DDA4230 Reinforcement learning Iterative methods

Lecture 9

Lecturer: Guiliang Liu *Scribe: Baoxiang Wang*

1 Goal of this lecture

In this lecture we will introduce the iterative approaches for discrete MDPs, including policy evaluation, policy iteration, value iteration, and the their connections to Q-learning. These iterative will eventually extends to deep settings and are thus very important in RL.
Suggested reading: Chapter 3, 4 and 5 of *Reinforcement learning: An introduction*; Chapter 1 and 2 of *Reinforcement learning: Theory and algorithms.*

2 Recap: Solving discrete MDPs directly

For policy evaluation, by the Bellman equation we can obtain the value function as

$$V = (I - \gamma P)^{-1} r\,.$$

When P and r are unknown, a model-based methods is to estimate them before computing V.

For value optimization, by the Bellman optimality equation it amounts to solve

$$\begin{aligned} \underset{V}{\text{minimize}} \quad & \mathbf{e}^T V \\ \text{subject to} \quad & (I - \gamma P_j)V - r_j \geq 0\,, \quad j = 1, \dots, m\,, \end{aligned}$$

where $\mathbf{e}$ is the all-one vector. The dual of the linear program describes policy optimization

$$\begin{aligned} \underset{\lambda_1,\dots,\lambda_m}{\text{maximize}} \quad & \sum_j \lambda_j^T r_j \\ \text{subject to} \quad & \sum_j (I - \gamma P_j^T)\lambda_j = \mathbf{e}\,, \\ & \lambda_j \geq 0, \quad j = 1, \dots, m\,. \end{aligned}$$

When P and r are unknown, a model-based methods is to estimate them before computing V and λ_i.

3 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. As such, when the reward function is deterministic and $\gamma < 1$, the error of the estimation decreases at a rate of γ^t (the update $V(s) = \sum_a \pi(a \mid s) \sum_{s',r} \mathbb{P}(s', r \mid s, a) \left[r + \gamma V(s')\right]$ forms a contraction, such that given V, V', we have $\|BV - BV'\|_\infty \leq \|V - V'\|_\infty$ where B denotes the update operator).

Algorithm 1: Iterative policy evaluation

```
**Input:** Policy $\pi$, 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
        $v = V(s)$
        $V(s) = \sum_a \pi(a \mid s) \sum_{s',r} \mathbb{P}(s', r \mid s, a) \left[r + \gamma V(s')\right]$
        $\Delta = \max(\Delta, |v - V(s)|)$
```

Algorithm 2: Iterative policy evaluation

```
**Input:** Policy $\pi$, 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
        $q = Q(s, a)$
        $Q(s, a) = \sum_{s',r} \mathbb{P}(s', r \mid s, a) \left[r + \gamma \sum_a' \pi(a' \mid s') Q(s', a')\right]$
        $\Delta = \max(\Delta, |q - Q(s, a)|)$
```

3.1 Dynamic programming

It worth mention that 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 also known as dynamic programming. These two algorithms look surprisingly similar, to the point that it is hard to tell the difference. For 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).

These equations immediately lend themselves to a dynamic programming solution whose pseudo-code is outlined in Algorithm 3. The algorithm takes as input a finite horizon Markov reward process $\mathcal{S}, \mathbb{P}, \mathcal{R}$ (which is defined as a Markov chain with reward but without a policy), and computes the value function for all states and at all times.

Algorithm 3: 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 $t = 0, \ldots, T$
```

4 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^*, as described in pseudo-code in Algorithm 4. The algorithm takes as input 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, and returns the optimal value function and an optimal policy.

It is clear that Algorithm 4 always 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). Thus the run-time complexity of this algorithm is $O(|\mathcal{A}|^{|\mathcal{S}|})$. It is possible to prove the correctness of the algorithm when $\epsilon = 0$, i.e. when in each iteration the policy evaluation is done exactly. In practice ϵ is set to a small number such as from 10^{-9} to 10^{-12}. We encourage the reader to work out the questions and try implementing the examples.

Lemma 1 *Algorithm 4 returns the optimal value function and an optimal policy when* $\epsilon = 0$.

Algorithm 4: 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 $s \in S$
```

Proof: Let π^* be an optimal policy, and thus $V^{\pi^*}(s) = V^*(s)$ for all states $s \in S$. Since the algorithm checks every policy in Π, it means that π^* must get selected at some iteration of the algorithm. Thus for the policies considered in future iterations the value function can no longer strictly increase. Future iterations may select a different policy with the same optimal value function, thus completing the proof. $\square$

For the MDP examples shown in LN3, we leave it to the reader to discuss (a) How many deterministic stationary policies does the agent have? (b) If $\gamma < 1$, is the optimal policy unique? (c) If $\gamma = 1$, is the optimal policy unique?

