Reinforcement Learning

Daniel Khashabi

Fall 2016
Last Update: November 27, 2016

1 Introduction

Reinforcement Learning (RL) is one of the sub-areas in machine learning which connects it to other fields like Control Theory, or Cognitive Science. The term in fact comes from study of animals, humans, society and how one can teach them certain behaviors via proper feedback/reward mechanisms [citation needed].

The general scenario is shown in Figure 1. Unlike the supervised learning in which the agent gets a *fixed* set of learning instances, in RL the learning agent receive *feedbacks/rewards* as a result of interacton with an *environment*. Given the feedbacks, the learning agent updates it's brain/strategy and *acts* in environment. Often times the strategy of the agent is dependent on the *state* of the environment (like whether agent has reached to its destination or not). Hence modelling states of the models and how transition between the states is done is a key issue.

Another key issue here is the way exploration of the states. Often times in real problems the size of the state-space is exponentially big (e.g. number of the ways one can reach to a corner of a room, starting from another corner). In such scenarios, a proper algorithms probably doesn't need (and shouldn't) explore all the possibel states, and instead only visit the representatives. Creating such a balance is called *exploration-exploitation* tradeoff.

2 Preliminaries

Here we formally define the basic notation and definitions:

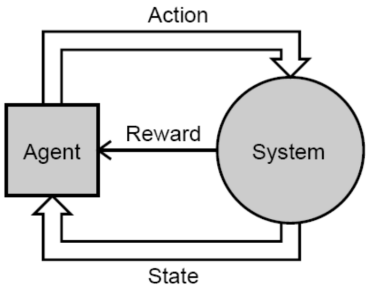

Figure 1: The general scenario in RL: the figure shows the interaction between the main components in an RL game

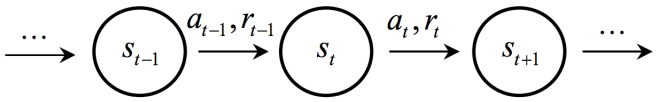

Figure 2: The transition between states, as a result of actions.

Definition 2.1 (Markov Definition Process (MDP))**.** *Here are the set of elements in an MDP:*

- *Possibly infinite set of labels S*
- *Initial state s_0*
- *Possibly infinite set of actions A*
- *Transition probably to state s', upon doing action a and starting from state s: $P(s'|s,a)$.*
- *Rewrds distribution according to the current action a and state s: $P(r|s,a)$.*

Definition 2.2 (Policy)**.** *A policy π is a function from states to actions:*

$$\pi : S \to A$$

A policy can be a function of time, i.e. change each at each iteration based on the reward recieved from environment (also called non-stationary policy). One other notion of policy is the *stationary policy* which is the policy in long run (after many iterations).

Definition 2.3 (Cumulative reward)**.** *The cumulative reward is the overall reward in an RL scenario. Often this is written as the sum of the rewards in each iterations:*

- *Finite time ($T < \infty$): $R_T = \sum_{t=0}^{T} r(s_t, \pi(s_t))$.*
- *Infinite time ($T = \infty$): $R_T = \sum_{t=0}^{T} \gamma^r r(s_t, \pi(s_t))$, for some $\gamma \in [0, 1)$.*

Definition 2.4 (Policy Value function)**.** *The function value function $V_\pi(s)$ is a function which contains the expected cumulative reward of a state s, for a given policy π. Here are the definitons, depending on the definition of the cumulative reward:*

- *Finite time:*

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{t'} r(s_t, \pi(s_t))|s_{t'} = s\right]$$

- *Infinite time:*

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{t'} \gamma^t r(s_t, \pi(s_t))|s_{t'} = s\right]$$

Intuitively speaking, one can think about finding the optimal policy, as the strategy that maximizes the policy value at all the states. In other words, the optimal policy π^* is defined as

$$V_{\pi^*}(s) = \max_\pi V_\pi(s)$$

For the rest of this write up, we will use $V_{\pi^*}(s)$ and $V^*(s)$ interchangeably.

Remark 2.1 (Bellman Equations). An intuitive observation about the policy value function is that, it is dependent on the policy values of the neighboring states. This is described by Bellman equation:

$$V_\pi(s) = \mathbb{E}\left[r(s, \pi(s))\right] + \gamma \sum_{s'} \mathrm{P}\left(s'|s, \pi(s)\right) V_\pi(s')$$

More compactly this can be written as

$$\mathbf{V} = \mathbf{R} + \gamma \mathbf{P}\mathbf{V} \tag{1}$$

which has a clear solution of $\mathbf{V}^* = (\mathbf{I} - \gamma\mathbf{P})^{-1}\mathbf{R}$. One can easly show that $(\mathbf{I} - \gamma\mathbf{P})$ is definitely invertible, by showing that $\|\gamma\mathbf{P}\| = \gamma < 1$ which imlpies that all the eigenvalues of $\gamma\mathbf{P}$ are less than one and $(\mathbf{I} - \gamma\mathbf{P})$ is definitely invertible

