IFT 6085 - Lecture 16
Basic results on reinforcement learning

This version of the notes has not yet been thoroughly checked. Please report any bugs to the scribes or instructor.

Scribes
Winter 2019: Eugene Vorontsov & Charles Guille-Escuret

Instructor: Ioannis Mitliagkas

1 Summary

Reinforcement learning is concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. An optimal *policy* defines actions by which an agent achieves its goal of maximizing its *rewards* in this environment. This lecture covers:

- Value iteration (compute states)
- Policy evaluation (expected return)
- Policy optimization (maximize expected return)

2 Basic definitions and assumptions

In this lecture, we consider state transitions and subsequent rewards induced by actions to have a Markov property: given an action in a state, the reward and the new state are independent of other states and actions. We assume the agent can be found in a finite set of states $\mathcal{S}$ with access to a set of actions $\mathcal{A}$. Furthermore, we assume that rewards are bounded: $\exists M \in \mathbb{R}$ such that $\forall s, s' \in \mathcal{S}, \forall a \in \mathcal{A}, \forall r > M, P(r, s'|s, a) = 0$.

Definition 1 (Markov Decision Process)**.** *The probability of transitioning from state $s \in \mathcal{S}$ to the new state $s' \in \mathcal{S}$ with a reward $r \in \mathbb{R}^+$ given an action $a \in \mathcal{A}$ is yielded by the distribution:*

$$P(r, s'|s, a)$$

In case only part of the state is observed, we have a *partially observed MDP* (POMDP).

Definition 2 (Partially Observed Markov Decision Process)**.** *Observe some y from s:*

$$y \sim P(y|s)$$

Observation : To work with an POMDP we can convert it to an MDP by assembling states from cumulative partial observations: $\tilde{s}_0 = \{y_0\}$, $\tilde{s}_1 = \{y_0, y_1\}$, $\tilde{s}_2 = \{y_0, y_1, y_2\}$, etc...

As an agent traces a trajectory of states, its goal is to find a policy for taking actions that maximize the reward along this trajectory.

Definition 3 (Policy)**.** *Actions $a \in \mathcal{A}$ are defined for a state $s \in \mathcal{S}$ with a policy:*

$$\pi(a|s)$$

States a are sampled according to the *probabilistic* policy $\pi(a|s)$. A policy is *deterministic* if $a = \pi(s)$.

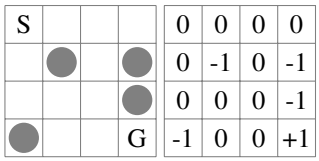

Figure 1: **Example** where an agent traverses a frozen lake from a starting point (S) to a goal (G). Moving along the grid has a 50% chance of moving in a move in a random direction. The lake grid is shown on the left and grid rewards are on the right. The agent is rewarded 1 for reaching the goal, -1 for falling in a hole (gray circles, left), and 0 elsewhere. Since an agent dies in a hole, this state is a *sink state* in that it no longer changes. One *episode* traces a trajectory from the start to the goal or a sink state.

3 Value Iteration

A dynamic programming approach called value iteration computes an optimal MDP policy and its value. This requires the computation of an expected return. Depending on the task, the expectation can be computed by dynamic programming over a finite horizon (task ends) or an infinite horizon (task continues).

3.1 Finite Horizon

In the case of a finite horizon, the trajectory reaches the goal or sink state in a finite number of steps.

Definition 4 (State Value Function)**.**

$$V_t^\pi(s) = \mathbb{E}\left[\sum_{\tau=0}^{\tau} r_\tau | s_0 = s\right]$$

Definition 5 (State-Action Value Function)**.**

$$V_t^\pi(s) = \mathbb{E}\left[\sum_{\tau=0}^{\tau} r_\tau | s_0 = s, a_0 = a\right]$$

Definition 6 (Expected Return)**.** *For a trajectory with T steps and reward r_i at each step $i \in [1, ..., T]$:*

$$\mathbb{E}\left[\sum_{i=0}^{T-1} r_i + V_T(s_T)\right],$$

To solve for the optimal policy and its value, we consider the state value function with the finite horizon expected return:

$$V(s_0) = \max_{\pi_0} \max_{\pi_1} ... \max_{\pi_{T-1}} \mathbb{E}[r_0 + r_1 + ... + r_{T-1} + V_T(s_T)]$$

Since for any i, r_i is independent of $\pi_{i+1}, \pi_{i+2}, ..., \pi_{T-1}$ (the reward does not depend of the decision we take in the future), we can successively take the max out of the expectation and obtain:

$$V(s_0) = \max_{\pi_0} \mathbb{E}[r_0 + \max_{\pi_1} \mathbb{E}[r_1 + ... + \max_{\pi_{T-1}} \mathbb{E}[r_{T-1} + V_T(s_T)]]]$$

We can then solve by applying for i following $\{T, ..., 1\}$:

$$\begin{aligned}
&\forall s \in \mathcal{S} \\
&\pi_{i-1}(s) = \operatorname*{argmax}_a(\mathbb{E}_{s_i}[r_{i-1} + V_i(s_i)] \\
&V_{i-1}(s) = \max_a(\mathbb{E}_{s_i}[r_{i-1} + V_i(s_i)]
\end{aligned}$$

Algorithm 1 Finite Horizon Value Iteration

for $t = T-1, T-2, .., 0$ **do**
  for $s \in \mathcal{S}$ **do**
    $\pi_t(s), V_t(s) = \underset{a}{\text{maximize}}(\mathbb{E}[r_t + V_{t+1}(s_{t+1})])$
  end
end

3.2 Infinite Horizon

In the case of an infinite horizon, future rewards are increasingly discounted (reward is bounded).

