Value Function Methods

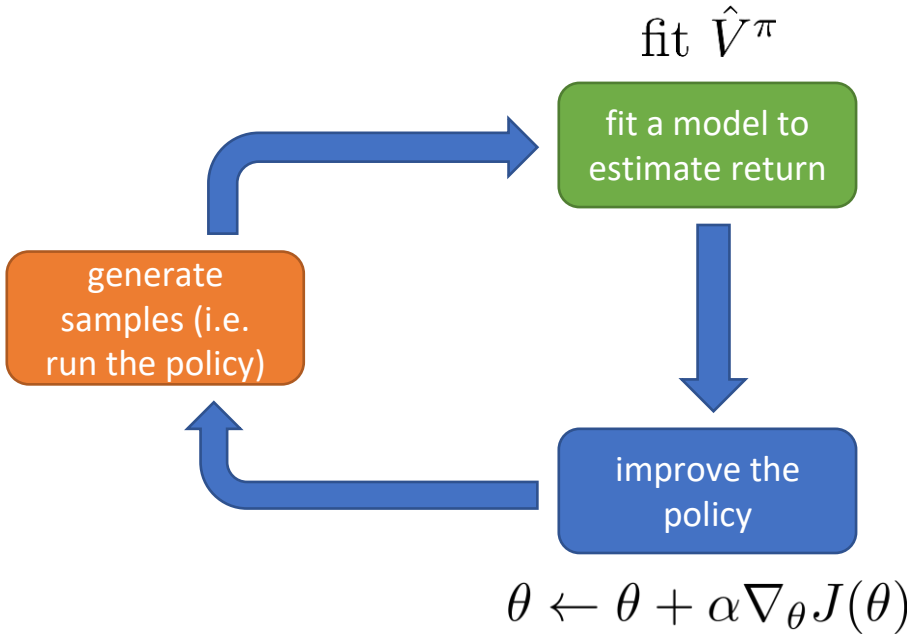
CS 285

Instructor: Sergey Levine
UC Berkeley

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 $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
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 $\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$

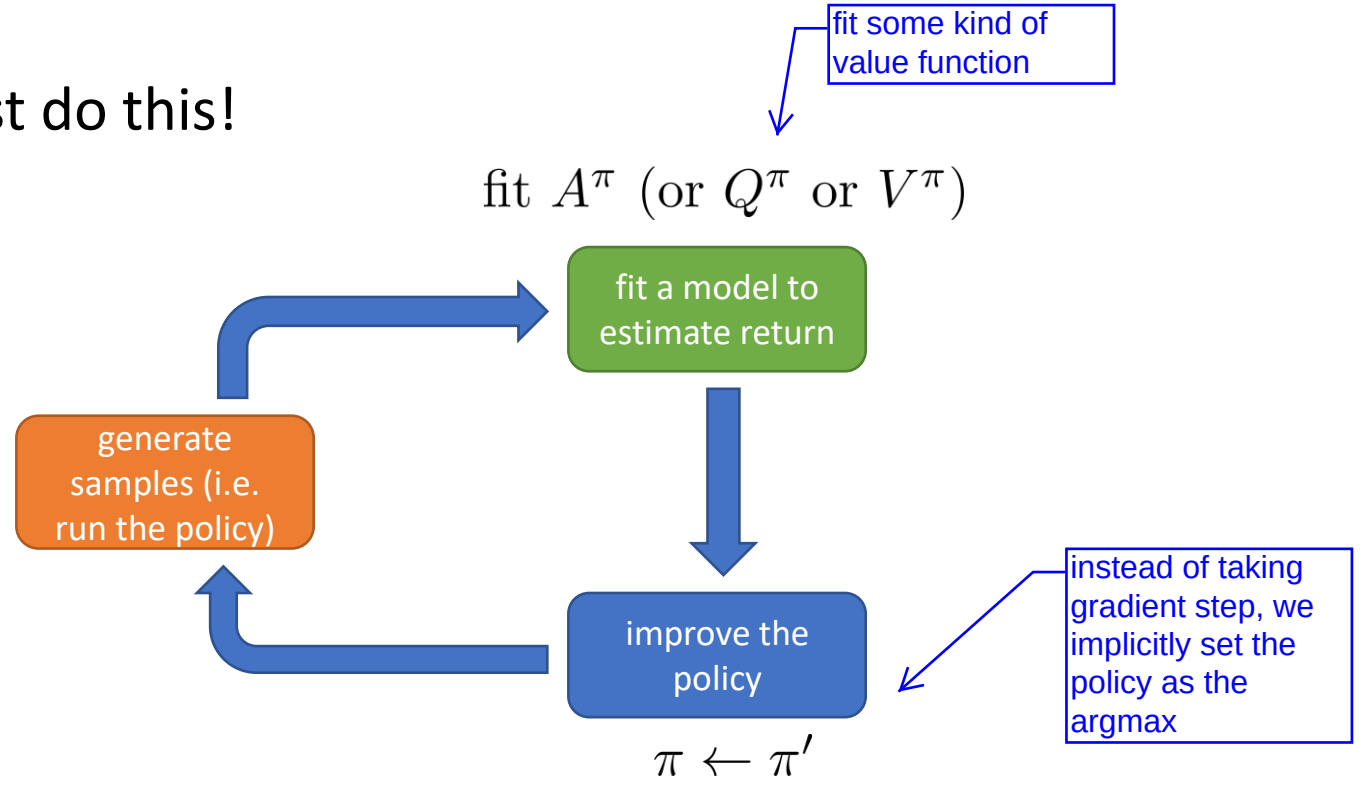
regardless of what $\pi(\mathbf{a}_t|\mathbf{s}_t)$ is!

