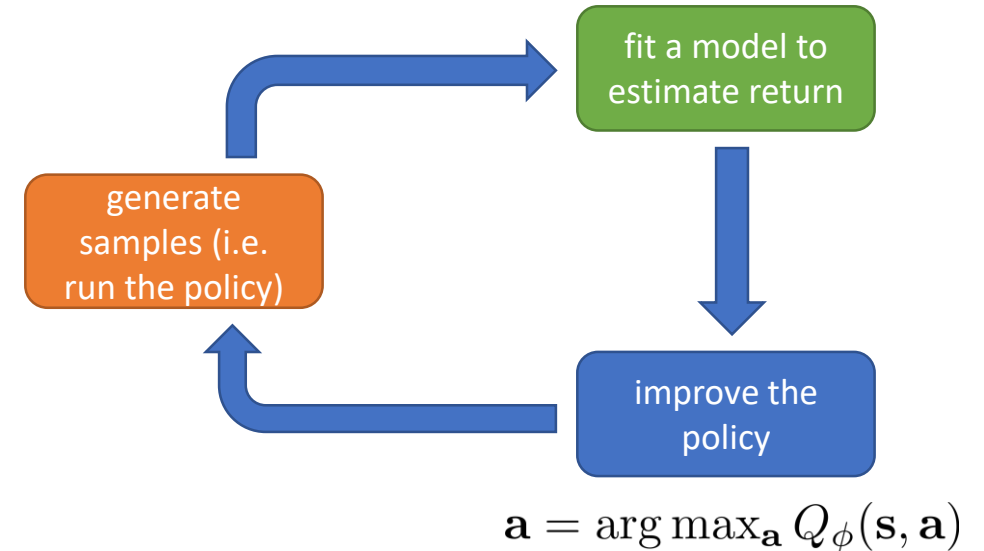
most guarantees are lost when we leave the tabular case (e.g., when we use neural network function approximation)

Online Q-learning algorithms

full fitted Q-iteration algorithm:

1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy
2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. set $\phi \leftarrow \arg \min_{\phi} \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

$$Q_\phi(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}', \mathbf{a}')$$



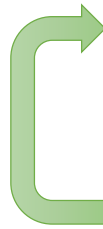
online Q iteration algorithm:

off policy, so many choices here!

1. take some action \mathbf{a}_i and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
2. $\mathbf{y}_i = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i)$

Exploration with Q-learning

online Q iteration algorithm:

- 
1. take some action \mathbf{a}_i and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
 2. $\mathbf{y}_i = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
 3. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i)$

$$\pi(\mathbf{a}_t | \mathbf{s}_t) = \begin{cases} 1 - \epsilon & \text{if } \mathbf{a}_t = \arg \max_{\mathbf{a}_t} Q_\phi(\mathbf{s}_t, \mathbf{a}_t) \\ \epsilon / (|\mathcal{A}| - 1) & \text{otherwise} \end{cases}$$

$$\pi(\mathbf{a}_t | \mathbf{s}_t) \propto \exp(Q_\phi(\mathbf{s}_t, \mathbf{a}_t))$$

final policy:

$$\pi(\mathbf{a}_t | \mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg \max_{\mathbf{a}_t} Q_\phi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 & \text{otherwise} \end{cases}$$

why is this a bad idea for step 1?

