Constrained Policy Optimization via Bayesian World Models

Vincent Meilinger

Technische Universität Berlin

November 17, 2022

Contents

Preliminaries

Preliminaries

Markov decision processes

- states $s_t \in \mathbb{R}^n$ with initial state distribution $s_0 \sim \rho(s_0)$
- actions $a_t \in \mathbb{R}^m$ sampled from policy distribution $\pi(\cdot|s_t)$
- transition distribution $s_{t+1} \sim p(\cdot|s_t, a_t)$ unknown
- given state r_t, agent observes reward generated by $r_t \sim p(\cdot|s_t, a_t)$
- the performance of a policy π and dynamics p is defined as follows:

$$J(\pi, p) = \mathbb{E}_{a_t \sim \pi,\ s_{t+1} \sim p,\ s_0 \sim \rho} \left[\sum_{t=0}^{T} r_t | s_0 \right]$$

Preliminaries

Constrained Markov decision processes

- in addition to the reward, the agent observes costs $c_t^i \sim p(\cdot|s_t, a_t)$ (i: distinct unsafe behaviours to avoid)
- the constraints of a policy π and dynamics p are defined as follows:

$$J^i(\pi, p) = \mathbb{E}_{a_t \sim \pi,\ s_{t+1} \sim p,\ s_0 \sim \rho} \left[\sum_{t=0}^{T} c_t^i | s_0 \right] \leq d^i, \forall i \in \{1, \ldots, C\}$$

where d^i are human-defined thresholds

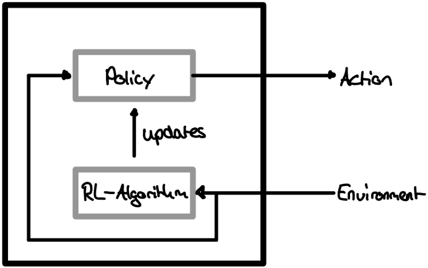
Preliminaries

Model based reinforcement learning

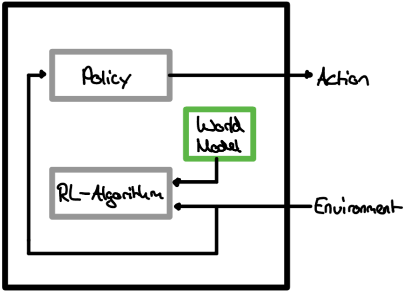
- Bayesian approaches to model based RL quantify uncertainty in model estimations
- agent stores observed transitions $\{s_{t+1}, s_t, a_t\}$ in data set $\mathcal{D}$
- the data set is fit to a statistical model $p(s_{t+1}|s_t, a_t, \theta)$
- the statistical model is used for planning future transitions (online MPC/offline policy optimization)
- model based reinforcement learning is *sample efficient*!

Preliminaries

Model based reinforcement learning



Model-Free Agent



Model-Based Agent

LAMBDA

LAMBDA

Experiment Setup

- multiple scenarios with random seeds
- world can be modeled by CMDP
- partially observable 3D-world, agent receives observation $o_t \sim p(\cdot|s_t)$ instead of state s_t
- unknown cost function & reward function
- one safety-constraint

LAMBDA

Langrangian Model-Based Agent (LAMBDA)

- Bayesian model based policy optimization approach to solve general constrained MDPs
  $\rightarrow$ allows for cheap generation of synthetic sequences of experience
  $\rightarrow$ probabilistic world model allows guided exploration
- unknown cost function & reward function are modeled by statistical models, similar to the transition distribution
- constrained optimization using Augmented Lagrangian, optimization based on optimistic and pessimistic bounds

$$\max_{\pi \in \Pi} \min_{\lambda \geq 0} \left[J(\pi) - \sum_{i=1}^{C} \lambda^i (J^i(\pi) - d^i) \right]$$

- LAMBDA can learn policies directly from observations (end-to-end, without prior knowledge)

LAMBDA

Bayesian world model

- *Recurrent State Space Model* used to infer transition density from observations
- it is used to:
  - generate trajectories
  - estimate an optimistic bound for the task objective
  - estimate pessimistic bounds for the constraints
- models the predictive distribution as a differentiable function $\rightarrow$ allows to perform constrained policy optimization by backpropagating gradients through the model

LAMBDA

Optimism/Pessimism

- greedy maximization of the predictive posterior distribution not always the best approach $\rightarrow$ concept of optimism and pessimism
- *optimism* describes "will" to explore $\rightarrow$ can lead to dangerous behaviours!
- *pessimism* is used to enforce safety constraints $\rightarrow$ can lead the agent to not explore enough
- results in constrained problem:

$$\max_{\pi \in \Pi} \max_{p_\theta \in \mathcal{P}} J(\pi, p_\theta)$$

s.t. $\max_{p_{\theta^i} \in \mathcal{P}} J^i(\pi, p_{\theta^i}) \leq d^i, \ \forall i \in \{1, \ldots, C\}$.

