A Lazy Introduction to Reinforcement Learning: A Supervised Learning Perspective

Thanh T. Nguyen

May 22, 2018

Abstract

[And16] does an excellent job in explaining Reinforcement Learning (RL) from a practical perspective. Inspired by that work, in this draft, we present an introduction to RL from a perspective of supervised learning (SL) in which the task is to optimize some distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ from i.i.d. sample $(\boldsymbol{x}_i)_i^n$.

1 Introduction

It is commonly believed that reinforcement learning (RL) is one of the main three paradigms that make up machine learning [Dav15]. The other two include supervised learning (SL) and unsupervised learning (UL). The three learning paradigms differ from one another in fundamental aspects: SL requires labeled samples to learn a good prediction, UL learns to discover structure of the unlabeled data, while RL learns through interaction with surrounding environments. Recently, [And16] excellently explains RL through the example of playing ping pong game from scratch and pixels, and points out a more subtle difference between RL and SL: SL uses a true label from the data while RL samples a "fake" label from the policy model. Inspired by that work, in this draft, we introduce the RL problem from the view of SL where we sample i.i.d. data and update a model using gradients computed over that sample.

For other reading on RL, please consider [SB14, SML+15, MKS+13, MKS+15].

2 RL Problem Formulation

We introduce here the major elements in RL and formulate RL problem.

- Observation space, $o \in \mathcal{O}$
- Action space, $a \in \mathcal{A}$
- An environment $\mathcal{E}$ that (possibly stochastically) produces an observation $o \in \mathcal{O}$ and a scalar reward $r \in \mathcal{R}$ based its current unobserved state $\psi \in \Psi$:

$$\mathcal{E} : \Psi \to (\mathcal{O}, \mathcal{R})$$

- An agent interacts with the environment $\mathcal{E}$, possibly alters the internal state of the environment via some unknown function λ, and then executes an action $a \in \mathcal{A}$ based on some policy $\pi \in \Pi$ that (possibly stochastically) maps the observation space onto the action space:

$$\pi : \mathcal{O} \to \mathcal{A}$$
$$\lambda : \mathcal{A} \to \Psi.$$

The goal of the agent is to choose an optimal policy such that the agent receives the maximum accumulated reward on average. At this point, there are at least two important concerns: (i) What do we exactly mean by "on average"? In other words, "on average" over what? (ii) How does the agent learn to get better only through interactions with the environment? These two concerns become clearer in the next section.

3 SL perspective

In practice, we usually parameterize the policy space by a family of distributions $p_\theta(a|o)$. To explain RL from SL perspective, we introduce here an important concept, namely a **game trajectory**, or a *game sample*. That said, a game trajectory is a sequence of consecutive triples of an observation, a reward, and an action from the initial state until the terminal state:

$$\boldsymbol{s} = (o_i, r_i, a_i)_{i=1}^T$$

where T is some finite integer indicating the time step at the terminal state. In other words, a game trajectory completely describes the temporal representation of the RL system when the terminal state is reached (e.g., in the Ping pong game, we have a game sample after a player wins or loses the game).

For each game trajectory $\boldsymbol{s}$, we define the *future total discounted reward* for each step $1 \leq t \leq T$ as:

$$R(t|\boldsymbol{s}) := \sum_{\tau=t}^{T} \gamma^{\tau-t} r(t)$$

Intuitively, this quantity represents how much the current action taken at time step t contributes to the ultimate result of the game at the final step T. The discounted sum loosely accounts for the fact that a more distant action has a lesser effect on the ultimate result at final step T.

As in SL, we need to evaluate whether the produced sample should be encouraged or discouraged via some score function that maps the sample into same scalar which is high if the agent has high total future rewards in this trajectory. One way that nicely implements such intuition is the *weighted*[1] future total discounted reward:

$$f(\boldsymbol{s}) := \sum_{t=1}^{T} p_{\boldsymbol{\theta}}(a_t|o_t) R(t|\boldsymbol{s})$$

Thus, the goal of RL now becomes the maximization of the expected score function:

$$E_{p(\boldsymbol{s})}[f(\boldsymbol{s})].$$

In that form, the RL problem now becomes more standard to be optimized by stochastic gradient descent (SGD). That is, we sample a game trajectory (or even of a batch of game trajectories), compute the gradients, and update the parameters. This is known as **policy gradients** as we compute the gradients of the score function to update the policy. In what follows, we present a detail derivation of policy gradients, and briefly explain why they even make sense in the RL context:

$$\frac{\partial E_{p(\boldsymbol{s})}[f(\boldsymbol{s})]}{\partial \boldsymbol{\theta}} = E_{p(\boldsymbol{s})}\left[\sum_{t=1}^{T} R(t|\boldsymbol{s}) \frac{\partial p_{\boldsymbol{\theta}}(a_t|o_t)}{\partial \boldsymbol{\theta}}\right] \tag{1}$$

$$= E_{p(\boldsymbol{s})}\left[\sum_{t=1}^{T} p_{\boldsymbol{\theta}}(a_t|o_t) R(t|\boldsymbol{s}) \frac{\partial \log p_{\boldsymbol{\theta}}(a_t|o_t)}{\partial \boldsymbol{\theta}}\right]. \tag{2}$$

Without the scaling factor $p_{\boldsymbol{\theta}}(a_t|o_t)R(t|\boldsymbol{s})$ in Equation (2), SGD would simply modify the weights $\boldsymbol{\theta}$ in such a way that improves the probability of producing the action trajectory $\boldsymbol{o} = (o_1, ..., o_T)$. But what if the action trajectory leads to a game loss in the end? That is where $p_{\boldsymbol{\theta}}(a_t|o_t)R(t|\boldsymbol{s})$ comes into a play. The scaling factor discourages the reoccurring of the action trajectory by discouraging each individual action of the trajectory. The degree of discouragement of each individual action is proportional to the contribution of the action to the ultimate loss and the frequency the action is produced at the current time step. If the action trajectory leads to a game win, SGD encourages the action trajectory the same way it discourages it in case of game loss. By sampling $\boldsymbol{s} \sim p(\boldsymbol{s})$, or equivalently playing many games, the agent exposes to many trajectories and is more likely to pick some good trajectories. Therefore, by playing more games, the agent in general becomes smarter.

[1] Some literature names this *expected* future total discounted reward, however $p_\theta(a_t|o_t)$ does not sum up to 1 as $1 \leq t \leq T$. Thus, it would be safer to name this *weighted* instead of *expected*.

References

[And16] Andrej Karpathy. Deep reinforcement learning: Pong from pixels. http://karpathy.github.io/2016/05/31/rl/, 2016. [Online; accessed 22-May-2018].

[Dav15] David Silver. Advanced topics 2015 (compm050/compgi13) reinforcement learning. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html, 2015. [Online; accessed 22-May-2018].

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[SB14] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *The MIT Press*, A Bradford Book, 2014.

[SML+15] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2015.