Safe Model-based Reinforcement Learning with Robust Cross-Entropy Method

Zuxin Liu, Hongyi Zhou, Baiming Chen, Sicheng Zhong, Martial Hebert, Ding Zhao

***Abstract*— This paper studies the safe reinforcement learning (RL) problem without assumptions about prior knowledge of the system dynamics and the constraint function. We employ an uncertainty-aware neural network ensemble model to learn the dynamics, and we infer the unknown constraint function through indicator constraint violation signals. We use model predictive control (MPC) as the basic control framework and propose the robust cross-entropy method (RCE) to optimize the control sequence considering the model uncertainty and constraints. We evaluate our methods in the Safety Gym environment. The results show that our approach achieves better constraint satisfaction than baseline safe RL methods while maintaining good task performance. Additionally, we are able to achieve several orders of magnitude better sample efficiency when compared to constrained model-free RL approaches. The code is available at `https://github.com/liuzuxin/safe-mbrl`.**

I. Introduction

Reinforcement learning (RL) has achieved great success in a wide range of applications, including solving Atari games [1], autonomous vehicles [2], and robot control [3], [4]. 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 safe reinforcement learning algorithms for real-world applications, which allow them to complete tasks while satisfying certain safety constraints.

Though some prior research has been proposed to achieve constrained reinforcement learning under certain conditions, there is an important problem that is seldom studied in the existing literature. Namely, how can we enforce safety constraints for an RL agent without the knowledge of an explicit analytical expression of the constraint function? One example problem that falls under this category is the `PointGoal` task setting in the Safety Gym simulation environment [5], 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 sensor data, such as a LiDAR point cloud, so it is hard to analytically express the mapping from observation space to the constraint violation. Another example could be an autonomous vehicle with only image data as the input. It is easy to collect historical accident data, but it would be difficult to directly analytically define which image represents an unsafe scenario because of numerous possibilities, such as hitting the road, hitting the tree, and so on. Thus we are interested in the hardest cases where both dynamics and constraints are needed to be learned from data without additional info.

The challenges of solving the above problem are threefold: First, pure model-free, constrained RL algorithms (such as Lagrangian-based methods [6], [7] and projection-based optimization methods [8]) are not sample efficient. They need to constantly violate safety constraints and collect a large number of unsafe data to learn the policy, 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 [9], [10]). 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 [5]. Most existing model-based safe 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 [11], [12], [13]). As far as we are aware, very little research has been done to investigate situations in which the dynamics and the constraint are both unknown.

In this paper, we present a model-based reinforcement learning algorithm with a robust cross-entropy (RCE) method to achieve near-optimal task performance with near-zero constraint violation rates. Our three main contributions are as follows:

1) We study the safe RL problem for the tasks where environment dynamics and the analytical expression

*(Corresponding author: Ding Zhao)

Zuxin Liu, Hongyi Zhou and Ding Zhao are with the Department of Mechanical Engineering, Carnegie Mellon University, USA (e-mail: {zuxinl, hzhou4, dingzhao}@andrew.cmu.edu)

Baiming Chen is with the School of Vehicle and Mobility, Tsinghua University, Beijing, China (e-mail: cbm17@mails.tsinghua.edu.cn)

Sicheng Zhong is with the Division of Engineering Science, University of Toronto, Canada (e-mail: sicheng.zhong@mail.utoronto.ca)

Martial Hebert is with the Robotics Institute, Carnegie Mellon University, USA (e-mail: hebert@cs.cmu.edu)

of the constraint function are unknown. We formulate the problem under the constrained Markov Decision Processes framework and propose a solution to learn both the dynamics model and constraint model from collected data with limited unsafe samples and weak constraint violation indicator signals.

