Lecture 8: Value Function Methods

Zhi Wang & Chunlin Chen

Department of Control and Systems Engineering
Nanjing University

Nov. 19th, 2020

Table of Contents

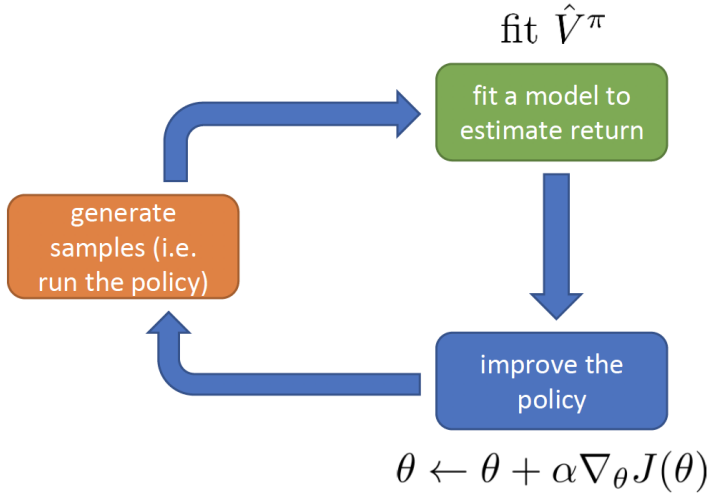
Contents and Goals

- What if we just use a critic, without an actor?
- Extracting a policy from a value function
- The fitted value iteration, fitted Q-iteration algorithms
- Goals
  - Understand how value functions give rise to policies
  - Understand the Q-learning with function approximation algorithm
  - Understand practical considerations for Q-learning

Recall: 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

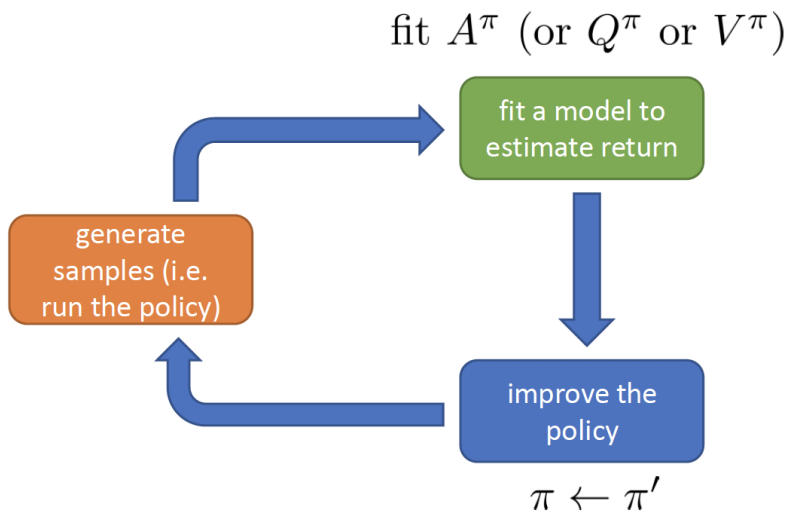
at *least* as good as any $\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$

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)



Recall the Policy Improvement Theorem

- Let π and π' be any pair of deterministic policies such that,

$$Q_\pi(s, \pi'(s)) \geq v_\pi(s), \quad \forall s \in \mathcal{S}.$$

Then the policy π' must be as good as, or better than, π.

Policy improvement theorem

$$
\begin{aligned}
v_\pi(s) &\leq Q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma Q_\pi(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) | S_t = s] \\
&\leq ... \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + ... | S_t = s] \\
&= v_{\pi'}(s)
\end{aligned}
$$

Policy improvement

- Consider the new **greedy** policy, π', selecting at each state the action that appears best according to $Q_\pi(s, a)$

$$\begin{aligned}\pi'(s) &= \arg\max_a Q_\pi(s, a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s', a} p(s', r | s, a)[r + \gamma v_\pi(s')]\end{aligned}$$

