Approximate Value and Policy Iteration

Sanjiban Choudhury
The story thus far …
The story thus far

We know how to define an MDP

If the MDP is **known** (i.e. I know my costs and my transition)

We know how to solve a MDP

What happens if the MDP is **unknown**?
Known MDP

*If I know the transition function, I could teleport to any state, try any action and know the next state*
I don't know the transition, I can only roll-out from start state, and see where I end up
Recall: How do we solve a known MDP?
Value Iteration

Initialize value function at last time-step

\[ V^*(s, T - 1) = \min_a c(s, a) \quad \forall s \]

for \( t = T - 2, \ldots, 0 \)

Compute value function at time-step \( t \)

\[ V^*(s, t) = \min_a \left[ c(s, a) + \gamma \sum_{s'} \mathcal{T}(s' | s, a)V^*(s', t + 1) \right] \quad \forall s \]
Q-Value Iteration

Initialize value function at last time-step

\[ Q^*(s, a, T - 1) = c(s, a) \quad \forall (s, a) \]

for \( t = T - 2, \ldots, 0 \)

Compute value function at time-step \( t \)

\[ Q^*(s, a, t) = c(s, a) + \gamma \sum_{s'} \mathcal{T}(s'|s, a) \min_{a'} Q^*(s', a', t + 1) \quad \forall s, a \]
Q-Value Iteration (Infinite horizon)

Initialize value function at last time-step

\[ Q^*(s, a) = c(s, a) \quad \forall (s, a) \]

While not converged

Update value function

\[ Q^*(s, a) = c(s, a) + \gamma \sum_{s'} \mathcal{T}(s'|s, a) \min_{a'} Q^*(s', a') \quad \forall (s, a) \]
Two Problems

Initialize value function at last time-step

\[ Q^*(s, a) = c(s, a) \quad \forall (s, a) \]

While not converged

Update value function

\[ Q^*(s, a) = c(s, a) + \gamma \sum_{s'} \mathcal{T}(s' \mid s, a) \min_{a'} Q^*(s', a') \quad \forall (s, a) \]

1) What happens when states are continuous?

2) What happens when I don’t know the MDP?

Are these known?

Can I do this?
Simple Idea

Can I collect roll-out data from the real world and just fit a Q function?
Step 1: First collect roll-out data

Data is a tuple of state, action, cost, next state

\[ \mathcal{D} = \{(s_i, a_i, c_i, s_{i+1})\}_{i=1}^{n} \]
Step 2: Fitted Q-Iteration

Regular Q-iteration

\[ Q(s,a) \leftarrow c(s,a) \]

while not converged do
  for \( s \in S, a \in A \)
    \[ Q^{\text{new}}(s,a) = c(s,a) + \gamma \mathbb{E}_{s',a'} \min Q(s',a') \]
    \[ Q \leftarrow Q^{\text{new}} \]

return \( Q \)
Step 2: Fitted Q-Iteration

Regular Q-iteration

\[ Q(s, a) \leftarrow c(s, a) \]

while not converged do
  for \( s \in S, a \in A \)
    \[ Q^{\text{new}}(s, a) = c(s, a) + \gamma \mathbb{E}_{s', a'} \min_{a'} Q(s', a') \]
  \( Q \leftarrow Q^{\text{new}} \)

return \( Q \)

---

Fitted Q-iteration

Init \( Q_\theta(s, a) \leftarrow 0 \)

while not converged do
  \( D \leftarrow \emptyset \)
  for \( i \in 1, \ldots, n \)
    input \( \leftarrow \{ s_i, a_i \} \)
    target \( \leftarrow c_i + \gamma \min_{a'} Q_\theta(s_i', a') \)
    \( D \leftarrow D \cup \{ \text{input}, \text{output} \} \)
  \( Q_\theta \leftarrow \text{Train}(D) \)

return \( Q_\theta \)

Given \( \{ s_i, a_i, c_i, s_i' \}_{i=1}^N \)
Step 2: Fitted Q-Iteration

**Fitted Q-Iteration**

Given \( \{s_i, a_i, c_i, s'_i\}_{i=1}^N \)

1. **Init** \( Q_\theta(s, a) \leftarrow 0 \)
2. **while** not converged **do**
   - \( D \leftarrow \emptyset \)
   - **for** \( i \in 1, \ldots, n \) **do**
     1. **Use old copy of** \( Q \)
     2. **to set target**
     3. **input** \( \leftarrow \{s_i, a_i\} \)
     4. **target** \( \leftarrow c_i + \gamma \min_a Q_\theta(s'_i, a') \)
     5. \( D \leftarrow D \cup \{\text{input, output}\} \)
   - \( Q_\theta \leftarrow \text{Train}(D) \)
3. **return** \( Q_\theta \)

---

**Training is a regression problem**

\[
\ell(\theta) = \sum_{i=1}^{n} (Q_\theta(s_i, a_i) - \text{target})^2
\]
Temporal Difference Error (TD Error)

Penalize violation of Bellman Equation

\[
\ell(\theta) = \left( c(s, a) + \gamma \min_{a'} Q_{\theta_{old}}(s', a') - Q_{\theta}(s_t, a_t) \right)^2
\]

\[\theta = \theta_{old} - \alpha \nabla_{\theta} l(\theta)\]
What policy do I use to collect data?

Do I explore randomly? Do I use my learnt Q function?
What policy do I use to collect data?

