Constrained Model-based Reinforcement Learning via Robust Planning

Zuxin Liu [1] **Zijian Guo** [1] **Ding Zhao** [1]

Abstract

This paper studies the safe reinforcement learning (RL) problem with sparse indicator signals for constraint violations. We propose a model-based approach to enable RL agents to effectively explore the environment with unknown system dynamics and environment constraints given a significantly small number of violation budgets. We employ the neural network ensemble model to estimate the prediction uncertainty and use model predictive control as the basic control framework. We propose the robust cross-entropy method to optimize the control sequence considering the model uncertainty over the constraints. We evaluate our methods in the Safety Gym environment. The results show that our approach learns to complete the tasks with better safety performance than baselines and achieves several orders of magnitude better sample efficiency when compared with constrained model-free RL approaches.

1. Introduction

Reinforcement learning (RL) has achieved great success in a wide range of applications. By setting a high-level reward function, an RL agent is able to learn a policy to maximize the reward signal received from the environment through trial and error. However, in the course of learning, it is usually hard to prevent the agent from getting into high-risk states which may lead to catastrophic results, especially for safety-critical applications. For example, if an RL algorithm is deployed on a real robot arm, it might hit fragile objects and surrounding people, which may break valuable property or cause injury. Therefore, it is important to develop constrained or safe reinforcement learning algorithms for real-world applications, which allow them to complete tasks while satisfying certain safety constraints.

Though some research has proposed to achieve constrained reinforcement learning with prior knowledge (Dalal et al., 2018; Koller et al., 2018), the assumptions may not hold for all applications. Therefore, we consider the most general settings of constrained RL — how can we enforce safety constraints for an RL agent without the knowledge of the system dynamics and an explicit expression of the constraint function? One example problem that falls under this category is the `Goal` task setting in the Safety Gym simulation environment (Ray et al., 2019), where a robot needs to navigate to the goal while avoiding all of the hazard areas. The dynamics model of the environment is unknown, and the robot only receives indicator signals when violating constraints. The observations of the robot are sensory data so it is hard to analytically express the mapping from observation space to the constraint violation.

The challenges of solving the above problem are threefold: First, pure model-free, constrained RL algorithms (such as Lagrangian-based methods (Stooke et al., 2020) and projection-based optimization methods (Achiam et al., 2017)) are not sample efficient. They need to constantly violate safety constraints while collecting a large number of unsafe data to learn the policy, and the final policy can hardly guarantee constraint satisfaction, which restrict the application in safety-critical environments. Second, the task objective and the safety objective of an RL agent may contradict each other, which may corrupt the policy optimization procedure for methods that simply transform the original reward optimization criteria to the combination of reward and constraint violation cost (such as risk-sensitive or uncertainty-aware methods (Geibel & Wysotzki, 2005; Gaskett, 2003)). As Fig. 1 (a) shows, the constraint violation signals give an opposite direction of the reward signal, which could cause oscillation behavior of the robot close to the dangerous flame area. Finally, the black-box constraint function and unknown environment dynamics model make the problem hard to optimize, especially for tasks with a high-dimensional observation space (Ray et al., 2019). Most existing model-based constrained RL approaches either assume a known prior dynamics model of the system or assume a known structure of the constraint function (which could be expressed by an analytical formula or a finite number of unsafe sets (Berkenkamp et al., 2017; Koller et al., 2018; Pham et al., 2018)).

[1]Carnegie Mellon University. Correspondence to: Zuxin Liu <zuxinl@andrew.cmu.edu>, Ding Zhao <dingzhao@cmu.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

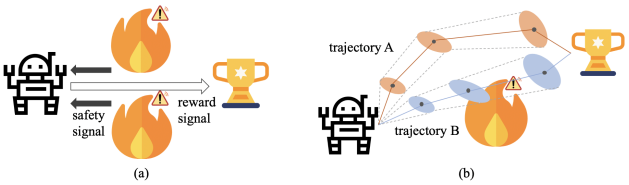


Figure 1. (a): The reward signal and safety signal may contradict to each other; (b): The trajectory sampling method for uncertainty-aware dynamics models.

The contributions of this paper are twofold: 1) We present a simple yet powerful constrained model-based reinforcement learning algorithm with continuous state and action spaces that can achieve near-optimal task performance with near-zero constraint violation rates. We formulate the problem under the constrained Markov Decision Processes framework and without additional assumptions regarding the system dynamics and constraint functions, which should be both learned from collected data with limited unsafe samples and sparse constraint violation indicator signals. 2) We propose a robust cross-entropy (RCE) optimization method that works with an uncertainty-aware dynamics model to deal with the dynamics prediction error that may lead to unsafe behaviors. We show that our RCE method is better than popular model-free and model-based baselines in terms of both sample efficiency and performance.

2. Related Work

Constrained reinforcement learning aims to learn policies that maximize the expected task reward while satisfying safety constraints (Garcıa & Fernández, 2015). One popular method to solve constrained RL problems is to transform the single reward optimization criteria to a combination of reward and constraint violation signals, such as using the notion of risk or uncertainty as one of the optimization loss terms (Gaskett, 2003). However, for some applications, it is better to separate the safety and performance specifications rather than combine them into a value and then optimize, because the reward signal and safety signal may conflict with each other, which could cause unstable performance (Ray et al., 2019) as we show in Fig. 1 (a). Furthermore, balancing the objective function between the performance metric and the safety metric is a difficult and domain-specific task (Garcıa & Fernández, 2015).

Recently, constrained optimization algorithms have attracted much attention. Achiam et al. (Achiam et al., 2017) proposed the Constrained Policy Optimization (CPO) algorithm based on the trust region method, which can be applied to high-dimensional tasks. However, the errors of gradient and Hessian matrix estimation may lead to poor performance on constraint satisfaction in practice (Wen & Topcu, 2018).