(J: obj. function, p_θ: predictive density, π: policy, d: constraint threshold)

LAMBDA

Estimating upper bounds

Simulate trajectories given posterior sample θ_j, to estimate $J(J(\pi, p_{\theta_j})$ and $J^i(\pi, p_{\theta_j})$. Choose the largest estimate for each objective. (N: #realizations)

Algorithm 1 Upper confidence bounds estimation via posterior sampling

```
Require: N, p(θ|D), p(s_{τ:τ+H} | s_{τ−1}, a_{τ−1:τ+H−1}, θ), s_{τ−1}, π(a_t, |s_t).
Initialize V = {}                                # Set of objective estimates, under different posterior samples.
for j = 1 to N do
    θ ∼ p(θ|D).                                   # Posterior sampling (e.g., via SWAG).
    s_{τ:τ+H} ∼ p(s_{τ:τ+H} | s_{τ−1}, a_{τ−1:τ+H−1}, θ) .   # Sequence sampling, see Appendix I.
    Append V ← V ∪ Σ_{t=τ}^{τ+H} V_λ(s_t).
end for
return max V.
```

LAMBDA

Algorithm

Algorithm 2 LAMBDA

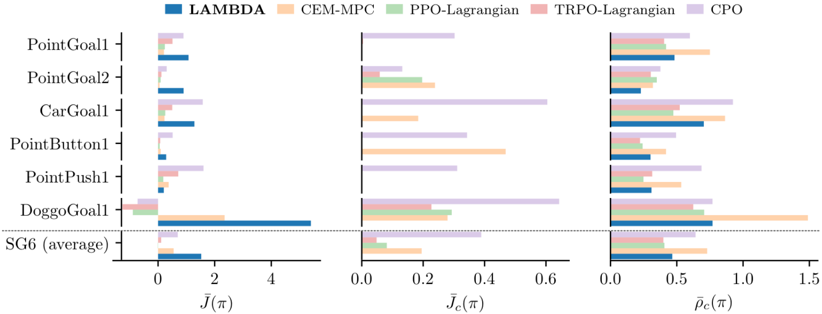
Initialize $\mathcal{D}$ by following a random policy or from an offline dataset.
while not converged **do**
  for $u = 1$ to U update steps **do**
    Sample B sequences $\{(\boldsymbol{a}_{\tau'-1:\tau'+L-1}, \boldsymbol{o}_{\tau':\tau'+L}, r_{\tau':\tau'+L}, c^i_{\tau':\tau'+L})\} \sim \mathcal{D}$ uniformly.
    Update model parameters θ and ϕ. # E.g., see Hafner et al. (2019a) for the RSSM.
    Infer $\boldsymbol{s}_{\tau':\tau'+L} \sim q_\phi(\cdot|\boldsymbol{o}_{\tau:\tau+L}, \boldsymbol{a}_{\tau'-1:\tau'+L-1})$.
    Compute $\sum_{t=\tau}^{\tau+H} \mathrm{V}_\lambda(\boldsymbol{s}_t), \sum_{t=\tau}^{\tau+H} \mathrm{V}^i_\lambda(\boldsymbol{s}_t)$ via Algorithm 1. Use each state in $\boldsymbol{s}_{\tau':\tau'+L}$ as an initial state for sequence generation.
    Update ψ and ψ^i via Equation (9) with $\sum_{t=\tau}^{\tau+H} \mathrm{V}_\lambda(\boldsymbol{s}_t)$ and $\sum_{t=\tau}^{\tau+H} \mathrm{V}^i_\lambda(\boldsymbol{s}_t)$.
    Update ξ according to Equation (10) with $\sum_{t=\tau}^{\tau+H} \mathrm{V}_\lambda(\boldsymbol{s}_t)$ and $\sum_{t=\tau}^{\tau+H} \mathrm{V}^i_\lambda(\boldsymbol{s}_t)$.
    Update λ^i via Equations (6) and (11).
  end for
  for $t = 1$ to T **do**
    Infer $\boldsymbol{s}_t \sim q_\phi(\cdot|\boldsymbol{o}_t, \boldsymbol{a}_{t-1}, \boldsymbol{s}_{t-1})$.
    Sample $\boldsymbol{a}_t \sim \pi_\xi(\cdot|\boldsymbol{s}_t)$.
    Take action $\boldsymbol{a}_t$, observe $r_t, c^i_t, \boldsymbol{o}_{t+1}$ received from the environment.
  end for
  Update dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{o}_{1:T}, \boldsymbol{a}_{1:T}, r_{1:T}, c^i_{1:T}\}$.
end while