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)

the policy is represented implicitly through the argmax



Policy iteration

High level idea:

policy iteration algorithm:

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

evaluate advantage of current policy

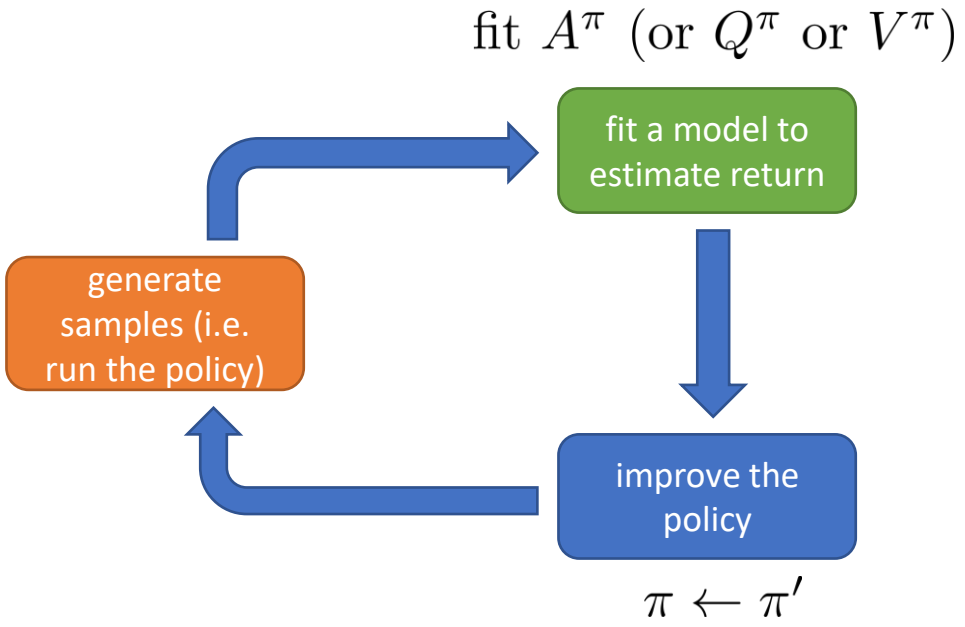
construct new policy

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

take action only if it's the argmax

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

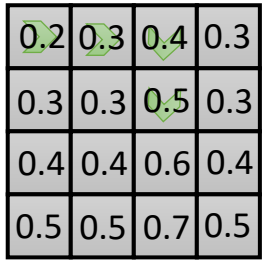
let's evaluate $V^\pi(\mathbf{s})$!



