Safe Reinforcement Learning by Imagining the Near Future

Anonymous Author(s)
Affiliation
Address
email

Abstract

Safe reinforcement learning has been a trending topic due to its high impact on applying reinforcement learning algorithms to real-world problems. In this work, we focus on the setting that unsafe states can be avoided by looking ahead several steps. We devise a model-based algorithm that heavily penalizes unsafe trajectories, and also derive theoretical evidence that our algorithm can avoid unsafe states under certain assumptions. Experiments demonstrate that our algorithm can achieve competitive rewards with fewer safety violations in several continuous control tasks.

1 Introduction

Reinforcement learning (RL) enables the discovery of effective policies for sequential decision-making tasks via trial and error [Mnih et al., 2015, Gu et al., 2016, Bellemare et al., 2020]. However, in domains such as robotics, healthcare, and autonomous driving, certain kinds of mistakes pose danger to people and/or objects in the environment. Hence there is an emphasis on the safety of the policy, both at execution time and while interacting with the environment during learning. This issue, referred to as *safe exploration*, is considered an important problem in AI safety [Amodei et al., 2016].

In this work, we advocate a model-based approach to safety, meaning that we estimate the dynamics of the system to be controlled and use the model for planning (or more accurately, policy improvement). The primary motivation for this is that a model-based method has the potential to anticipate safety violations *before they occur*. Often in real-world applications, the engineer has an idea of what states should be considered violations of safety: for example, a robot colliding rapidly with itself or surrounding objects, a car driving on the wrong side of the road, or a patient's blood glucose levels spiking.Yet model-free algorithms typically lack the ability to incorporate such prior knowledge and must encounter some safety violations before learning to avoid them.

We begin with the premise that in practice, forward prediction for relatively few timesteps is sufficient to avoid safety violations. Consider the didactic example in Figure 1, in which an agent navigates a 2D environment while avoiding obstacles. Note that there is an upper bound on how far into the future the agent would have to plan to foresee and (if possible) avoid the collision.

Assuming that the horizon required for detecting unsafe situations is not too large, we show how to construct a reward function with the property that an optimal policy will never incur a safety violation. A short prediction horizon is also beneficial for model-based RL, as the well-known issue of *compounding error* plagues long-horizon prediction [Asadi et al., 2019]: imperfect predictions are fed back into the model as inputs (possibly outside the distribution of inputs in the training data), leading to progressively worse accuracy as the prediction horizon increases.

Our main contribution is a model-based algorithm that utilizes a reward penalty – the value of which is prescribed by our theoretical analysis – to guarantee safety (under some assumptions). Experi-

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

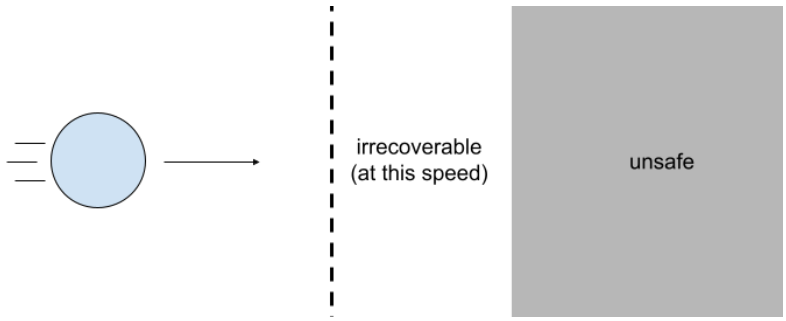


Figure 1: A didactic example. The agent navigates its environment and must avoid obstacles. If it is traveling toward an obstacle and its speed is high, then once it gets too close to the obstacle there is no way to avoid it. If the agent planned several steps into the future, it could avoid take evasive action or reduce its speed to avoid this "irrecoverable" region.

ments indicate that the practical instantion of our algorithm, Safe Model-Based Policy Optimization, effectively reduces the number of safety violations on several continuous control tasks, achieving a comparable performance with far fewer safety violations compared to several model-free safe RL algorithms.

2 Background

Consider a Markov decision process (MDP) $M = (\mathcal{S}, \mathcal{A}, T, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ the reward function, and $\gamma \in [0, 1)$ the discount factor. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ determines what action to take at each state. A trajectory is a sequence of states and actions $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$ where $s_{t+1} = T(s_t, a_t)$ and $r_t = r(s_t, a_t)$.

Typically, the goal is to find a policy which maximizes the expected discounted return $\eta(\pi) = \mathbb{E}^\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$. The notation $\mathbb{E}^\pi$ denotes that actions are sampled according to $a_t \sim \pi(s_t)$. The initial state s_0 is drawn from an initial distribution which we assume to be fixed and leave out of the notation for simplicity.