On the other hand, Lagrangian-based methods aim to transform the original constrained optimization problem to an unconstrained form by adding the Lagrangian multiplier, which achieves relatively better performance than CPO in a recent empirical comparison in the Safety Gym environment (Ray et al., 2019). The Lagrangian multiplier can be regarded as a dynamic weight coefficient that balances the weight between the performance and safety metrics, and can be optimized via gradient descend together with the policy parameters. Nevertheless, a target constraint violation rate must be set in advance, which is not flexible to transfer a trained policy to different tasks. We use CPO and a Lagrangian-based method as part of our baselines.

To achieve safety constraint satisfaction, several model-based approaches have been proposed. Pham et al. (Pham et al., 2018) and Dalal et al. (Dalal et al., 2018) combined unconstrained model-free methods with model-based safety checks to guarantee constraint satisfaction for the output. Similar action projection ideas are also used in some Lyapunov function-based methods (Chow et al., 2019). To guarantee safe exploration of the environment, Gaussian Processes (GPs) are usually used to model the dynamics because of their ability to estimate uncertainty (Berkenkamp et al., 2017; Koller et al., 2018). However, these methods either assume prior knowledge of the environment such as a prior dynamics model, or require a known constraint function structure that is analytically expressed or defined by a set of states. Furthermore, although GP-based approaches perform well in low-dimensional simple tasks, they do not scale well as the data dimension and amount increases, and struggle to represent complicated and discontinuous dynamics models (Chua et al., 2018).

3. Preliminaries

3.1. Constrained Markov Decision Process

We investigate the constrained RL problem in the constrained Markov decision process (CMDP) framework, which is defined by a tuple $(\mathcal{S}, \mathcal{A}, f, r, c, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $f : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ is a deterministic state transition function, $r : \mathcal{S} \mapsto \mathbb{R}$ is the reward function, and $c : \mathcal{S} \mapsto \{0, 1\}$ is an indicator cost function, where 0 means safe and 1 represents constraint violation, and $\gamma \mapsto [0, 1]$ is the discount factor.

We assume the dynamics f and the cost function c are both unknown, and should be learned from data. The policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ is a mapping from the state space to the action space. Let $J_r(\pi)$ denote the expected return of policy π w.r.t the reward function r and $J_c(\pi)$ denote the expected return of policy π w.r.t the cost function c. We have $J_r(\pi) = \mathbb{E}_{\mathcal{T}\sim\pi}[\sum_{t=0}^{T} r(s_{t+1})]$, $J_c(\pi) = \mathbb{E}_{\mathcal{T}\sim\pi}[\sum_{t=0}^{T} c(s_{t+1})]$, where T is the time horizon, and

$\mathcal{T} = \{s_0, a_0, s_1, a_1, ...\}$ is the trajectory collected by π.

Some model-free constrained RL methods, such as Lagrangian-based methods (Stooke et al., 2020), aim to maximize the cumulative reward while limiting the cost incurred from constraint violations to a target constraint violation value $d \in (0, +\infty)$. The problem can then be expressed as

$$\pi^* = \arg\max_{\pi} J_r(\pi), \quad s.t. \quad J_c(\pi^*) \leq d$$

where π^* is the optimal policy. Setting $d = 0$ represents perfect constraint satisfaction, which is usually desired in many safety-critical applications.

3.2. Cross-Entropy Method for Optimization

The cross-entropy method (CEM) is a sampling-based stochastic optimization approach, which has been used in a series of reinforcement learning problems recently (Chua et al., 2018). In CEM, we assume the n dimensional solution $\mathcal{X} \in \mathbb{R}^n$ is sampled from a distribution that is parameterized by Θ. The distribution is assumed to be a n-dimensional factorized multivariate Gaussian, which is one of the most common choices in the RL literature. Then we have $\mathcal{X} \sim \mathcal{N}(\Theta)$, where $\Theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$. $\boldsymbol{\mu}$ is an n dimensional vector, and $\boldsymbol{\Sigma}$ is an n dimensional diagonal covariance matrix. The basic idea is to sample solutions iteratively from a distribution that is close to previous samples which have resulted in high rewards. The iteration's stopping criterion is often determined by a predefined maximum iteration number and a threshold on the covariance. For more details on CEM methods and their applications, refer to (Botev et al., 2013).

4. Approach

4.1. Model Learning

As we introduced in section 3.1, the dynamics model (deterministic state transition function) $f(s_t, a_t)$ and the cost model (constraint violation indicator function) $c(s_{t+1})$ are both unknown. We need to infer them from collected data. For model-based RL, the choice of dynamics model is crucial, as even a small prediction error may influence the performance of the controller significantly (Chua et al., 2018). Therefore, using an uncertainty-aware dynamics prediction model is necessary, especially in safety-critical scenarios.

As a particular instance of this paper, we adopt a neural network ensemble model to learn the dynamics and estimate the epistemic uncertainty (subjective uncertainty due to a lack of data) of the input data, which is similar to the ensemble model that was proposed by Chua et. al (Chua et al., 2018). We choose the ensemble model because of its scalability, implementation simplicity, and reasonable uncertainty estimation in complex environments. It could be replaced by any other uncertainty-aware prediction models in our framework.