4.1 Policy iteration

We now discuss a more efficient algorithm than policy search called policy iteration. The algorithm is a straightforward application of the Bellman operator, which states that given any stationary policy π, we can find a deterministic stationary policy that is no worse than the existing policy. In particular the theorem also applies to deterministic policies. This simple step has a special name called policy improvement, whose pseudo-code is presented in Algorithm 5.

Algorithm 5: Policy improvement

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

The output of Algorithm 5 is always guaranteed to be 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. When performed iteratively with the policy evaluation algorithm (Algorithm 1), this gives rise to the policy iteration algorithm. The pseudo-code of policy iteration is outlined in Algorithm 6.

Note that the algorithm will always terminate as there are a finite number of stationary deterministic policies, as a conclusion of the following lemma. This also establishes the worst-case run-time complexity of the algorithm.

Algorithm 6: 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 $s \in S$
```

Lemma 2 *Consider an infinite horizon MDP with $\gamma < 1$. The following statements hold.*

1. *When Algorithm 6 is run with $\epsilon = 0$, 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 $s \in S$.*

5 Value iteration

Value iteration is yet another technique that can be used to compute the optimal value function and an optimal policy given a known MDP. To motivate this method we will need the following theorem.

Suppose we are now given an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, \mathcal{R}, \gamma)$ and consider the finite dimensional Banach space $\mathbb{R}^n$ equipped with the infinity norm $\|\cdot\|_\infty$. Then for every element $U \in \mathbb{R}^n$ the Bellman optimality backup operator B^* is defined as

$$(B^*U)(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s,a) U(s') \right], \ \ \forall\, s \in S\ . \tag{2}$$

This operator converts the LFS of the Bellman optimality equation to its RHS, while the function to be converted could be any value function (not necessarily the optimal value function).

Theorem 3 *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*

$$\pi^*(s) = \arg\max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s,a) V^*(s') \right], \ \forall\, s \in S \tag{3}$$

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^*}.

In particular this implies that there exists a stationary deterministic policy π^* whose value function is the fixed point of B^*. Moreover, π^* is an optimal policy.

Proof: We start by noting that π^* as defined in (3) is a stationary deterministic policy, and so we can conclude that $R^{\pi^*}(s) = R(s, \pi^*(s))$ and $\mathbb{P}^{\pi^*}(s' \mid s) = \mathbb{P}(s' \mid s, \pi^*(s))$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

As V^* is the fixed point of B^*, namely $B^*V^* = V^*$, by (2) and (3) we can write for all $s \in \mathcal{S}$

$$\begin{aligned} V^*(s) &= \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s, a) V^*(s') \right] \\ &= R(s, \pi^*(s)) + \gamma \sum_{s' \in S} \mathbb{P}(s' \mid s, \pi^*(s)) V^*(s') \\ &= R^{\pi^*}(s) + \gamma \sum_{s' \in S} \mathbb{P}^{\pi^*}(s' \mid s) V^*(s') \\ &= V^{\pi^*}(s)\,. \end{aligned}$$

This completes the proof of the first part of the theorem.

To prove that π^* is an optimal policy, we show that if an optimal policy exists then its value function must be a fixed point of the operator B^*. Assume that an optimal policy exists. Then we can specify a stationary deterministic policy to be optimal, and let us denote it as μ and the corresponding optimal value function as V^μ. Suppose that V^μ is not a fixed point of B^*. Then there exists $s \in S$ such that $V^\mu(s) \neq (B^*V^\mu)(s)$, which implies that $V^\mu(s) > (B^*V^\mu)(s)$. Then the greedy policy $\hat{\pi}$ induced by $V^\mu(s)$ implies that there exists a policy $\hat{\pi}$ which is strictly better than μ, which leads to a contradiction. This proves that V^μ must be the unique fixed point of B^*. Combining this fact with the first part implies that V^* must be the optimal value function and π^* is an optimal policy. This completes the proof. □

Theorem 3 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. The pseudo-code of this algorithm is given in Algorithm 7, which takes as input an infinite horizon MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, \mathcal{R}, \gamma)$ and a tolerance ϵ, and returns the optimal value function and an optimal policy.

If Algorithm 7 is run with $\epsilon = 0$, we can recover the optimal value function and an optimal policy exactly. However in practice, ϵ is set to be a small number such as from 10^{-9} to 10^{-12}.

Acknowledgement

This lecture notes partially use material from *Reinforcement learning: An introduction*, *Reinforcement learning: Theory and algorithms*, and *CS234: Reinforcement learning* from Stanford.

Algorithm 7: Value iteration

```
Input: $\epsilon$
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 $s \in S$
```