Experiment

Experiment

SG6 Benchmark

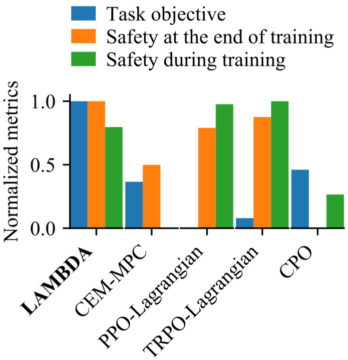
LAMBDA performance in different scenarios against various other methods:



($\hat{J}(\pi)$: undiscounted episodic return for E episodes, $\hat{J}_c(\pi)$: undiscounted episodic cost return for E episodes, $p_c(\pi)$: normalized sum of costs during training)

Experiment

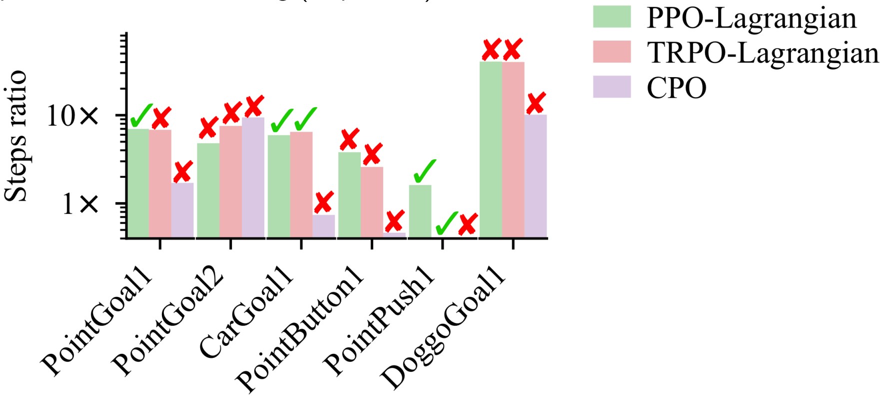
Safety & Performance
LAMBDA's ability to trade-off average performance vs. safety metrics (average across all SG6 tasks):



Experiment

Sample Efficiency

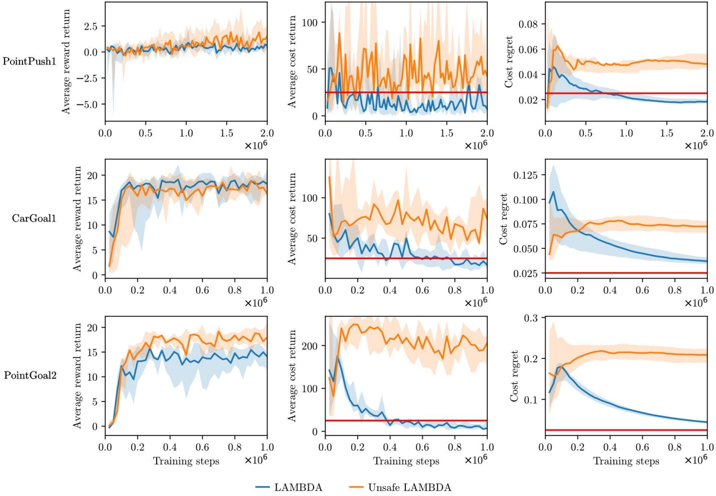
Average number of steps required by model-free methods to match LAMBDA's performance after training (steps ratio).



Experiment

Unsafe LAMBDA

LAMBDA achieves similar performance to ”unsafe” LAMBDA:



Summary

Summary

- LAMBDA: Bayesian model-based policy optimization algorithm
- generates world model to generate trajectories, estimates optimistic task bound and pessimistic constraint bounds
- policy search using Augmented Lagrangian method to solve the optimization problem
- performs equally good or better than model-free competitors
- end-to-end

Prospect

Prospect

- current approach does not incorporate prior knowledge/assumptions about the environment
- authors express the potential to learn a policy without ever violating any constraints!
- comparison of LAMBDA vs. other model-based approaches
- experiments incorporating multiple constraints

Thank you for listening!

Sources:

1 Yarden As, Ilnura Usmanova, Sebastian Curi and Andreas Krause. Constrained Policy Optimization via Bayesian World Models. ICLR 2022

2 Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565, 2019a