2) We propose MPC-RCE, a model-based safe reinforcement learning algorithm for environments with continuous state and action spaces. We propose a robust cross-entropy optimization method towards the dynamics model prediction error that works with an uncertainty-aware dynamic model to achieve safe explorations of the environment.
3) Our approach is evaluated in the Safety Gym environment [5], and the results show that our method is able to achieve state-of-the-art performance in terms of constraint violation rate and accumulated expected reward when compared to existing constrained model-free and model-based approaches. Compared with model-free methods, our approach is much more sample efficient, as we use a data buffer to memorize unsafe states, while model-free methods need to constantly violate constraints in order to optimize the policy.

II. Related Work

Safe reinforcement learning aims to learn policies that maximize the expected task reward while satisfying safety constraints throughout the learning and/or the deployment processes [14]. Safe RL problems are usually modeled under the constrained Markov decision processes (CMDPs) [7] framework, where the agents are enforced with restrictions on expected auxiliary constraint violation costs. One popular method to solve safe 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 [14], [10], [15]. 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 [5] 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 [14].

Recently, several constrained model-free RL algorithms have attracted much attention. Achiam et al. [8] 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 [16]. 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 [5], [6]. 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. [13] and Dalal et al. [17] 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 [18], [19]. To guarantee safe exploration of the environment, Gaussian Processes (GPs) are usually used to model the dynamics because of their ability to estimate uncertainty [11], [12], [20]. 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 [21], [22]. By contrast, neural network models can scale well with high-dimensional data. However, a single neural network is not good at estimating uncertainty. In this paper, we will use an ensemble of neural networks to model the dynamics of the environment, which provides accurate dynamics prediction as well as uncertainty estimation.

III. Preliminaries

A. Constrained Markov Decision Process

We investigate the safe RL problem in the constrained Markov decision process (CMDP) framework, which is defined by a tuple $(\mathcal{S}, \mathcal{A}, f, r, c)$, 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.

We assume the state transition function f and the cost function c are both unknown, and can only be learned from data collected by interacting with the environment. 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 reward function r and $J_c(\pi)$ denote the expected return of policy π w.r.t 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 the policy π.

Some model-free constrained RL methods, such as Lagrangian-based methods [6], 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

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

Algorithm 1 CEM for RL

Input: Initial distribution parameter Θ ; number of samples N; number of elite samples k

Output: Solution $\mathcal{X}^*$ with the highest reward

1: **while** The stop criteria is not satisfied **do**
2: Draw N samples from the distribution: $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_N \sim \mathcal{N}(\Theta)$
3: Evaluate each sample $\mathcal{X}_i$ by the reward function $r(\mathcal{X}_i)$
4: Sort $\{\mathcal{X}_i\}_{i=1}^N$ in descending order w.r.t the reward. Let Λ_k be the first k elements
5: Update Θ by maximizing the likelihood given Λ_k: $\Theta \leftarrow \arg\max_\theta \prod_{\mathcal{X} \in \Lambda_k} p(\mathcal{X}; \theta)$
6: **end while**
7: **return** $\mathcal{X}^*$ with highest reward in Λ_k

where π^* is the optimal policy. We aim to achieve perfect constraint satisfaction, so the final objective function becomes

$$\pi^* = \arg\max_\pi J_r(\pi), \quad s.t. \quad J_c(\pi^*) = 0$$

B. 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 [23], [21]. 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. Denote the reward function as $r(\mathcal{X}) : \mathcal{S} \mapsto \mathbb{R}$, and the probability density function as $p(\mathcal{X}; \Theta)$. The CEM algorithm commonly used in RL is described in Algorithm 1. For more details on CEM methods and their applications, refer to [24].

IV. Approach

A. Model Learning

As we introduced in section III-A, the dynamic 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 [21]. Both Bayesian models, such as Gaussian processes (GPs) [12], [11], [22], [23], and neural networks [25] have been used to learn the dynamics. However, GP-based approaches usually make additional assumptions on the smoothness of the environment dynamics, and their performance relies heavily on the choice of kernel. Additionally, the model capacity and computational complexity limit the application of GPs in high-dimensional environments with complex dynamics [21]. On the other hand, the approaches with a single neural network are not good at representing epistemic uncertainty, which may lead to accumulated prediction errors along the planning horizon and thus results in worse performance compared to model-free RL algorithms [25].