The Q function $Q^\pi(s, a) = \mathbb{E}^\pi[\sum_{t=0}^{\infty} \gamma^t r_t \,|\, s_0 = s, a_0 = a]$ quantifies the conditional performance of a policy π assuming it starts in a specific state s and takes action a, and the value function $V^\pi(s) = \mathbb{E}_{a \sim \pi(s)}[Q^\pi(s, a)]$ averages this quantity over actions. The values of the best possible policies are denoted $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ and $V^*(s) = \max_\pi V^\pi(s)$. The function Q^* has the important property that any optimal policy $\pi^* \in \arg\max_\pi \eta(\pi)$ must satisfy $a^* \sim \pi^*(s) \implies \mathbb{P}(a^* \in \arg\max_a Q^*(s, a)) = 1$ for all states s. Q^* is the unique fixed point of the Bellman operator

$$\mathcal{B}^* Q(s, a) = r(s, a) + \gamma \max_{a'} Q(s', a') \quad \text{where } s' = T(s, a) \tag{1}$$

In model-based RL, the algorithm estimates a dynamics model $\widehat{T}$ using the data observed so far, then uses the model for planning or data augmentation. The theoretical justification for model-based RL is typically based some version of the "simulation lemma", which roughly states that if $\widehat{T} \approx T$ then $\hat{\eta}(\pi) \approx \eta(\pi)$ [Kearns and Singh, 2002, Luo et al., 2018].

3 Method

In this work, we train safe policies by modifying the reward function to penalize safety violations. We assume that the engineer provides a Boolean function $\textsc{Unsafe} : \mathcal{S} \to \{\text{true}, \text{false}\}$ which specifies whether or not visiting a given state should be considered a safety violation.

We must also account for the existence of states which are not themselves unsafe, but lead inevitably to unsafe states regardless of what actions are taken.

Definition 1. *A state s is said to be*

- ***unsafe*** *if* $\textsc{Unsafe}(s)$.
- ***irrecoverable*** *if* $\neg\textsc{Unsafe}(s)$ *but for any sequence of actions* $a_0, a_1, a_2, \ldots$, *the trajectory defined by* $s_0 = s$ *and* $s_{t+1} = T(s_t, a_t)$ *for all* $t \in \mathbb{N}$ *satisfies* $\textsc{Unsafe}(s_t)$ *for some* $t \in \mathbb{N}$.
- ***safe*** *if it is neither unsafe nor irrecoverable.*

Crucially, we do not assume that the engineer specifies which states are (ir)recoverable, as that would be a lot to ask. However, we do assume that a safety violation must come fairly soon after entering an irrecoverable region:

Assumption 1. *There exists a horizon* $H^* \in \mathbb{N}$ *such that, for any irrecoverable states* s, *any sequence of actions* $a_0, \ldots, a_{H^*-1}$ *will lead to an unsafe state. That is, if* $s_0 = s$ *and* $s_{t+1} = T(s_t, a_t)$ *for all* $t \in \{0, \ldots, H^* - 1\}$, *then* $\textsc{Unsafe}(s_t)$ *for some* $t \in \{1, \ldots, H^*\}$.

This assumption rules out the possibility that a state leads inevitably to termination but takes an arbitrarily long time to do so. The implication of this assumption is that a perfect lookahead planner which considers the next H steps into the future can avoid not only the unsafe states, but also any irrecoverable states, with some positive probability.

3.1 Reward penalty framework

Now we present a reward penalty framework for guaranteeing safety. Let $\widetilde{M} = (\mathcal{S}, \mathcal{A}, \widetilde{T}, \tilde{r}, \gamma)$ be an MDP with reward and dynamics

$$\left(\tilde{r}(s,a), \widetilde{T}(s,a)\right) = \begin{cases} (r(s,a), T(s,a)) & \neg\textsc{Unsafe}(s) \\ (-C, s) & \textsc{Unsafe}(s) \end{cases} \tag{2}$$

where the terminal cost $C \in \mathbb{R}$ is a constant (more on this below). That is, unsafe states are "absorbing" in that they transition back into themselves and receive the reward of $-C$ regardless of what action is taken.

The basis of our approach is to determine how large C must be so that the Q values of actions leading to unsafe/irrecoverable states are less than the Q values of safe actions.

Proposition 1. *Suppose that Assumption 1 holds, and let*

$$C > \frac{r_{\max} - r_{\min}}{\gamma^{H^*}} - r_{\max} \tag{3}$$

Then for any state s, *if* a *is a safe action (i.e.* $T(s,a)$ *is a safe state) and* a' *is an unsafe action (i.e.* $T(s,a)$ *is unsafe or irrecoverable), it holds that* $\widetilde{Q}^*(s,a) > \widetilde{Q}^*(s,a')$, *where* $\widetilde{Q}^*$ *is the* Q^* *function for the MDP* $\widetilde{M}$.

Proof. Since a' is unsafe, it leads to an unsafe state in at most H^* steps by assumption. Thus the discounted reward obtained is at most

$$\sum_{t=0}^{H^*-1} \gamma^t r_{\max} + \sum_{t=H^*}^{\infty} \gamma^t (-C) = \frac{r_{\max}(1-\gamma^{H^*}) - C\gamma^{H^*}}{1-\gamma} \tag{4}$$

By comparison, the safe action a leads to another safe state, where it can be guaranteed to never encounter a safety violation. The reward of staying within the safe region forever must be at least $\frac{r_{\min}}{1-\gamma}$. Thus, it suffices to choose C large enough that

$$\frac{r_{\max}(1-\gamma^{H^*}) - C\gamma^{H^*}}{1-\gamma} < \frac{r_{\min}}{1-\gamma} \tag{5}$$

Rearranging, we arrive at the condition stated. □

The important consequence of this result is that an optimal policy for this MDP will always take safe actions. However, in practice we cannot compute $\widetilde{Q}^*$ without knowing T. Therefore we extend our result to the model-based setting where the dynamics are imperfect.