Denote B as the number of ensemble models. Denote $\tilde{f}_{\theta_b}$ as the b-th neural network ($b \in \{1, 2, ..., B\}$) parameterized by θ_b. Given state s_t, action a_t, next state s_{t+1} tuples of data $\mathcal{D}$, where t represents the time, we train each neural network by minimizing the mean square error (MSE) loss as $\text{loss}(\theta) = \mathbb{E}_{(s_t, a_t, s_{t+1}) \in \mathcal{D}_b} \left[||s_{t+1} - \tilde{f}_{\theta_b}(s_t, a_t)|| \right]$, where $\mathcal{D}_b$ is a subset of the whole data $\mathcal{D}$ to prevent each model from overfitting. After we train all base models, we define the predictive distribution as a multivariate Gaussian with mean $\tilde{\mu} = \frac{1}{B} \sum_{b=1}^{B} \tilde{f}_{\theta_b}$ and variance $\tilde{\Sigma} = \frac{\sum_{b=1}^{B} (\tilde{f}_{\theta_b} - \tilde{\mu})^2}{B}$, where $\tilde{\Sigma}$ can be regarded as the epistemic uncertainty estimation.

Since the unknown cost model $c(s_{t+1})$ is an indicator function of constraint violations, any classification model may be used to approximate it. However, the unsafe data that violate safety constraints may only make up a small portion of the collected data, which induces an imbalanced data classification problem (Sun et al., 2009; Wang et al., 2019). For single-neural-network-based classification models, the results could be biased towards safe data, meaning the model will still achieve high prediction accuracy overall, even if the model falsely determines all of the input data to be safe. Therefore, in light of robustness towards imbalanced data, as well as low computational burden, we adopt a state-of-the-art gradient boosting decision tree-based ensemble method - LightGBM (Ke et al., 2017) - as a classifier to approximate the indicator cost function. In addition, we separate the entirety of our data into two buffers - one for safe data and another for unsafe data - in order to control the maximum ratio of safe data to unsafe data used for training. The data management tricks can reduce the bias towards safe data as much as possible.

4.2. Model Predictive Control with Learned Dynamics and Cost Model

We use Model Predictive Control (MPC) as the basic control framework for our constrained model-based RL approach (Okada & Taniguchi, 2020; Drews et al., 2017). The objective of MPC is to maximize the accumulated reward w.r.t a sequence of actions $\mathcal{X} = (a_0, ..., a_T)$, where T is the planning horizon. After the first action is applied to the system, new observations are received, and the same optimization is performed again. In our CMDP setting, additional constraints are introduced so that the original objective becomes a constrained optimization problem. Denote s_t as the observation at time t. We aim to solve the following problem:

$$\mathcal{X} = \arg\max_{a_0,...,a_T} \quad \mathbb{E}\big[\sum_{t=0}^{T} \gamma^t r(s_{t+1})\big]$$
$$s.t. \quad s_{t+1} = f(s_t, a_t), c(s_{t+1}) = 0, \quad \forall t \in \{0, 1, ..., T-1\} \tag{1}$$

where γ is the discount factor, $r(s_{t+1})$ is the reward function, $f(s_t, a_t)$ is the dynamics model, and $c(s_{t+1}) \in \{0, 1\}$ is the indicator cost function. Both f and c are learned from data and can be viewed as black-box functions.

4.3. Robust Cross-Entropy Method for Planning

Algorithm 1 Robust Cross-Entropy Method for RL

Input: Initial distribution parameter Θ ; number of samples N; number of elites k; initial state s_0
Output: Sample $\mathcal{X}^*$ with the highest reward

while The stop criteria is not satisfied **do**
  Draw N samples from the initial distribution: $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_N \sim \mathcal{N}(\Theta)$
  Evaluate each sample $\mathcal{X}_i$ by Eq.2 to get the estimation of reward $r(\mathcal{X}_i; s_0)$ and cost $c(\mathcal{X}_i; s_0)$
  Select the feasible set $\Omega \in \{\mathcal{X}_i\}_{i=1}^N$ based on the cost estimation
  if Ω is empty **then**
    Sort $\{\mathcal{X}_i\}_{i=1}^N$ in ascending order w.r.t the cost. Let Λ_k be the first k elements
  else
    Sort Ω in descending order w.r.t the reward.
    Let Λ_k be the first k elements of Ω if $|\Omega| > k$, otherwise let Λ_k be Ω
  end if
  Update Θ by maximizing the likelihood given Λ_k: $\Theta \leftarrow \arg\max_\theta \prod_{\mathcal{X} \in \Lambda_k} p(\mathcal{X}; \theta)$
end while
return: $\mathcal{X}^*$ with highest reward in Λ_k

To directly solve the constrained optimization problem in Eq. 1, we propose the robust cross-entropy method (RCE) by using the trajectory sampling (TS) technique (Chua et al., 2018) to estimate reward and constraint violation cost. We define the solution $\mathcal{X} = (a_0, a_1, ..., a_{T-1})$ to as an action sequence with length of planning horizon T. Given the initial state s_0, the learned dynamics model $\tilde{f}_\theta$, the learned indicator constraint function $c(s) \in \{0, 1\}$, we can evaluate the accumulated reward and cost of the solution by:

$$r(\mathcal{X}; s_0) = \sum_{t=0}^{T} \gamma^t \big(\frac{1}{B} \sum_{b=1}^{B} r(s_{t+1}^b)\big),$$
$$c(\mathcal{X}; s_0) = \sum_{t=0}^{T} \beta^t \max_b c(s_{t+1}^b) \tag{2}$$

where $s_{t+1}^b = \tilde{f}_{\theta_b}(s_t^b, a_t), \forall t \in \{0, ..., T-1\}, \forall b \in \{1, ..., B\}$, γ and β are discounting factors, and B is the ensemble size of the dynamics model. The reward $r(s)$ could either be predefined or learned together with the dynamics model from data (as an additional dimension of the state). The intuition behind the TS estimation method is shown in Fig. 1 (b), where the dots on the blue line and the dots on the orange line represent two real trajectories, and the ellipses represent the uncertainty of the dynamics model prediction based on the initial observation and action sequences. From the figure, we can see that the reward for trajectory B should be higher than trajectory A because choosing B will result in the goal being reached faster. However, trajectory A is preferred because a robot following trajectory B may pass through the flames and violate the safety constraint. Without TS, trajectory B could potentially be predicted as a safe route because of the dynamics model prediction error. With TS, the uncertainty estimate of our dynamics model has a slight chance to cover the unsafe area, so the trajectory B will be classified as unsafe. Because TS estimates the cost of a trajectory with the worst-case scenario among all sampled routes, it is more robust when the dynamics model prediction is not highly accurate.

