EE 290 Theory of Multi-armed Bandits and Reinforcement Learning Lecture 16 - 3/11/2021

Lecture 16: Planning Algorithms for Markov Decision Processes

Lecturer: Jiantao Jiao *Scribe: Ilgin Dogan, Kathy Jang*

In this lecture, we present various Markov Decision Process (MDP) formulations, including finite horizon and indefinite horizon MDPs. We also begin discussion on well-known planning algorithms for MDPs, including policy iteration and value iteration.

1 MDP Formulations

Continuing from March 9's lecture introducing Bellman equations and MDP formulations, we continue our discussion on the latter. As a reminder, MDP formulations we have discussed thus far include: indefinite-horizon, undiscounted; infinite-horizon, discounted. To recap:

- **Indefinite-horizon, undiscounted**: Some states are terminal, the length of the trajectory is not fixed, and for terminal states s, the value function is $V(s) = 0$.
- **Infinite-horizon, discounted**: The length of the trajectory is infinite. The value function is discounted according to γ^{t-1}, where t denotes the timestep. This formulation encourages finishing.

Next, we move onto finite-horizon MDPs. In finite-horizon MDPs, all trajectories stop in precisely H steps. There is no "natural" terminal state, but termination is enforced "externally" via this H parameter. In particular, we discuss two variants: **finite-horizon, time-invariant** and **finite-horizon, time-varying**. In both of these variations, the MDP can be expressed as $(\mathcal{S}, \mathcal{A}, R, P, H)$, where H denotes the length of the finite time horizon. Note that that in the infinite-horizon version of this problem, we would have γ instead, the discount factor.

Now, we further discuss the two variations of finite-horizon MDP:

1.1 Finite-horizon, time-invariant

In this variation, the transition and reward have no dependence on time. That is, the transition function is given by $p(s \mid s, a)$, and the reward function is given by $r(s, a)$. This variation is in general considered to be quite unnatural, as an optimal policy is in general non-stationary.

1.2 Finite-horizon, time-variant

This variation is a special case of the above (finite-horizon, time-invariant). The main idea of this reduction is: decompose $S = \cup_{i=1}^{H} S_i$, where S_i are disjoint sets. We add the condition that at time t, you are only allowed to be in S_i. A couple caveats on this:

- If the cardinality of S_i is $\mid S_i \mid = m$, then $\mid S \mid = Hm$.
- Due to the instated condition, states cannot be revisited.
- This is considered the easiest variation to solve; caution should be heeded to papers citing success in this specific variation, as it does not imply success in more salient MDP formulations, such as finite-horizon, time-variant; or infinite-horizon, discounted.

These two finite-horizons complete the set of 3 major categories of MDP variations:

1. Infinite-horizon, discounted

2. Finite-horizon, time-invariant

3. Finite-horizon, time-variant

1.3 Comparing indefinite and fixed-horizon MDPs

How do we reduce an indefinite-horizon problem into a fixed-horizon problem?
This can be done by setting the horizon to be H. Add a dummy state, and if the indefinite-horizon problem ends early, loop in the dummy state such that termination occurs at exactly H.

How do we reduce a fixed-horizon problem into an indefinite-horizon problem?
This can be done by augmenting the state space. We create a new state space as $S' = Sx[H]$. Note that through the act of augmenting, we are losing the assumption that state transition functions are the same across time and that states are revisited. Thus, this reduction is not exact and is not so frequently used.

1.4 Uniqueness of indefinite-horizon: Sensitivity to Reward Shifting

In this section, we show that the indefinite-horizon is sensitive to reward shifting. Before that, we demonstrate how two other variants are insensitive to reward shifting.

First consider the infinite-horizon, discounted problem with $R(s,a) \rightarrow R(s,a) + c$. Does the optimal policy change? The answer is no. This is because the relative values of the value functions which are used to calculate the optimal policy do not change.

$$V^{\pi}(s) \rightarrow V^{\pi}(s) + \frac{C}{1-\gamma}$$

Second, consider either of the fixed-horizon cases. Does the optimal policy change? The answer is no for the exact same reason.

