Trust Region Policy Optimization and Proximal Policy Optimization

Manan Tomar : ED14B023

Abstract

This work introduces a policy optimization method with monotonic improvement guarantees. This is then proposed as a practical algorithm, called Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) by using a few approximations and is shown to work on continuos control Mujoco tasks as well as on the Atari domain with fairly robust performance across multiple environments. A slight variant of TRPO which is based on modifications to the originally derived objective is also proposed, called Proximal Policy Optimization (PPO) (Schulman et al., 2017). PPO offers simplicity in terms of implementation and is shown to perfom as well as TRPO if not better.

1. Preliminaries

The expected discounted reward for the reinforcement learning problem is defined as :

$$\eta(\pi) = \mathbb{E}_{s_0,a_0,\ldots}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t)\right] \tag{1}$$

Moreover, state-action value function, state value function and advantage function are defined as :

$$Q_\pi(s_t,a_t) = \mathbb{E}_{s_{t+1},a_{t=1},\ldots}\left[\sum_{l=0}^{\infty}\gamma^l r(s_{t+l})\right] \tag{2}$$

$$V_\pi(s_t) = \mathbb{E}_{a_t,s_{t+1},\ldots}\left[\sum_{l=0}^{\infty}\gamma^l r(s_{t+l})\right] \tag{3}$$

$$A_\pi(s_t,a_t) = Q_\pi(s_t,a_t) - V_\pi(s_t) \tag{4}$$

Kakade and Langford (Kakade & Langford, 2002) show that η for an updated policy $\tilde{\pi}$ can be written in terms of the performance of an old policy π as

$$\eta_(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t \bar{A}(s_t)\right] \tag{5}$$

, where the expected advantage $\bar{A}(s) = \mathbb{E}_{a\sim\tilde{\pi}(.|s)}\left[A_\pi(s,a)\right]$. Now, in such a case, performance improvement is guaranteed if the expected advantage is positive for every state. However, such is not the case in the function approximation setting. Morevoer, using trajectories sampled from $\tilde{\pi}$ is difficult to optimize directly.

2. Surrogate Loss

Due to the above mentioned reasons, a surrogate loss is introduced as :

$$L_\pi(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t \bar{A}(s_t)\right] \tag{6}$$

Such a loss differs from the original loss η only in terms of the trajectories being sampled using the old policy π instead of $\tilde{\pi}$. Moreover, L_π and η match to the first order. Therefore a sufficiently small step that improves L_π will also improve η. Essentially, providing a lower bound on the difference in the original and surrogate loss will allow improving the original loss, given surrogate loss is improved. To ensure this, Kakade and Langford (Kakade & Langford, 2002) propose conservative policy iteration which defines a mixture of policies to ensure taking small steps.

However, in practical, mixture policies are quite restrictive in nature a thus a method which constraints the step size in a better way is required. To this end, the authors propose using a constraint on the KL divergence between the old and updated policies. This results in the following lower bound :

$$\eta(\tilde{\pi}) \geq L_\pi(\tilde{\pi}) - CD_{KL}^{max}(\pi,\tilde{\pi}) \tag{7}$$

, where $C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$, $D_{KL}^{max}(\pi,\tilde{\pi}) = max_s D_{KL}(\pi,\tilde{\pi})$

3. Practical Considerations

The above provides theoretical guarantees. However, it does not provide a practical algorithm to work with. In order to ensure this, the authors mainly include two modifications to the above objective :

- Since, in practise, the step size C is very small, a hard constraint is introduced as follows :

$$\underset{\theta}{\text{maximize}}\ L_{\theta_{old}}(\theta)$$

subject to $D_{KL}^{max}(\theta_{old}, \theta) \leq \delta (8)$

- The KL term is currently constrained for all states, which is impractical to optimize. Therefore, a single constraint on the average KL is used instead :

$$\mathbb{E}_{s\sim\rho}\left[D_{KL}(\theta_{old}, \theta)\right] \leq \delta \tag{9}$$