Do I explore randomly? Do I use my learnt Q function?
Q-learning: Learning off-policy

$(s, a, s', c)$

$\pi(a|s)$ (e.g., $\epsilon$-greedy)

dataset of transitions ("replay buffer")

Fitted Q iteration
QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation
Training time

Reward: Grasp success determined by subtracting pre and post-drop images

Distributed RL

State, Action, Reward

Learned weights

Inference time

State: 472x472 Image and gripper aperture

Critic Function

Q(State, Action)

Q-Values

Action proposals

Cross-Entropy Method

arg max Q(State, Action)
Large-scale Q-learning with continuous actions (QT-Opt)

stored data from all past experiments \{ (s_i, a_i, s'_i) \}_i

live data collection

training buffers

off-policy \((s, a, s', r)\)

on-policy \((s, a, s', r)\)

labeled \((s, a, Q_T(s, a))\)

training threads

\[
\min_\theta \|Q_\theta(s, a) - Q_T(s, a)\|^2
\]

Bellman updaters

compute \(Q_T(s, a) = r + \max_{a'} Q_\theta(s', a')\)

Kalashnikov, Irpan, Pastor, Ibarz, Herzong, Jang, Quillen, Holly, Kalakrishnan, Vanhoucke, Levine. QT-Opt: Scalable Deep Reinforcement Learning of Vision-Based Robotic Manipulation Skills
So does approximate value iteration work?
A simple example: Gridworld

Continuous Gridworld

Optimal path

True value function

What happens when we run value iteration with a \textit{quadratic}?
What happens when we run value iteration with a quadratic?
What happens when we run value iteration with a \textit{quadratic}?
Another Example: Mountain Car!

Figure 8.1.4 shows the car-on-hill example.

Figure 8.1.5 shows that a two layer MLP can also diverge to underestimate the costs.
What happens when we run value iteration with a 2 Layer MLP?

Iteration 11

Figure 8.1.4 shows the car-on-hill example.

Figure 8.1.5 shows that a two layer MLP can also diverge to underestimate the costs.
What happens when we run value iteration with a 2 Layer MLP?

Figure 8.1 shows the car-on-hill example.

Car-on-the-Hill $J^*(\text{pos, vel})$

Figure 8.1.5 shows that a two layer MLP can also diverge to underestimate the costs.

Iteration 101

Training with neural network.
What happens when we run value iteration with a 2 Layer MLP?

Iteration 201
The problem of Bootstrapping!

max()
The problem of Bootstrapping!

Errors in approximation are amplified.

Key reason is the minimization

Minimization operation visit states where approximate values is less than the true value of that state – that is to say, states that look more attractive than they should.

Typically states where you have very few samples
What about policy iteration?
Policy Iteration

Policy Evaluation

Policy Improvement
Policy Iteration

Init with some policy $\pi$

Repeat forever

Evaluate policy

$$Q^\pi(s, a) = c(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s, a)} Q^\pi(s', \pi(s'))$$

Improve policy

$$\pi^+(s) = \arg \min_a Q^\pi(s, a)$$
Fitted Policy Iteration

Fitted policy evaluation

Given \( \{s_i, a_i, c_i, s'_i\}_{i=1}^{N} \)

Init \( Q_\theta(s, a) \leftarrow 0 \)

while not converged do
  \( D \leftarrow \emptyset \)
  for \( i \in 1, \ldots, n \)
    input \( \leftarrow \{s_i, a_i\} \)
    target \( \leftarrow c_i + \gamma Q_\theta(s'_i, \pi(s'_i)) \)
    \( D \leftarrow D \cup \{\text{input, output}\} \)
  \( Q_\theta \leftarrow \text{Train}(D) \)

return \( Q_\theta \)

Policy Improvement

Collect data using current policy \( \pi \)

This remains the same!

\( \pi^+(s) = \arg\min_a Q^\pi(s, a) \)
Fitted Policy Iteration

**Fitted policy evaluation**

Given \( \{s_i, a_i, c_i, s_i'\}_{i=1}^N \)

Init \( Q_\theta(s, a) \leftarrow 0 \)

while not converged do

This is fine..

for \( i \in 1, \ldots, n \)

input \( \leftarrow \{s_i, a_i\} \)

target \( \leftarrow c_i + \gamma Q_\theta(s'_i, \pi(s'_i)) \)

\( D \leftarrow D \cup \{\text{input, output}\} \)

\( Q_\theta \leftarrow \text{Train}(D) \)

return \( Q_\theta \)

\[ \pi^+(s) = \arg \min_a Q^{\pi}(s, a) \]

Policy Improvement

**Fitted policy evaluation**

Given \( \{s_i, a_i, c_i, s_i'\}_{i=1}^N \)

Init \( Q_\theta(s, a) \leftarrow 0 \)

while not converged do

No min()

for \( i \in 1, \ldots, n \)

input \( \leftarrow \{s_i, a_i\} \)

target \( \leftarrow c_i + \gamma Q_\theta(s'_i, \pi(s'_i)) \)

\( D \leftarrow D \cup \{\text{input, output}\} \)

\( Q_\theta \leftarrow \text{Train}(D) \)

return \( Q_\theta \)

\[ \pi^+(s) = \arg \min_a Q^{\pi}(s, a) \]
Fitted Policy Iteration

Fitted policy evaluation

Given \( \{s_i, a_i, c_i, s'_i\}_{i=1}^N \)

Init \( Q_\theta(s, a) \leftarrow 0 \)

while not converged do

This is fine..

for \( i \in 1, \ldots, n \)

input \( \leftarrow \{s_i, a_i\} \)

target \( \leftarrow c_i + \gamma Q_\theta(s'_i, \pi(s'_i)) \)

\( D \leftarrow D \cup \{\text{input}, \text{output}\} \)

\( Q_\theta \leftarrow \text{Train}(D) \)

return \( Q_\theta \)

But this has the \( \text{min}() \) step!

\[ \pi^+(s) = \arg\min_a Q_\pi(s, a) \]