Reward Constrained Policy Optimization

PeiChieh Sung
Department of Information management and finance
National Yang Ming Chiao Tung University
pjs5956.mg07@nycu.edu.tw

1 Introduction

1.1 The main research challenges :

1. In Multi-Objective RL, in order to maximize total rewards, additional rewards will be given to guide the agent.

Problem : The coefficients, which controls the different signals from Multi-Objective RL, cannot be cross-domain. This cause different combinations of factors lead to different optimal solutions.

2. Using constraints becomes a better choice. We don't need to optimize the parameters, the only thing is to make them satisfy some constraints.

Problem : The existed methods can only work as using parameterized policy work; however, some Q-learning algorithms such as DQN is not supported by such training algorithms. Also, some methods do not support mean value constraints.

3. The paper came up with the solution "Reward Constrained Policy Optimization", which added the constraint as a penalty term in reward function and solved the problem.

1.2 The high-level technical insights into the problem of interest :

Since the "Reward Constrained Policy Optimization" combined Constrained MDP's and policy optimization, the paper chose to solve by using the Lagrange relaxation technique, which was a two-timescale method. It meant that in the fast timescale, the parameter of neural network was continuously updating in order to maximize the reward J_R. On the other side, in the slow timescale, the Lagrange multiplier grow larger to minimize $-\lambda J_C$, which meant to maximize the penalty.

Also, the methods applied the modified Actor-Critic, which considered the discounted penalty in order to satisfy the recursive property required to train a critic. So the total methods used three timescale approach. First, on the fast timescale, updated critic by using TD-critic, hence θ and λ were static and got the results $v_k -> v(\theta, \lambda)$. Then, on the second timescale, using policy gradient to updated θ_k. Lastly, on the slow timescale, the Lagrangian was updated by ascending on the original constraint.

1.3 The main contributions :

The paper had three major benefits. First, it not only supported discounted sum constraints, but also supported mean value constraints. Second, the work was reward agnostic, that is, there was no requirement for the magnitude of the reward. Third, no prior knowledge was used.

1.4 My personal perspective on the method :

The paper proposed a method for learning constraint satisfaction policies with complex constraints. Not only the method could solve the problem with multiple complex constraints but guarantee the solution was always feasible.

I think the method of this paper is helpful to better achieve the method of CMDP. Taking the recommended advertising system as an example, usually advertisers cannot know the best solution for the budget on the investment, that is to say, how much money invested in each user will bring the best return, or which one has the highest rate of return (ROI). These two cases do not necessarily occur in the same solution, where it is usually not a priori feasible to evaluate tradeoffs so that an appropriate budget can be set.

Moreover, in different tasks, limited resources must be shared among multiple tasks, and each formed by a different MDP. In this case, it is important to compare the value that can be achieved by a single specific MDP with the value of multiple MDPs across tasks for resource allocation.

The method successfully solve all of the problem above.

2 Problem Formulation

2.1 Markov decision process (MDPs)

Markov decision process (MDPs)included the tuple $(S, A, R, P, \mu, \gamma)$. The followings are the definitions :

- Set of states S
- Set of action A
- Rewards $R(s, a, s')$
- Transitions probability $P(s')$
- The initial state distribution $\mu S- > [0, 1]$
- The discount factor for future rewards $\gamma \subseteq [0, 1)$
- A Policy $\pi(a \mid s)$ is the probability that the agent will take a at state s.
- π is a probability distribution over actions.