Chua et. al [21] proposed an ensemble of probabilistic neural network models with task performance comparable to model-free methods, better sample efficiency, and reasonable uncertainty estimation in complex environments. We adopt a similar 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. We use deterministic neural networks to represent deterministic environment dynamics, without the need to estimate the aleatoric uncertainty (inherent variance of the observed data) that can be captured by probabilistic neural network models [21].

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 [26], [27]. 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 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 [28] - 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.

B. Model Predictive Control with Learned Dynamics and Cost Model

We use Model Predictive Control (MPC) as the basic control framework for our safe model-based RL approach [29], [30]. 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. 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:

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

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, as we introduced in section IV-A, and can be viewed as black-box functions. Without the cost function, the Eq. 1 can be solved with many optimization methods, such as random shooting and the cross-entropy method (CEM) which have successfully been applied to many model-based RL tasks [25], [21], [23]. However, very few papers have investigated the problem with additional black-box constraints. One natural way to extend existing unconstrained optimization methods is to add large penalties to the objective for unsafe states. Then the optimization becomes:

$$\begin{aligned} \mathcal{X} = \arg\max_{a_0,\ldots,a_T} \quad & \mathbb{E}\big[\sum_{t=0}^{T} \gamma^t \big(r(s_{t+1}) - \lambda c(s_{t+1})\big)\big] \\ s.t. \quad & s_{t+1} = f(s_t, a_t), \forall t \in \{0, 1, ..., T-1\} \end{aligned} \tag{2}$$

where λ is a large positive value that encourages the solution to satisfy the constraints. We can then use random shooting [25] to sample a batch of sequences and select the best one, or use the CEM method [21] that is introduced in section III-B to solve Eq. 2.

C. Robust Cross-Entropy Method for Planning

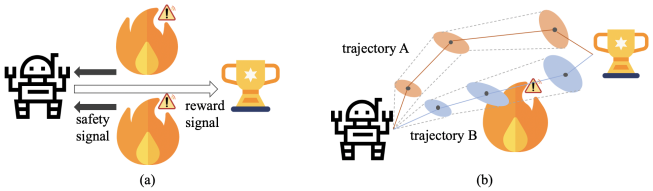


Fig. 1: (a): The reward signal and safety signal may contradict to each other; (b): The trajectory sampling method for uncertainty-aware dynamic models.

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 [21] to estimate reward and constraint violation cost. We define the solution $\mathcal{X} = (a_0, a_1, ..., a_{T-1})$ to be an action sequence with length of planning horizon T. Given the initial state s_0, we can

Algorithm 2 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.3 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

evaluate the accumulated reward and cost of the solution by:

$$\begin{aligned} r(\mathcal{X}; s_0) &= \sum_{t=0}^{T} \gamma^t \big(\frac{1}{B} \sum_{b=1}^{B} r(s_{t+1}^b)\big), \\ c(\mathcal{X}; s_0) &= \sum_{t=0}^{T} \beta^t \max_b c(s_{t+1}^b) \end{aligned} \tag{3}$$

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 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 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 dynamic model prediction is not highly accurate.

Our RCE method first selects the feasible set of solutions that satisfy the constraints based on the estimated cost in Eq. 3. 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. A similar idea is adopted in [16]. 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 expectation. 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 RCE algorithm is shown in Algorithm 2. The entire training pipeline of our MPC with RCE is presented in Algorithm 3.

