IIIS 2018 Spring: ATCS - Selected Topics in Optimization
Lecture date: June 4th, 2018
Instructor: Jian Li **Scribe: Zihan Zhang**

1 Definition of Reinforcement Learning

Generally, we formulate reinforcement learning problem by Markov decision process(MDP).

Definition1 (MDP) A MDP is a 5-tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where

- $\mathcal{S}$ is a set of states which we assume finite here;
- $\mathcal{A}_s$ is a set of actions available from state s and when it's clear we often omit the subscript;
- $P(s'|s,a)$ is the probability that action a in state s will lead to state s' in the next timestep;
- $r(s', s, a)$ is the reward the one would get after executing action a in state s and arriving s';
- $\gamma \in [0, 1]$ is the discount factor.

The agent is asked to find a series actions $\{a_t\}_{t=0}^{T}$ such that the long term reward is maximized, i.e. we aim to maximize

$$E_{a_{0:\infty}}[\sum_{t=0}^{T} \gamma^t r_t]$$

where $T \in N \cup \{\infty\}$ is the horizon.

A policy π is a mapping from $\mathcal{S} \times [T]$ to $\mathcal{A}$. We say the agent follows π when the agent chooses actions according to π.

Definition2 (Value function and Q-function). Given a policy π, then value function and Q-function are defined as:

$$V_\pi(s) = E_\pi[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s]$$

$$Q_\pi(s,a) = E_\pi[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a]$$

and the optimal value function and Q-function are defined as:

$$V^*(s) = \sup_\pi V_\pi(s)$$

$$Q^*(s) = \sup_\pi Q_\pi(s,a)$$

It's trivial to find the iteration of V and Q:

$$V_\pi(s) = E_\pi[r_0 + \gamma V(s')]$$

$$Q_\pi(s,a) = E[r_0] + \gamma \sum_{s'} P(s'|s,a) V_\pi(s')$$

$$V^*(s) = max_a Q^*(s,a) = max_{a \in \mathcal{A}_S} E_a[r_0] + \gamma \sum_{s'} P(s'|s,a) V^*(s')$$

$$Q^*(s,a) = E[r_0] + \sum_{s'} P(s'|s,a) V^*(s)$$

Remark1. Here we consider the discountted case, i.e. $\gamma < 1$. Actually when the horizon T is finite, we could do nothing more than dynamic programming. In the case $T = \infty$, to keep the long-term reward converge, we have to select $\gamma < 1$. Alternatively, we could consider the average reward per step. For more details, please refer to [1]

2 Basic Algorithms

Definition2 (Bellman Operator) Suppose $\mathcal{S} = [n]$, then $\mathcal{T} : R^n \to R^n$ is defined as:

$$\mathcal{T} v_i = max_{a \in A_i} \sum_{j=1}^{n} P(j|i,a)(r(j,i,a) + \gamma v_j)$$

when π is a policy independent of t, then $\mathcal{T}_\pi$ is defined as:

$$\mathcal{T}_\pi v_i = \sum_{j=1}^{n} P(j|i,\pi)(r(j,i,\pi) + \gamma v_j)$$

$$\mathcal{T}_\pi v = r_\pi + \gamma P_\pi v$$

where $P(j|i,\pi) = \sum_{a \in \mathcal{A}_i} Pr(a|\pi) P(j|i,a)$ and $r(j,i,\pi) = \sum_{a \in \mathcal{A}_i} Pr(a|\pi) r(j,i,a)$.

It's easy to see that $\mathcal{T}$ is a contraction with factor $\gamma < 1$ in L^∞ space. So we have following claims:

- $v = \mathcal{T} v \iff v$ is the optimal value function and we denote it as v^*;
- $\lim_{k \to \infty} \mathcal{T}^k v = v^*$ for any $v \in R^n$;
- π is optimal iff $\mathcal{T} v = \mathcal{T}_\pi v = v^*$

To solve v^*, we list 3 algorithms.

Algorithm 1: Value Iteration (VI)

```
v_0 ← 0
for t = 0, 1, 2, ..., T do
    v_{t+1} ← T v_t
return v_T
```

Note that if we select $m_t = 1$ for all t, then MPI turns to be VI, and if $m_t = \infty$, MPI becomes PI. So we only need to show convergence for MPI.

Algorithm 2: Policy Iteration (PI)

$v_0 \leftarrow 0$
for $t = 0, 1, 2, \ldots, T$ **do**
  find π_t such that $\mathcal{T}_{\pi_t} v_t = \mathcal{T} v_t$
  $v_{t+1} \leftarrow lim_{k\to\infty} \mathcal{T}_{\pi_t}^k v_t = v_{\pi_t}$
return v_T

Algorithm 3: Modified Policy Iteration (MPI)

$v_0 \leftarrow 0$
for $t = 0, 1, 2, \ldots, T$ **do**
  find π_t such that $\mathcal{T}_{\pi_t} v_t = \mathcal{T} v_t$
  $v_{t+1} \leftarrow \mathcal{T}_{\pi_t}^{m_t} v_t$
return v_T