In Markov Decision Process, the value function is $V_R^\pi(s) = \mathbb{E}^\pi[r(s_t, a_t) + \gamma V_R^\pi(s') \mid s]$, and we are going to maximize the expectation function $\underset{\pi \subseteq \Pi}{max} J_R^\pi$ under the distribution μ.

$$maxJ_R^\pi = \mathbb{E}_{s\sim\mu}^\pi[\sum_{t=0}^{\infty} \gamma^t r_t] = \sum_{s\subseteq S} \mu(s) V_R^\pi(s) \tag{1}$$

2.2 Constrained Markov decision processes(CMDP)

The Constrained Markov Decision Processes are extensions to Markov decision process (MDPs), which is an enhanced version of MDP constraints and limits the feasible strategies of MDP. Specifically, MDP will be subject to a series of cost functions C_1, ... ,C_N. $C : S \times A \times S' \to R$ maps a conversion tuple to the cost, just like the reward function in MDP. And similarly $J_{C_i}(\Pi), i \subseteq [0, N]$ represents the cumulative discount cost. The goal is to find a cost-constrained strategy that can make cumulative mapping Π with the largest discount reward function. However, there is some problem in CMDPs. First, CMDPs are solved with linear programs only, and dynamic programming, which applys on MDP does not work. Second, the stationary policies usually suffice in CMDPs.Third, it is hard to check if the solution is feasible or not in CMDP.

As the that Constrained Markov Decision Processes we known, the total expectation constraint is: $J_C^\pi = \mathbb{E}_{s\sim\mu}^\pi[C(s)]$. So the formula becomes:

$$\underset{\pi \subseteq \Pi}{max} J_R^\pi,\ s.t.\ J_C^\pi \leq \alpha.\ \alpha \subseteq [0, 1]. \tag{2}$$

As the mention above, the paper came up with two assumption:

Assumption 1: The value function $V_R^{\pi}(s)$is bounded for all π.
Assumption 2: All of the local minima of the objective $J_C^{\pi_\theta}$ is one of feasible solutions.

The first and second assumptions were to ensure that all the constraints could converge which satisfied the policy by making the assumptions above. As for the second assumption, it was the minimum assumption to guarantee convergence. Because in the method of gradient descent, if it met local solution, we could not make sure that this local solution was feasible; therefore, we should only view constraint as regularizing term.

2.3 The optimization problem of interest

2.3.1 Constrained Policy Optimization

The reason to use Constrained Policy Optimization is to ensure that the agent satisfies the constraints at each step in the learning process. Specifically, we try to satisfy constraints on cost, as the designer assigns a cost and a limit to each outcome that the agent should avoid, and the agent learns to make all costs are kept within limits. The paper solved the problem by using the Lagrange relaxation technique

$$\min_{\lambda \geq 0} \max_{\theta} L(\lambda, \theta) = \min_{\lambda \geq 0} \max_{\theta} [J_R^{\pi_\theta} - \lambda \cdot (J_C^{\pi_\theta} - \alpha)] \tag{3}$$

There was a two-timescale method. It meant that in the fast timescale, the parameter of neural network was continuously updating in order to maximize the reward J_R. On the other side, in the slow timescale, the Lagrange multiplier grow larger to minimize $-\lambda J_C$, which meant to maximize the penalty. Besides, the paper simulated the MDP by sampling; hence, the following are the algorithms to estimate the gradient. The Γ_λ project λ into $[0, \lambda_m ax]$. The Γ_θ project θ onto a compact and convex set :

1. $\lambda_{k+1} = \Gamma_\lambda[\lambda_k - \eta_1(k) \nabla_\lambda L(\lambda_k, \theta_k)]$

   $\nabla_\theta L$ is solved by the trick of log-likelihood :
   $\nabla_\theta L(\lambda, \theta) = \nabla_\theta \mathbb{E}_{s\sim\mu}^{\pi_\theta}[\log \pi(s, a; \theta)[R(s) - \lambda \cdot C(s)]]$

2. $\theta_{k+1} = \Gamma_\theta[\theta_k + \eta_2(k) \nabla_\theta L(\lambda_k, \theta_k)]$

   $\nabla_\lambda L$ is solved by the trick of log-likelihood :
   $\nabla_\lambda L(\lambda, \theta) = -(\mathbb{E}_{s\sim\mu}^{\pi_\theta}[C(s) - \alpha])$

Assumption 3:
$\sum_{k=0}^{\infty} \eta_1(k) = \sum_{k=0}^{\infty} \eta_2(k) = \infty, \sum_{k=0}^{\infty} (\eta_1(k)^2 + \eta_2(k)^2) < \infty \; and \; \frac{\eta_1(k)}{\eta_2(k)} \to 0$

In the formulation above, η_1 meant the learning rate of λ and η_2 meant the learning rate of θ. Due to the assumption, we supposed that λ was updated by slow timescale; therefore, $\frac{\eta_1}{\eta_2} = 0$, which meant we asked η_1 to be a tiny number and η_2 to be a large number.