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)

tabular RL



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

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

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:

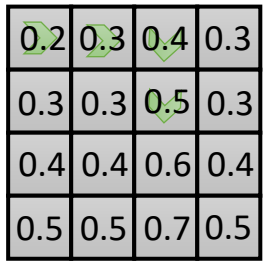
1. evaluate $V^\pi(\mathbf{s})$
2. set $\pi \leftarrow \pi'$

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

recursive!

policy evaluation:

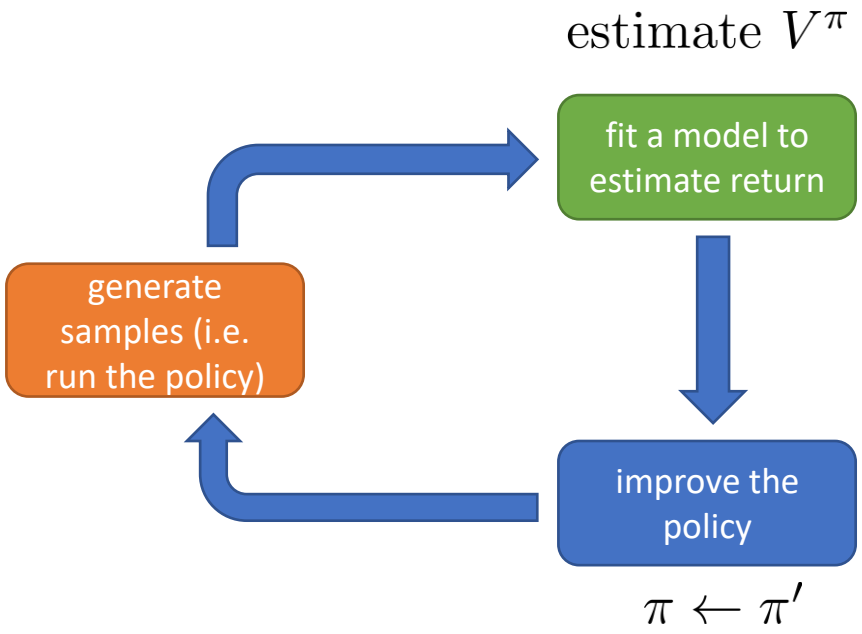
$$V^\pi(\mathbf{s}) \leftarrow r(\mathbf{s}, \pi(\mathbf{s})) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \pi(\mathbf{s}))}[V^\pi(\mathbf{s}')]$$



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

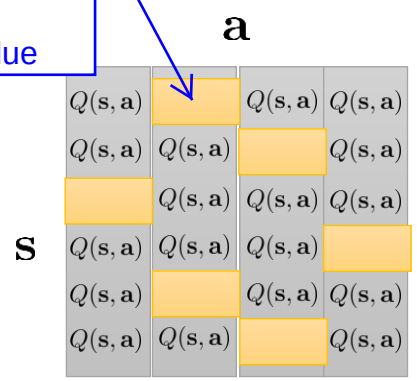


Even simpler dynamic programming

Construct table of q-values

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

select the action that has the highest Q-value

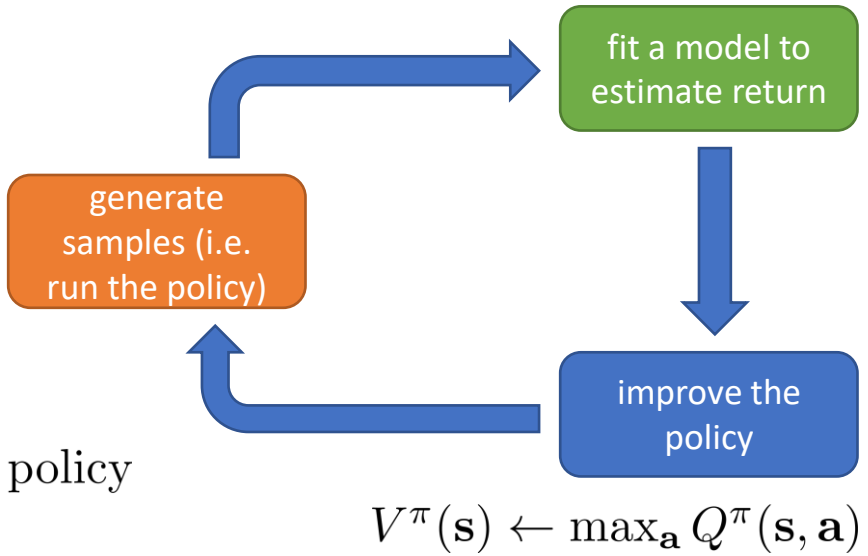


$$A^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma E[V^\pi(\mathbf{s}')] - V^\pi(\mathbf{s})$$