3.2 Extension to model-based rollouts

We prove safety for the following theoretical setup. Suppose we have a dynamics model that outputs *sets* of states $\widehat{T}(s,a) \subseteq \mathcal{S}$ to account for uncertainty.

Definition 2. *We say that a set-valued dynamics model $\widehat{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})^1$ is **calibrated** if $T(s,a) \in \widehat{T}(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.*

We define the *Bellmin* operator:

$$\underline{\mathcal{B}}^* Q(s,a) = \tilde{r}(s,a) + \gamma \min_{s' \in \widehat{T}(s,a)} \max_{a'} Q(s',a') \tag{6}$$

Lemma 1. *The Bellmin operator $\underline{\mathcal{B}}^*$ is a γ-contraction in the ∞-norm.*

The proof is deferred to Appendix A.1. As a consequence Lemma 3.2 and Banach's fixed-point theorem, $\underline{\mathcal{B}}^*$ has a unique fixed point $\underline{Q}^*$ which can be obtained by iteration. This fixed point is a lower bound on the true Q function if the model is calibrated:

Lemma 2. *If $\widehat{T}$ is calibrated in the sense of Definition 3.2, then $\underline{Q}^*(s,a) \leq \widetilde{Q}^*(s,a)$ for all (s,a).*

Proof. Let $\tilde{\mathcal{B}}^*$ denote the Bellman operator with reward function $\tilde{r}$. First, observe that for any $Q, Q' : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, $Q \leq Q'$ pointwise implies $\underline{\mathcal{B}}^* Q \leq \mathcal{B}^* Q'$ pointwise because we have $\tilde{r}(s,a) + \gamma \max_{a'} Q(s',a') \leq \tilde{r}(s,a) + \gamma \max_{a'} Q'(s',a')$ pointwise and the min defining $\underline{\mathcal{B}}^*$ includes the true $s' = T(s,a)$.

Now let Q_0 be any inital Q function. Define $\widetilde{Q}_k = (\tilde{\mathcal{B}}^*)^k Q_0$ and $\underline{Q}_k = (\underline{\mathcal{B}}^*)^k Q_0$. An inductive argument coupled with the previous observation shows that $\underline{Q}_k \leq \widetilde{Q}_k$ pointwise for all $k \in \mathbb{N}$. Hence, taking the limits $\widetilde{Q}^* = \lim_{k\to\infty} \widetilde{Q}_k$ and $\underline{Q}^* = \lim_{k\to\infty} \underline{Q}_k$, we obtain $\underline{Q}^* \leq \widetilde{Q}^*$ pointwise. □

Now we are ready to present our main theoretical result.

Theorem 1. *Let $\widehat{T}$ be a calibrated dynamics model and $\pi^*(s) = \arg\max_a \underline{Q}^*(s,a)$ the greedy policy with respect to $\underline{Q}^*$. Assume that Assumption 1 holds. Then for any $s \in \mathcal{S}$, if there exists an action a such that $\underline{Q}^*(s,a) \geq \frac{r_{\min}}{1-\gamma}$, then $\pi^*(s)$ is a safe action.*

Proof. Lemma 3.2 implies that $\underline{Q}^*(s,a) \leq \widetilde{Q}^*(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.

As shown in the proof of Proposition 3.1, any unsafe action a' satisfies

$$\underline{Q}^*(s,a') \leq \widetilde{Q}^*(s,a') \leq \frac{r_{\max}(1-\gamma^{H^*}) - C\gamma^{H^*}}{1-\gamma} \tag{7}$$

Similarly if $\underline{Q}^*(s,a) \geq \frac{r_{\min}}{1-\gamma}$, we also have

$$\frac{r_{\min}}{1-\gamma} \leq \underline{Q}^*(s,a) \leq \widetilde{Q}^*(s,a) \tag{8}$$

so a is a safe action. Taking C as in inequality (3) guarantees that $\underline{Q}^*(s,a) > \underline{Q}^*(s,a')$, so the greedy policy π^* will choose a over a'. □

This theorem gives us a way to establish safety using only short-horizon predictions. The conclusion conditionally holds for any state s, but for s far from the observed states, we expect that $\widehat{T}(s,a)$ likely has to contain many states in order to satisfy the assumption that it contains the true next state, so that $\underline{Q}^*(s,a)$ will be very small and we may not have any action such that $\underline{Q}^*(s,a) \geq \frac{r_{\min}}{1-\gamma}$. However, it is plausible to believe that there can be such an a for the set of states in the replay buffer, $\{s : (s,a,r,s') \in \mathcal{D}\}$.