Then, we arrive at the indefinite-horizon case, which is in fact sensitive to reward shifting. Consider a Gridworld example to illustrate this concept. As a reminder, in Gridworld, the reward is -1 before reaching the goal. After reaching the goal, the reward is 0. What if we changed $-1 \rightarrow 1$ and $0 \rightarrow 2$? Now, the agent learns to maximize its time in Gridworld before reaching the goal. This is an example of how the positivity of reward actually has meaning.

2 MDP Planning Algorithms

The *planning* problem refers to the problem of computing the optimal policy π^* for a completely known MDP specification $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. A particular example is AlphaZero which is modelled as a multi-player planning problem in which both opponents know the rules, transitions and final reward of the game exactly.

An open concern that has not been well-addressed in the literature is the difference between a planning problem and a (reinforcement) learning problem which are two effective approaches in MDP optimization. In [1], the key difference between planning and RL is stated based on the accessibility to the MDP enviroment. RL can be regarded as operating under an "unknown/irreversible model" whereas planning assumes a "known/reversible model".

One of the traditional planning algorithms is dynamic programming (DP) that is a well-studied approach mostly in the Operations Research literature. However, the Bellman's equation that is the core of DP suffers from "the curse of dimensionality" in terms of the dimensions of the state space, the outcome space, and the action space. Thus, as a practical proxy to DP, approximate dynamic programming approaches have been proposed. Approximate DP approach is basically avoids enumerating and utilizes an approximation of the value function over a forward propagation process. For further information on these approaches, we

refer the reader to [2, 3]. However, approximate DP has been recently dominated by learning and statistical analyses perspectives. Therefore, we start by discussing the two other famous MDP planning algorithms: policy iteration and value iteration. Both of them are special cases of the fixed-point iteration method.

2.1 Policy Iteration

The *policy iteration* algorithm starts by initializing an arbitrary policy π_0. Let Q^{π_k} be the Q function of policy π_k. Then, the following iterative procedure is repeated for $k = 1, 2, \ldots$:

1. *Policy evaluation:* Compute $Q^{\pi_{k-1}}$
2. *Policy improvement:* $\pi_k = \pi_{Q^{\pi_{k-1}}}$

For the policy evaluation step, the value of Q^{π_k} can be easily computed using the Bellman equation which gives us the following relation

$$Q^{\pi_k} = (I - \gamma P^{\pi_k})^{-1} r \tag{1}$$

where I is the identity matrix in appropriate dimensions and P^{π_k} is the transition matrix on state-action pairs induced by the stationary policy π_k.

The policy improvement step gives us the next policy which can be interpreted as the greedy policy that is controlled by the Q function obtained in the latest iteration. In other words, we have

$$\pi_k(s) = \arg\max_{a \in \mathcal{A}} Q^{\pi_{k-1}}(s, a) \tag{2}$$

The following result analyzes the convergence of the policy iteration algorithm.

Theorem 1. *The policy iteration algorithm guarantees the improvement in the policy, i.e.,* $Q^{\pi_k}(s) \geq Q^{\pi_{k-1}}(s)$ $\forall s \in \mathcal{S}, k \geq 1$, *and the improvement is strictly positive for at least one state* $s \in \mathcal{S}$, *i.e.,* $Q^{\pi_k}(s) > Q^{\pi_{k-1}}(s)$ *for at least one* $s \in \mathcal{S}, \forall k \geq 1$, *until the optimal policy* π^* *is found.*

We omit the proof of this theorem and refer the reader to [4]. Now, we analyze the upper bound on the number of iterations required for the policy iteration algorithm to converge. Given S states and A actions, the number of policies is given as A^S. It is known that the number of iterations required for the convergence of the policy iteration algorithm is bounded by the number of policies, A^S, in the worst case. However, in practice, the average performance of the algorithm is observed to be much faster than the worst case. For instance, for a grid world problem, the policy iteration is shown to converge much faster than the value iteration algorithm. Besides, other variants of policy iteration have been proposed in the literature, such as the policy iteration with entropy regularization [5], that are shown to be computationally more tractable than the vanilla algorithm presented here.