Denote the reward function as $r(\mathcal{X}) : \mathcal{S} \mapsto \mathbb{R}$, and the probability density function as $p(\mathcal{X}; \Theta)$. The RCE algorithm is shown in Algorithm 1. We first select the feasible set of solutions that satisfy the constraints based on the estimated cost in Eq. 2. Then, we sort the solutions in the feasible set and select the top k samples to use when calculating the parameters of the sampling distribution for the next iteration. If all the samples violate at least one constraint, we select the top k samples with the lowest costs.

Algorithm 2 MPC with RCE

Input: Initial collected data $\mathcal{D}$; RCE parameters $\mathcal{P}$

while The performance is not converged **do**
  Train the dynamics $\tilde{f}$ and cost model $\tilde{c}$ given $\mathcal{D}$
  for Time $t = 0$ to EpisodeLength **do**
    Observe state s_t from the environment
    Optimize actions by Alg. 1: $\{a_i^*\}_{i=t}^{t+T} \leftarrow \text{RCE}(\mathcal{P}, s_t)$
    Apply the first action a_t^* in $\{a_i^*\}_{i=t}^{t+T}$ to the system
    Observe next state s_{t+1} and cost signal $c(s_{t+1})$
    Update data buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, a_t, s_{t+1}, c(s_{t+1})\}$
  end for
end while

A similar idea is adopted in (Wen & Topcu, 2018). Our approach differs from theirs in two aspects. First, we consider the worst-case cost and aim to minimize the maximum cost in order to select the feasible set while they calculate the ex-

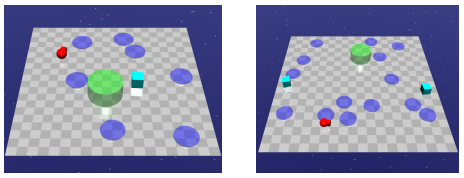
pectation. Second, their primary application is to optimize the policy parameters for model-free RL, while we directly optimize the action sequence within the planning horizon in the model-based RL setting. The entire training pipeline of our MPC with RCE is presented in Algorithm 2.

5. Experiments

In this section, we aim to answer two questions: 1) how sample efficient of our method than popular primal-dual constrained optimization approaches? 2) what is the role of RCE in the MPC framework? Since RL agents learn by trial and error, and we assume no prior knowledge of the environment, it is inevitable to violate the constraints in the early stage of training. However, our goal is to reduce the unsafe samples as much as possible and increasing the efficiency of using the unsafe data because collecting unsafe data could be expensive in some cases.

5.1. Simulation Environment

We evaluate our safe RL approach in the OpenAI Safety Gym environment (Ray et al., 2019), which is shown in Fig. 2. Each experiment setting involves a robot (red object in Fig. 2) that must navigate a clustered environment to accomplish a task while avoiding contact with obstacles. When the robot enters the goal circle (green circle), the goal location is randomly reset. A bonus of $r_t = 1$ is given to the robot for reaching the goal. Hazards (blue circles) are dangerous areas to avoid. Vases (teal cube) are objects initialized to be stationary but movable upon touching. The agent is penalized for entering Hazards or touching Vases. If the agent violates the safety constraint at time step t, it will receive a cost $c(s_t) = 1$, otherwise the cost is 0. We have two types of robot – Point and Car, and two levels – level 2 is more difficult than level 1 task since more constraints are presented. We name the 4 tasks as `Point-Goal1`, `Point-Goal2`, `Car-Goal1` and `Car-Goal2`.



(a) Point Goal1 Env (b) Point Goal2 Env

(c) Car Goal1 Env (d) Car Goal2 Env

Figure 2. Experiment Environments

5.2. Baselines and Experiment Setting

Model-free baselines. Inspired by the official benchmark of Safety Gym, we use 4 Lagrangian-based approaches – `TRPO-Lag`, `PPO-Lag`, `SAC-Lag`, `DDPG-Lag` – and a constrained optimization approach `CPO` (Achiam et al., 2017) as the model-free constrained RL baselines. The Lagrangian baselines are augmented based on TRPO (Schulman et al., 2015), PPO (Schulman et al., 2017), SAC (Haarnoja et al., 2018), DDPG (Lillicrap et al., 2015), which we have introduced in section 2. Note that SAC and DDPG are off-policy methods, so they should be more sample efficient. We also use two unconstrained RL methods – TRPO and PPO – to show the performance and constraint violations when we only optimize for the task reward.

Model-based baselines A simple way to solve Eq. 1 is to add large penalties to the objective function for constraint violations. So we extend two popular model-based methods to solve the safe RL problem: CEM and random shooting, which have successfully been applied to many model-based RL tasks (Nagabandi et al., 2018; Chua et al., 2018). The two model-based baseline methods will adopt the same trajectory sampling technique and the same models as we used in RCE. The only difference is the optimization procedure. We name the two methods as `MPC-random` and `MPC-CEM`.