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

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

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



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

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

construct value function by taking the max

approximates the new value!

skip the policy and compute values directly!

value iteration algorithm:

1. set $Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E[V(\mathbf{s}')]$ ← construct Q-value table
2. set $V(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$ set the value to be the max

we can simplify this even more by representing the entire thing w.r.t to the Q-function, and eliminate V

Fitted Value Iteration & Q-Iteration

Fitted value iteration

this fails due to the curse of dimensionality! Cannot store this many entries. For continuous state-space, it won't work at all

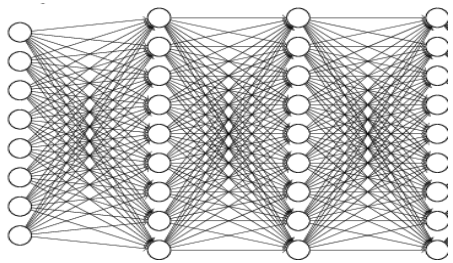
fixes the curse of dimensionality

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

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

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

$$\mathbf{s} = 0 : V(\mathbf{s}) = 0.2$$
$$\mathbf{s} = 1 : V(\mathbf{s}) = 0.3$$
$$\mathbf{s} = 2 : V(\mathbf{s}) = 0.5$$

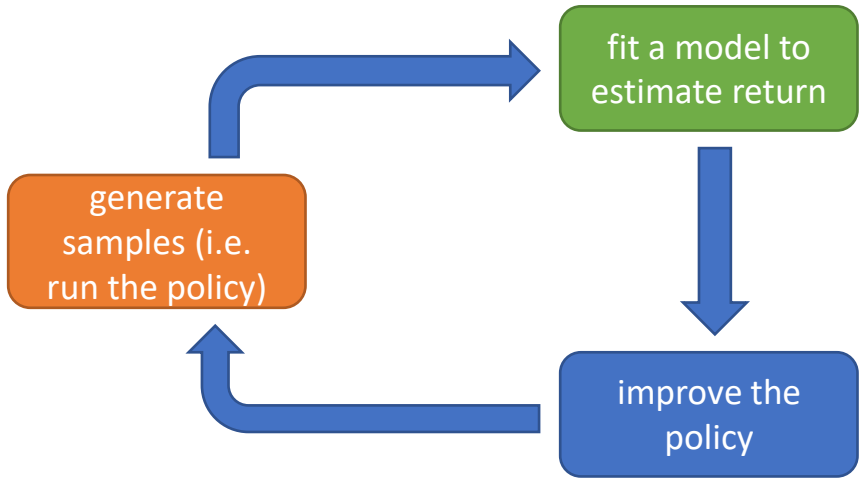
$\mathbf{s}$  $V(\mathbf{s})$

parameters ϕ

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

(more than atoms in the universe)

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



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

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

curse of dimensionality

for every state we have to find this value for every possible action

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 $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \left\| V_\phi(\mathbf{s}_i) - \mathbf{y}_i \right\|^2$

Step 1: Create targets by calculating the expected returns for every possible action given the state. Then take the max.

Step 2: Regress over those targets

What if we don't know the transition dynamics?

so one issue is that we can't actually try every single action b/c this would require us to be able to reset to the state. So if we don't know the transition dynamics, we can't do this.