- The process of making a new policy that improves on an original policy, by making greedy w.r.t. the value function of the original policy, is called **policy improvement**
  - The greedy policy meets the conditions of the policy improvement theorem

The Generalized Policy Iteration framework

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

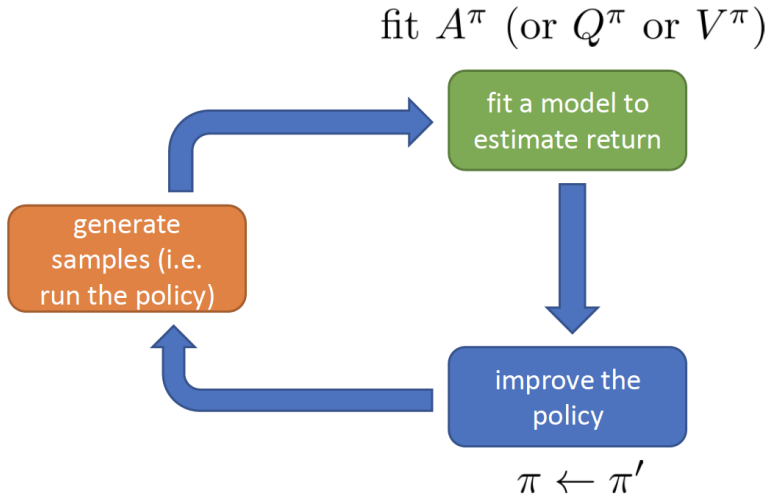
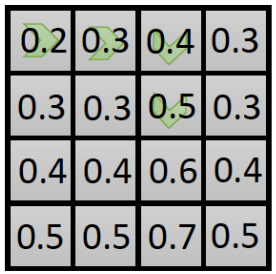


Table of Contents

Dynamic programming (model-based)

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



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

Recall policy iteration

- Using policy improvement theorem, we can obtain a sequence of monotonically improving policies and value functions

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \ldots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

- This process is guaranteed to converge to an optimal policy and optimal value function in a finite number of iterations
  - Each policy is guaranteed to be a strictly improvement over the previous one unless it is already optimal
  - A finite MDP has only a finite number of policies

Policy iteration algorithm

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \,|\, s, \pi(s)) \big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \,|\, s, a) \big[r + \gamma V(s')\big]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Dynamic programming with policy iteration

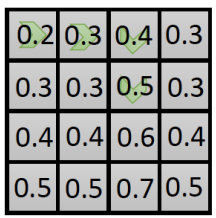
policy iteration:

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:

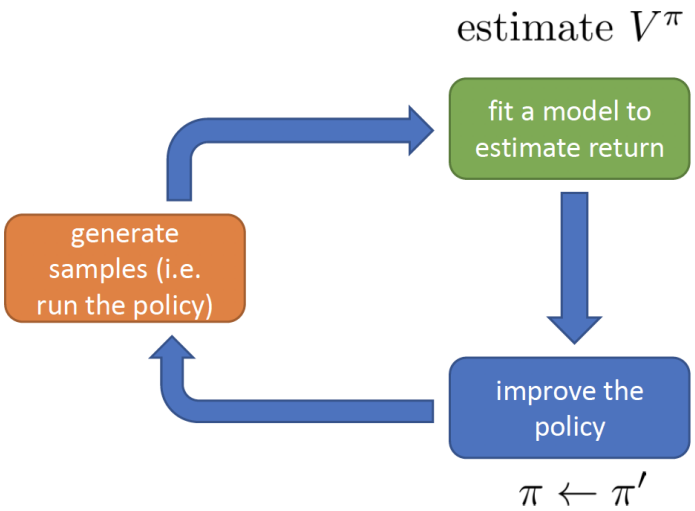
$$V^\pi(\mathbf{s}) \leftarrow r(\mathbf{s}, \pi(\mathbf{s})) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \pi(\mathbf{s}))}[V^\pi(\mathbf{s}')]$$