Algorithm 3 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. 2:
    $\{a_i^*\}_{i=t}^{t+T} \leftarrow \text{RCE}(\mathcal{P}, s_t)$
    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

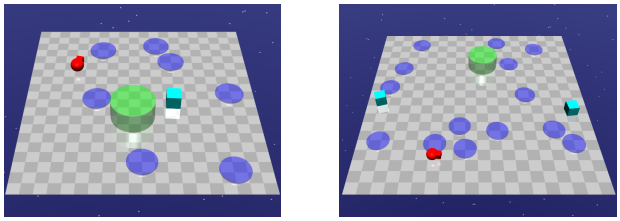
V. Experiments

A. Simulation Environment

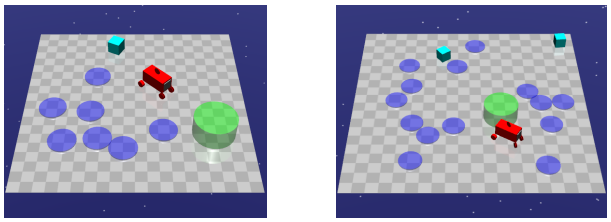
We evaluate our safe RL approach in the OpenAI Safety Gym environment [5]. Each experiment setting involves a robot that must navigate a clustered environment to accomplish a task while avoiding contact with the surrounding objects. The robotic agent in the environment perceives the world through its sensors and performs actions with its actuators. In our experiment, we consider two robots: Point (red object in Fig. 2a), a simple pre-made robot included in Safety Gym, and Car (red object in Fig. 2c), a customized four wheel car model with differential drive and Ackermann steering that has one actuator for turning the front wheels and another for driving the back wheels forward/backward. Both robots have two-dimensional continuous action spaces and observation spaces ranging from 24 to 40 dimensions depending on the robot type and difficulty level.

We choose the Goal task from the Safety Gym environment (Fig. 2a), which requires the robot to move to a series of goal positions. When the robot enters the goal circle (green circle), the goal location is randomly reset, while the rest of the layout is kept the same in one episode. The entire layout is reset after an episode ends. Dense reward option gives the robot agent a small bonus for moving towards the goal [5], a bonus of $r_t = 1$ is given to the robot for reaching the goal.

We use two Safety Gym constraints elements: Hazards and Vases. As Fig. 2 shows, 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. The cost function follows the default Safety Gym setting: $c_t = 1$ if the agent violates the safety constraint at the current time-step. The Goal task is categorized into 2 levels; level 2 tasks (Fig. 2b) are more difficult to solve than level 1 (Fig. 2a) task since there are more constraints presented.



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



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

Fig. 2: Experiment Environments

B. Baselines and Experiment Setting

Baselines. We use the official baseline methods provided by the Safety Gym environment [5]. The trust region policy optimization (TRPO) algorithm [31] is an unconstrained baseline that could give us intuition about the performance and constraint violations when we only care about the reward and not the cost of violating constraints. We use the Lagrangian version of TRPO (TRPO-Lagrangian) and the constrained policy optimization (CPO) [8] as two model-free constrained reinforcement learning baselines, which we have introduced in section II. For model-based safe RL baselines, we adopt the constrained extension of existing random shooting and the cross-entropy method (CEM) together with the MPC framework, which are introduced in section IV-B. We name the two methods as MPC-random and MPC-CEM.

Metrics. We compare different approaches in terms of episodic accumulated reward and episodic cost, which is defined as the accumulated constraint violation counts in each episode. We also compare the sample efficiency based on the number of total interactions needed to converge, and compare the total cost during training.

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 [5]. For each algorithm, we evaluate them for each task with 3 different random seeds. For the detail about specific hyper-parameters used in our experiments, please refer to the supplementary material and code. Our code has been released as a public benchmark for further research on safe RL.