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:

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

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

can fit this using samples

this allows us to do policy iteration without having to know the transition probabilities. it just requires us to use 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 $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)]$
2. set $\phi \leftarrow \arg\min_\phi \frac{1}{2}\sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

approxiate $E[V(\mathbf{s}'_i)] \approx \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$

doesn't require simulation of actions!

now all we need are samples (s, a, s') which we can achieve by rolling out our policy

+ 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)

e.g. neural network

Fitted Q-iteration

full fitted Q-iteration algorithm:

parameters

1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy

dataset size N, collection policy

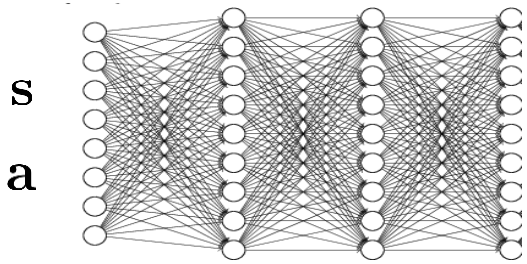
$K\times$

2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$ for every transition that we sampled, calculate the target

iterations K

3. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$ update our Q-function

gradient steps S



$Q_\phi(\mathbf{s}, \mathbf{a})$

parameters ϕ

another architecture is to input the state and then output the Q-values for every possible action a

Review

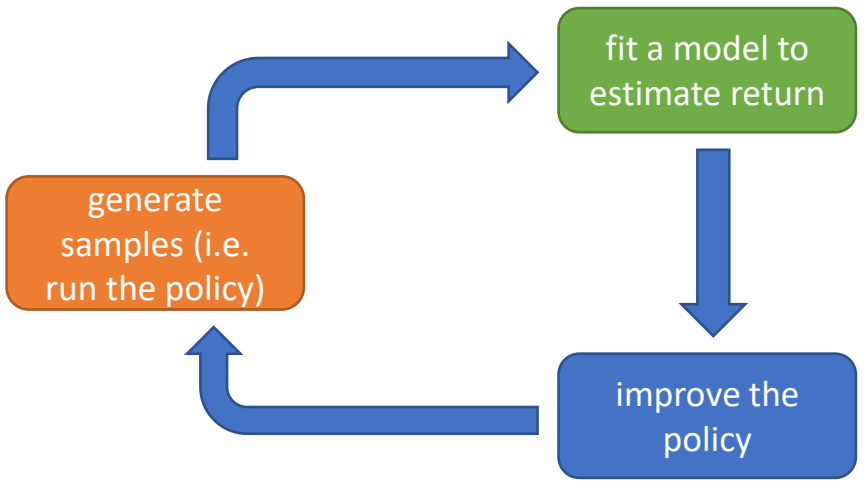
- Value-based methods
  - Don't learn a policy explicitly
  - Just learn value or Q-function
- If we have value function, we have a policy
- Fitted Q-iteration

represented by a table or NN

if we have a value function, we can recover a policy by using the argmax

removes the need for us to know the transition probabilities

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



$$\mathbf{a} = \arg\max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a})$$

From Q-Iteration to Q-Learning

Why is this algorithm off-policy?

full fitted Q-iteration algorithm:

