Images haven't loaded yet. Please exit printing, wait for images to load, and try to print again.

https://github.com/khuangaf LinkedIn: https://goo.gl/7DEfvr
Jan 17 · 11 min read

Introduction to Various Reinforcement Learning Algorithms. Part II (TRPO, PPO)

In the first part of this series *Introduction to Various Reinforcement Learning Algorithms. Part I (Q-Learning, SARSA, DQN, DDPG)*, I talked about some basic concepts of Reinforcement Learning (RL) as well as introducing several basic RL algorithms. In this article, I will continue to discuss two more advanced RL algorithms, both of which were just published last year. In the end, I am going to briefly make a comparison between each of the algorithm I have discussed.

. . .

1. Getting Started

Definition:

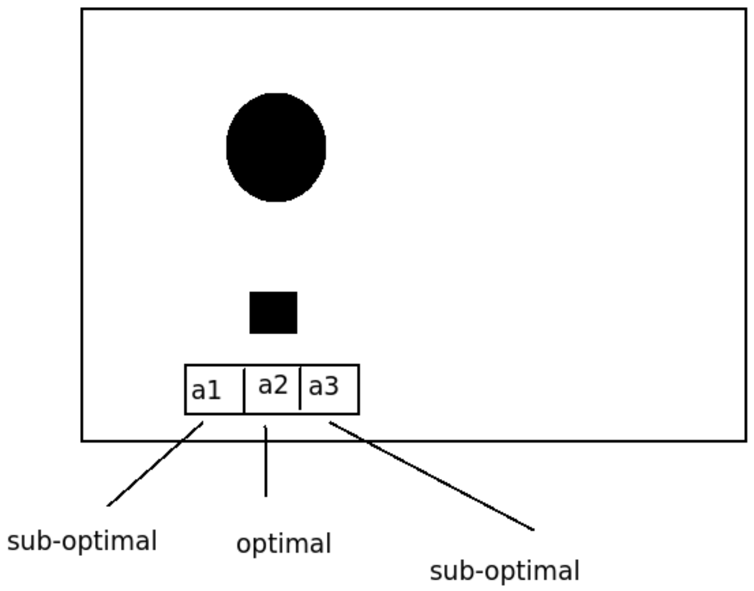
- Advantage (A): A(s, a) = Q(s, a)- V(s)

Advantage is a term that is commonly used in numerous advanced RL algorithms, such as A3C, NAF, and the algorithms that I am going to discuss (perhaps I will write another blog post for these two algorithms). To view it in a more intuitive manner, think of it as how good an action is compared to the average action for a specific state.

But why do we need advantage? Isn't Q-value good enough?

I will use an example posted in this forum to illustrate the idea of advantage.

Have you ever played a game called "Catch"? In the game, fruits will be dropping down from the top of the screen. You need to move the basket right or left in order to catch them.



Catch (https://datascience.stackexchange.com/questions/15423/understanding-advantage-functions)

The above image shows a sketch of the game. The circle on the top represents a fruit, whereas the small rectangle below is a basket. There are three actions, a1, a2, and a3. Apparently, the best action is a2, not moving at all, as the fruit will directly fall into the basket. Now, assume that there is no negative reward for any action. In this case, the agent does not have the incentive to choose the optimal action, a2 in the above scenario. Why? Let's use Q*(s, a) to denote the optimal Q value for state s and action a. Then we will have:

