Value Function Methods

CS 285: Deep Reinforcement Learning, Decision Making, and Control
Sergey Levine
Class Notes

1. Homework 2 is due in one week (next Monday)
2. Remember to start forming final project groups and writing your proposal!
   • Proposal due 9/25, this Wednesday!
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 \( \{s_i, a_i\} \) from \( \pi_\theta(a|s) \) (run it on the robot)
2. fit \( \hat{V}_\phi(s) \) to sampled reward sums
3. evaluate \( \hat{A}_\pi(s_i, a_i) = r(s_i, a_i) + \hat{V}_\phi(s_i') - \hat{V}_\phi(s_i) \)
4. \( \nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}_\pi(s_i, a_i) \)
5. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)
Can we omit policy gradient completely?

$A^\pi(s_t, a_t)$: how much better is $a_t$ than the average action according to $\pi$

arg max$_{a_t} A^\pi(s_t, a_t)$: best action from $s_t$, if we then follow $\pi$

at least as good as any $a_t \sim \pi(a_t|s_t)$

regardless of what $\pi(a_t|s_t)$ is!

fit $A^\pi$ (or $Q^\pi$ or $V^\pi$)

$\pi'(a_t|s_t) = \begin{cases} 1 \text{ if } a_t = \text{arg max}_a A^\pi(s_t, a_t) \\ 0 \text{ otherwise} \end{cases}$

as good as $\pi$

(probably better)

忘记策略，我们就这样做吧！

generate samples (i.e. run the policy)

fit a model to estimate return

improve the policy

$\pi \leftarrow \pi'$
Policy iteration

High level idea:

policy iteration algorithm:

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

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

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

let’s evaluate $V^{\pi}(s)$!
Dynamic programming

Let’s assume we know $p(s'|s,a)$, and $s$ and $a$ are both discrete (and small)

![Transition probabilities](image)

16 states, 4 actions per state

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

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

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

just use the current estimate here

$\pi'(a_t|s_t) = \begin{cases} 1 & \text{if } a_t = \text{arg max}_a A^\pi(s_t,a_t) \\ 0 & \text{otherwise} \end{cases}$

deterministic policy $\pi(s) = a$

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

Policy iteration:
1. evaluate $V^\pi(s)$
2. set $\pi \leftarrow \pi'$

$$\pi'(a_t|s_t) = \begin{cases} 
1 & \text{if } a_t = \text{arg max}_{a_t} A^\pi(s_t, a_t) \\
0 & \text{otherwise}
\end{cases}$$

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

16 states, 4 actions per state

16 × 16 × 4 tensor

Generate samples (i.e. run the policy)

Fit a model to estimate return

Improve the policy

$\pi \leftarrow \pi'$
Even simpler dynamic programming

\[ \pi'(a_t|s_t) = \begin{cases} 
1 & \text{if } a_t = \arg \max_{a_t} A^\pi(s_t, a_t) \\
0 & \text{otherwise} 
\end{cases} \]

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

\[ \arg \max_{a_t} A^\pi(s_t, a_t) = \arg \max_{a_t} Q^\pi(s_t, a_t) \]

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

skip the policy and compute values directly!

value iteration algorithm:

1. set \( Q(s, a) \leftarrow r(s, a) + \gamma E[V(s')] \)
2. set \( V(s) \leftarrow \max_a Q(s, a) \)

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

arg \( \max_a Q(s, a) \) \rightarrow \text{policy}

argmax_a Q(s, a) \rightarrow \text{policy}

approximates the new value!

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

fit a model to estimate return

generate samples (i.e. run the policy)

improve the policy

\[ V^\pi(s) \leftarrow \max_a Q^\pi(s, a) \]
Fitted value iteration

how do we represent $V(s)$?

big table, one entry for each discrete $s$

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

\[
\begin{align*}
  s = 0 &: V(s) = 0.2 \\
  s = 1 &: V(s) = 0.3 \\
  s = 2 &: V(s) = 0.5
\end{align*}
\]

parameters $\phi$

\[
L(\phi) = \frac{1}{2} \left\| V_\phi(s) - \max_a Q^\pi(s, a) \right\|^2
\]

fitted value iteration algorithm:

