Lecture 9 - Iterative methods

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

DDA4230: Reinforcement Learning
Course Page: [Click]

DDA 4230 Resources

Check our course page.

Please post your question on the discussion board in the BlackBoard (BB) system.

- Step 1: Search for existing questions.
- Step 2: Create a thread.
- Step 3: Post your question.

Course Page Link (all the course relevant materials will be posted here):

https://guiliang.github.io/courses/cuhk-dda-4230/dda_4230.html

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Recap: Discrete-time Markov Decision Process (MDP)

Discrete-time Markov decision process (MDP), denoted as the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0, \gamma)$.

- $\mathcal{S}$ the state space;
- $\mathcal{A}$ the action space. $\mathcal{A}$ can depend on the state s for $s \in \mathcal{S}$;
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ the environment transition probability function;
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$ the reward function;
- $\rho_0 \in \Delta(\mathcal{S})$ the initial state distribution;
- $\gamma \in [0, 1]$ the discount factor.

Note that $\Delta(\mathcal{X})$ denotes the set of all distributions over set $\mathcal{X}$.

Recap: Discrete-time Markov Decision Process (MDP)

A stationary MDP follows for $t = 0, 1, \ldots$ as below, starting with $s_0 \sim \rho_0$.

- The agent observes the current state s_t;
- The agent chooses an action $a_t \sim \pi(a_t \mid s_t)$;
- The agent receives the reward $r_t \sim P_{\mathcal{R}}(s_t, a_t)$;
- The environment transitions to a subsequent state according to the Markovian dynamics $s_{t+1} \sim P_{\mathcal{T}}(s_t, a_t)$.

This process generates the sequence $s_0, a_0, r_0, s_1, \ldots$ indefinitely. The sequence up to time t is defined as the trajectory indexed by t, as $\tau_t = (s_0, a_0, r_0, s_1, \ldots, r_t)$.

Recap: Discrete-time Markov Decision Process (MDP)

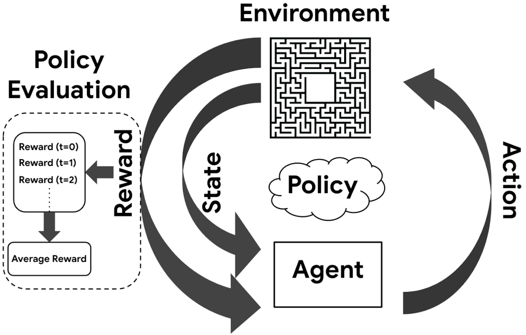
The goal is to optimize the expected discounted cumulative return

$$\mathbb{E}_{s_t, a_t, r_t, t \geq 0}[R_0] = \mathbb{E}_{s_t, a_t, r_t, t \geq 0}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

over the agent's policy π.

Policy Evaluation

Policy Evaluation (PE): compute the value function given a fixed policy.



Recap: The Bellman Equation

- State-value Bellman equation (named after Richard E. Bellman):

$$V(s_t) = \mathbb{E}[r_t + \gamma V(s_{t+1})] \quad \text{and} \quad V(s_T) = \mathbb{E}[r_T]\,.$$

  for non-terminal and terminal states, respectively.

- Action-value Bellman equation:

$$Q(s_t, a_t) = \mathbb{E}[r_t + \gamma Q(s_{t+1}, a) \mid a \sim \pi(a \mid s_{t+1})] \quad \text{and} \quad Q(s_T, a_T) = \mathbb{E}[r_T]$$

  for non-terminal and terminal states, respectively.

Iterative Policy Evaluation

The iterative policy evaluation algorithm constructs a contraction when $\gamma < 1$, which gives an arbitrarily close value function estimation of a given policy.

- The update $V(s) = \sum_a \pi(a \mid s) \sum_{s',r} \mathbb{P}(s', r \mid s, a) [r + \gamma V(s')]$ forms a contraction, such that given V, V', $\|BV - BV'\|_\infty \leq \|V - V'\|_\infty$ where B denotes the operator.