Algorithm 1 SMBPO

Require: Horizon H

Initialize empty buffers $\mathcal{D}$ and $\widehat{\mathcal{D}}$, an ensemble of probabilistic dynamics $\{\widehat{T}_{\theta_i}\}_{i=1}^N$, policy π_ϕ, critic Q_ψ.
Collect initial data using random policy, add to $\mathcal{D}$.
for episode $1, 2, \ldots$ **do**
  Collect episode using π_ϕ; add the samples to $\mathcal{D}$. Let ℓ be the length of the episode.
  Re-fit models $\{\widehat{T}_{\theta_i}\}_{i=1}^N$ by several epochs of SGD on $L_{\widehat{T}}(\theta_i)$ defined in (9)
  Compute empirical $r_{\min}$ and $r_{\max}$, and update C according to (3).
  for ℓ times **do**
    for n_{rollout} times (in parallel) **do**
      Sample $s \sim \mathcal{D}$.
      Startin from s, roll out H steps using π_ϕ and $\{\widehat{T}_{\theta_i}\}$; add the samples to $\widehat{\mathcal{D}}$.
    for n_{actor} times **do**
      Draw samples from $\mathcal{D} \cup \widehat{\mathcal{D}}$.
      Update Q_ψ by SGD on $L_Q(\psi)$ defined in (10) and target parameters $\bar{\psi}$ according to (11).
      Update π_ϕ by SGD on $L_\pi(\phi)$ defined in (12).

3.3 Practical algorithm

Based (mostly) on the framework described in the previous section, we develop a deep model-based RL algorithm. We build on practices established in previous deep model-based algorithms, particularly MBPO [Janner et al., 2019] a state-of-the-art model-based algorithm (which does not emphasize safety).

The algorithm is described in Algorithm 1. It follows a common pattern used by online model-based algorithms: alternate between collecting data, re-fitting the dynamics models, and improving the policy.

Following prior work [Chua et al., 2018, Janner et al., 2019], we employ an ensemble of (diagonal) Gaussian dynamics models $\{\widehat{T}_i\}_{i=1}^N$, where $\widehat{T}_i(s,a) = \mathcal{N}(\mu_i(s,a), \operatorname{diag}(\sigma_i^2(s,a)))$, in an attempt to capture both aleatoric and epistemic uncertainties. Each model is trained via maximum likelihood on all the data observed so far:

$$L_{\widehat{T}}(\theta_i) = -\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}} \log \widehat{T}_{\theta_i}(s', r \mid s, a) \tag{9}$$

However, random differences in initialization and mini-batch order while training lead to different models.

The models are used to generate additional samples for fitting the Q function and updating the policy. In MBPO, this takes the form of short model-based rollouts, starting from states in $\mathcal{D}$, to reduce the risk of compounding error. At each step in the rollout, a model $\widehat{T}_i$ is randomly chosen from the ensemble and used to predict the next state. The rollout horizon H is chosen as a hyperparameter, and ideally exceeds the (unknown) H^* from Assumption 1. In principle, one can simply increase H to ensure it is large enough, but this increases the opportunity for compounding error.

MBPO is based on the soft actor-critic (SAC) algorithm, a widely used off-policy maximum-entropy actor-critic algorithm [Haarnoja et al., 2018a]. The Q function is updated by taking one or more SGD steps on the objective

$$\begin{aligned} L_Q(\psi) &= \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}\cup\widehat{\mathcal{D}}}[(Q_\psi(s,a) - (r + \gamma V_{\bar{\psi}}(s'))^2] \\ \text{where} \quad V_{\bar{\psi}}(s') &= \begin{cases} -C/(1-\gamma) & \textsc{Unsafe}(s') \\ \mathbb{E}_{a'\sim\pi(s')}[Q_{\bar{\psi}}(s',a') - \alpha \log \pi_\phi(a' \mid s')] & \neg\textsc{Unsafe}(s') \end{cases} \end{aligned} \tag{10}$$

Here $\bar{\psi}$ are parameters of a "target" Q function which is updated via an exponential moving average towards ψ:

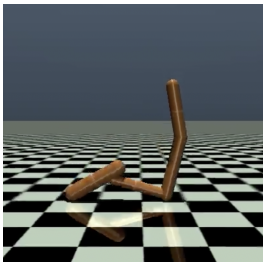
$$\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi} \tag{11}$$

for a hyperparameter $\tau \in (0,1)$ which is often chosen small, e.g., 0.005. This is a common practice used to promote stability in deep RL, originating from [Lillicrap et al., 2015]. The scalar α is a

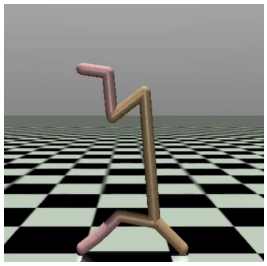
[1] $\mathcal{P}(X)$ is the powerset of a set X.