1. set $y_i \leftarrow \max_{a_i} (r(s_i, a_i) + \gamma E[V_\phi(s'_i)])$
2. set $\phi \leftarrow \arg \min_{\phi} \frac{1}{2} \sum_i \left\| V_\phi(s_i) - y_i \right\|^2$

cause of dimensionality

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

(more than atoms in the universe)

\[
Q^\pi(s, a) \leftarrow r(s, a) + \gamma E_{s' \sim p(s' | s, a)}[V^\pi(s')]
\]
What if we don’t know the transition dynamics?

fitted value iteration algorithm:
1. set $y_i \leftarrow \max_{a_i} (r(s_i, a_i) + \gamma E[V_{\phi}(s_i')])$
2. set $\phi \leftarrow \arg\min_{\phi} \frac{1}{2} \sum_i \|V_{\phi}(s_i) - y_i\|^2$

need to know outcomes for different actions!

Back to policy iteration...

policy iteration:
1. evaluate $Q^\pi(s, a)$
2. set $\pi \leftarrow \pi'$

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

policy evaluation:

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

$Q^\pi(s, a) \leftarrow r(s, a) + \gamma \mathbb{E}_{s' \sim p(s'|s, a)}[Q^\pi(s', \pi(s'))]$
Can we do the “max” trick again?

**Policy Iteration:**
1. Evaluate $V^\pi(s)$
2. Set $\pi \leftarrow \pi'$

**Fitted Value Iteration Algorithm:**
1. Set $y_i \leftarrow \max_{a_i} (r(s_i, a_i) + \gamma E[V_\phi(s'_i)])$
2. Set $\phi \leftarrow \arg \min_{\phi} \frac{1}{2} \sum_i \|V_\phi(s_i) - y_i\|^2$

Forget policy, compute value directly.

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

**Fitted Q Iteration Algorithm:**
1. Set $y_i \leftarrow r(s_i, a_i) + \gamma E[V_\phi(s'_i)]$ \(\approx \max_{a'_i} Q_\phi(s'_i, a'_i)\)
2. Set $\phi \leftarrow \arg \min_{\phi} \frac{1}{2} \sum_i \|Q_\phi(s_i, a_i) - y_i\|^2$

Approximate $E[V(s'_i)]$ 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 \( \{(s_i, a_i, s'_i, r_i)\} \) using some policy
2. set \( y_i \leftarrow r(s_i, a_i) + \gamma \max_{a'_i} Q_\phi(s'_i, a'_i) \)
3. set \( \phi \leftarrow \arg \min_\phi \frac{1}{2} \sum_i \|Q_\phi(s_i, a_i) - y_i\|^2 \)

parameters

dataset size \( N \), collection policy
iterations \( K \)
gradiant steps \( S \)
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
Break
Why is this algorithm off-policy?

full fitted Q-iteration algorithm:

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

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

3. set $\phi \leftarrow \arg \min_{\phi} \frac{1}{2} \sum_i \|Q_\phi(s_i, a_i) - y_i\|^2$

this approximates the value of $\pi'$ at $s'_i$

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

given $s$ and $a$, transition is independent of $\pi$
What is fitted Q-iteration optimizing?

full fitted Q-iteration algorithm:

1. collect dataset \{ (s_i, a_i, s'_i, r_i) \} using some policy

2. set \( y_i \leftarrow r(s_i, a_i) + \gamma \max_{a'_i} Q_\phi(s'_i, a'_i) \)

3. set \( \phi \leftarrow \arg \min_\phi \frac{1}{2} \sum_i \| Q_\phi(s_i, a_i) - y_i \|^2 \)

error \( E \)

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

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

this is an optimal Q-function, corresponding to optimal policy \( \pi' \):

\[
\pi'(a_t | s_t) = \begin{cases} 
1 & \text{if } a_t = \arg \max_{a_t} Q_\phi(s_t, a_t) \\
0 & \text{otherwise}
\end{cases}
\]

maximizes reward sometimes written \( Q^* \) and \( \pi^* \)

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

**online Q iteration algorithm:**

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

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

fit a model to estimate return

generate samples (i.e. run the policy)

improve the policy

\[ a = \arg \max_a Q_\phi(s, a) \]

off policy, so many choices here!
Exploration with Q-learning

online Q iteration algorithm:
1. take some action $a_i$ and observe $(s_i, a_i, s'_i, r_i)$
2. $y_i = r(s_i, a_i) + \gamma \max_{a'} Q_\phi(s'_i, a'_i)$
3. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(s_i, a_i)(Q_\phi(s_i, a_i) - y_i)$

final policy:

\[
\pi(a_t|s_t) = \begin{cases} 
1 & \text{if } a_t = \arg\max_{a_t} Q_\phi(s_t, a_t) \\
\frac{\epsilon}{(|A| - 1)} & \text{otherwise}
\end{cases}
\]

why is this a bad idea for step 1?