1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy

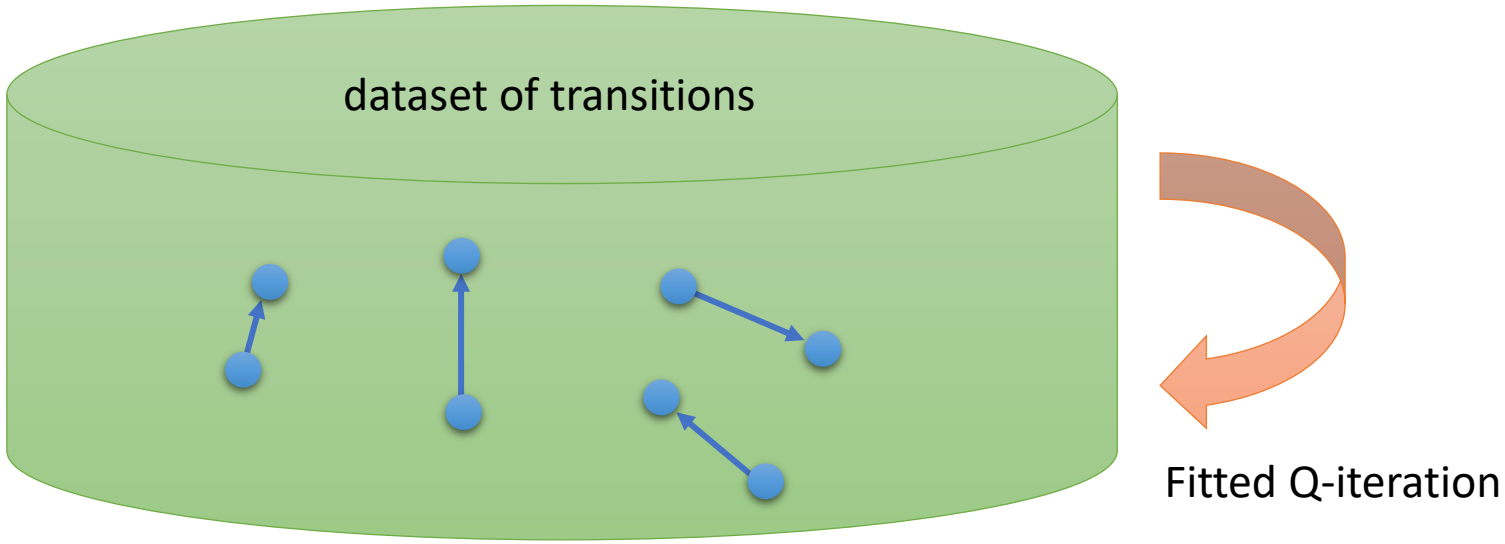
$K\times$

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 $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

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

this approximates the value of π' at $\mathbf{s}'_i$

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



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

$K\times$

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 $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \left\| Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i \right\|^2$

in tabular learning, we could just write the y's in our table. but in deep learning, we need to optimize the network

error $\mathcal{E}$

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

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':

but using NN, the error is never 0, so we can't guarantee we have the optimal policy

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

maximizes reward

sometimes written $Q^\star$ and $\pi^\star$

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

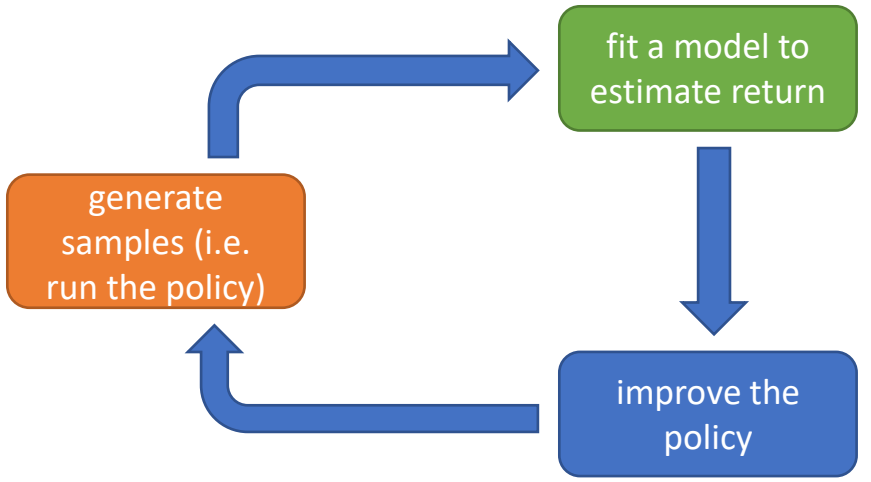
Online Q-learning algorithms

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

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 $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

($K\times$)



$\mathbf{a} = \arg\max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a})$

off policy, so many choices here!

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. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i)$

TD error

This is the Watkin's Q-learning algorithm

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. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i)$

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?

because we won't explore! Our initial Q-function will be pretty bad, so we will commit our policy to take the same action every time we enter a particular state. If this action isn't good, then we have a bad result

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

“epsilon-greedy”

with probability (1 - epsilon) take the max. With probability epsilon) take another action uniformly at random. You can vary epsilon over the course of training.