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



Recall: Value iteration = Truncate policy evaluation for one sweep

- In policy iteration, stop policy evaluation after just one sweep

$$v_{k+1}(s) = \sum_{s',r} p(s', r|s, \pi_k(s))[r + \gamma v_k(s')]$$

$$\pi_{k+1}(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_{k+1}(s')]$$

- Combine into one operation, called **value iteration** algorithm

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_k(s')]$$

  - For arbitrary v_0, the sequence $\{v_k\}$ converges to v_* under the same conditions that guarantee the existence of v_*

Value iteration

- Bellman optimality equation

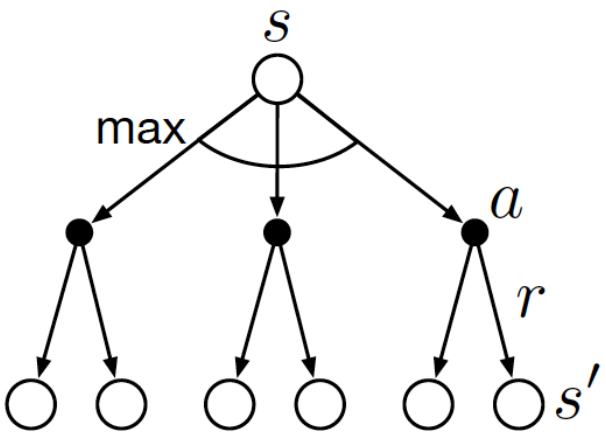
$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]$$

- Value iteration

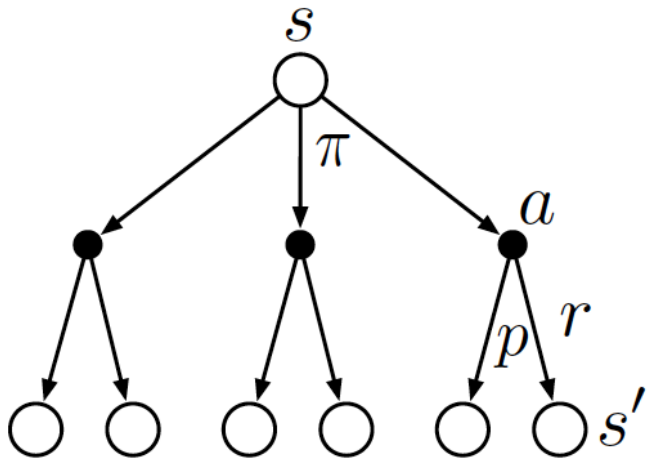
$$v_{k+1}(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_k(s')]$$

  - Turn Bellman optimality equation into an update rule
  - Directly approximate the optimal state-value function, v_*

Value iteration vs. policy evaluation



Backup diagram for value iteration



Backup diagram for policy evaluation

Value iteration algorithm

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $\quad v \leftarrow V(s)$
| $\quad V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
| $\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

- One sweep = one sweep of policy evaluation + one sweep of policy improvement

Dynamic programming with value iteration

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

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

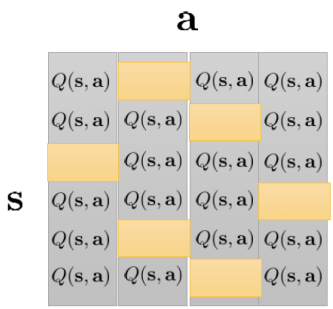
$$\arg\max_{\mathbf{a}_t} A^\pi(\mathbf{s}_t, \mathbf{a}_t) = \arg\max_{\mathbf{a}_t} Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$$

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

skip the policy and compute values directly!

value iteration algorithm:

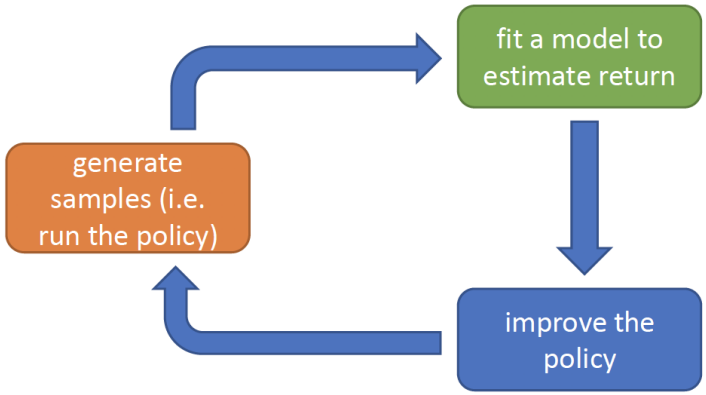
1. set $Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E[V(\mathbf{s}')]$
2. set $V(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$



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

approximates the new value!