“epsilon-greedy”

\[
\pi(a_t|s_t) \propto \exp(Q_\phi(s_t, a_t))
\]

“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

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

- fit a model to estimate return
- generate samples (i.e. run the policy)
- improve the policy
- \( a = \arg \max_a Q_\phi(s, a) \)
Value function learning theory

value iteration algorithm:

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

does it converge? and if so, to what?

stacked vector of rewards at all states for action $a$

define an operator $B$: $BV = \max_a r_a + \gamma T_a V$

matrix of transitions for action $a$ such that $T_{a,i,j} = p(s' = i|s = j, a)$

$V^*$ is a fixed point of $B$ $V^*(s) = \max_a r(s, a) + \gamma E[V^*(s')]$, so $V^* = BV^*$

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 $Q(s, a) \leftarrow r(s, a) + \gamma E[V(s')]$
2. set $V(s) \leftarrow \max_a Q(s, a)$

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

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

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

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

gap always gets smaller by $\gamma$!
(with respect to $\infty$-norm)

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

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

1. $V \leftarrow \PiBV$

fitted value iteration algorithm:

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

what does this do?

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

$\Pi$ is a projection onto $\Omega$ (in terms of $\ell_2$ norm)

updated value function

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

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

set $\Omega$ (e.g., neural nets)
Non-tabular value function learning

fitted value iteration algorithm (using $\mathcal{B}$ and $\Pi$):
1. $V \leftarrow \Pi \mathcal{B} V$

$\mathcal{B}$ is a contraction w.r.t. $\infty$-norm ("max" norm)

$\Pi$ is a contraction w.r.t. $\ell_2$-norm (Euclidean distance)

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 $y_i \leftarrow r(s_i, a_i) + \gamma E[V_\phi(s_i')]$
2. set $\phi \leftarrow \arg \min_\phi \frac{1}{2} \sum_i \|Q_\phi(s_i, a_i) - y_i\|^2$

define an operator $B$: $BQ = r + \gamma T \max_a Q$

max now after the transition operator

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

fitted Q-iteration algorithm (using $B$ and $\Pi$):

1. $Q \leftarrow \Pi BQ$

$B$ is a contraction w.r.t. $\infty$-norm ("max" norm)

$\Pi$ is a contraction w.r.t. $\ell_2$-norm (Euclidean distance)

$\Pi B$ 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 \( a_i \) and observe \( (s_i, a_i, s'_i, r_i) \)
2. \( y_i = r(s_i, a_i) + \gamma \max_{a'} Q(\phi)(s'_i, a'_i) \)
3. \( \phi \leftarrow \phi - \alpha \frac{dQ(\phi)}{d\phi} (s_i, a_i) (Q(\phi)(s_i, a_i) - y_i) \)

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

Q-learning is not gradient descent!

\[
\phi \leftarrow \phi - \alpha \frac{dQ(\phi)}{d\phi} (s_i, a_i) (Q(\phi)(s_i, a_i) - [r(s_i, a_i) + \gamma \max_{a'} Q(\phi)(s'_i, a'_i)])
\]

no gradient through target value
A sad corollary

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

\( \ell_\infty \) contraction \( \mathcal{B} \) (but without max)

\[ y_{i,t} \approx r(s_{i,t}, a_{i,t}) + \gamma \hat{V}_\phi^\pi(s_{i,t+1}) \]

\[ \mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}_\phi^\pi(s_i) - y_i \right\|^2 \]

\( \ell_2 \) contraction \( \Pi \)

fitted bootstrapped policy evaluation doesn’t converge!

An aside regarding terminology

\( V^\pi \): value function for policy \( \pi \)
this is what the critic does

\( V^* \): value function for optimal policy \( \pi^* \)
this is what value iteration does
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

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

fit a model to estimate return

generate samples (i.e. run the policy)

improve the policy

a = \arg\max_a Q_\phi(s, a)