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

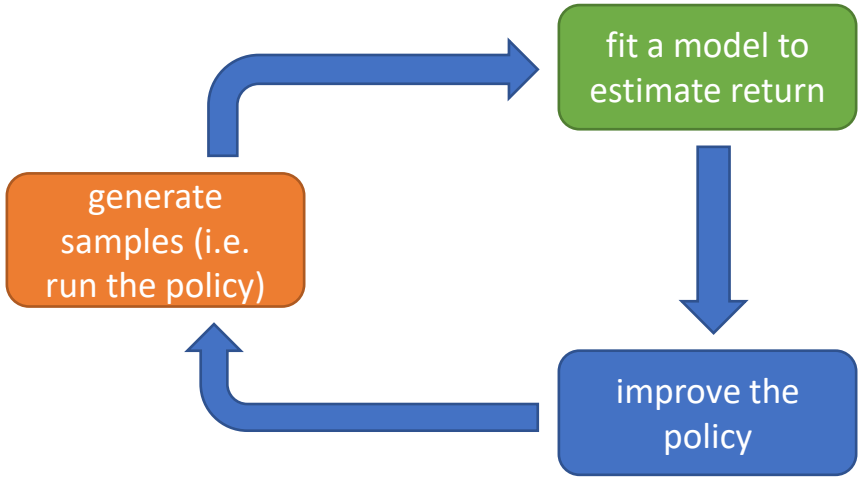
“Boltzmann exploration”

We’ll discuss exploration in 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

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



$$\mathbf{a} = \arg\max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a})$$

Value Functions in Theory

Value function learning theory

value iteration algorithm:

1. set $Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E[V(\mathbf{s}')]$ construct table of Q-values
2. set $V(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$ set the value function to be the max along the rows of that table

V(s1) = 0.4	0.2	0.3	0.4	0.3
V(s2) = 0.5	0.3	0.3	0.5	0.3
V(s3) = 0.6	0.4	0.4	0.6	0.4
V(s4) = 0.7	0.5	0.5	0.7	0.5

does it converge? and if so, to what?

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

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

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

$V^\star$ is a *fixed point* of $\mathcal{B}$ $V^\star(\mathbf{s}) = \max_{\mathbf{a}} r(\mathbf{s}, \mathbf{a}) + \gamma E[V^\star(\mathbf{s}')]$, so $V^\star = \mathcal{B}V^\star$

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

...but will we reach it?

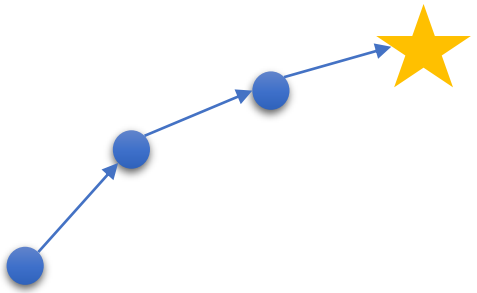
Value function learning theory

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

$V^\star$ is a *fixed point* of $\mathcal{B}$ $\quad V^\star(\mathbf{s}) = \max_{\mathbf{a}} r(\mathbf{s}, \mathbf{a}) + \gamma E[V^\star(\mathbf{s}')]$, so $V^\star = \mathcal{B}V^\star$



we can prove that value iteration reaches $V^\star$ 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 $V^\star$ as $\bar{V}$? $\mathcal{B}V^\star = V^\star$!

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

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?

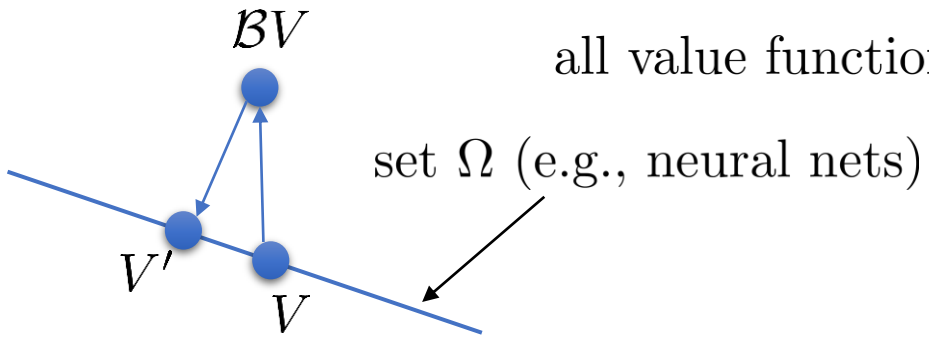
define new operator Π: $\Pi V = \arg\min_{V' \in \Omega} \frac{1}{2}\sum \|V'(\mathbf{s}) - V(\mathbf{s})\|^2$