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



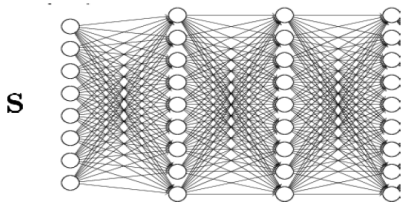
$$V^\pi(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})$$

Fitted value iteration with function approximation

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

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

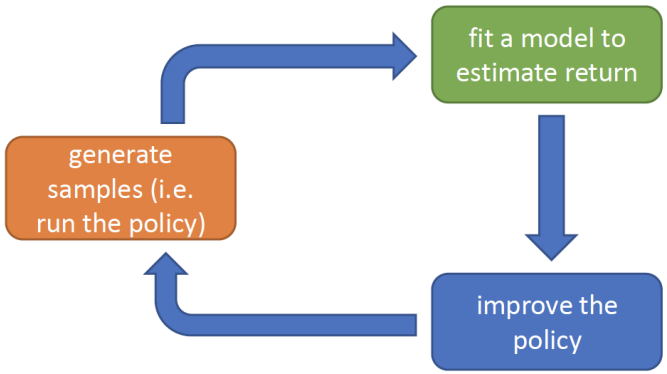
(more than atoms in the universe)

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

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

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



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

Table of Contents

Determine optimal policies from optimal value functions

- For v_*: a one-step search
  - Actions that appear best after one-step search will be optimal actions
- For Q_*: no need to do a one-step-ahead search
  - $a_* = \arg\max_a Q_*(s, a)$
  - The optimal action-value function allows optimal actions to be selected without having to know anything about possible successor states and their values, i.e., without having to know anything about the environment's dynamics

What if we don't know the transition dynamics?

- From prediction problems to control problems
- From state-value functions to action-value functions

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

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 $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)]$ ← approxiate $E[V(\mathbf{s}'_i)] \approx \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
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 (FQI)

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

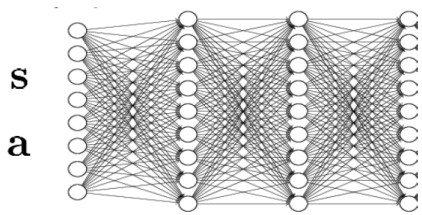
$K\times$

2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$

iterations K

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

gradient steps S



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

On-policy vs. Off-policy

Target policy $\pi(a\|s)$	Behavior policy $b(a\|s)$
To be evaluated or improved	To explore to generate data
Make decisions finally	Make decisions in training phase

- **On-policy** methods: $\pi(a|s) = b(a|s)$
  - Evaluate or improve the policy that is used to make decisions during training
  - e.g., SARSA
- **Off-policy** methods: $\pi(a|s) \neq b(a|s)$
  - Evaluate or improve a policy different from that used to generate the data
  - Separate exploration from control
  - e.g., Q-learning

Q-learning vs. SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

- Q-learning approximates the optimal action-value function for an optimal policy, $Q \approx Q_* = Q_{\pi_*}$
  - The target policy is greedy w.r.t Q, $\pi(a|s) = \arg\max_a Q(s, a)$
  - The behavior policy can be others, e.g., $b(a|s) = \varepsilon$-greedy

---

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

- SARSA approximates the action-value function for the behavior policy, $Q \approx Q_\pi = Q_b$
  - The target and the behavior policy are the same, e.g., $\pi(a|s) = b(a|s) = \varepsilon$-greedy