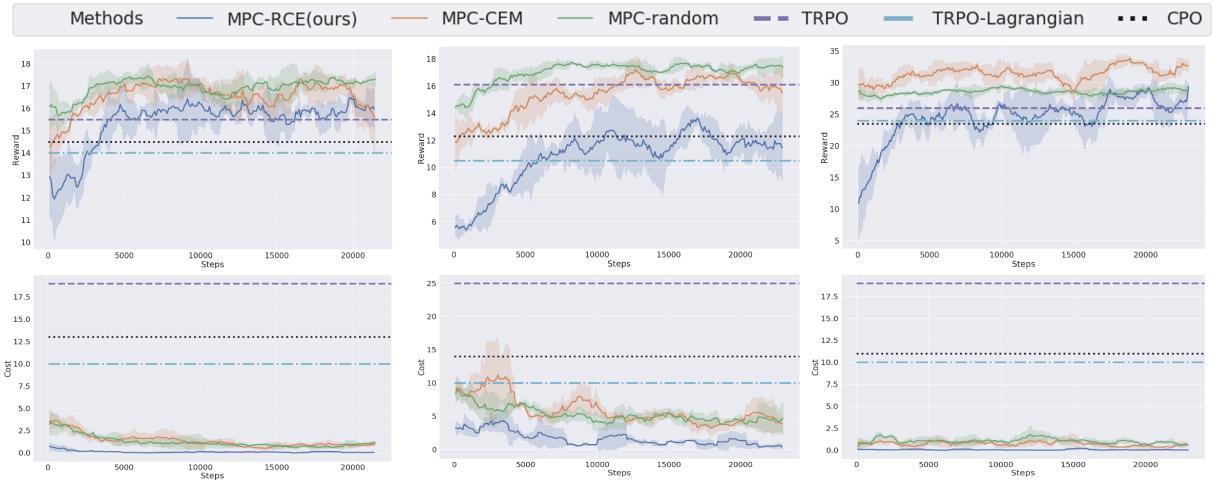


Fig. 3: Learning curves. The upper row figures are the reward trends and the lower row figures are the cost trends. From left to right the tasks are: Point Goal1, Point Goal2, and Car Goal1.

C. Results

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 because collecting unsafe data could be expensive in some cases. Such setting is practical in real-world applications. Here we provide two examples: 1. The cost to enter an 'unsafe' zone is not deadly but is not desired. For example, a tomato-picking-up robot damages tomatoes, which is not harmful to itself and people, but we still regard it as unsafe because it should be avoided in reality. 2. We can train the robot by historical unsafe data (such as public dataset) or in the simulator, which is not harmful during training. Then we can deploy trained safe policies to real-world situations.

Fig. 3 shows the learning curves of reward and constraint violation cost in Point Goal1, Point Goal2, and Car Goal1 tasks. The reward and cost are averaged among 3 experiments with the same hyper-parameters but different random seeds. The solid line is the mean value, and the light shade represents the area within one standard deviation. We use dashed lines to represent the value at convergence for model-free approaches, as they require several orders of magnitude more interaction steps to obtain satisfactory performance. Taking the Point Goal1 environment as an example, the model-based approaches converge after 5e3 steps while TRPO requires 4e6 steps, which means our approach is 80 times more sample efficient compared to model-free methods.

From the figure, it is apparent that our MPC-RCE approach learns the underlying constraint function very quickly to avoid unsafe behaviors during the exploration and achieves the lowest constraint violation rate, though its reward is slightly lower than other methods. For constrained model-free methods, a target cost value is required to be set in advance; it is set to be 10 in Fig. 3. We observed that if we set the target constraint violation rates to be a small value, the reward will decrease dramatically. In addition, these model-free methods constantly violate constraints during training to optimize the policy, which is not efficient and could be dangerous in practice. More experimental results and discussion on constrained model-free RL are presented in the supplementary material.

TABLE I: Comparison of total/normalized constraint violation number for the first 10000 steps.

Task \ Method	MPC-RCE	MPC-CEM	MPC-random
Point Goal 1	**16.00 / 1.00**	184.33 / 11.50	169.0 / 10.56
Point Goal 2	**231.00 / 1.00**	746.00 / 3.23	600.67 / 2.60
Car Goal 1	**7.00 / 1.00**	103.67 / 14.81	139.33 / 19.90
Car Goal 2	**96.00 / 1.00**	578.00 / 6.02	509.33 / 5.31

Table I 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, where the first number in each cell is the total number of constraint violations and the second number is the normalized value based on our MPC-RCE approach. For constrained model-free RL methods, the total number of constraint violations is several orders of magnitude larger than model-based approaches, so we do not list the result here. From the table, we can see our approach achieves much lower cost than the other methods for the first 10,000 training steps, which means the MPC-RCE agent requires the minimum number of samples to converge and is able to explore the environment in a safer way.