(a) Navigation1

(b) Navigation2



(c) Hopper



(d) Cheetah-no-flip

Figure 2: The tasks to be solved. For hopper and cheetah-no-flip, we show examples of safety violations.

hyperparameter of SAC which controls the tradeoff between entropy and reward. We learn α using the procedure suggested by Haarnoja et al. [2018b].

Note that in (10), we are fitting to the average TD target across models, rather than the min, even though we proved Theorem 3.2 using the Bellmin operator. We found that taking the average worked better empirically, likely because the min was overly conservative and harmed exploration.

The policy is updated by taking one or more steps to minimize

$$L_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}\cup\widehat{\mathcal{D}},a\sim\pi_\phi(s)}[\alpha \log \pi_\phi(a \mid s) - Q_\psi(s,a)]. \tag{12}$$

4 Experiments

In the experimental evaluation, we compare our algorithm to several model-free safe RL algorithms, as well as MBPO, on several continuous control tasks. Additional experimental details, including hyperparameter selection, are given in the Appendix.

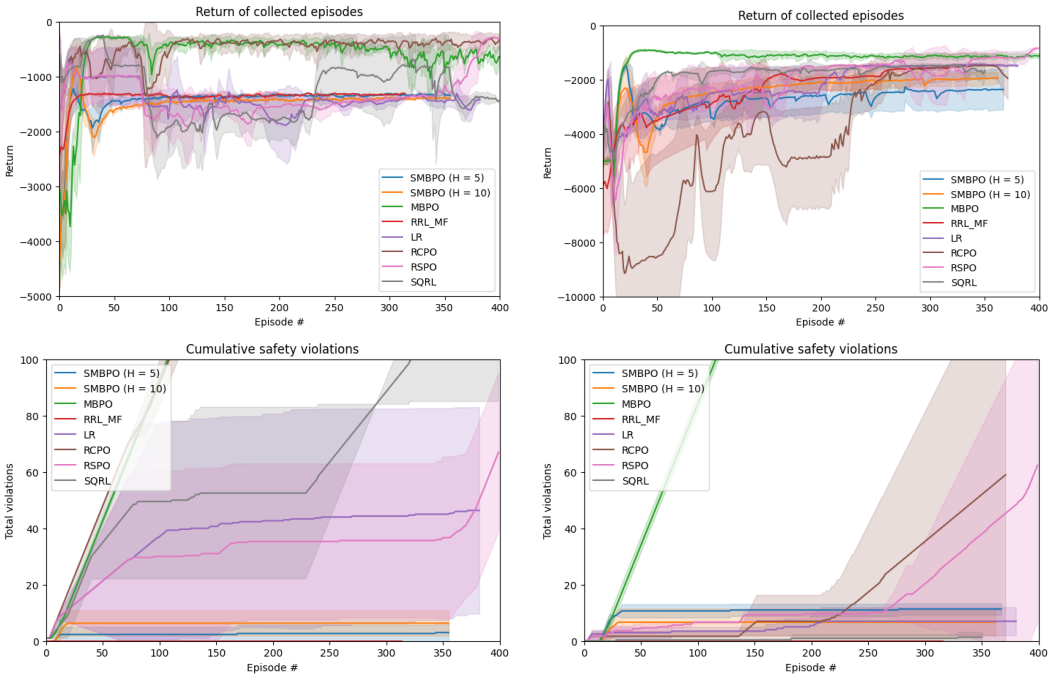
4.1 Tasks

The tasks are described below:
Navigation1 and **Navigation2** from Thananjeyan et al. [2020]: The agent controls a 2D pointmass towards a goal position. The reward is the negative Euclidean distance to the goal, and the safety condition is to avoid obstacles which have been placed in the environment.
Hopper: The standard hopper environment from OpenAI Gym, except with the "alive bonus" (a constant of +1) removed from the reward so that the task reward does not implicitly encode the safety objective. The reward is the change in x position minus a small penalty on the magnitude of the action, and the safety condition is the usual termination condition for this task, which corresponds to the robot falling over.
Cheetah-no-flip: The standard half-cheetah environment from OpenAI Gym. The reward is the change in x position minus a small penalty on the magnitude of the action, and the safety condition being that the robot's head does not come into contact with the ground.

For all of the tasks, the episode is terminated once the safety requirement is violated.

4.2 Algorithms

We compare against the following algorithms:
MBPO: Corresponds to SMBPO with $C = 0$.
Recovery RL, model-free (RRL-MF): Trains a critic to estimate the safety separately from the reward, as well as a recovery policy which is invoked when the critic predicts risk of a safety



(a) Navigation1

(b) Navigation2

Figure 3: Results for the navigation tasks. We run 3 seeds for each algorithm independently and average the results. The curves indicate mean of different seeds and the shaded areas indicate one standard deviation centered at the mean. Our algorithm achieves very few safety violations, despite being given only 1000 initial samples compared to 20000 for the model-free algorithms.

violation.

Lagrangian relaxation (LR): Forms a Lagrangian to implement a constraint on the risk, updating the dual variable via dual gradient descent.

Safety Q-functions for RL (SQRL): Also formulates a Lagrangian relaxation, and uses a filter to reject actions which are too risky according to the safety critic.