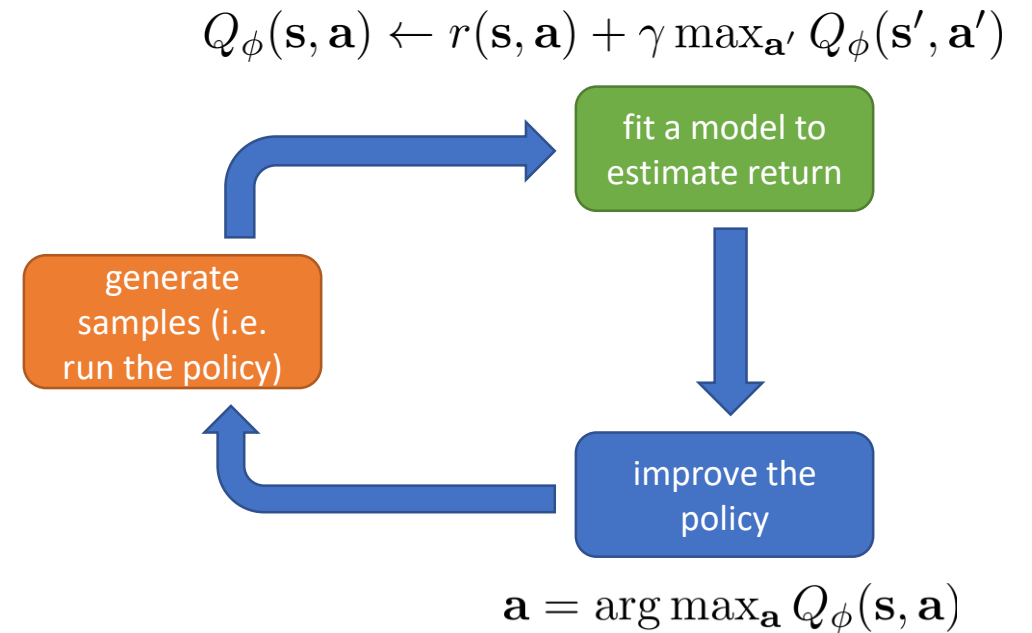
“epsilon-greedy”

“Boltzmann exploration”

We'll discuss exploration in more detail in a later lecture!

Review


- Value-based methods
 - Don't learn a policy explicitly
 - Just learn value or Q-function
- If we have value function, we have a policy
- Fitted Q-iteration
 - Batch mode, off-policy method
- Q-learning
 - Online analogue of fitted Q-iteration



Break

Value function learning theory

value iteration algorithm:

- 
1. set $Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E[V(\mathbf{s}')$
 2. set $V(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$

0.2	0.3	0.4	0.3
0.3	0.3	0.5	0.3
0.4	0.4	0.6	0.4
0.5	0.5	0.7	0.5

does it converge?

and if so, to what?

define an operator \mathcal{B} : $\mathcal{B}V = \max_{\mathbf{a}} r_{\mathbf{a}} + \gamma \mathcal{T}_{\mathbf{a}}V$

stacked vector of rewards at all states for action \mathbf{a}

matrix of transitions for action \mathbf{a} such that $\mathcal{T}_{\mathbf{a},i,j} = p(\mathbf{s}' = i | \mathbf{s} = j, \mathbf{a})$

V^* is a *fixed point* of \mathcal{B}


$$V^*(\mathbf{s}) = r(\mathbf{s}, \mathbf{a}) + \gamma E[V^*(\mathbf{s}')], \text{ so } V^* = \mathcal{B}V^*$$

always exists, is always unique, always corresponds to the optimal policy

...but will we reach it?

Value function learning theory

value iteration algorithm:

- 
1. set $Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E[V(\mathbf{s}')$
 2. set $V(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$

0.2	0.3	0.4	0.3
0.3	0.3	0.5	0.3
0.4	0.4	0.6	0.4
0.5	0.5	0.7	0.5

V^* is a *fixed point* of \mathcal{B}

$$V^*(\mathbf{s}) = r(\mathbf{s}, \mathbf{a}) + \gamma E[V^*(\mathbf{s}')], \text{ so } V^* = \mathcal{B}V^*$$