VI. Conclusion

We introduce a safe 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 good task performance compared with other safe RL baselines. Potential directions for future work include improving the cost model to provide a measure of risk towards longer horizons rather than only a single step prediction.

ACKNOWLEDGMENT

The authors acknowledge the support from the Manufacturing Futures Initiative at Carnegie Mellon University made possible by the Richard King Mellon Foundation. The authors would like to thank Keegan Harris, Peide Huang and Zixin Ye for their valuable suggestions of this paper.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[2] A. Filos, P. Tigas, R. McAllister, N. Rhinehart, S. Levine, and Y. Gal, "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?" *arXiv preprint arXiv:2006.14911*, 2020.

[3] B. Chen, M. Xu, Z. Liu, L. Li, and D. Zhao, "Delay-aware multi-agent reinforcement learning," *arXiv preprint arXiv:2005.05441*, 2020.

[4] Z. Liu, B. Chen, H. Zhou, G. Koushik, M. Hebert, and D. Zhao, "Mapper: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments," *arXiv preprint arXiv:2007.15724*, 2020.

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

[6] A. Stooke, J. Achiam, and P. Abbeel, "Responsive safety in reinforcement learning by pid lagrangian methods," *arXiv preprint arXiv:2007.03964*, 2020.

[7] E. Altman, "Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program," *Mathematical methods of operations research*, vol. 48, no. 3, pp. 387–417, 1998.

[8] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," *arXiv preprint arXiv:1705.10528*, 2017.

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

[10] C. Gaskett, "Reinforcement learning under circumstances beyond its control," 2003.

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

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

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

[14] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[15] A. Tamar, H. Xu, and S. Mannor, "Scaling up robust mdps by reinforcement learning," *arXiv preprint arXiv:1306.6189*, 2013.

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

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

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

[19] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," in *Advances in neural information processing systems*, 2018, pp. 8092–8101.

[20] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with gaussian processes," in *International Conference on Machine Learning*, 2015, pp. 997–1005.

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

[22] A. Wachi and Y. Sui, "Safe reinforcement learning in constrained markov decision processes," in *International Conference on Machine Learning (ICML)*, 2020.

[23] M. Xu, W. Ding, J. Zhu, Z. Liu, B. Chen, and D. Zhao, "Task-agnostic online reinforcement learning with an infinite mixture of gaussian processes," *arXiv preprint arXiv:2006.11441*, 2020.

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

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

[26] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *International journal of pattern recognition and artificial intelligence*, vol. 23, no. 04, pp. 687–719, 2009.

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

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

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

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

[31] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[32] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

APPENDIX

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 [5]. 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 4 and Fig 5. 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. Compared with model-based approaches, CPO and TRPO-Lagrangian can hardly achieve comparable task performance with the same level of constraint violation rate.

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. 6. 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