Moreover, the expected advantage can be written as

$$\mathbb{E}_{a\sim\tilde{\pi}}\left[A_{\pi}(s,a)\right] = \mathbb{E}_{a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}A_{\theta_{old}}(s,a)\right] \tag{10}$$

using importance sampling, where q is the sampling distribution. Therefore, expanding L_π from above and using the importance sampling expression for the expected advantage gives :

$$\underset{\theta}{\text{maximize}}\ \mathbb{E}_{s\sim\rho_{\theta_{old}}, a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta old}(s,a)\right]$$
$$\text{s.t. } \mathbb{E}_{s\sim\rho_{\theta_{old}}}\left[D_{KL}(\pi_{\theta_{old}}, \pi_\theta)\right] \leq \delta \tag{11}$$

4. Sampling Schemes

To estimate the Q values in the above objective using trajectory samples, two sampling schemes are introduced.

- **Single Path** Simulating $\pi_{\theta_{old}}$ to generate trajectory $s_0, a_0, s_1, a_1, ..., s_T$. Q is computed for each action pair.
- **Vine** : Simulating multiple policies π_i and choosing a random subset of N states. From each of these N states, k actions are chosen and rollouts are performed are performed to estimate Q. Since this method requires multiple trajectories with rollouts stemming from different states, it assumes that the system can be reset to any desired state, which is quite restrictive in nature. The benefit here is in terms of a Q estimate with reduced variance.

5. Optimization

Eventually, we are left with the following optimization problem :

$$\text{maximize}\, L(\theta)\ s.t.\ \bar{D}_{KL}(\theta_{old}, \theta) \leq \delta \tag{12}$$

This is done using conjugate gradient descent in the following three steps :

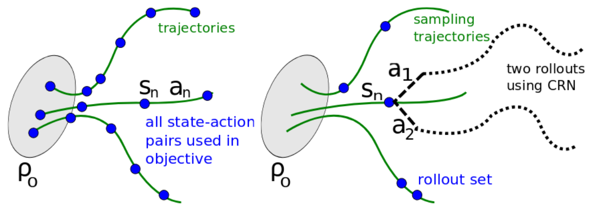


Figure 1. The *single* and *vine* sampling schemes used to estimate Q.

- Compute a search direction $s = F^{-1}g$, while approximating objective as linear (g) and constraint as quadratic (F), where g is the gradient of the objective term and F is the Fischer Information matrix.
- Compute maximal step length β given search direction from above such that KL is less than δ.
- Start with maximal step length from above and shrink it until the objective improves.

6. Proximal Policy Optimization

From above, using the importance sampling ratio, the authors define :

$$r_t(\theta) = \frac{\pi_\theta(s,a)}{\pi_{\theta_{old}}(s,a)} \tag{13}$$

Since the main idea is to have the correct step size at every optimization step, the authors propose a clipping loss as below instead of a constrained objective.

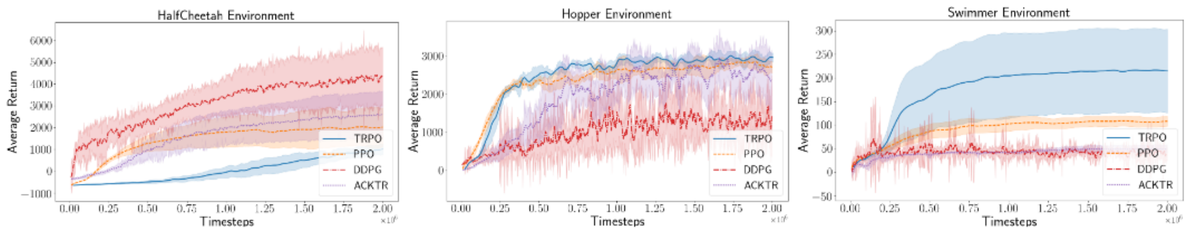
$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[min(r_t(\theta)\hat{A}_t,\ clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{14}$$

, where r is clipped to $1+\epsilon$ when $\hat{A}$ is positive and to $1-\epsilon$ when it is negative. Such a loss is simple to train on and implement, and ensures that the updated policy is always close to the old policy by clipping it to an ϵ neighbourhood.