Proof. Proving this is streightforward:

$$\begin{aligned} V_\pi(s) &= \sum_{t=1}^{t'} \gamma^t \mathbb{E}\left[r(s_t, \pi(s_t))|s_{t'} = s\right] \\ &= r(s_0, \pi(s_0)) + \gamma \sum_{t=0}^{t'-1} \gamma^t \mathbb{E}\left[r(s_{t+1}, \pi(s_{t+1}))|s_{t'-1} = s\right] \\ &= \mathbb{E}\left[r(s, \pi(s))\right] + \gamma \sum_{s'} \mathrm{P}\left(s'|s, \pi(s)\right) V_\pi(s') \end{aligned}$$

■

Definition 2.5. *State-action function The state-action function $Q(s, a)$ contains the expected value of the cumulative rewards per-action per-state. Here are the definitons, depending on the definition of the cumulative reward:*

- *Finite time:*

$$Q_\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{t'} r(s_t, \pi(s_t))|s_{t'} = s, \pi(s_{t'}) = a\right]$$

- *Infinite time:*

$$Q_\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{t'} \gamma^t r(s_t, \pi(s_t))|s_{t'} = s, \pi(s_{t'}) = a\right]$$

Similar to Bellman equation for value function, one can write something similar for the optimal state-action function:

$$Q^*(s, a) = \mathbb{E}\left[r(s, a)\right] + \gamma \sum_{s'} \mathrm{P}\left(s'|s, a\right) V_\pi^*(s')$$

And it should intuitively make sense that the optimal state-action function and the optimal policy value functions are related via a simple connection:

$$V^*(s) = \max_{a \in A} Q^(s, a)$$

And indeed this is achieved by the optimal policy:

$$\pi^*(s) = \arg\max_{a \in A} Q^(s, a)$$

3 RL algorithms

GIven the basic defintions of the previous section, the goal is to find the policy for an RL agent, in a given environment. Each of the following algorithms try to calculate a certain function: some work based on policy value function, some work based on the state-action function, etc, which has resulted in different algorithms. But in essense, since all these functions are related to each other, they all result in an estimated policy.

3.1 Linear Programming

Suppose we have non-infintely action and non-infinite states. We can use the Belllman equation for pocliy value and model the problem as a Linear Program:

$$\begin{cases} \min_{\mathbf{V}} \sum_{s \in S} \alpha(s) V(s) \\ \text{Subject to: } V(s) \geq \mathbb{E}\left[r(s,a)\right] + \gamma \sum_{s'} \mathrm{P}\left(s'|s,a\right) V(s'), \forall s \in S, a \in A \end{cases}$$

where $\boldsymbol{\alpha}$ is a vector of constants.

The dual of this program is:

$$\begin{cases} \max_{\mathbf{x}} \sum_{s \in S, a \in A} \mathbb{E}\left[r(s,a)\right] x(s,a) \\ \text{Subject to: } \sum_{a \in A} x(s,a) = \alpha(s) + \gamma \sum_{s' \in S, a \in A} \mathrm{P}\left(s'|s,a\right) x(s',a), \forall s \in S \\ \forall s \in S, a \in A, x(s,a) \geq 0 \end{cases}$$

Here $x(s,a)$ can be interpreted as the probabilty of being in a state s and taking action a.

Note that LP is just a way of modelling the problem, and by itself does not refer to any specific algorithms. Now on the other hand, there is a wide range of algorithm for solving/approximating large LPs, for example Simple algorithm, interior point methods, and variety of special-purpose algorithms.

3.2 Value Iteration

Value iteration is a way to approximate the value function based on the Bellman equation. Define the function $\boldsymbol{\Phi}(\mathbf{V}) : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$:

$$\boldsymbol{\Phi}(\mathbf{V}) \triangleq \max_{\pi} \left\{ \mathbf{R}_{\pi} + \gamma \mathbf{P}_{\pi} \mathbf{V} \right\}$$

which is a short form for:

$$\left[\boldsymbol{\Phi}(\mathbf{V})\right](s) = \max_{a \in A} \left\{ \mathbb{E}\left[r(s,a)\right] + \gamma \sum_{s'} \mathrm{P}\left(s'|s,a\right) V(s') \right\}, \forall s \in S$$

Algorithm 1: Value iteration

$\mathbf{V} \leftarrow \mathbf{V}_0$;
while $\|\mathbf{V} - \boldsymbol{\Phi}(\mathbf{V})\| \geq \frac{1-\gamma}{\gamma}\epsilon$ **do**
  $\mathbf{V} \leftarrow \boldsymbol{\Phi}(\mathbf{V})$;
return $\boldsymbol{\Phi}(\mathbf{V})$;

Theorem 3.1. *For any initial value* $\mathbf{V}_0$ *the sequence definef by* $V_{n+1} = \mathbf{\Phi}(\mathbf{V})(V_n)$ *converges to* V^*.