Definition 7 (State Value Function)**.**

$$V_t^{\pi}(s) = \mathbb{E}\left[\sum_{\tau=0}^{\tau} \gamma^{\tau} r_{\tau} | s_0 = s\right]$$

Definition 8 (State-Action Value Function)**.**

$$V_t^{\pi}(s) = \mathbb{E}\left[\sum_{\tau=0}^{\tau} \gamma^{\tau} r_{\tau} | s_0 = s, a_0 = a\right]$$

Definition 9 (Expected Return)**.** *For a trajectory with T steps and reward r_i at each step $i \in [1, ..., T]$:*

$$\mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_i\right],$$

Observation : We can interpret the rewards in a discounted setting as the non-discounted rewards from an MDP, by adding a sink state $\tilde{s}$ that traps the agent indefinitely with reward 0, and using the transition:

$$\tilde{P}(s'|s,a) = \begin{cases} P(s'|s,a) & \text{with probability } \gamma \\ \tilde{s} & \text{with probability } 1-\gamma \end{cases}$$

Algorithm 2 Infinite Horizon Value Iteration

Initialize $V^{(0)}$ arbitrarily
for $n = 0, 1, 2, ...$ *until termination condition* **do**
  for $s \in \mathcal{S}$ **do**
    $\pi^{(n+1)}(s), V^{(n+1)}(s) = \underset{a}{\text{maximize}}(\mathbb{E}_{s_T}[r_{T-1} + \gamma V^n(s_T)])$
  end
end

Theorem 10. *By assuming a horizon $T \in \mathbb{N}$, an infinite horizon value iteration problem can be converted into a finite horizon one with error bounded by:*

$$\epsilon \leq r_{max} \frac{\gamma^T}{(1-\gamma)}$$

Proof.

$$\begin{aligned}
\sum_{t=T}^{\inf} \gamma r_t &\leq \sum_{t=T}^{\inf} \gamma r_{max} \\
&= \sum_{t=0}^{\inf} \gamma r_{max} - \sum_{t=0}^{T} \gamma r_{max} \\
&= r_{max} \left(\frac{1}{1-\gamma} - \frac{1-\gamma^T}{1-\gamma} \right) && \text{Geometric series.} \\
&= r_{max} \frac{\gamma^T}{1-\gamma}
\end{aligned}$$

□

Since we are dealing with discounted rewards where $\gamma < 1$, we can interpret a value iteration update as application of an operator that has a fixed point to which iteration converges at the limit.

Definition 11 (Backup Operator)**.**

$$\mathcal{T} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$$

$$[\mathcal{T}V](s) = \max_a \mathbb{E}_{s'|s,a}[r + \gamma V(s')]$$

Theorem 12. *Backup operator $\mathcal{T}$ is a contraction with modulus γ under ∞-norm*

$$||\mathcal{T}V - \mathcal{T}W||_\infty \leq \gamma ||V - W||_\infty$$

Theorem 13. *The backup operator $\mathcal{T}$ has a fixed point V^* and :*

$$(\mathcal{T}^i V) \underset{i \to \infty}{\to} V^*$$

Proof : This is a direct consequence of Theorem 12 and Banach's Fixed Point theorem.

We can then rewrite Algorithm 2 in a cleaner manner:

Algorithm 3 Infinite Horizon Value Iteration with operator

Initialize $V^{(0)}$ arbitrarily
for $n = 0, 1, 2, ...$ *until termination condition* **do**
| $V^{(n+1)} = \mathcal{T}V^{(n)}$
end

4 Policy Evaluation and Iteration

While value iteration allows us to evaluate the return from states according to the optimal policy, policy iteration seeks to update each policy to increase the expected return. We again use a backup operator but this time without maximizing over actions.

Definition 14 (Backup Policy Operator)**.**

$$\mathcal{T}^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$$

$$[\mathcal{T}^\pi V](s) = \mathbb{E}_{s'|s,a)\pi(s)}[r + \gamma V(s')]$$

Instead, we evaluate the expected return for every state, induced by each policy, and then select the actions that maximize the expected return for that policy. We obtain the state valuation induced by π by using $V^* = \mathcal{T}^\pi V^*$. This induces a linear equation that can be solved exactly for *policy evaluation*:

$$V(s) = \sum_{s'} P(s'|s, a = \pi(s))[r(s, a, s') + \gamma V(s')]$$

Policy iteration is then performed by alternating between policy evaluation for each policy π and a greedy update of the policies actions:

Algorithm 4 Policy Iteration

Initialize $\pi^{(0)}$ arbitrarily
for $n = 1, 2, ...$ **do**
  $V^{\pi^{(n-1)}} = Solve[V = \mathcal{T}^{\pi^{(n-1)}} V]$ $\pi^{(n)} = \underset{a}{\text{argmax}}(\mathbb{E}_{s'|s,a}[r + V^{\pi^{(n-1)}}(s')])$
end

Note that for a finite MDP, Algorithm 4 should converge in a finite number of iterations since the number of policies is finite [1].

References

[1] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. 2011.