Algorithm 1: Iterative policy evaluation

Input: Policy π, threshold $\epsilon > 0$
Output: Value function estimation $V \approx V^\pi$
Initialize $\Delta > \epsilon$ and V arbitrarily
while $\Delta > \epsilon$ **do**
    $\Delta = 0$
    for $s \in \mathcal{S}$ **do**
        $v = V(s)$
        $V(s) = \sum_a \pi(a \mid s) \sum_{s',r} \mathbb{P}(s', r \mid s, a) [r + \gamma V(s')]$
        $\Delta = \max(\Delta, |v - V(s)|)$

Iterative Policy Evaluation

The iterative policy evaluation algorithm constructs a contraction when $\gamma < 1$, which gives an arbitrarily close value function estimation of a given policy.

- similarly, we can replace the "state-value Bellman equation" with the "action-value Bellman equation".

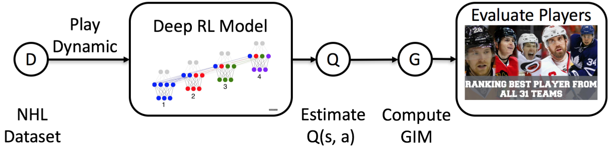
Algorithm 2: Iterative policy evaluation

Input: Policy π, threshold $\epsilon > 0$
Output: Action-Value function estimation $Q \approx Q^\pi$
Initialize $\Delta > \epsilon$ and V arbitrarily
while $\Delta > \epsilon$ **do**
  $\Delta = 0$
  for $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
    $q = Q(s, a)$
    $Q(s, a) = \sum_{s', r} \mathbb{P}(s', r \mid s, a) \left[r + \gamma \sum_a' \pi(a' \mid s') Q(s', a') \right]$
    $\Delta = \max(\Delta, |q - Q(s, a)|)$

Iterative Policy Evaluation

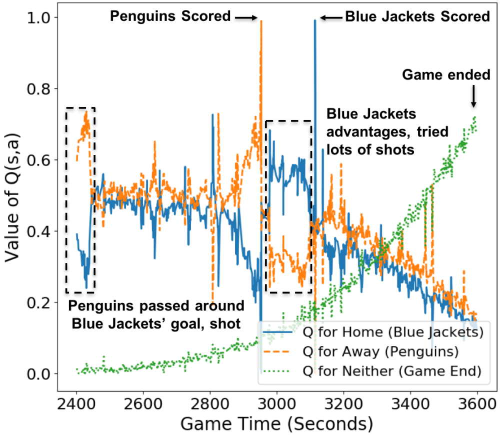
Application: Player evaluation in Sports Analytics. Players are rated by their observed performance over a set of games. Given dynamic game tracking data:

- Apply policy evaluation to estimate the *value* function $V(s)$ and the *action value* function $Q(s, a)$.
- Compute the player evaluation metric based on the aggregated impact (GIM, i.e., advantages) of their actions over the entire game or season.



Iterative Policy Evaluation

Temporal visualization of Q values over a game:



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Dynamic programming

For **a finite horizon MDP**, the iterative policy evaluation algorithm requires the iteration to go through the index with a non-stationary value function. This process is known as dynamic programming. By the Bellman equation,

$$
\begin{aligned}
& V_t(s) = R(s) + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s, \pi) V_{t+1}(s') \ , \ \forall\, t = 0, \dots, H-1 \, , \\
& V_T(s) = 0 \, .
\end{aligned} \tag{1}
$$

For episodic MDPs, R and $\mathbb{P}$ can be stochastic and we run this process for many episodes (usually denoted as T/H episodes with horizon H).

Dynamic programming

Algorithm 2: Iterative policy evaluation with finite horizon

Input: $\mathcal{S}, \mathbb{P}, \mathcal{R}, T$

For all states $s \in \mathcal{S}$, $V_T(s) \leftarrow 0$

$t \leftarrow T - 1$

while $t \geq 0$ **do**

  For all states $s \in S$, $V_t(s) = \sum_a \pi(a \mid s) \sum_{s',r} \mathbb{P}(s', r \mid s, a) \left[r + \gamma V_{t+1}(s')\right]$

  $t \leftarrow t - 1$

return $V_t(s)$ for all $s \in S$ and $t = 0, \ldots, T$

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

Iterative Policy Search

The policy evaluation algorithm immediately renders itself to a brute force algorithm called policy search to find the optimal value function V^* and an optimal policy π^*.