2.3.2 Reward Constrained policy Optimization

The paper applied the modified Actor-Critic, which considered the discounted penalty in order to satisfy the recursive property required to train a critic. It came up to the method "Simple Reward Shaping", which used a simple way that viewed a lambda variable as weighted variable and substituted two critics, respectively; one of the critics was in charge of integrating the return of reward and another one was responsible for the return of cost:
$\hat{r}(\lambda, s, a) \triangleq r(s, a) - \lambda c(s, a)$,

$$\begin{aligned}\hat{V}^{\pi}(\lambda, s) &\triangleq \mathbb{E}^{\pi}[\sum_{t=0}^{\infty} \gamma^t \hat{r}(\lambda, s_t, a_t) \mid s_0 = s] \\ &= \mathbb{E}^{\pi}[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) - \lambda c(s_t, a_t)) \mid s_0 = s] \\ &= V_R^{\pi}(s) - \lambda V_{C_\gamma}^{\pi}(s)\end{aligned} \tag{4}$$

Assumption 4: $\Theta_\gamma \subseteq \Theta$

For the forth assumption, it was the most important one. We supposed all of the local-minimum of $J_C^{\pi_\theta}$ as Θ_γ satisfied the limited feasible solutions($\Theta = \theta : J_C^{\pi_\theta} \leq \alpha$) because we knew that by(3), when λ became infinity, the process converged to a feasible solution.
Else if the forth assumption did not hold, we could try to use Monte-Carlo method in order to approximate the gradient since it was unsafe to use J_C gradient under this circumstance. Yet, this situation could not decrease variance and improved sample efficiency because of the lack of critic. So the forth assumption must be existed.

3 Theoretical Analysis

3.1 Theorem 1

Under Assumption 3 and the standard stability assumption in (Borkar et al., 2008), iteration (θ_n, λ_n) almost surely converges to a fixed point.

Before proving the theorem, it did not make any assumptions for the structure of constraints so that the value of λ_{max} would be finite. First, in order to tackle the converge of θ, because of using the method of projection, the parameters of θ was stable. It could be known that we could view λ as a constant on the fastest timescale; therefore, we could do partial differential on θ, as below:

$$\dot{\theta}_t = \Gamma_\theta(\nabla_\theta L(\lambda, \theta_t)) \tag{5}$$

Γ_θ was a projection so that θ would be project on a compact and convex set. And the equation became

$$\theta_{k+1} = \Gamma_\theta[\theta_k + \eta_2(k)(\mathbb{E}_{s\sim\mu}^{\pi_\theta}[C(s) - \alpha])],$$

it would be approach to limit no matter what the value of λ was on the slowest timescale. Second, we are going to proof λ would converge and given that λ in the equation(3) converged, the overall convergence of (θ_k, λ_k) is a local saddle point $(\theta_*(\lambda_*, \lambda_*))$ of $L(\lambda, \theta)$.

$$\lambda_{k+1} = \Gamma_\lambda[\lambda_k - \eta_1(k)\nabla_\lambda L(\lambda_k, \theta(\lambda_k))]$$

As the equation above, λ_k was related to θ-recursion. Doing partial differential on λ, as below:

$$\dot{\lambda}_t = \Gamma_\lambda(\nabla_\lambda L(\lambda_t, \theta(\lambda_t))) \tag{6}$$

According to the (Borkar et al., 2008), we knew that it would be converged.