Why FQI is off-policy?

full fitted Q-iteration algorithm:

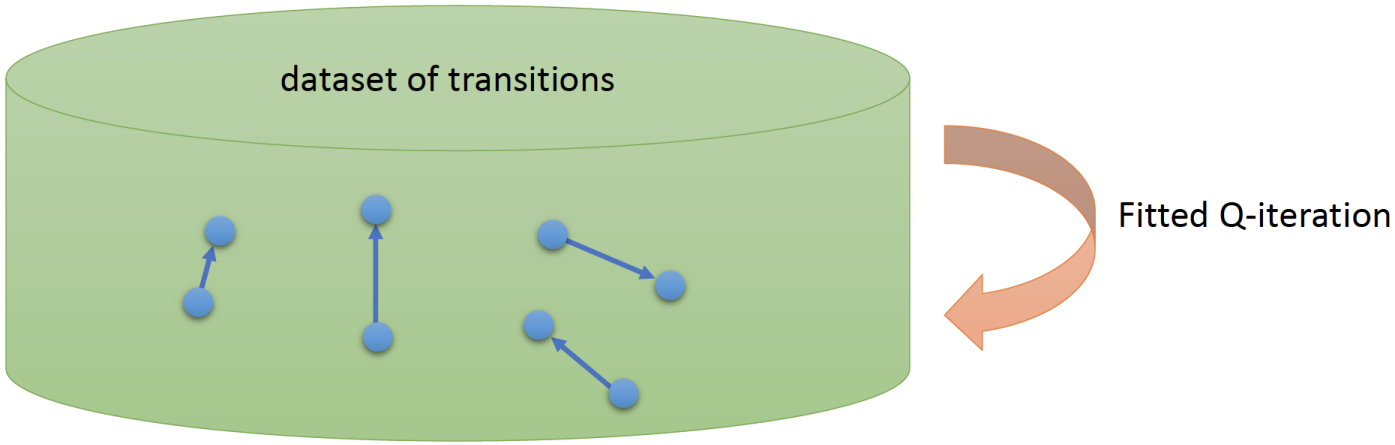
1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy
2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i,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

($K\times$)

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

↑ error $\mathcal{E}$

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

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':

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

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

Batch-mode (offline) vs. online

- Batch-model (offline) algorithms
  - Collect a batch of samples using some policy
  - Fit the state- or action-value function iteratively
- Online algorithms
  - Take some action to collect one sample
  - Fit the value function
  - Iteratively alternate the above two steps

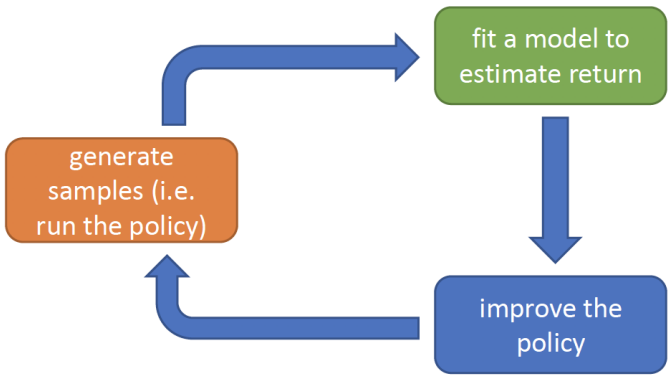
Online Q-learning algorithms

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

($K\times$)

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



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

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

Exploration strategies

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?

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

“epsilon-greedy”

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

“Boltzmann exploration”

Exploitation-exploration dilemma

- ϵ-greedy: $\pi(a_t|s_t) =$
  - $1-\epsilon$: the exploitation part
  - $\epsilon/(|\mathcal{A}|-1)$: the exploration part
  - balance/trade-off between exploitation and exploration: decrease ϵ as the learning proceeds