- The input is an infinite horizon MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, \mathcal{R}, \gamma)$ with arbitrary initial state distribution ρ_0 and a tolerance ε for accuracy of policy evaluation,

Algorithm 3: Policy search

```
Input: M, ε
Π ← All stationary deterministic policies of M
π* ← Randomly choose a policy π ∈ Π
V* ← POLICY EVALUATION (M, π*, ε)
for π ∈ Π do
    V^π ← POLICY EVALUATION (M, π, ε)
    if V^π(s) ≥ V*(s) ∀ s ∈ S then
        V* ← V^π
        π* ← π
return V*(s), π*(s) for all s ∈ S
```

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

Iterative Policy Search

The policy evaluation algorithm immediately renders itself to a brute force algorithm called policy search to find the optimal value function V^* and an optimal policy π^*.

- The Algorithm terminates as it checks all $|\Pi| = |\mathcal{A}|^{|\mathcal{S}|} = m^n$ deterministic stationary policies (Recall that we are assuming that there exists an optimal policy and in this case there is a deterministic stationary policy that is optimal).
- The run-time complexity of this algorithm is $O(|\mathcal{A}|^{|\mathcal{S}|})$.

Lemma

Policy Search returns the optimal value function and an optimal policy when $\varepsilon = 0$.

Recap: The Bellman Optimality Equation

The Bellman optimality equation, named after Richard E. Bellman, is a necessary condition for a value function to be optimal:

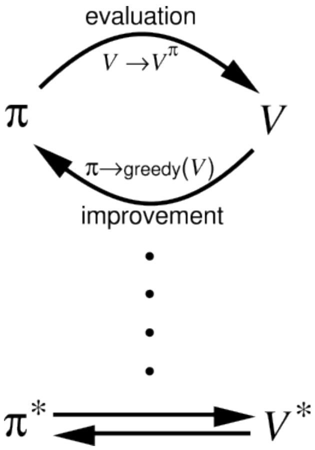
$$V^*(s_t) = \max_a \mathbb{E}[r_t + \gamma V^*(s_{t+1}) \mid a_t = a].$$

The Bellman optimality equation $\neq$ The Bellman equation.

- The Bellman equation describes an arbitrary policy's value function $V(s_t) = \mathbb{E}[r_t + \gamma V(s_{t+1})]$ (expected w.r.t. $\pi(a_t|s_t)$).
- The Bellman optimality equation takes the maximum overall actions (no policy in the expectation).

Recap: The Bellman Optimality Equation

Can we try iterative policy evaluation and improvement?



Policy Iteration

The policy iteration algorithm applies the Bellman operator (Bellman optimality equation and Bellman equation), which shows that given any stationary policy π, we can find a deterministic stationary policy that is no worse than the existing policy.

Algorithm 4: Policy improvement

Input: V^π

$\hat{\pi}(s) \leftarrow \arg\max_{a \in \mathcal{A}} \left[R(s,a) + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s,a) V^\pi(s') \right], \forall\, s \in S$

return $\hat{\pi}(s)$ for all $s \in S$

The output of Algorithm 4 is at least as good as the policy π corresponding to the input value function V^π, and represents a greedy attempt to improve the policy.

Algorithm 5: Policy iteration

Input: $\mathcal{M}, \epsilon$

$\pi \leftarrow$ Randomly choose a policy $\pi \in \Pi$

while *true* **do**

  $V^\pi \leftarrow$ POLICY EVALUATION $(\mathcal{M}, \pi, \epsilon)$

  $\pi^* \leftarrow$ POLICY IMPROVEMENT $(\mathcal{M}, V^\pi)$

  if $V^{\pi^*} = V^\pi$ **then**

    break

  else

    $\pi \leftarrow \pi^*$

$V^* \leftarrow V^\pi$

return $V^*(s)$, $\pi^*(s)$ for all $s \in S$

g, Shenzhen

Policy Iteration

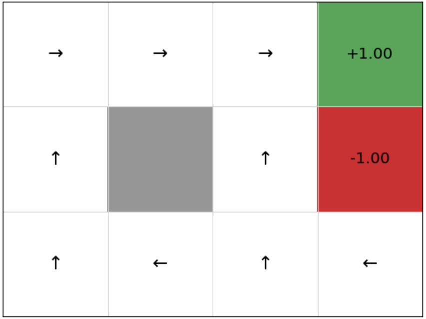
Lemma

Consider an infinite horizon MDP with $\gamma < 1$. The following statements hold.