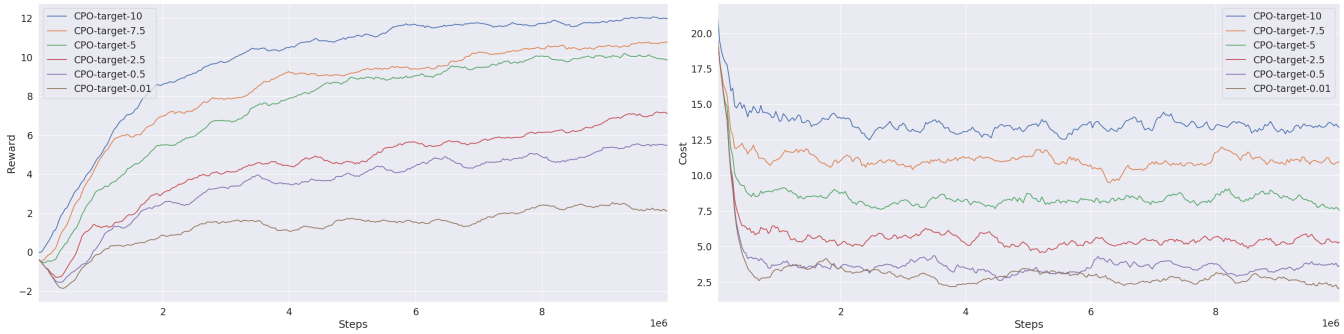


Fig. 4: Learning curves of reward and cost for CPO with different target cost value.

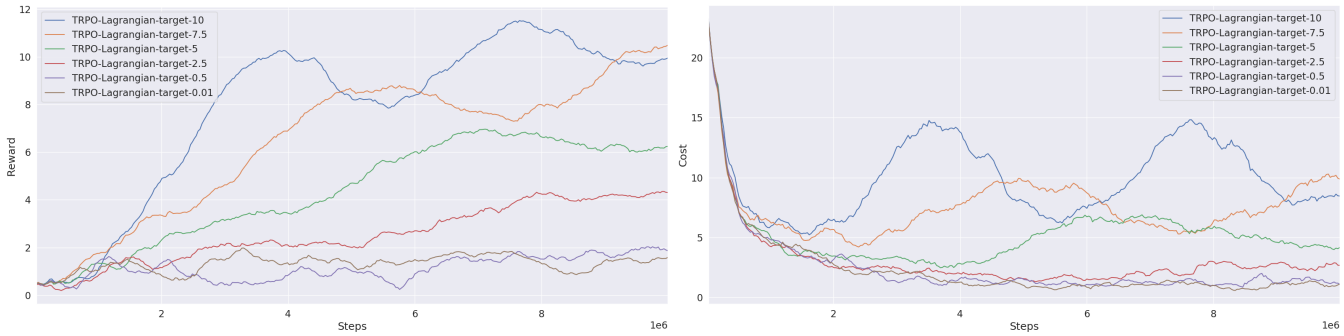


Fig. 5: Learning curves of reward and cost for TRPO-Lagrangian with different target cost value.

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.

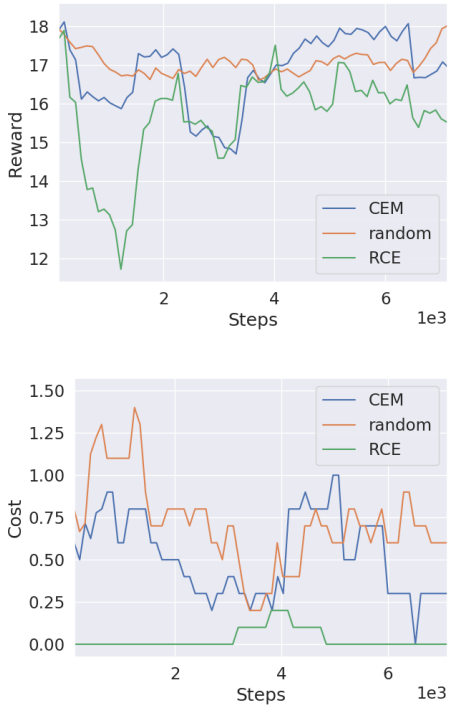


Fig. 6: Testing curves of reward and cost with fixed learned models.

Uncertainty-aware dynamics model selection. The dynamics model module in this paper can be replaced by any other uncertainty-aware models in principle, and there are some alternatives other than the ensemble method, such as the dropout-based approximate Bayesian inference method [32]. We implemented and tested the dropout-based method but found that the prediction performance is worse than the ensemble method, so we use the ensemble method in our cases.