A Minimal Introduction to Reinforcement Learning

November 9, 2024

With its latest successes on fine-tuning language models as well as generalizing robotic agents, Reinforcement Learning (RL) has regained attentions ever since the prevalence of AlphaGO. Unlike the other canonical ML paradigms, RL follows a different set of terminology which, as I've experienced several times, sets back the path of going from zero to hero. Nonetheless, in this blog I will explain RL through the lens of supervised learning, and hopefully it could make the learning curve steeper for interested amateurs.

TL;DR: RL is identical to multi-class regression.

1 Example

Imagine finding knowledge on Google Search. You type some keywords in the search bar, and Google returns some relevant pages. You evaluate the proximity between your desired outcome and the displayed pages, and type calibrated keywords accordingly for another try. You repeat the above procedure till it shows the desired webpages, or until the time is through.

2 The Formulation of RL

By definition, RL is about making better decisions based on the insights pertinent to the observations that are obtained from its interaction with a system. Through calculated trial and error, it formulated a strategy which reinforces what it did right, and suppresses what it did wrong.

The key components in RL are marked in different but consistent colors, for both of the example and definition above. In the terminology of RL

- observations $\rightarrow$ state s
- decisions $\rightarrow$ action a
- insights $\rightarrow$ reward $r(s, a)$

- strategy $\rightarrow$ policy $\pi(a|s)$
- system $\rightarrow$ environment $\mathcal{P}(s'|s,a)$, that returns the next state s' given current state s and action a

Here[1], the RL interacts with the environment by following a policy $\pi(a|s)$ for T times, and obtains a sequence of information (episode) as

$$s_0, a_0, r_0,\ s_1, a_1, r_1,\ \cdots\ s_T, a_T, r_T$$

Given this information, the goal of RL is to learn the policy that maximizes the cumulative rewards

$$\max_{\pi} \sum_{t=0}^{T} r(s_t, a_t) \tag{1}$$

where the action a_t follows the policy, i.e., $a_t = \pi(\cdot|s_t)$. Essentially, the policy can be obtained through a neural network $Q(s,a)$,

$$Q(s_t, a_t) = \sum_{t=0}^{T} r(s_t, a_t) \tag{2}$$

such that $\pi(\cdot|s_t) = \arg\max_a Q(s_t, a_t)$. For problem with m discrete possible actions, it is identical to m-class classification where the neural network returns score for each action (class), and yield the one with the highest value as output.

In summary, the goal of RL is to learn a policy $\pi(\cdot|s_t)$ that maximizes the cumulative reward (1) through a neural network $Q(s,a)$ (2). In such manner, to learn a policy is much the same as a **multi-class regression problem**.

3 The Connections

For supervised learning, the model $f_\theta(x)$ is parameterized by θ and the parameters need to be learned through supervision.

In regression problem, the supervision is formulated as minimizing the least squares error, $\min\limits_{\theta}\ (f_\theta(x) - y)^2$, where y are the **static** targets. In essence, the parameters are learned by forcing the model $f_\theta(x)$ to **converge** to the targets. For example, we can learn θ gradually through gradient descent:

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta\ (f_\theta(x) - y)^2$$

Similarly, in RL the model $Q_\theta(s,a)$ is parameterized by θ, and we can also learn the parameters through regression.

[1]Don't know why but I found the term "Here" is kind of a terminology in RL literatures

But what are the targets to converge to? What error to minimize?

In my perspective, the formulation of the target and the error function is the most intriguing part in RL. Given the definition of Q function in Eq (2) and a tuple of (s, a, s', r)[2] from an episode, we can utilize the well-known *Bellman Equation* and have:

$$\underbrace{Q_\theta(s,a)}_{f_\theta(x)} \sim \underbrace{r(s,a) + \max_{a'} Q_\theta(s', a')}_{\text{y}} \tag{3}$$

where the squared difference between the left-hand side and right-hand side in Eq (3) is the **temporal difference error** to be minimized. Unlike supervised regression, the targets y are obtained from interactions with the environment and are no longer static. In other words, the targets are **dynamic** as they are collected by the policy during the learning process. Specifically, the training loop is as follows:

```
for each epoch:
  - collect $(s, a, s', r)$ through $\arg\max_a Q_\theta(s,a)$
  - update $Q_\theta(s,a)$ by minimizing the TD error
    as in Eq (3)
```

Similar to regression problem, we can use gradient descent to iteratively update the parameters θ at each epoch as:

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta \left(\underbrace{Q_\theta(s,a)}_{f_\theta(x)} - \underbrace{r(s,a) + \max_{a'} Q_\theta(s', a')}_{\text{y}} \right)^2$$

Interestingly, in this setup we implicitly require the **convergence** of Q function to an optimality governed by itself.

To conclude, RL can be regarded as a specific case of regression problem, in which the parameters are learned through minimizing the temporal difference error, and the targets y are dynamically generated throughout training.

Reference

- Hugging Face tutorial: for beginner
- OpenAI tutorial: for amateur
- PyTorch DQN: sample code

[2]s' is the next state in episode