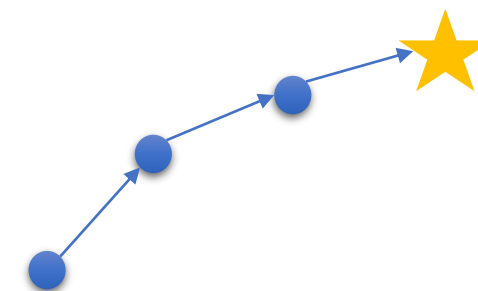
we can prove that value iteration reaches V^* because \mathcal{B} is a *contraction*

contraction: for any V and \bar{V} , we have $\|\mathcal{B}V - \mathcal{B}\bar{V}\|_{\infty} \leq \gamma \|V - \bar{V}\|_{\infty}$

gap always gets smaller by γ !
(with respect to ∞ -norm)

what if we choose V^* as \bar{V} ? $\mathcal{B}V^* = V^*$!

$$\|\mathcal{B}V - V^*\|_{\infty} \leq \gamma \|V - V^*\|_{\infty}$$



Non-tabular value function learning

value iteration algorithm (using \mathcal{B}):

↪ 1. $V \leftarrow \mathcal{B}V$

fitted value iteration algorithm (using \mathcal{B} and Π):

↪ 1. $V \leftarrow \Pi\mathcal{B}V$

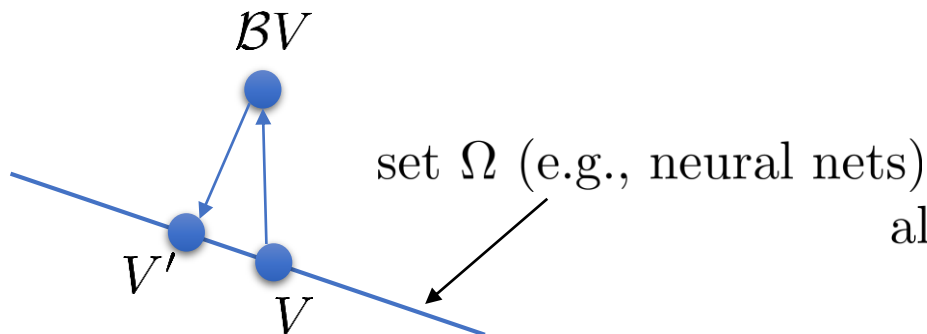
fitted value iteration algorithm:

↪ 1. set $\mathbf{y}_i \leftarrow \max_{\mathbf{a}_i} (r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)])$
2. set $\phi \leftarrow \arg \min_{\phi} \frac{1}{2} \sum_i \|V_\phi(\mathbf{s}_i) - \mathbf{y}_i\|^2$

what does this do?

define new operator Π : $\Pi V = \arg \min_{V' \in \Omega} \frac{1}{2} \sum \|V'(\mathbf{s}) - V(\mathbf{s})\|^2$

Π is a *projection* onto Ω (in terms of ℓ_2 norm)



updated value function

$V' \leftarrow \arg \min_{V' \in \Omega} \frac{1}{2} \sum \|V'(\mathbf{s}) - (BV)(\mathbf{s})\|^2$

all value functions represented by, e.g., neural nets

Non-tabular value function learning

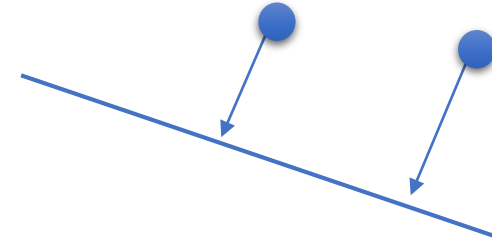
fitted value iteration algorithm (using \mathcal{B} and Π):

↻ 1. $V \leftarrow \Pi \mathcal{B} V$

\mathcal{B} is a contraction w.r.t. ∞ -norm (“max” norm)