Proof. Here is the proof idea: first show that the $\mathbf{\Phi}$ function is γ-Lipchitz, using its definition: $\mathbf{\Phi}(\mathbf{V})(s) - \mathbf{\Phi}(\mathbf{U})(s) \leq \gamma \left\|\mathbf{V} - \mathbf{U}\right\|_\infty, \forall s$ or $\left\|\mathbf{\Phi}(\mathbf{V}) - \mathbf{\Phi}(\mathbf{U})\right\|_\infty \leq \gamma \left\|\mathbf{V} - \mathbf{U}\right\|_\infty$, which implies that

$$\left\|\mathbf{\Phi}(\mathbf{V})^* - \mathbf{\Phi}(\mathbf{V})_{n+1}\right\|_\infty \leq \gamma^{n+1} \left\|\mathbf{V}^* - \mathbf{V}_0\right\|_\infty$$

■

Also one can show that the convergence of this algorithm is $O\left(\log \frac{1}{\epsilon}\right)$, where ϵ is the distance to the (unknown) optimal policy function:

$$\left\|\mathbf{V}^* - \mathbf{V}_{n+1}\right\|_\infty \leq \epsilon$$

Also note that since $\left\|\mathbf{V}^* - \mathbf{V}_{n+1}\right\|_\infty \leq \gamma \left\|\mathbf{V}^* - \mathbf{V}_{n+1}\right\|_\infty + \gamma \left\|\mathbf{V}_{n+1} - \mathbf{V}_n\right\|_\infty$, hence:

$$\left\|\mathbf{V}^* - \mathbf{V}_{n+1}\right\|_\infty \leq \frac{\gamma}{1-\gamma} \left\|\mathbf{V}_{n+1} - \mathbf{V}_n\right\|_\infty$$

We can't bound $\left\|\mathbf{V}^* - \mathbf{V}_{n+1}\right\|_\infty$, but we can bound $\left\|\mathbf{V}_{n+1} - \mathbf{V}_n\right\|_\infty$ instead. Hence we try to find the number of the iterations needed to get $\left\|\mathbf{V}_{n+1} - \mathbf{V}_n\right\|_\infty \leq \frac{1-\gamma}{\gamma}\epsilon$. From earlier analyze of γ-Lipchitz-ness we had $\left\|\mathbf{\Phi}(\mathbf{V})_{n+1} - \mathbf{\Phi}(\mathbf{V})_n\right\|_\infty \leq \gamma^{n+1} \left\|\mathbf{\Phi}(\mathbf{V}_0) - \mathbf{V}_0\right\|_\infty$. Hence $n \in O\left(\log \frac{1}{\epsilon}\right)$.

3.3 Policy Iteration

A slightly different algorithm is Policy Iteration, which works based on iterating over the policy function π. At each iteration it greedily chooses the best policy according to the Bellman function.

Algorithm 2: Policy iteration

$\pi \leftarrow \pi_0$;
$\pi' \leftarrow NIL$;
while $\pi \neq \pi'$ **do**
  $\mathbf{V} \leftarrow \mathbf{V}_\pi$ // evaluete the policy, e.g. by solving $(\mathbf{I} - \gamma \mathbf{P}_\pi)\mathbf{V} = \mathbf{R}_\pi$;
  $\pi' \leftarrow \pi$;
  $\pi \leftarrow \arg\max_\pi \{\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V}\}$
return π;

One can show that the convergence of policy iteration is as bad as value iteration in the worst case.

Theorem 3.2. *Suppose* $\mathbf{U}_n$ *is the sequence of values generated by value iteration, and* $\mathbf{U}_n$ *is the sequence of values generated by value policy iteration. If* $\mathbf{V}_0 = \mathbf{U}_0$ *then,*

$$\mathbf{U}_n \leq \mathbf{V}_n \leq \mathbf{V}^*, \forall n$$

Proof is done with induction on n.

4 Beyond basic algorithms

Up to now we assumed that we know everying about the environmen, namely we know the transition probabilities between the states as well as the reward funcon.

4.1 Sochastic Approximation

4.2 Q-learning

[6]

4.3 Temporal Difference Learning

[5]

4.4 SARSA algorithm

5 Bibliographical notes

The literature for RL is intertwined with the existenting literature for other fields like Stochastic Control, Optimal Control, etc, and hence it's hard to pinpoint the real person behind some of the ideas in the field. The first proof of convergence for Q-learning was given by Watkins&Dayan [6]. The interested reader could refer to wellknown surveys and textsbooks in the area [3, 4, 2, 1].

References

[1] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Found. Trends Mach. Learn.*, 8(5-6):359–483, November 2015.

[2] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[3] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[4] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2010.

[5] Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[6] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.