Metrics and comparison. We follow the metrics and comparison method in the Safety Gym paper (Ray et al., 2019). We compare different approaches in terms of episodic accumulated reward and episodic cost, which is defined as the total constraint violation number in each episode. Method A is better than B if A's episodic cost is lower than B's. If their costs are similar, then the one with higher episodic reward is better. We also compare the sample efficiency of exploiting constraint violations and the cost after convergence.

Training. We use the same hyper-parameters for the model-based methods (MPC-RCE, MPC-CEM, and MPC-random) and the same hyper-parameters for the model-free baselines provided by the Safety Gym official benchmark (Ray et al., 2019). For the detail about hyper-parameters used in our experiments, please refer to the appendix 6 and code.

5.3. Comparison of Model-free Baselines

Fig. 3 shows the experiment results in the four tasks, where the first two rows are the learning curves of reward and constraint violation cost respectively, and the last row is the maximum reward versus cumulative cost plot that characterize the efficiency of exploiting costs. The dashed line in the second row is the target constraint violation threshold. The figures help us to address the first question – how sample efficient of our method? **Note that the horizontal axis (total**

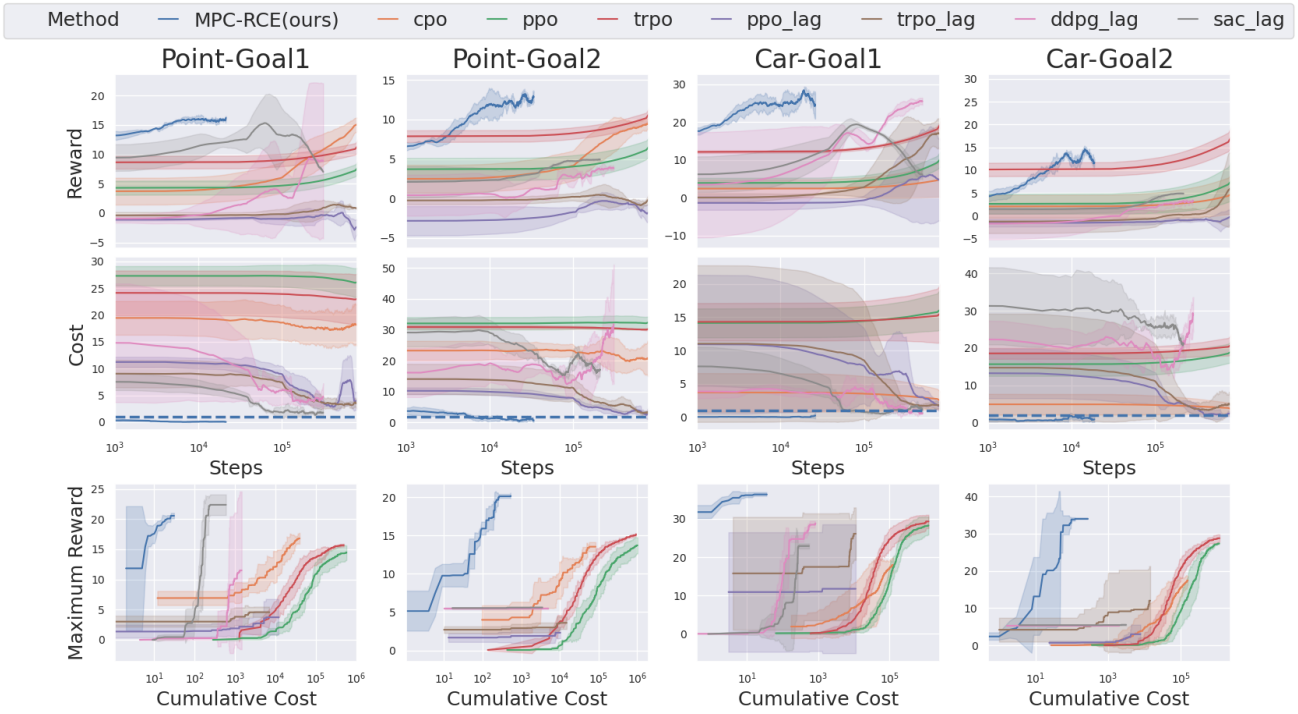


Figure 3. Learning curves in the logarithmic scale of steps. Each column is a task. The first row figures are the reward trends, the second row figures are the cost trends, and the last row figures are the maximum reward versus the cumulative cost. All plots are averaged among 3 random seeds. The solid line is the mean value, and the light shade represents the area within one standard deviation.

interaction steps) of all the plots are in the logarithmic scale, since our method requires significantly fewer samples to converge than baseline approaches. From the figure, it is apparent that our MPC-RCE method learns the underlying constraint function quickly to avoid unsafe behaviors during the exploration and achieves the lowest constraint violation rate throughout the training. During the training, the reward begins increasing almost one magnitude episodes earlier for our method and the number of constraint violations remains low. After training, our method can achieve **nearly zero constraint violations**, while baseline methods struggle to meet the target constraint threshold. In addition, our method maintains high task rewards, while we could observe a clear reward decreasing of the baseline Lagrangian-based approaches after certain steps along with the cost, which is caused by the oscillation behavior and instability of the dual variables (see Appendix for more details).

The last row's figures show the effectiveness of utilizing each cost – how much task rewards we could obtain given a budget of constraint violations. The curves that are close to the upper left would be better because they require less unsafe samples to achieve high rewards. Though the off-policy baselines (SAC-Lag, DDPG-Lag) are generally more sample efficient regarding cumulative costs than on-policy baselines, our method still outperforms them with large margin among all tasks. For simple tasks, such as `Point-Goal1` and `Car-Goal1`, our method use 1 or 2 magnitude less unsafe samples to achieve the same task rewards as the best baseline approach, while the margin is larger for more challenging `Point-Goal2` and `Car-Goal2` tasks, where MPC-RCE could reach the same performance with at least 100 times fewer unsafe samples. As far as we are aware, our method can achieve the best constraint satisfaction performance in these Safety Gym tasks without prior knowledge of them. It is more clear in the last row figures, the cost it takes to reach high rewards is much lower for our method.

