Lecture 10 - Iterative methods

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

DDA4230: Reinforcement Learning

Course Page: [Click]

DDA 4230 Resources

Please join our Slack group.

https://join.slack.com/t/slack-us51977/shared_invite/zt-22g8b40v8-0qSs9o0G3~8hXHwWydlCpw

Please check our course page.

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

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

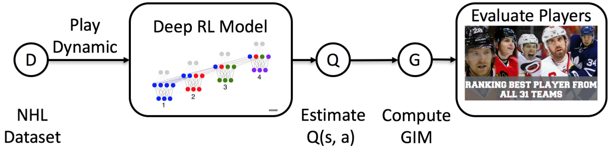
Input: Policy π, threshold $\epsilon > 0$
Output: Value function estimation $V \approx V^\pi$
Initialize $\Delta > \epsilon$ and V arbitrarily
while $\Delta > \epsilon$ **do**
    $\Delta = 0$
    for $s \in \mathcal{S}$ **do**
        $v = V(s)$
        $V(s) = \sum_a \pi(a \mid s) \sum_{s',r} \mathbb{P}(s',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

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

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, we:

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



港中文大學(深圳)
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,

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

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).

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

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 $t = 0, \ldots, T$

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: $\mathcal{M}, \epsilon$

$\Pi \leftarrow$ All stationary deterministic policies of M

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

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

for $\pi \in \Pi$ **do**

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

  if $V^\pi(s) \geq V^*(s)\ \forall\ s \in S$ **then**

    $V^* \leftarrow V^\pi$

    $\pi^* \leftarrow \pi$

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

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 $\varepsilon = 0$.

Policy Iteration

The policy iteration algorithm applies the Bellman operator, 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

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

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 $s \in S$

Policy Iteration

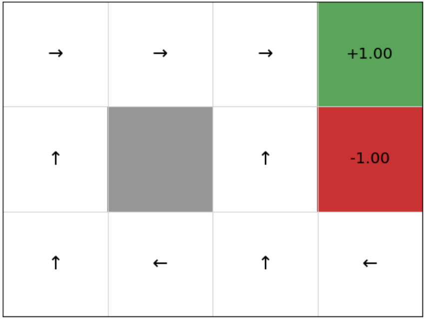
Lemma

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.



Value Iteration

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

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

Value Iteration

Theorem

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

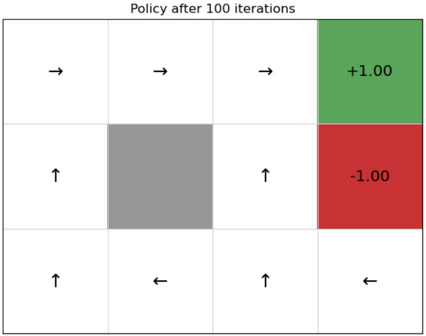
$V^* \leftarrow V$ for all $s \in S$

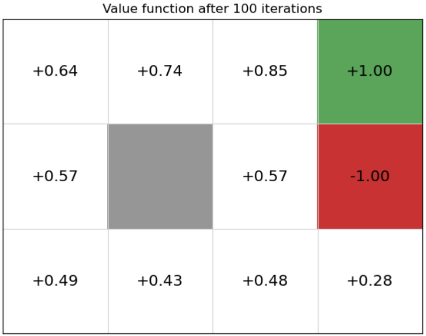
$\pi^* \leftarrow \arg\max\limits_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s,a) V^*(s')\right] \quad, \forall\, s \in S$

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

Value Iteration

Value Iteration in Grid World.





Question and Answering (Q&A)