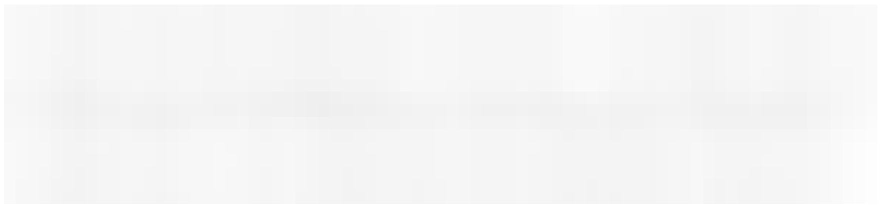
$$Q^*(s_t, a_3) \approx r(a_3) + \gamma \ [max_i \ Q^*(s_{t+1}, a_i)]$$

$$Q^*(s_t, a_3) \approx r(a_3) + \gamma[r(a_1) \ + \gamma \ [max_i \ Q^*(s_{t+2}, a_i)]]$$

$$Q^*(s_t, a_3) \approx r(a_3) + \gamma r(a_1) \ + \gamma^2 \ [Q^*(s_t, a_2)]$$

(https://datascience.stackexchange.com/questions/15423/understanding-advantage-functions)

Assume the discount factor γ is only slightly smaller than 1. We can get



(https://datascience.stackexchange.com/questions/15423/understanding-advantage-functions)

Since there is no negative reward, r(a3) and r(a1) are both greater or equal to 0, implying that Q*(s, a3) and Q*(s, a2) are not very different. Thus, the agent will only have little preference of a2 over a3 in this situation.

To solve this problem, we can compare the Q-value for each action with the average of them so that we know how good an action is relative to each other. Recall from the last blog that the average Q-value of a state is defined as Value (V). Essentially, we coin a new operator called ***advantage***, which is define by subtracting the Q-value for each of the action with the Value of that state.

2. Algorithms Illustration

2.1 Trust Region Policy Optimization (TRPO)

Deep Deterministic Policy Gradient (DDPG) discussed in the last post was a break through that allows agent to perform actions in a continuous space while maintaining a descent performance. However, the main issue of DDPG is that you need to pick the step size that falls into the right range. If it is too small, the training progress will be extremely slow. If it is too large, conversely, it tends to be overwhelmed by the noise, leading to tragic performance. Recall that the target for calculating the Temporal Difference (TD) error is the following:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$$

Target for TD error (https://arxiv.org/pdf/1509.02971.pdf)

If the step size is selected inappropriately, the target *yi* derived from the networks or function estimators will not be good, leading to a even worse sample and worse estimate of the value function.

Therefore, what we need is a way to update parameters that guarantees policy improvement. Namely, we want the **expected discounted long-term reward η** to be always increasing.

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right]$$

Expected Discounted Long-term Reward (https://arxiv.org/pdf/1509.02971.pdf)

> ***WARNING:*** *There will be numerous mathematical equations and formulas in this section. If you are not comfortable with that, feel free to jump to the end of this section.*

Similar to DDPG, TRPO also belongs to the category of policy gradient. It adopts the **actor-critic** architecture, but modifies how the policy parameters of the actor are updated.

For a new policy π', η(π') can be viewed as the the expected return of policy π' in terms of the advantage over π, which is the old policy. (Since I cannot find π with a curve on it on my keyboard, I will use π' in the following paragraphs)

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \cdots \sim \tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right]$$

η For New Policy π' (https://arxiv.org/pdf/1509.02971.pdf)

You might wonder why advantage is used. Intuitively, you can think of it as measuring how good the new policy is with regard to the average performance of the old policy. η of the new policy can be rewrite into the following form, where **ρ** is the **discounted visitation frequencies**.

η Rewrite & ρ (https://arxiv.org/pdf/1509.02971.pdf)

However, the above formula is hard to be optimized since ρ is highly dependent on the new policy π'. Therefore, the paper introduced an approximation to η(π'), Lπ(π'):

Approximation of η (https://arxiv.org/pdf/1509.02971.pdf)

Note that we replace ρπ with ρπ', assuming state visitation frequency is not too different for the new and the old policies. With this equation, we can combine with the well-know policy update method:

.

$$\pi_{\text{new}}(a|s) = (1-\alpha)\pi_{\text{old}}(a|s) + \alpha\pi'(a|s).$$

CPI (https://arxiv.org/pdf/1509.02971.pdf)

Here, π_{old} is the current policy, while π' is the argument max of the policy that maximizes L_{πold}. We will then obtain the following theorem (Let's use Theorem 1 to denote it).