Theorem1. MPI algorithm converges when $T \to \infty$.
Proof. Define $D = \{v \in R^n, \mathcal{T} v \geq v\}$ where $x \geq y$ means $x_i \geq y_i$ for all $i \in [n]$. Denote $\mathcal{T}^k v_0 = u_k$, we'll prove that $v_k \geq u_k$ and $v_k \in D$ by induction. For $k = 0$, the induction hypothesis is satisfied. Assume now it holds for $k = 0, 1, 2, .., n$, we have $v_{n+1} = \mathcal{T}_{\pi_n}^{m_n} v_n = v_n + \sum_{i=0}^{m_n - 1} \mathcal{T}_{\pi_n}^i (\mathcal{T}_{\pi_n} v_n - v_n) \geq \mathcal{T} v_n \geq \mathcal{T} u_n = u_{n+1}$ and $\mathcal{T} v_{n+1} - v_{n+1} \geq \mathcal{T}_{\pi_n}^{m_n} (\mathcal{T} v_n - v_n) \geq 0$. Thus the hypothesis is satisfied for $k = n + 1$. At last, because $v^* \geq lim_{t\to\infty} v_t \geq lim_{t\to\infty} u_t = v^*$, we have $\lim_{t\to\infty} v_t = v^*$.

3 LP-solution

Consider following LP problem:

$$min \sum_s u_0(s) v(s)$$

s.t.

$$v(s) \geq \sum_{s'} P(s'|s, a)(r(s', s, a) + \gamma v(s'))$$

for any $s \in \mathcal{S}$ and $a \in \mathcal{A}_s$.
Claim. If $u_0(s) > 0$ for any s, then the solution of this LP problem is the optimal value function.
Proof. We only need to show $\mathcal{T} v = v$. Suppose $\mathcal{T} v(s) > v(s)$ for some $s = s_0$, then we could decrease $v(s_0)$ with a proper positive stepsize keeping all the constraints still satisfied. On the other side, the target $\sum_s u_0(s) v(s)$ is strictly decreased because $u_0(s_0) > 0$, which is contrary to the optimality of v.

4 Techniques to Evaluate Value Function

In previous discussion, we assume that the transition probability P and the reward function r is known, but in practice we have to estimate this random variables by sampling. Here we're going

to talk about how to evaluate value function given a fixed policy π (which we often omit when it's clear). We have following algorithms:

Algorithm 4: α-MC

$v(s) \leftarrow 0$ for all s
for $t = 0, 1, 2, \ldots, T$ **do**
    Play a sequence starting from s_t until terminal to get $\{r_j\}_{j=0}^{\infty}$;
    $v(s_t) \leftarrow (1-\alpha)v(s_t) + \alpha \sum_{j=0}^{\infty} \gamma^j r_j$;
return v_T

Algorithm 5: α-TD

$v(s) \leftarrow 0$ for all s
for $t = 0, 1, 2, \ldots, T$ **do**
    Play one step starting from s_t to get $\{s_{t+1}, r_t\}$;
    $v(s_t) \leftarrow (1-\alpha)v(s_t) + \alpha(r_t + \gamma v(s_{t+1}))$;
return v_T

5 Learn Q-function

Instead of solve value function, we can try to optimize for Q-function. We lists two algorithms here: Q-learning (off-policy TD control) and Sarsa (on-policy TD control)

Algorithm 6: Q-learning

$Q(s, a) \leftarrow 0$ for all s and a
for $t = 0, 1, 2, \ldots, T$ **do**
    option1: **for** $a \in \mathcal{A}_{s_t}$ **do**
        $Q(s_t, a) \leftarrow \sum_{s'} P(s'|s_t, a)(r(s', s, a) + \gamma \max_{a'} Q(s', a'))$
    option2: **for** $a \in \mathcal{A}_{s_t}$ **do**
        $Q(s_t, a) \leftarrow (1-\alpha)Q(s_t, a) + (1-\alpha) \sum_{s'} P(s'|s_t, a)(r(s', s, a) + \gamma \max_{a'} Q(s', a'))$
    option3: play one step starting from s_t with action $a_t = a$ and get $\{s_{t+1}, r_t\}$
        $Q(s_t, a) \leftarrow (1-\alpha)Q(s_t, a) + \alpha(r_t + \max_{a'} Q(s_{t+1}, a'))$
return Q_T

One could replace the 6-th line of algorithm7 with

$$Q(s, a) \leftarrow (1-\alpha)Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a'))$$

in the off-policy case.

Algorithm 7: Sarsa

Input: H: number of episodes
for $i = 0, 1, 2, \ldots, H$ **do**
    Initialize the state to be s_0
    while *s is not the terminal state* **do**
        Take action a , observe r and s', and derive a' according to $Q(s')$ (here we do not specify the method to derive a', and a possible choice is ϵ-greedy method);
        $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(r + \gamma Q(s', a'))$;
        $s \leftarrow s'$, $a \leftarrow a'$
return Q

References

[1] Puterman M L. Markov decision processes: discrete stochastic dynamic programming[M]. John Wiley, Sons, 2014.