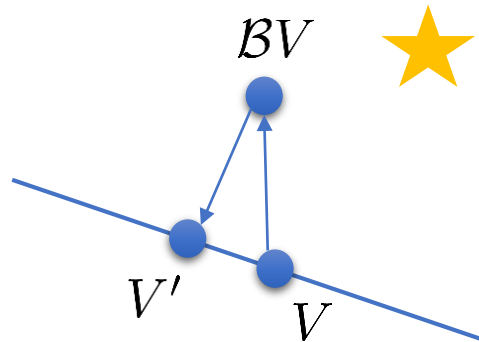
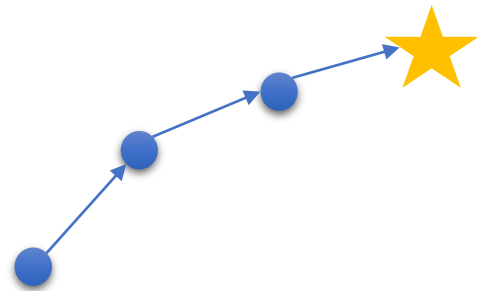
Π is a contraction w.r.t. ℓ_2 -norm (Euclidean distance)

but... $\Pi \mathcal{B}$ is not a contraction of any kind



$$\|\mathcal{B}V - \mathcal{B}\bar{V}\|_\infty \leq \gamma \|V - \bar{V}\|_\infty$$

$$\|\Pi V - \Pi \bar{V}\|^2 \leq \|V - \bar{V}\|^2$$



Conclusions:

value iteration converges (tabular case)


fitted value iteration does **not** converge

not in general

often not in practice

What about fitted Q-iteration?

fitted Q iteration algorithm:

- 
1. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)]$
 2. set $\phi \leftarrow \arg \min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

define an operator \mathcal{B} : $\mathcal{B}Q = r + \gamma \mathcal{T} \max_{\mathbf{a}} Q$

max now after the transition operator

define an operator Π : $\Pi Q = \arg \min_{Q' \in \Omega} \frac{1}{2} \sum \|Q'(\mathbf{s}, \mathbf{a}) - Q(\mathbf{s}, \mathbf{a})\|^2$

fitted Q-iteration algorithm (using \mathcal{B} and Π):

- 
1. $Q \leftarrow \Pi \mathcal{B} Q$

\mathcal{B} is a contraction w.r.t. ∞ -norm (“max” norm)

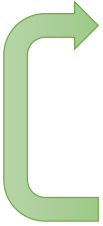
Π is a contraction w.r.t. ℓ_2 -norm (Euclidean distance)

$\Pi \mathcal{B}$ is not a contraction of any kind

Applies also to online Q-learning

But... it's just regression!

online Q iteration algorithm:

- 
1. take some action \mathbf{a}_i and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
 2. $\mathbf{y}_i = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
 3. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i)$

isn't this just gradient descent? that converges, right?

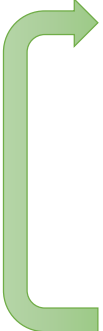
Q-learning is *not* gradient descent!

$$\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - (r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)))$$

no gradient through target value

A sad corollary

batch actor-critic algorithm:


- 
1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
 2. fit $\hat{V}_\phi^\pi(\mathbf{s})$ to sampled reward sums
 3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
 4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
 5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

An aside regarding terminology


V^π : value function for policy π
this is what the critic does

V^* : value function for optimal policy π^*
this is what value iteration does

ℓ_∞ contraction \mathcal{B} (but without max)


$$y_{i,t} \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1})$$

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}_\phi^\pi(\mathbf{s}_i) - y_i \right\|^2$$



ℓ_2 contraction Π

fitted bootstrapped policy evaluation doesn't converge!

Review

- Value iteration theory
 - Linear operator for backup
 - Linear operator for projection
 - Backup is contraction
 - Value iteration converges
- Convergence with function approximation
 - Projection is also a contraction
 - Projection + backup is **not** a contraction
 - Fitted value iteration does not in general converge
- Implications for Q-learning
 - Q-learning, fitted Q-iteration, etc. does not converge with function approximation
- But we can make it work in practice!
 - Sometimes – tune in next time