Theorem 1 (https://arxiv.org/pdf/1509.02971.pdf)

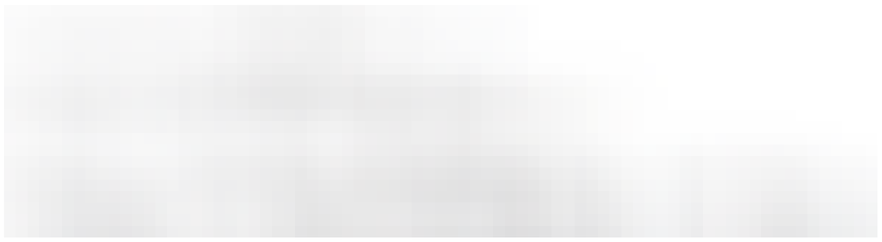
C represents the penalty coefficient, whereas D^{max}_{KL} denotes the maximum KL divergence of the two parameter for each of the state. The concept of KL divergence was originated from information theory, describing the information loss. Simply put, you can view it as how different these two parameters, π and π', are.

The above formula implies that the expected long-term reward η is monotonically improving as long as the right-hand-side term is maximized. Why? Let's define the right-hand-side of the inequality to be M_{i}.

$$M_i\left(\pi\right) = L_{\pi_i}\left(\pi\right) - CD_{KL}^{\max}\left(\pi_i, \pi\right)$$

(https://arxiv.org/pdf/1509.02971.pdf)

We can then prove the following inequality.



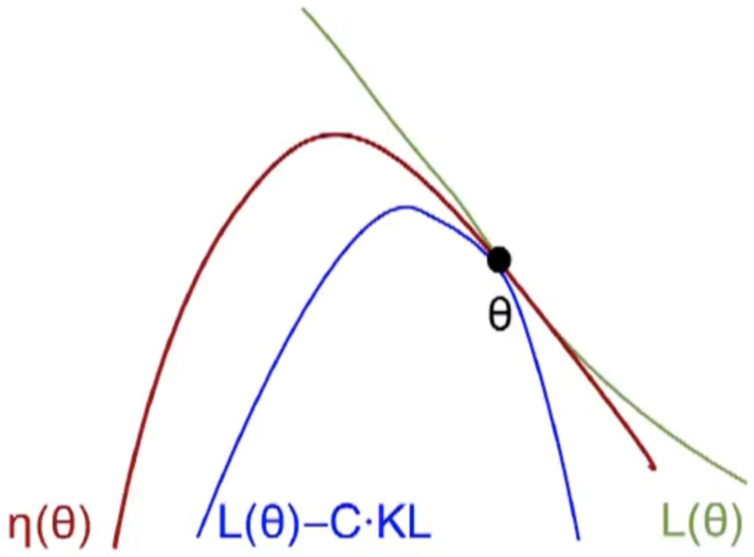
(https://arxiv.org/pdf/1509.02971.pdf)

The first line can be obtained by simply plugging the definition of M_{i} into Theorem 1. The second line holds because the KL divergence between π_{i} and π_{i} is 0. Combining the first and the second line, we will get the third line. This shows that as long as M_{i} is maximized at every iteration, the objective function η is always improving. (I think the last term atthe end of the third line should be Mi instead of M. Not sure if it is a typo if the paper). Therefore, the complex problem that we are trying to solve now boils down to maximizing *Mi.* Namely,

$$\underset{\theta}{\text{maximize}}\left[L_{\theta_{\text{old}}}(\theta) - C D_{\text{KL}}^{\max}(\theta_{\text{old}}, \theta)\right]$$

Objective Function 1 (https://arxiv.org/pdf/1509.02971.pdf)

The following graph visually illustrates the approximation of η with L:



Visual Illustration of The Approximation (https://www.youtube.com/watch?v=xvRrgxcpaHY&t=363s)

In practice, if penalty coefficient is included in the objective function, the step size will be very small, leading to long training time. Consequently, a constraint on the KL divergence is used to allow a larger step size while guarantee robust performance.