Reward constrained policy optimization (RCPO): Uses policy gradient to optimize a reward function which is penalized according to the safety critic.

Risk-sensitive policy optimization (RSPO): Forms a Lagrangian, but uses a fixed schedule for the dual variable rather than updating it via dual gradient descent.

All of the above algorithms except for MBPO are as implemented in the Recovery RL paper [Thananjeyan et al., 2020] and its publicly available codebase[2]. For the navigation task, we use the hyperparameters provided by [Thananjeyan et al., 2020], while for the other tasks we follow the hyperparameter tuning procedure described in their paper. A recent work [Bharadhwaj et al., 2020] can also serve as a baseline but the code has not been released.

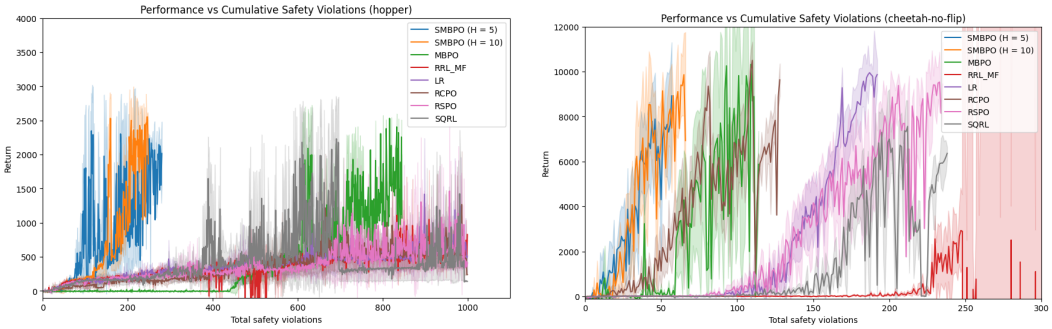
Our algorithm requires very little hyperparameter tuning. We use $\gamma = 0.99$ in all experiments. We try both $H = 5$ and $H = 10$ and usually find out $H = 10$ works slightly better.

4.3 Results

The main criterion in which we are interested is performance (return) vs. the cumulative number of safety violations. The results are plotted in Figure 3 and Figure 4. We see that our algorithm performs favorably compared to model-free alternatives in terms of this tradeoff.

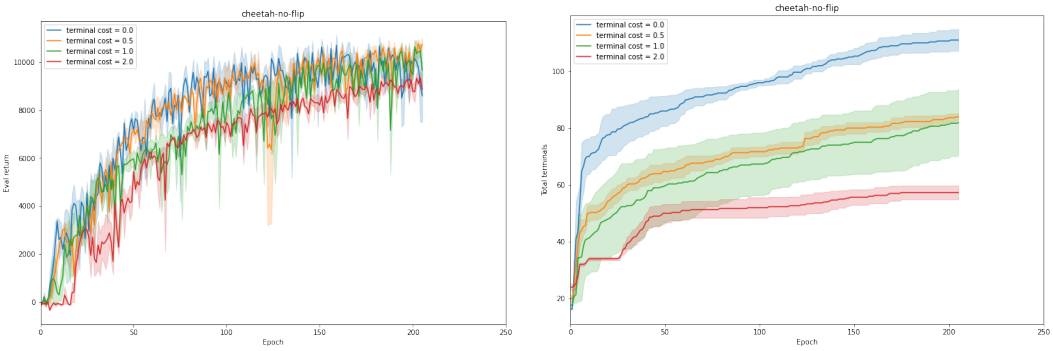
We also show in Figure 5 that hard-coding the value of C leads to an intuitive tradeoff between performance and safety violations. With a larger C, SMBPO incurs substantially fewer safety violations, although the total rewards are learned slower.

[2] `https://github.com/abalakrishna123/recovery-rl`



(a) Hopper (b) Cheetah-no-flip

Figure 4: Undiscounted return of episodes collected vs. total safety violations. We run 3 seeds for each algorithm independently and average the results. The curves indicate mean of different seeds and the shaded areas indicate one standard deviation centered at the mean. Note that the model-free baselines employ additional pretraining data which we do *not* count towards their violation count in these plots (i.e. they get some "free" violations). Even so, SMBPO yields the best results.



(a) Performance (b) Cumulative safety violations

Figure 5: Increasing C makes exploration more conservative, leading to fewer safety violations but potentially harming performance. This is what we expected: A larger C focuses more on the safety requirement and learns more conservatively.

5 Related Work

Safe Reinforcement Learning Many of the prior work correct the action locally, that is, changing the action when the action is detected to lead to unsafe state. Dalal et al. [2018] linearizes the dynamics and adds a layer on the top of the policy for correction. Bharadhwaj et al. [2020] uses rejection sample to ensure the action meets the safety requirement. Thananjeyan et al. [2020] either trains a backup policy which is only used to guarantee safety, or uses model-predictive control (MPC) to find the best action sequence. MPC could also be applied in the short-horizon setting that we consider here, but it involves high runtime cost that may not be acceptable for real-time robotic control. Also, MPC only optimizes for rewards under the short horizon and can lead to suboptimal performance on tasks that require longer-term considerations [Tamar et al., 2017].