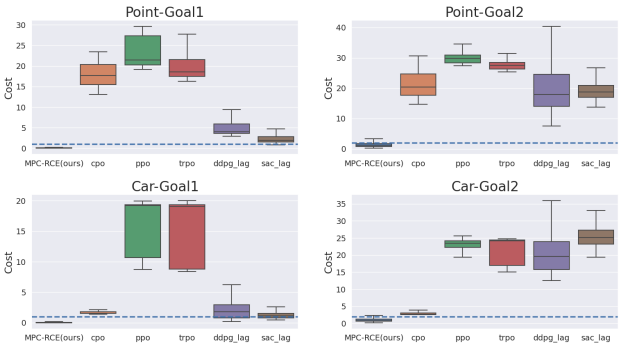


Figure 4. Comparison of constraint violation of model-free methods after convergence

The constraint violation after convergence for model-free baseline comparison is demonstrated in Fig. 4. From the figure, we can see that our approach realizes the lowest cost among all the baselines. Besides, it is clear that the

maximum of converged cost of our approach is below the minimum cost of all the baselines. In addition, the variance is smaller for RCE, which indicates that our method is more robust than baselines.

5.4. Comparison of Model-based Baselines

In order to address the second question – what is the role of RCE in the algorithm, we compare RCE with CEM and random shooting under the same control framework. Note that all the other training variables, such as hyper-parameters, are the same as RCE for the model-based baselines. The only difference is the optimizer, where we use a robust estimation mechanism to estimate the risk of rollout trajectories. Table 1 demonstrates the constraint satisfaction performance during the training procedure. We compare the total number of constraint violations for the first 10,000 steps of training. From the table, we can see our approach achieves much lower cost than baselines, which means that the robust estimation scheme is the crucial factor to ensure safety – MPC-RCE agent requires the minimum number of samples to converge and is able to explore the environment in a safer way.

Table 1. Comparison of total constraint violation number for the first 10000 training steps.

Method / Task	MPC-RCE	MPC-CEM	MPC-random
Point Goal 1	**16.00**	184.33	169.0
Point Goal 2	**231.00**	746.00	600.67
Car Goal 1	**7.00**	103.67	139.33
Car Goal 2	**96.00**	578.00	509.33

Fig. 5 demonstrates the box plot of constraint violation when the training is converged. Here, the convergence is denoted by the last 10% of the total training epochs. The box plot – also named whisker plot – displays the minimum, first quartile, median, third quartile, and maximum of the cost after convergence. From the figure, we can see that our approach realizes the lowest cost among all baselines (see appendix for the complementary results for model-free baselines). Interestingly, we could observe that even **the third quartile of converged cost of our approach is below the minimum cost** of the baselines, which indicates that the robust planning could improve safety with a large margin.

As we have shown in the results, RCE can achieve better constraint satisfaction performance than CEM, although they use the same dynamics model, constraint model, and hyper-parameters. We provide an intuition about why RCE is better — CEM is more likely to converge to unsafe action sequences than RCE when the agent is close to the dangerous areas. Denote the elite sample threshold as k. Consider the t-th iteration during the optimization procedure. Suppose that in the sampled action sequences, only $q < \frac{k}{2}$ samples are safe, and the remaining samples will cause constraint violations, which is likely to happen when the agent is close to unsafe areas. For RCE method, the feasible set selection phase will help to discard all the unsafe samples in the elites, so that only the remaining safe samples will be used to update the sampling distribution at the $(t+1)$-th iteration. However, for the CEM method that simply adds cost penalties to the reward for unsafe samples, all the k samples will be used to calculate the mean. Therefore, the mean of the $(t+1)$-th sampling distribution will be biased towards unsafe samples, which causes more unsafe samples in the $(t+1)$-th iteration.

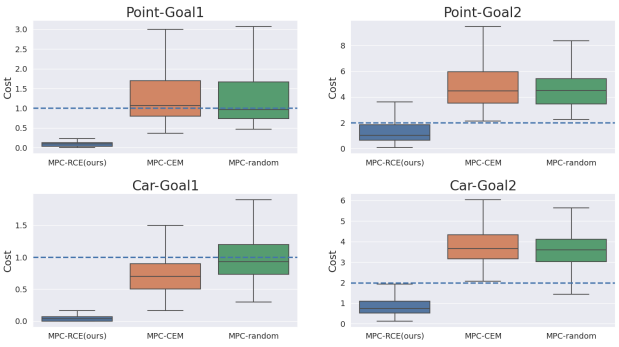


Figure 5. Comparison of constraint violation of model-based methods after convergence

6. Conclusion

We introduce a simple yet effective constrained model-based RL algorithm without any prior assumptions on the system dynamics or the constraint function. We propose the robust cross-entropy method (RCE) to optimize the action under the MPC framework in light of the model uncertainty and underlying constraints. Our method is evaluated in the Safety Gym environment and achieves better constraint satisfaction while maintaining high task performance compared with other constrained RL baselines. In addition, RCE is much more sample efficient and has better exploitation capability of unsafe samples than baseline approaches. We also show that the robust estimation mechanism is indeed the crucial factor to ensure safety in the overall RL framework, and simply extending previous model-based RL approaches to the safe RL domain may suffer from poor constraint satisfaction. We believe our approach could inspire more work along this direction to further improve the robustness and safety of RL agents.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International Conference on Machine Learning*, pp. 22–31. PMLR, 2017.