$$\underset{\theta}{\text{maximize}}\; L_{\theta_{\text{old}}}(\theta)$$
$$\text{subject to } D_{\text{KL}}^{\max}(\theta_{\text{old}}, \theta) \leq \delta$$

Objective Function 2 (https://arxiv.org/pdf/1509.02971.pdf)

The KL divergence constraint is imposed on every state in the state space, the maximum of which should be smaller than a small number δ. Unfortunately, it is not solvable as there are a infinitely large number of states. The paper proposed a solution which provides a heuristic approximation with the expected KL divergence over states, as opposed to finding the maximum KL divergence.

$$\overline{D}_{\text{KL}}^{\rho}(\theta_1, \theta_2) := \mathbb{E}_{s\sim\rho}\left[D_{\text{KL}}(\pi_{\theta_1}(\cdot|s) \parallel \pi_{\theta_2}(\cdot|s))\right]$$

KL Divergence With State Visitation Frequency ρ (https://arxiv.org/pdf/1509.02971.pdf)

Now, the objective function becomes the following when we expand the first line:

[illegible]

Objective Function 3 (https://arxiv.org/pdf/1509.02971.pdf)

By replacing Σ over states with expectation and Σ over the actions with importance sampling estimator, which is equal to the old policy if adopting single path method, we can rewrite the above as:

Final Objective Function (https://arxiv.org/pdf/1509.02971.pdf)

The objective function is also called a “surrogate” objective function as it contains a probability ratio between current policy and the next policy. TPRO successfully addresses the problem imposed by DDPG that the performance does not improve monotonically. The subset of region lies within the constraint is called trust region. As long as the policy change is reasonably small, the approximation is not much different from the true objective function. By choosing the new policy parameters which maximizes the expectation subject to the KL divergence constraint, a lower bound of the expected long-term reward η is guaranteed. This also implies that you don’t need to worry too much about the step size with TRPO.

2.2 Proximal Policy Optimization (PPO, OpenAI version)

Although TRPO has achieved great and consistent high performance, the computation and implementation of it is extremely complicated. In TRPO, the constraint imposed on the surrogate objective function is the KL divergence between the old and the new policy.

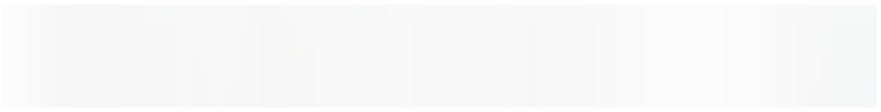
Fisher Information Matrix, a second-order derivative of KL divergence, is used to approximate the KL term. This results in computing several second-order matrixes, which requires a great amount of computation. In the TRPO paper, Conjugate Gradient (CG) algorithm was used to solve the constrained optimization problem so that the Fisher Information Matrix does not need to be explicitly computed. Yet, CG makes implementation more complicated.

PPO gets rid of the computation created by constrained optimization as it proposes a clipped surrogate objective function.

Let rt($\boldsymbol{\theta}$) denotes the ratio between the new and the old policy. The surrogate objective function used for approximating long-term reward η for TRPO becomes the following. Note the subscript describes the conservative policy iteration (CPI) methods that TRPO is based on.

TRPO Objective Function (https://arxiv.org/pdf/1707.06347.pdf)

The idea of TRPO's constraint is disallowing the policy to change too much. Therefore, instead of adding a constraint, PPO slightly modifies TRPO's objective function with a penalty for having a too large policy update.



Clipped Objective Function (https://arxiv.org/pdf/1707.06347.pdf)

On the right you can see that the probability ratio rt(θ) is clipped between [1- ϵ, 1+ϵ]. This indicates that if rt(θ) causes the objective function to increase to a certain extent, its effectiveness will decrease (be clipped). Let's discuss two different cases:

- Case 1: When the advantage Ât is greater than 0

If Ât is greater than 0, it means that the action is better than the average of all the actions in that state. Therefore, the action should be encouraged by increasing rt(θ) so that this action has a higher chance to be adopted. Since the denominator of rt(θ) is constant, the old policy, increasing rt(θ) also implies increasing the new policy πθ(a, s). Namely, increase the chance for taking that action in the given state. However, because of the clip, rt(θ) will only grows to as much as 1+ϵ.

