Safe reinforcement learning?

Image source

Projected Proximal Policy Optimization for Safe Continuous-State Reinforcement Learning

Sergei Volodin

École Polytechnique Fédérale de Lausanne,
Theory and Methods of Reinforcement Learning EE-618

sergei.volodin@epfl.ch

July 22, 2021

Outline

1. What is the problem: CMDP
2. Why is this problem important (AI safety)
3. Current results
4. Environments
5. A need for a set of benchmarks
6. Existing Algo 1: Constrained Policy Optimization
7. Existing Algo 2: Lyapunov-based safe RL
8. Intuition behind the improvement
9. Proposed method: Projected PPO
10. Experimental results
11. Conclusion

Motivation: AI safety

Machine Learning is gaining momentum and affecting our lives, sometimes causing unintended side effects

1. Short-term (right now): adversarial examples (1), data poisoning
2. Long term (next 5-10 years): **safe exploration**(2), scalable oversight
3. Longer term (?): Artificial General Intelligence, value alignment (3)

At each level there is a need for a trade-off between right performance (solving the problem) and causing no harm

Practical goal: developing systems which learn without causing harm to the environment (e.g. a copter)

Problem setting: Constrained MDP (4)

1. Continuous set of states $\mathcal{S}$
2. Finite set of actions $\mathcal{A}$
3. Environment transition probabilities $p(s'|s, a)$
4. (Stochastic, stationary) policy: mapping $\pi\colon S \to \Delta\mathcal{A}^*$
5. Reward: a function $R\colon \mathcal{S} \times \mathcal{A} \to \Delta\mathbb{R}$
6. Return for reward:

$$J_R(\pi) = \mathbb{E}_{p\leftrightarrow\pi} \sum_{t=0}^{\infty} \gamma^t R_t$$

---

*Distribution over $\mathcal{A}$

Problem setting: Constrained MDP (4)

1. Continuous set of states $\mathcal{S}$
2. Finite set of actions $\mathcal{A}$
3. Environment transition probabilities $p(s'|s, a)$
4. (Stochastic, stationary) policy: mapping $\pi\colon S \to \Delta\mathcal{A}^*$
5. Reward: a function $R\colon \mathcal{S} \times \mathcal{A} \to \Delta\mathbb{R}$
6. Return for reward:

$$J_R(\pi) = \mathbb{E}_{p\leftrightarrow\pi} \sum_{t=0}^{\infty} \gamma^t R_t$$

---

7. Cost: a function $C\colon \mathcal{S} \times \mathcal{A} \to \Delta\mathbb{R}$
8. Return for cost:

$$J_C(\pi) = \mathbb{E}_{p\leftrightarrow\pi} \sum_{t=0}^{\infty} \gamma^t C_t$$

---

*Distribution over $\mathcal{A}$

Problem setting: Constrained MDP (4)

1. Continuous set of states $\mathcal{S}$
2. Finite set of actions $\mathcal{A}$
3. Environment transition probabilities $p(s'|s, a)$
4. (Stochastic, stationary) policy: mapping $\pi \colon S \to \Delta\mathcal{A}^*$
5. Reward: a function $R \colon \mathcal{S} \times \mathcal{A} \to \Delta\mathbb{R}$
6. Return for reward:

$$J_R(\pi) = \mathbb{E}_{p \leftrightarrow \pi} \sum_{t=0}^{\infty} \gamma^t R_t$$

---

7. Cost: a function $C \colon \mathcal{S} \times \mathcal{A} \to \Delta\mathbb{R}$
8. Return for cost:

$$J_C(\pi) = \mathbb{E}_{p \leftrightarrow \pi} \sum_{t=0}^{\infty} \gamma^t C_t$$