Other works aim to solve the constrained MDP more efficiently and better, with Lagrangian methods being applied widely. The Lagrangian multipliers can be a fixed hyperparameter, or adjusted by the algorithm [Tessler et al., 2018, Stooke et al., 2020]. The policy training might also have issues. The issue that the policy might change too fast so that it's no longer safe is addressed by building a trust region of policies [Achiam et al., 2017] and further projection to a safer policy [Yang et al., 2020], and another issue of too optimistic policy is address by Bharadhwaj et al. [2020] by using conservative policy updates. Expert information can greatly improve the training-time safety. Srinivasan et al. [2020], Thananjeyan et al. [2020] are provided offline data, while Turchetta et al. [2020] is provided

interventions which are invoked at dangerous states and achieves zeros safety violations during training.

Returnability is also considered by Eysenbach et al. [2018] in practice, which trains a policy to return to the initial state, or by Roderick et al. [2021] in theory, which designs a PAC algorithm to train a policy without safety violations. Bansal et al. [2017] gives a brief overview of Hamilton-Jacobi Reachability and its recent progress.

Model-based Reinforcement Learning Model-based reinforcement learning, which additionally learns the dynamics model, has gained its popularity due to its superior sample efficiency. Kurutach et al. [2018] uses an ensemble of models to produce imaginary samples to regularize leaerning and reduce instability. The use of model ensemble is further explored by Chua et al. [2018], which studies different methods to sample trajectories from the model ensemble. Based on Chua et al. [2018], **?** combines policy networks with online learning. Luo et al. [2019] derives a lower bound of the policy in the real environment given its performance in the learned dynamics model, and then optimizes the lower bound stochastically. Our work is based on Janner et al. [2019], which shows the learned dynamics model doesn't generalize well for long horizon and proposes to use short model-generated rollouts instead of a full episodes. Dong et al. [2020] studies the expressivity of Q function and model and shows that at some environments, the model is much easier to learn than the Q function.

6 Conclusion

We consider the problem of safe exploration in reinforcement learning, where the goal is to discover a policy that maximizes the expected return, but additionally desire the training process to incur minimal safety violations. In this work, we assume access to a user-specified function which can be queried to determine whether or not a given state is safe. We have proposed a model-based algorithm that can exploit this information to anticipate safety violations before they happen and thereby avoid them. Our algorithm compares favorably to state-of-the-art model-free safe exploration methods in terms of the tradeoff between performance and total safety violations, and in terms of sample complexity.

7 Limitations

- The true H^*, if there even exists such an H^* (it is hard to verify), is not known in general, forcing us to choose it as a hyperparameter.
- The model-based approach relies on the model making reasonably accurate predictions. We had trouble scaling our algorithm to tasks with higher-dimensional state spaces, where prediction is more challenging.

8 Negative Social Impact

We believe that most of the impacts of this work are positive, but there are some risks:

- Improved automation technology will likely lead to elimination of some kinds of jobs. (Although it may create others.)
- Labeling an algorithm as "safe" may lead people to trust it more than they otherwise would. It is important for practitioners of safe RL to understand the limitations of the algorithms they use.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michael L. Littman. Combating the compounding-error problem with a multi-step model. *arXiv preprint*, abs/1905.13320, 2019. URL `http://arxiv.org/abs/1905.13320`.

Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

Marc G. Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C. Machado, Subhodeep Moitra, Sameera S. Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. page 77–82, 2020.

Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Kefan Dong, Yuping Luo, Tianhe Yu, Chelsea Finn, and Tengyu Ma. On the expressivity of neural networks for deep reinforcement learning. In *International Conference on Machine Learning*, pages 2627–2637. PMLR, 2020.

B Eysenbach, S Gu, J Ibarz, and S Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *6th International Conference on Learning Representations (ICLR 2018)*. OpenReview. net, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

Shixiang Gu, Ethan Holly, Timothy P. Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation. abs/1610.00633, 2016. URL `http://arxiv.org/abs/1610.00633`.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. In *International Conference on Machine Learning*, pages 1861–1870, 2018b.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2019.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. *arXiv preprint arXiv:1807.03858*, 2018.

Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BJe1E2R5KX`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Melrose Roderick, Vaishnavh Nagarajan, and Zico Kolter. Provably safe pac-mdp exploration using analogies. In *International Conference on Artificial Intelligence and Statistics*, pages 1216–1224. PMLR, 2021.

Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603*, 2020.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020.

Aviv Tamar, Garrett Thomas, Tianhao Zhang, Sergey Levine, and Pieter Abbeel. Learning from the hindsight plan — episodic mpc improvement. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 336–343, 2017. doi: 10.1109/ICRA.2017.7989043.

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *arXiv preprint arXiv:2010.15920*, 2020.