- Case 2: When the advantage Ât is smaller than 0

By contrast, if Ât is smaller than 0, then that action should be discouraged. As a result, rt(θ) should be decreased. Similarly, due to the clip, rt(θ) will only decreases to as little as 1-ϵ.

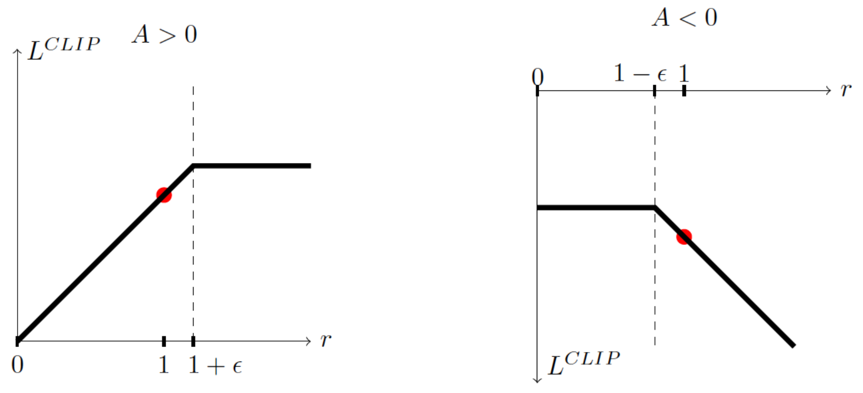


Illustration of The Clip (https://arxiv.org/pdf/1707.06347.pdf)

Essentially, it restricts the range that the new policy can vary from the old one; thus, removing the incentive for the probability ratio rt(*θ*) to move outside the interval.

In practice, loss function error and entropy bonus should also be considered during implementation as shown below. However, I am not going into details of them as the most innovative and important part is still the clipped objective function.

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right]$$

PPO Objective Function (https://arxiv.org/pdf/1707.06347.pdf)

Comparing the objective function L^{CPI} and L^{CLIP}, we can observe that L^{CLIP} is in fact a lower bound of the former one. It also removes the KL divergence constraint. Consequently, the computation for optimizing this PPO objective function is much less than that of TRPO's. Empirically, it also proves that PPO's performance is better than TRPO. In fact, thanks to its lightness and ease of implementation, PPO has become the default RL algorithm of OpenAI (https://blog.openai.com/openai-baselines-ppo/).

3. Comparison of Discussed Algorithms

Various RL Algorithms I Have Discussed

All the discussed RL algorithms are model-free. That is, non of them are trying to estimate the objective function. Instead, they update their knowledge based on trial-and-error. Among all of them, only SARSA is on-policy, learning value based on its current action. DQN was a huge improvement from a discrete observation space to a continuous one, allowing the agent to handle unseen state. DDPG was another break through that enables agent to perform continuous actions with policy gradient, broadening the application of RL to more tasks such as control. TRPO improves the performance of DDPG as it introduces a surrogate objective function and a KL divergence constraint, guaranteeing non-decreasing long-term reward. PPO further optimizes TRPO by modifying the surrogate objective function, which improves the performance as well as decreasing the complexity of implementation and computation.

Conclusion

To sum up, I have introduced two more advanced RL algorithms and compared all the RL algorithms that I have discussed. Nevertheless, in TRPO, the mathematical formula is very complicated. Although I have tried my best to explain it, I believe it might still seems confusing to some of you. Please feel free to leave a comment below if you have any question or drop me an email: khuangaf@connect.ust.hk. If you like my article, I will write another blog to discuss model-based algorithms which allow for planning ahead or imagination for the agents.