3.2 Theorem 2

The policy obtained using this method (RCPO) would almost certainly converge to a fixed point: $(\theta^*(\lambda^*, v^*), v^*(\lambda^*), (\lambda^*))$

The difference between Theorem 1 was that it proved that the value function also converges together. In this case, we suggested that the algorithm followed the three timescale approximation. Due to the algorithm, on the fast timescale, updated critic by using TD-critic, hence θ and λ were static and got the results $v_k -> v(\theta, \lambda)$. Then, on the second timescale, using policy gradient to updated θ_k. Critic v was already converged and λ was static. Lastly, on the slow timescale, Lagrangian was updated by ascending on the original constraint. $(\lambda_n, \theta_n, v_n) \rightarrow (\lambda(\theta^*), (\theta^*), v(\theta^*))$

3.3 Lemma 1.

Under assumptions 1 and 2, the fixed point of Theorem 1 was a feasible solution.

According to the proof of Theorem 1 above, the assumption 2 and the knowledge of the first order method could be converged to a local minima, since λ_{max} equaled to ∞, the whole process would converge to a fixed point($\theta^*(\lambda^*), \lambda^*$).

3.4 Small mistake:

There is a mistake in page 5 where is at algorithm 1 line 10. It must be equation 5 not 8.

4 Conclusion

1. **The Final Results**
   As for the experiment, it was tested on grid world and mujoco. The effect of grid world was to prove that the method of RCPO was based on the basic Primal-Duathe method; and for the mujoco, it showed that RCPO was better than the normal reward shaping method.
   Especially, in Mujoco, the method was based on PPO, it had good results on every environments except Walker2d-v2. Although reward of RCPO was not the highest, it satisfied all of the constraints. Here are the interest conclusions:
   - It was suitable to use large lambda to help those easy tasks which had high rewards to reduce costs.
   - Large lambda would increase the loss function for the difficult tasks.
   - Since the reward of the value function J_r kept increasing in training, a non-adaptive method was unavailable.

   Also, the paper compare RCPO to the common "reward shaping", with the difference λ. First, the the task of common reward shaping is difficult. Second, successful costs are hard to transfer across domains. Third, RCPO can find a solution that satisfies the constraints.

2. **The potential future research directions**
   For the future works, we can focus more on about how to impose constraints on multiple initial distribution. For example, in some cases, the solution of CMDP reliance on the initial distribution; however, it is also a disadvantage. For example, suppose an advertiser wants a strategy to attract customers online, but the initial state distribution depends on the amount of customer. Compare to compute separate policies for each initial distribution, it is more desirable to have value functions and policies that are easily adaptable to different initial distributions. To achieve the target, we can adopt neural network to predict parameters, and the input can be the multiple initial state distributions. Besides, we can design a efficient exploration method for CMDPs.

3. **Technical limitations**
   The paper updated the policy by a mirror-descent like update, which could be highly unstable, since it used gradient descend that the oscillations of θ and λ would appear. I think it will be better if adding some constraints or thresholds to control the value of θ and λ.

4. **Any latest results on the problem of interest**
   In the paper of Zheng and Ratliff [2020], it set the reward function and the constraints by upper confidence reinforcement learning, which were unknown and did not need prior knowledge.This setup was good for overcoming complexity and unsafe environments. It can be used in many application, which helps decision maker to select a perfect policy under constraints. Compared to the original paper, the method it used is the combination of lagrangian-based actor-critic algorithm and Constrained Policy Optimization. However, this kind of gradient-based method only guaranteed safety if the learning period is sufficient. The problem successfully solved by the paper "Constrained Upper Confidence Reinforcement

Learning,2020". It interacted with the stochastic environment as well as ensured performance guarantees.

This project is mainly introducing Tessler et al. [2018]

References

Liyuan Zheng and Lillian J. Ratliff. Constrained upper confidence reinforcement learning. *CoRR*, abs/2001.09377, 2020. URL `https://arxiv.org/abs/2001.09377`.

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. *CoRR*, abs/1805.11074, 2018. URL `http://arxiv.org/abs/1805.11074`.