7. Adaptive KL Penalty

The authors also introduce a different objective as opposed to the clipped loss, one which uses a penalty term for the KL divergence instead of a hard constraint (as in TRPO). This is done by ensuring an adaptive penalizing term.

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(s,a)}{\pi_{\theta_{old}}(s,a)}\hat{A}_t - \beta KL(\pi_{\theta_{old}}, \pi_\theta)\right] \tag{15}$$



Environment	DDPG	ACKTR	TRPO	PPO
HalfCheetah-v1	5037 (3664, 6574)	3888 (2288, 5131)	1254.5 (999, 1464)	3043 (1920, 4165)
Hopper-v1	1632 (607, 2370)	2546 (1875, 3217)	2965 (2854, 3076)	2715 (2589, 2847)
Walker2d-v1	1582 (901, 2174)	2285 (1246, 3235)	3072 (2957, 3183)	2926 (2514, 3361)
Swimmer-v1	31 (21, 46)	50 (42, 55)	214 (141, 287)	107 (101, 118)

Table 3: Bootstrap mean and 95% confidence bounds for a subset of environment experiments. 10k bootstrap iterations and the pivotal method were used.

Figure 2. TRPO and PPO performance as reported in (Henderson et al., 2017), which reproduces both the original works and compares performance with others such as Deep Deterministic Policy Gradients. The environments are Half-Cheetah, Hopper, Walker and Swimmer.

β is updated based on how KL is away from a target distance d_{targ}

$$d = \hat{\mathbb{E}}_t[KL(\pi_\theta, \pi_\theta)] \tag{16}$$

- if $d < d_{targ}/1.5$, update $\beta \leftarrow \beta/2$

- if $d > d_{targ} \times 1.5$, update $\beta \leftarrow \beta \times 2$

8. Experiments and Results

Both TRPO and PPO are shown to work for various Mujoco tasks as well as for the Atari domain. For implementing this, neural network policies are used in both cases. For the continuous action setting, a learnable covariance is used along with the mean given by a neural network, to model a gaussian policy. The experiments largely show that both these methods can be widely applied across different environments, reaching on par performance with other popular methods. Moreover, both methods have been shown to be quite robust against training across multiple seeds, showing very less variance as compared to other works (Duan et al., 2016) (Henderson et al., 2017). There are some issues, in terms of the performance varying substantially on some of the tasks. For instance, TRPO does much better than any other method in the Swimmer environment while performing really poorly on Half-Cheetah at the same time. There also does not seem to be a clearly better method when comparing TRPO and PPO head to head across 4-5 environments. This again points to how sensitive each environment is to the step size used.

9. Related Work and Conclusion

TRPO resembles most prominently with *Natural Policy Gradient* (Kakade, 2002) method. This can be obtained by assuming a linear and quadratic approximation to the objective and the constraint terms in the surrogate loss and solving using a fixed step size. Other works (Rajeswaran et al., 2017) have also proposed using a adaptive step size in this case, by using a normalized step which depends on the gradient as well as the Fischer Information matrix. This has yielded results comparable to PPO. This indicates that the step size is probably the hardest of parameters to tune when searching directly in the policy space and can affect performance significantly.

References

Duan, Yan, Chen, Xi, Houthooft, Rein, Schulman, John, and Abbeel, Pieter. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338, 2016.

Henderson, Peter, Islam, Riashat, Bachman, Philip, Pineau, Joelle, Precup, Doina, and Meger, David. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

Kakade, Sham and Langford, John. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pp. 267–274, 2002.

Kakade, Sham M. A natural policy gradient. In *Advances in neural information processing systems*, pp. 1531–1538, 2002.

Rajeswaran, Aravind, Lowrey, Kendall, Todorov, Emanuel V, and Kakade, Sham M. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pp. 6550–6561, 2017.

Schulman, John, Levine, Sergey, Abbeel, Pieter, Jordan, Michael, and Moritz, Philipp. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.