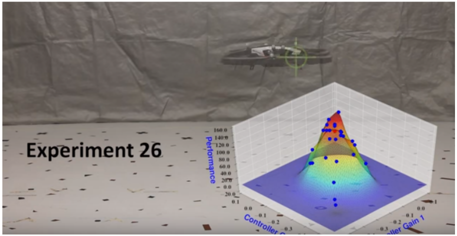
9. Want to solve: $\max_{\pi} J_R(\pi)$ s.t. $J_C(\pi) \leq C_{\max}$

---

*Distribution over $\mathcal{A}$

Existing results

Success story: automatic quadcopter controller tuning(5)



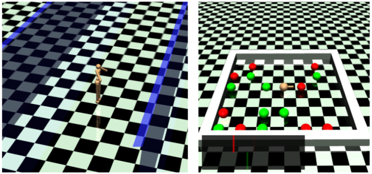
Project proposal

1. Creating a set of benchmark environments
2. Implementing existing algorithms
3. Comparing existing algorithms
4. Improving one of them and testing it

Environments used

In papers (6):

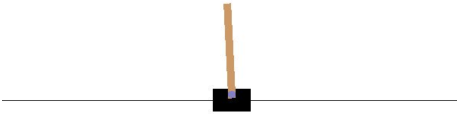
1. Circle: reward for running in a circle, constraint: stay in a smaller circle
2. Gather: collecting green apples, avoiding red bombs
3. Point, Ant, Humanoid from MuJoCo



(a) Humanoid-Circle (b) Point-Gather

Figure 2. The Humanoid-Circle and Point-Gather environments. In Humanoid-Circle, the safe area is between the blue panels.

I use CartPole, InvertedPendulum, InvertedDoublePendulum because it is faster to train.



Planned to switch to more complex environments

Benchmarks

No unified set of environments, everybody codes their own. Need an open-source extension for Gym?

RL: OpenAI Gym

Safe RL: **?**

Benchmarks

No unified set of environments, everybody codes their own. Need an open-source extension for Gym?

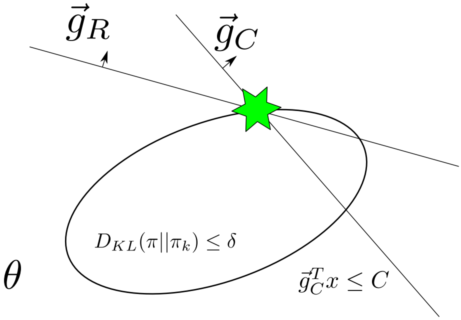
RL: OpenAI Gym

Safe RL: **?**

Have modular code for safe CartPole right now

Constrained Policy Optimization (6)

Have estimated policy gradient for reward g_r and for constraint g_c. Have estimated constraint return J_C
Trivial solution: $g_r^T(\theta-\theta_0) \to \max$ s.t. $g_c^T(\theta-\theta_0)+J_C \leq C_{\max}$, $\theta^T\theta \leq \varepsilon$.
Using KL-divergence gradient instead: $g_r^T(\theta-\theta_0) \to \max$ s.t. $g_c^T(\theta-\theta_0)+J_C \leq C_{\max}$, $\frac{\partial^2}{\partial\theta^2}D_{KL}(\pi(\theta)||\pi(\theta_k)) \leq \varepsilon$.



CPO

1. Has a theoretical guarantee of the form "if δ is low enough, and second (third) order terms are small, the algorithm finds an improvement"
2. Dual problem is low-dimensional, but still quadratic. Explicit solution is quite cumbersome
3. Fallback option (following natural gradient of the constraint to decrease it)
4. Existing implementation in RLLab, own implementation

Lyapunov-based methods (7)

Using Lyapunov functions: $T_\pi[L](x) \leq L(x)$.