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

updated value function

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

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

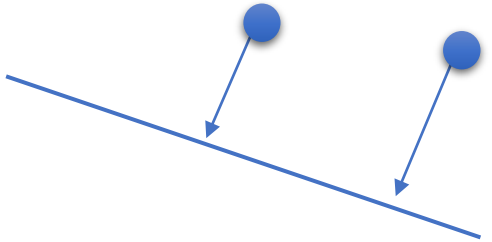


set Ω (e.g., neural nets)

Non-tabular value function learning

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

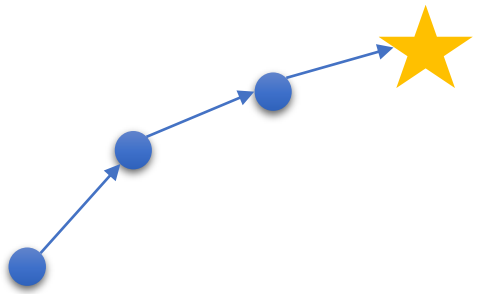
1. $V \leftarrow \Pi\mathcal{B}V$

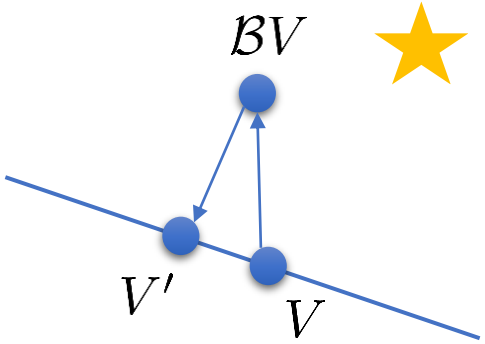


$\mathcal{B}$ is a contraction w.r.t. ∞-norm ("max" norm) $\quad \|\mathcal{B}V - \mathcal{B}\bar{V}\|_\infty \leq \gamma\|V - \bar{V}\|_\infty$

Π is a contraction w.r.t. ℓ_2-norm (Euclidean distance) $\quad \|\Pi V - \Pi\bar{V}\|^2 \leq \|V - \bar{V}\|^2$

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





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 $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
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

$V^\star$: value function for optimal policy $\pi^\star$
this is what value iteration does

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

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

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

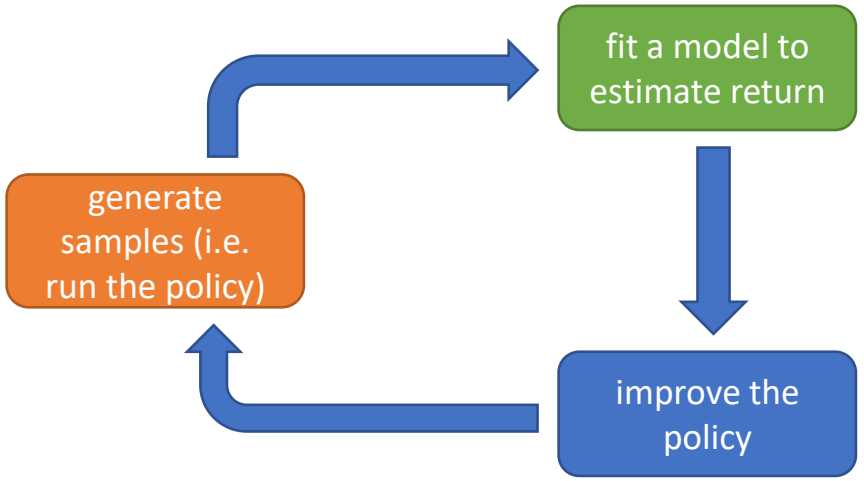
ℓ_2 contraction Π

fitted bootstrapped policy evaluation doesn't converge!

Review

- Value iteration theory
  - Operator for backup
  - 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

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



$$\mathbf{a} = \arg\max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a})$$