Berkenkamp, F., Turchetta, M., Schoellig, A. P., and Krause, A. Safe model-based reinforcement learning with stability guarantees. *arXiv preprint arXiv:1705.08551*, 2017.

Botev, Z. I., Kroese, D. P., Rubinstein, R. Y., and L'Ecuyer, P. The cross-entropy method for optimization. In *Handbook of statistics*, volume 31, pp. 35–59. Elsevier, 2013.

Chow, Y., Nachum, O., Faust, A., Duenez-Guzman, E., and Ghavamzadeh, M. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pp. 4754–4765, 2018.

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., and Tassa, Y. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Drews, P., Williams, G., Goldfain, B., Theodorou, E. A., and Rehg, J. M. Aggressive deep driving: Combining convolutional neural networks and model predictive control. In *Conference on Robot Learning*, pp. 133–142, 2017.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Gaskett, C. Reinforcement learning under circumstances beyond its control. 2003.

Geibel, P. and Wysotzki, F. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pp. 3146–3154, 2017.

Koller, T., Berkenkamp, F., Turchetta, M., and Krause, A. Learning-based model predictive control for safe exploration. In *2018 IEEE Conference on Decision and Control (CDC)*, pp. 6059–6066. IEEE, 2018.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566. IEEE, 2018.

Okada, M. and Taniguchi, T. Variational inference mpc for bayesian model-based reinforcement learning. In *Conference on Robot Learning*, pp. 258–272, 2020.

Pham, T.-H., De Magistris, G., and Tachibana, R. Optlayer-practical constrained optimization for deep reinforcement learning in the real world. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6236–6243. IEEE, 2018.

Ray, A., Achiam, J., and Amodei, D. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7, 2019.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Stooke, A., Achiam, J., and Abbeel, P. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pp. 9133–9143. PMLR, 2020.

Sun, Y., Wong, A. K., and Kamel, M. S. Classification of imbalanced data: A review. *International journal of pattern recognition and artificial intelligence*, 23(04): 687–719, 2009.

Wang, Y., Gan, W., Yang, J., Wu, W., and Yan, J. Dynamic curriculum learning for imbalanced data classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Wen, M. and Topcu, U. Constrained cross-entropy method for safe reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 7450–7460, 2018.

A. Discussion about the Safety Gym Environment and the Task Setting

Safety Gym environments use the MuJoCo physics engine as the backbone simulator. Each environment and task is inspired by a practical safety issue in robotics control. The observation spaces used in the original Safety Gym environment includes standard robot sensors (accelerometer, gyroscope, magnetometer, and velocimeter) and pseudo-lidar (each lidar sensor perceives objects of a single kind and is computed by filling bins with appropriate values). The observation space used in our approach is different from the default Safety Gym options in that we pre-process the sensor data to get rid of some noisy and unstable sensors, such as the z-axis data of accelerometer. We use the relative coordinates of the perceived objects instead of the pseudo-lidar readings because the former representation is more friendly to dynamics model learning, which is important for model-based RL.

Both robots used in our experiment have two-dimensional continuous action spaces and all actions are linearly scaled to $[-1, +1]$. We also performed careful hand-tuning of some MuJoCo actuator parameters during sensor analysis, since robust and responsive control is critical to robot operations in both the simulation environment and the real world.

Our work in the Safety Gym environment has implications for real-world applications. The Goal task in our experiment resembles the setting of the delivery robot and other domestic robots, where the robot has to navigate around static obstacles such as furniture to reach the goal. Additionally, since the state representation in our experiments is directly derived from sensor information and the control input of our environment to the robot is very similar to that of real-world situations, our model-based RL approach in the simulation environment could serve as an important pre-training for the real-world applications. Given that a certain amount of unsafe data is required to train our model, it would be unrealistic to have the real robot repeatedly violate the constraints to collect such data. Therefore, the training in the simulator is an important step for the model to be transferable to real wold safety-critical applications.

B. Training Detail

Dynamics model: We use the same architecture and hyper-parameters of each neural network in the ensemble dynamics model. Each neural network is of 3 layers with ReLU activation and each layer is of 1024 neurons. All the training parameters for one task are the same for MPC-RCE, MPC-CEM, and MPC-random. The batch size is 256, the learning rate is 0.001, the training epochs are 70, and the optimizer is Adam. The ensemble number is 4 for Point robot-related tasks and is 5 for Car robot-related tasks. Each neural network model in the ensemble is trained with 80% of the training data to prevent overfitting.

LightGBM classifier: We use LightGBM to predict the constraint violation given a state in our RCE method and all the model-based baselines. We use the default `gdbt` boosting type and 400 base estimators. Each base estimator has a maximum depth of 8 and 12 leaves. The learning rate is 0.3 and all other hyper-parameters are the default value.

RCE, CEM, and random optimizer: We use the same hyper-parameters for RCE and CEM except that RCE has a discount value of $\gamma = 0.98$ for reward and discount value of $\beta = 0.4$ for cost while CEM only has one discount value $\gamma = 0.98$ for the combination of reward and cost. We sample $N = 500$ solutions for each iteration of RCE and CEM and select top $k = 12$ elite samples to estimate the distribution parameters for the next iteration. If the iteration number exceeds 8 or the sum of the variance of elite samples is less than $\epsilon = 0.01$, the optimization procedure stops and returns the best solution that has been found so far. To fairly compare with RCE and CEM, we use 5000 samples for the random shooting method so that the maximum number of samples is at the same order of magnitude. The planning horizon is $T = 8$ for all methods.