1. *When Algorithm 5 is run with $\varepsilon = 0$, it finds the optimal value function and an optimal policy.*
2. *If the policy does not change during a policy improvement step, then the policy cannot improve in future iterations.*
3. *The value functions corresponding to the policies in each iteration of the algorithm form a non-decreasing sequence for every $s \in S$.*

Policy Iteration

Policy iteration in Grid World.



Recap: The Bellman Optimality Equation

The Bellman optimality equation, named after Richard E. Bellman, is a necessary condition for a value function to be optimal:

$$V^*(s_t) = \max_a \mathbb{E}[r_t + \gamma V^*(s_{t+1}) \mid a_t = a].$$

Value Iteration (VI). If we replace V^* by a not-necessarily optimal value function V, VI assigns RHS to V and repeats the iteration:

$$V(s_t) \leftarrow \max_a \mathbb{E}[r_t + \gamma V(s_{t+1}) \mid a_t = a].$$

This leads to improvements of the current value for each iteration and V will converge to the optimal value function under some conditions.

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Value Iteration

Value Iteration computes the optimal value function and an optimal policy given a known MDP. For every element $V \in \mathbb{R}^n$ the Bellman optimality backup operator B^* is defined as:

$$(B^* V)(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s,a) V(s') \right], \ \forall\, s \in S\,. \tag{1}$$

Value Iteration

Theorem

For an MDP with $\gamma < 1$, let the fixed point of the Bellman optimality backup operator B^* be denoted by $V^* \in \mathbb{R}^n$. Then the policy given by

$$\pi^*(s) = \arg\max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s, a) V^*(s') \right], \forall\, s \in S \tag{1}$$

will be a stationary deterministic policy. The value function of this policy V^{π^*} satisfies the identity $V^{\pi^*} = V^*$, and V^* is also the fixed point of the operator B^{π^*}.

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Value Iteration

The above theorem suggests a straightforward way to calculate the optimal value function V^* and an optimal policy π^*. The idea is to run fixed point iterations to find the fixed point of B^*. Once we have V^*, an optimal policy π^* can be extracted using the $\arg\max$ operator in the Bellman optimality equation.

Algorithm 6: Value iteration

Input: ϵ

For all states $s \in S$, $V'(s) \leftarrow 0$, $V(s) \leftarrow \infty$

while $\|V - V'\|_\infty > \epsilon$ **do**

  $V \leftarrow V'$

  For all states $s \in S$, $V'(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s,a) V(s') \right]$

$V^* \leftarrow V$ for all $s \in S$

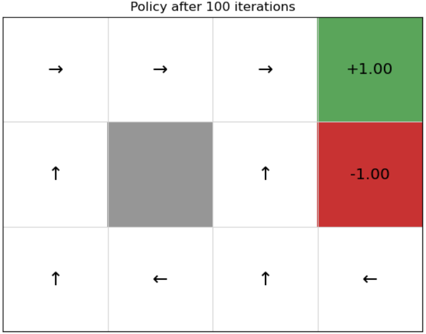
$\pi^* \leftarrow \arg\max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s,a) V^*(s') \right] \quad , \forall\, s \in S$

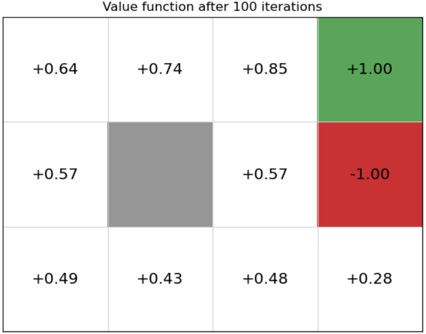
return $V^*(s)$, $\pi^*(s)$ for all $s \in S$

香 港 中 文 大 學 (深 圳)
The Chinese University of Hong Kong, Shenzhen

Value Iteration

Value Iteration in Grid World.





Question and Answering (Q&A)