- Boltzmann strategy, softmax strategy:

$$\pi(a_t|s_t) = \frac{e^{Q(s_t,a_t)/\tau}}{\sum_a e^{Q(s_t,a)/\tau}}$$

  - τ: temperature parameter
  - decrease τ as the learning proceeds

Looking back from DRL

- Tabular RL algorithms
  - R. S. Sutton and A. G. Barto, **Reinforcement Learning: An Introduction**, 2nd Edition.
- RL with linear function approximation
  - M. G. Lagoudakis and R. Parr, **Least-Squares Policy Iteration**, *Journal of Machine Learning Research*, 2003.
- RL with nonlinear function approximation, e.g., neural network
  - J. A. Boyan, et al., **Generalization in reinforcement learning: Safely approximating the value function**, NIPS 1995.
  - R. S. Sutton, et al., **Policy gradient methods for reinforcement learning with function approximation**, NIPS 2000.
  - Martin Riedmiller, **Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method**, ECML 2005.
- RL with deep neural networks
  - **Human-level performance, the milestone of artificial intelligence**

Review

- Value-based methods
  - Don't learn a policy explicitly
  - Just learn state- or action-value 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

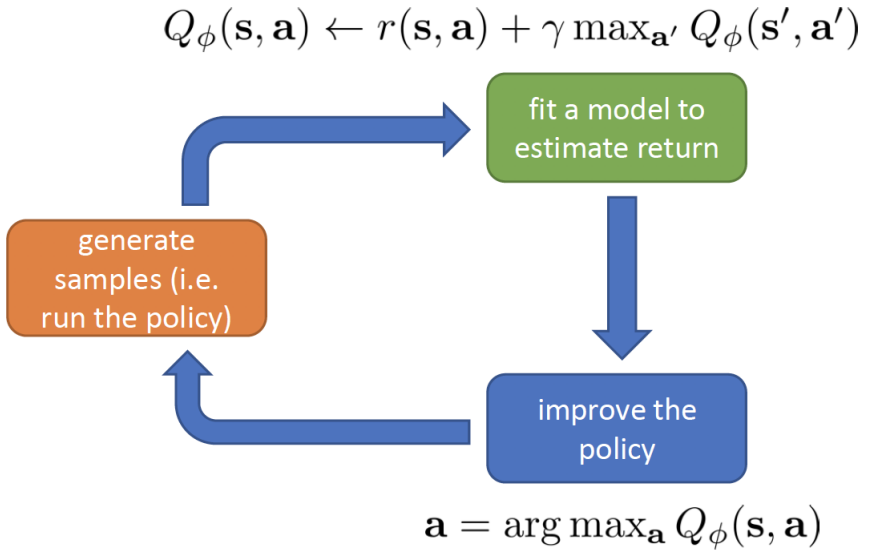


Table of Contents

Value iteration theory in tabular cases

- Bellman optimality equation

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]$$

- Value iteration

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_k(s')]$$

  - Turn Bellman optimality equation into an update rule
  - Directly approximate the optimal state-value function, v_*
  - For arbitrary v_0, the sequence $\{v_k\}$ converges to v_* under the same conditions that guarantee the existence of v_*

Value iteration

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?

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

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

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?

Bellman operator is a contraction

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 $V^\star = \mathcal{B}V^\star$

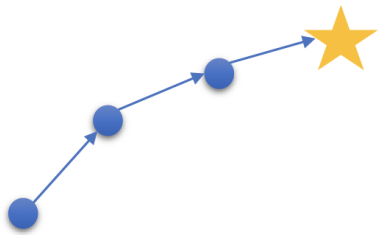
we can prove that value iteration reaches $V^\star$ because $\mathcal{B}$ is a *contraction*

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 $\bar{V}$? $\mathcal{B}V^\star = V^\star$!