Given this iteration complexity of the policy iteration, there is a linear convergence bound on its performance for computing a good policy rather than finding the exact optimal policy. The following theorem formalizes this bound.

Theorem 2. *Let* Q^* *be the (unique) optimal value function for a given MDP specification* $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. *Then,*

$$||Q^* - Q^{\pi_{k+1}}||_\infty \leq \gamma ||Q^* - Q^{\pi_k}||_\infty \tag{3}$$

where the value difference matrix $Q^* - Q^{\pi_{k+1}}$ *is an* $S \times A$ *dimensional matrix and the supremum norm of an* $m \times n$ *matrix is defined as the maximum absolute column sum of the matrix, i.e.,* $||X||_\infty = \max_{1 \leq j \leq n} \sum_{j=1}^{m} |x_{ij}|$.

This result gives a linear convergence bound as $\gamma \in [0, 1)$ and implies that the Q function converges exponentially fast to the optimal Q function value. We also note that the convergence result of policy iteration is expressed in terms of the Q function instead of the policy π. That is because the optimal Q

function value is unique whereas there might be alternative optimal policies corresponding the this unique optimal Q value.

To prove Theorem 2, we need to provide two additional results which will be covered in the next lecture. Before presenting these results, here we first introduce the *Bellman optimality operator*.

Definition 3. *Let f be an $S \times A$ dimensional matrix, then the Bellman operator $\mathcal{T}$ is defined such that*

$$\mathcal{T}f(s,a) = R(s,a) + \gamma \langle P(\cdot|s,a), V_f(s) \rangle \tag{4}$$

where the $V_f(s') = \max_{a \in \mathcal{A}} f(s',a)$.

For $f = Q^*$, then the Bellman operator gives us the sum of two terms: i) instantaneous reward obtained by implementing action a at state s, and ii) expectation of the future values under the policy induced by function f. Next, we define the Bellman operator for an arbitrary policy π, $\mathcal{T}^\pi$.

Definition 4. *Let f be an $S \times A$ dimensional matrix, then the Bellman operator for an arbitrary policy π $\mathcal{T}^\pi$ is defined such that*

$$\mathcal{T}^\pi f(s,a) = R(s,a) + \gamma \langle P(\cdot|s,a), V_f^\pi(s) \rangle \tag{5}$$

where the $V_f^\pi(s') = f(s', \pi(s))$.

Both of these Bellman operators are helpful for explicitly writing down the Bellman equations for policy evaluation and optimization in policy iteration algorithm. Specifically, the Bellman optimality operator $\mathcal{T}$ is used for policy optimization whereas the Bellman operator $\mathcal{T}^\pi$, for an arbitrary policy π, is used for policy evaluation. The respective Bellman equations for policy evaluation and optimization are given as

$$Q^* = \mathcal{T}Q^* \quad \textit{(policy optimization)} \tag{6}$$

$$Q^\pi = \mathcal{T}^\pi Q^\pi \quad \textit{(policy evaluation)} \tag{7}$$

References

[1] T. M. Moerland, J. Broekens, and C. M. Jonker, "A framework for reinforcement learning and planning," *ArXiv*, vol. abs/2006.15009, 2020.

[2] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality.* John Wiley & Sons, 2007, vol. 703.

[3] W. Powell, "What you should know about approximate dynamic programming," *Naval Research Logistics*, vol. 56, pp. 239–249, 2009.

[4] A. Agarwal, N. Jiang, and S. M. Kakade, "Reinforcement learning: Theory and algorithms," 2019.

[5] A. Abdolmaleki, J. T. Springenberg, J. Degrave, S. Bohez, Y. Tassa, D. Belov, N. Heess, and M. Riedmiller, "Relative entropy regularized policy iteration," *arXiv preprint arXiv:1812.02256*, 2018.