1. Have a safe policy π_k as a network
2. Estimating Q_R (reward), Q_C (cost) and Q_T (discounted stopping time) as networks via Bellman updates
3. Constructing a Lyapunov function via $Q_L = Q_C + \varepsilon Q_T$ with $\varepsilon = C_{\max} - \frac{\pi_k^T Q_D}{\pi_k^T Q_T}$
4. At each step, solving for $\pi^T Q_R \to \max$ s.t. $(\pi - \pi_k)^T Q_L \leq \varepsilon$ (linear program)
5. Making a supervised step $D_{JSD}(\pi|\pi_k) \to \min^\dagger$

---

†Jensen-Shannon Divergence: $D_{JSD}(p, q) = \frac{1}{2}(D_{KL}(p||r) + D_{KL}(q||r))$ for $r = \frac{1}{2}(p + q)$

Lyapunov-based methods (7) II

1. No implementation with the paper, own implementation
2. After each rollout, need to call TF twice: once to get Q_R, Q_L and then to do the JSD step
3. Chicken-and-egg problem: to train good π, need good Q and vice versa.
4. I train first Qs with greedy action-selection and then switch on π training
5. The paper does not describe how to deal with this
6. Very unstable, unlearns everything when switching to the next phase.
7. Approximation to exact problem, so no guarantee for this version
8. In case of failure, only doing Bellman update

Common elements

Boiling down to constrained optimization problem after some approximation (1st or 2nd order)

CPO and L. have in common:

1. Some step for reward maximization
2. Some first-order hard constraint for cost
3. Some δ to stay close (implicit in PPO)

Other methods:

1. Lagrangian method: simply combining $R - \lambda C$ with a learnable λ. Problem: unstable
2. TRPO

Proposal: Projected PPO

1. Existing algorithms are quite complex
2. PPO solves the issue of closeness by not encouraging large deviations
3. $\Rightarrow$ Making PPO safe makes sense

Proposed method (PPPO):

1. Estimate A for current policy π, constraint gradient g_C and return J_C
2. Optimize $L_{PPO} \to \min\limits_{\theta}$ s.t. $g_C^T(\theta - \theta_k) + J_C \leq C_{\max}$ using Projected Gradient Descent[‡]
3. Fallback option: policy gradient for constraint in case if current is not safe

Advantage: easier to implement than CPO and Lyapunov, no inner optimization

[‡]Projection on a half-plane is easy $\theta' = \theta - \frac{g}{g^T g}(g^T x - c)$

Experimental results

1. Considering the problem solved if constraint was violated $< 1\%$ of the training time and a reward of at least 175 was achieved.
2. Agents are compared by mean over repetitions and max over training reward for which cost was satisfactory < 100
3. Lyapunov did not converge
4. CPO should show better results, a problem might be in my implementation

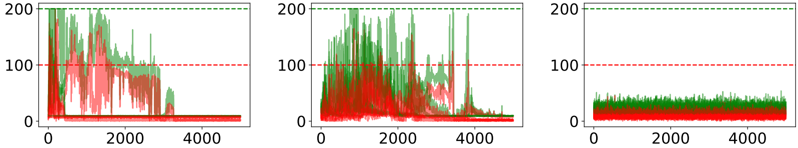


Figure: Agents: CPO, PPPO, Random on Cartpole-v0, best hyperparameters. 5 repetitions of a single experiment are shown on the same plot.

Conclusion & Future directions

1. Safe continuous-state RL in CMDPs
2. Proposal to standartize benchmarks
3. Re-implementation of existing algorithms and comparison
4. Projected PPO proposal and evaluation on toy experiments

Next:

1. Finalizing the safe environment list and publishing it.
2. Theoretical guarantees for PPPO
3. Releasing the code for Lyapunov safe RL (so far they do not provide it).
4. Testing new PPO in more demanding environments

References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[2] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[3] T. Everitt, G. Lea, and M. Hutter, "Agi safety literature review," *arXiv preprint arXiv:1805.01109*, 2018.

[4] E. Altman, *Constrained Markov decision processes*, vol. 7. CRC Press, 1999.

[5] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in neural information processing systems*, pp. 908–918, 2017.

[6] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained Policy Optimization," 2017.

[7] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based Approach to Safe Reinforcement Learning," 2018.