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



Norm (mathematics)

- A norm is a function from a real or complex vector space to the nonnegative real numbers
  - behave in certain ways like the distance from the origin: it commutes with scaling, obeys a form of the triangle inequality, and is zero only at the origin
  - the length or size of a vector in a certain space

- p-norm: $||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{(1/p)}$
  - $l1$-norm: $||x||_1 = \sum_{i}^{n} |x_i|$
  - $l2$-norm, the Euclidean norm: $||x||_2 = \sqrt{x_1^2 + ... + x_n^2}$
  - The infinity norm: $||x||_\infty = \max_i |x_i|$

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

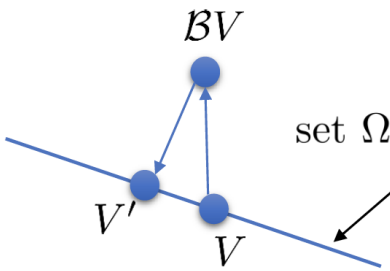
fitted value iteration algorithm:

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)



set Ω (e.g., neural nets)

updated value function

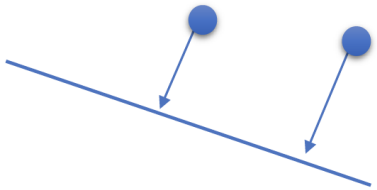
$V' \leftarrow \arg\min_{V' \in \Omega} \frac{1}{2}\sum \|V'(\mathbf{s}) - (\mathcal{B}V)(\mathbf{s})\|^2$

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

Non-tabular value function learning

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

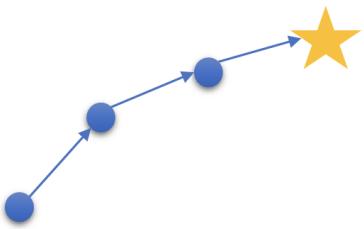
1. $V \leftarrow \Pi\mathcal{B}V$

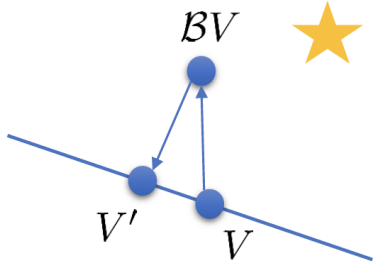


$\mathcal{B}$ is a contraction w.r.t. ∞-norm ("max" norm) $\quad \|\mathcal{B}V - \mathcal{B}\bar{V}\|_\infty \leq \gamma\|V - \bar{V}\|_\infty$

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

Applies also to online Q-learning

$\mathcal{B}$ 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

Fitted Q-iteration – Regression, but not gradient descent

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)

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

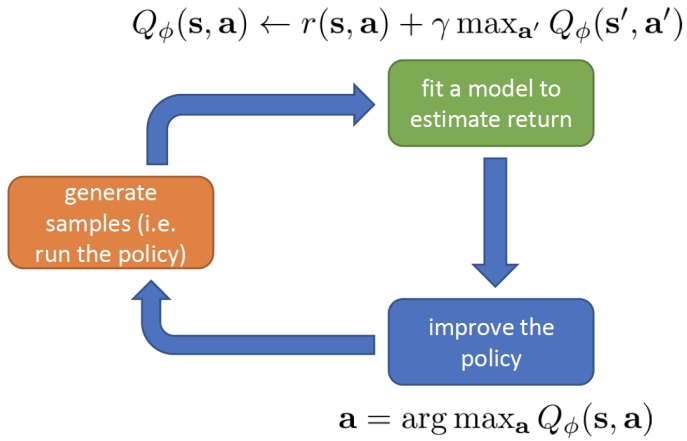
$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}^\pi_\phi(\mathbf{s}_i) - y_i \right\|^2$$

ℓ_2 contraction Π

fitted bootstrapped policy evaluation doesn't converge!

Review

- Value iteration theory
  - 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!



Learning objectives of this lecture

- You should be able to...
  - Extend discrete value iteration to fitted value iteration with function approximation
  - Understand fitted Q-iteration (FQI)
  - Be aware of some theories of value function methods

Suggested readings

- Lecture 7 of CS285 at UC Berkeley, **Deep Reinforcement Learning, Decision Making, and Control**
  - `http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-7.pdf`

- Classical papers
  - Lagoudakis. 2003. **Least-squares policy iteration**: linear function approximation
  - Riedmiller. 2005. **Neural fitted Q-iteration**: batch mode Q-learning with neural networks

THE END