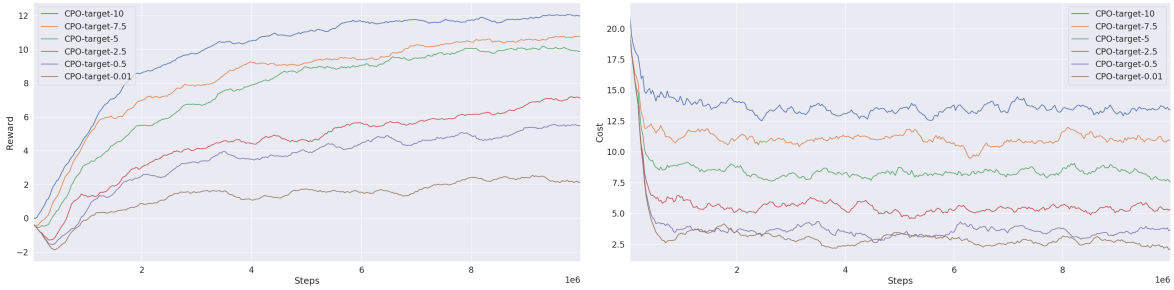
TRPO, TRPO-Lagrangian, and CPO: We use the same hyper-parameters offered in the open-sourced code from the baseline method for the Safety Gym simulation environment (Ray et al., 2019). All hyperparameters are kept the same for all three model-free baseline methods. The actor-critic neural network model has 2 linear layers of 256 hidden neurons in each. The discount factor $\gamma = 0.99$. The target cost limit is 10 with penalty term λ initialized to be 1 and a penalty term learning rate of 0.05. The target KL divergence is 0.01, and for the value function learning, the learning rate is 0.001 with 80 iterations. For each experiment, the total number of environment interactions is $1e7$ and $3e4$ steps for each training epoch. More hyper-parameters information can be found in the source code.

C. More Results

The influence of the target cost value for TRPO-Lagrangian and CPO. Since the target cost limit value must be set in advance before training, we empirically study the performance of TRPO-Lagrangian and CPO with different target values

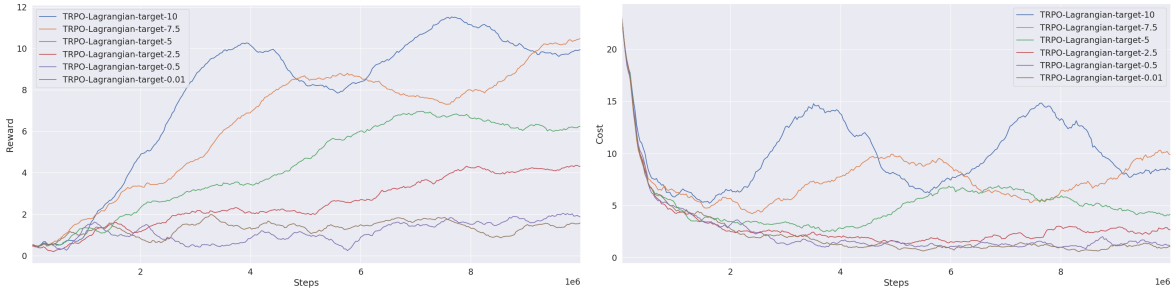
in the Point Goal2 environment. The learning curves are shown in Fig 6 and Fig 7. We can see that the task performance is negatively correlated with the target cost, and there is a dramatic task performance drop if we limit the target cost to a small value that is comparable with our method's performance. Compared with model-based approaches, CPO and TRPO-Lagrangian can hardly achieve comparable task performance with the same level of constraint violation rate. In addition, we could observe obvious oscillation behavior of the training curves, which induces many unstable factors that might affect the final performance.

RCE, CEM, and random optimizer comparison. To better compare the performance of RCE, CEM, and random optimizer, we fix the dynamics model, cost model, and random seed to test in the same Point Goal1 environment. The smoothed reward and cost curves are shown in Fig. 8. We can see our RCE approach achieves the lowest cost throughout the testing phase while maintaining comparable task performance compared to CEM and random. It is interesting to note that there is a reward drop for RCE and a cost jump for CEM and random methods at around 1000 steps, which means the environment layout in this episode is difficult. Compared to CEM and random methods, which fail to explore the environment safely, our RCE approach is able to achieve zero constraint violation.



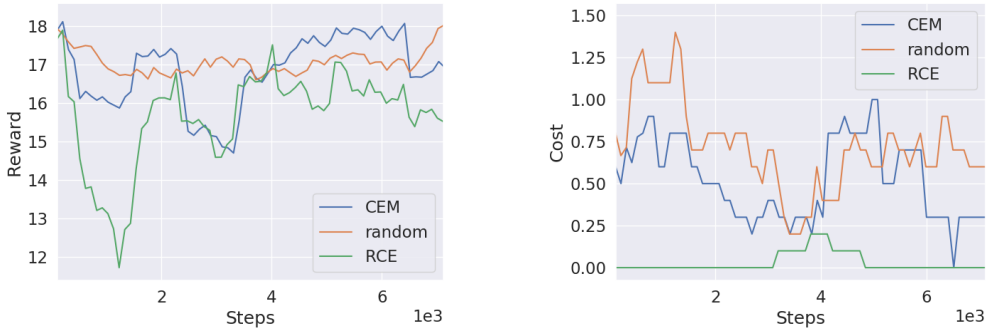
(a) Reward (b) Cost

Figure 6. Learning curves of reward and cost for CPO with different target cost value.



(a) Reward (b) Cost

Figure 7. Learning curves of reward and cost for TRPO-Lagrangian with different target cost value.



(a) Reward (b) Cost

Figure 8. Testing curves of reward and cost with fixed learned models.