Matteo Turchetta, Andrey Kolobov, Shital Shah, Andreas Krause, and Alekh Agarwal. Safe reinforcement learning via curriculum induction. *arXiv preprint arXiv:2006.12136*, 2020.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Accelerating safe reinforcement learning with constraint-mismatched policies. *arXiv preprint arXiv:2006.11645*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In Section 8.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]

(b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In Appendix A.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In Appendix A.2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Proofs

Proof of Lemma 3.2 ($\underline{\mathcal{B}}^*$ is a γ-contraction in ∞-norm). First observe that for any functions f and g,

$$|\max_x f(x) - \max_x g(x)| \leq \max_x |f(x) - g(x)| \tag{13}$$

To see this, suppose $\max_x f(x) > \max_x g(x)$ (the other case is symmetric) and let $\tilde{x} = \arg\max_x f(x)$. Then

$$|\max_x f(x) - \max_x g(x)| = f(\tilde{x}) - \max_x g(x) \leq f(\tilde{x}) - g(\tilde{x}) \leq \max_x |f(x) - g(x)| \tag{14}$$

We also note that (13) implies

$$|\min_x f(x) - \min_x g(x)| \leq \max_x |f(x) - g(x)| \tag{15}$$

since $\min_x f(x) = -\max_x(-f(x))$. Thus for any $Q, Q' : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

$$\begin{aligned}
\|\underline{\mathcal{B}}^* Q - \underline{\mathcal{B}}^* Q'\|_\infty &= \sup_{s,a} |\underline{\mathcal{B}}^* Q(s,a) - \underline{\mathcal{B}}^* Q'(s,a)| \\
&= \gamma \sup_{s,a} \left| \min_{s' \in \widehat{T}(s,a)} \max_{a'} Q(s',a') - \min_{s' \in \widehat{T}(s,a)} \max_{a'} Q'(s',a') \right| \\
&\leq \gamma \sup_{s,a} \max_{s' \in \widehat{T}(s,a)} \left| \max_{a'} Q(s',a') - \max_{a'} Q'(s',a') \right| \\
&\leq \gamma \sup_{s',a'} |Q(s',a') - Q'(s',a')| \\
&= \gamma \|Q - Q'\|_\infty
\end{aligned}$$

Hence $\underline{\mathcal{B}}^*$ is indeed a γ-contraction. □

A.2 Implementation details and hyperparameters

In this appendix we provide additional details regarding the algorithmic implementation, including hyperparameter selection.

Here are some additional details regarding the (S)MBPO implementation:

- All neural networks are implemented in PyTorch [Paszke et al., 2019] and optimized using the Adam optimizer [Kingma and Ba, 2014] and batch size 256.
- The dynamics models use a branched architecture, where a shared trunk computes an intermediate value $z = h_{\theta_1}([s, a])$ which is then passed to branches $\mu_{\theta_2}(z)$ and $\sigma_{\theta_3}(z)$. All three networks are implemented as multi-layer perceptrons (MLPs) with ReLU activation and 200 hidden width. The h_θ network has 3 layers (with ReLU on the final layer too), while μ_{θ_2} and σ_{θ_3} each have one hidden layer (no ReLU on final layer).
- When re-fitting the models, we train for 50 epochs, linearly annealing the learning rate from 3e-4 to 1e-4.
- The networks for the Q functions and policies all have two hidden layers of width 256.
- We use a learning rate of 3e-4 for the Q function and 1e-4 for the policy.
- Following Fujimoto et al. [2018], we store two copies of the weights for Q (and $\bar{Q}$), trained the same way but with different initializations. When computing the target $\bar{Q}$ in equation (10) and when computing Q in equation (12), we take the minimum of the two copies' predictions. When computing the Q in equation (10), we compute the loss for both copies of the weights and add the two losses.
- When sampling batches of data from $\mathcal{D} \cup \widehat{\mathcal{D}}$, we take 10% of the samples from $\mathcal{D}$ and the remainder from $\widehat{\mathcal{D}}$.

The model-free algorithms have their own hyperparameters, but all share γ_{safe} and ϵ_{safe}. We use the hyperparameters suggested by Thananjeyan et al. [2020] for the navigation tasks but tune the model-free algorithms on the hopper and cheetah tasks. Following Thananjeyan et al. [2020], we tune γ_{safe} and ϵ_{safe} for recovery RL first, then hold those fixed for all algorithms and tune any remaining algorithm-specific hyperparameters. All these hyperparameters are given in the tables below:

Name	Which algorithm(s)?	Choices	Selected (hopper, cheetah)
γ_{safe}	all	0.65, 0.7, 0.75, 0.8, 0.85	0.65, 0.7
ϵ_{safe}	all	0.1, 0.15, 0.2, 0.25, 0.3	0.15, 0.15
ν	LR, SQRL	10, 20, 50, 100, 200, 500, 1000, 2000, 5000	100, 5000
ν_0	RSPO	10, 20, 50, 100, 200, 500, 1000, 2000, 5000	500, 5000
λ	RCPO	10, 20, 50, 100, 200, 500, 1000, 2000, 5000	10, 100

We run our experiments using a combination of NVIDIA GeForce GTX 1080 Ti, TITAN Xp, and TITAN RTX GPUs from our internal cluster. A single run of (S)MBPO takes as long as 72 hours on a single GPU.